SPDK NVMe-oF TCP (Target & Initiator) Performance Report Release 20.07

Testing Date: August 2020

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)

Benjamin Walker (benjamin.walker@intel.com)

Seth Howell (seth.howell@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

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)
Memory	12 x 32GB Hynix HMA84GR7AFR4N-VK, DDR4, 2666MHz Total of 384GB
Operating System	Fedora 30
BIOS	3.1a
Linux kernel version	5.4.14-100.fc30
SPDK version	SPDK 20.07
Storage	**OS:** 1x 120GB Intel SSDSC2BB120G4 **Storage Target**: 16x Intel® SSD DC P4610™ 1.6TB (FW: QDV10190) (8 on each CPU socket)
NIC	2x 100GbE Mellanox ConnectX-5 NICs. Both ports connected. 1 NIC per CPU socket.

Initiator 1 Configuration

Item	Description
Server Platform	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)
Memory	6 x 32GB Micron M393A1G40EB1-CRC, DDR4, 2933MHz Total 192GBs
Operating System	Fedora 30
BIOS	02.01.0008 03/19/2019
Linux kernel version	5.4.14-100.fc30
SPDK version	SPDK 20.07
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

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)
Memory	6 x 32GB Micron M393A1G40EB1-CRC, DDR4, 2933MHz Total 192GBs
Operating System	Fedora 30
BIOS	02.01.0008 03/19/2019
Linux kernel version	5.4.14-100.fc30
SPDK version	SPDK 20.07
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

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 Fedora 5.4.14-100.fc30 kernel version available from DNF repository with default patches for spectre-meltdown issue enabled.

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

Introduction to SPDK NVMe-oF (Target & Initiator)

The NVMe over Fabrics (NVMe-oF) protocol extends the parallelism and efficiencies of the NVM Express* (NVMe) block protocol over network fabrics such as RDMA (iWARP, RoCE), InfiniBand™, Fibre Channel 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).

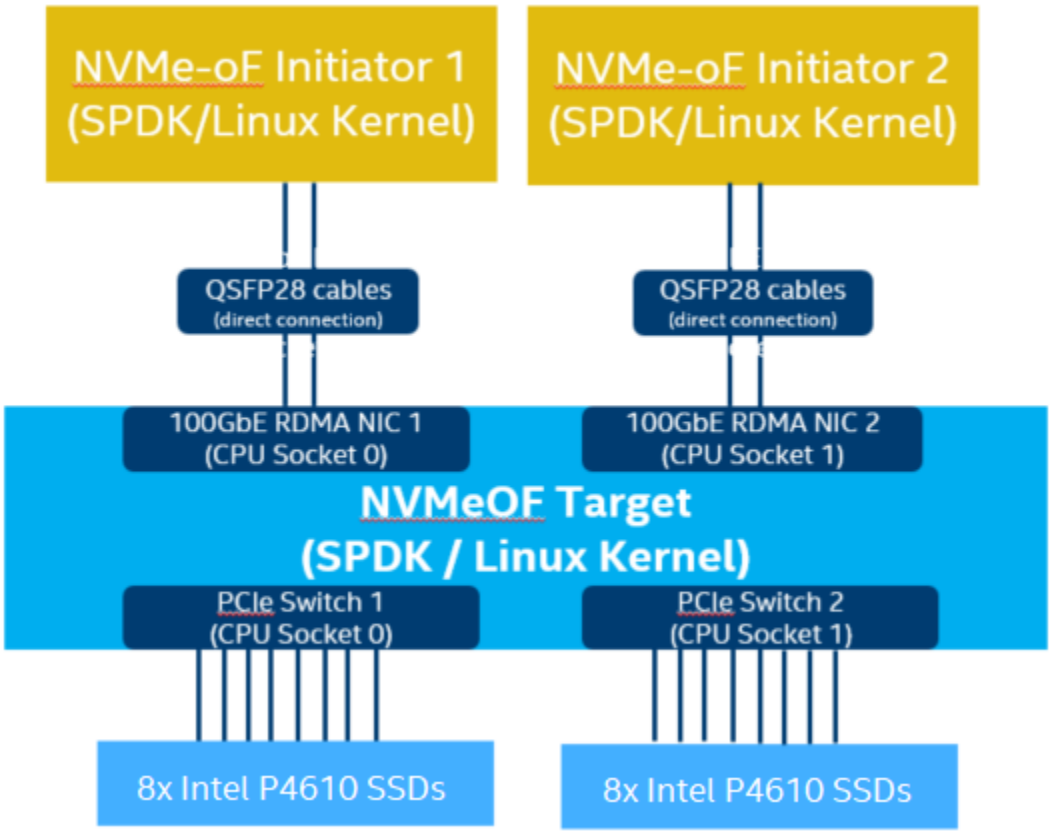

Figure 1: High-Level NVMe-oF TCP performance testing setup

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 write and 4KB 70% read 30% write we ran preconditioning once before running all of the workload to ensure that NVMe devices reached higher IOPS so that we can saturate the network.

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

	construct_nvme_bdev -t PCIe -b Nvme7 -a 0000:67:00.0
	construct_nvme_bdev -t PCIe -b Nvme8 -a 0000:b5:00.0
	construct_nvme_bdev -t PCIe -b Nvme9 -a 0000:b6:00.0
	construct_nvme_bdev -t PCIe -b Nvme10 -a 0000:b7:00.0
	construct_nvme_bdev -t PCIe -b Nvme11 -a 0000:b8:00.0
	construct_nvme_bdev -t PCIe -b Nvme12 -a 0000:b9:00.0
	construct_nvme_bdev -t PCIe -b Nvme13 -a 0000:ba:00.0
	construct_nvme_bdev -t PCIe -b Nvme14 -a 0000:bb:00.0
	construct_nvme_bdev -t PCIe -b Nvme15 -a 0000:bc:00.0
	
	nvmf_create_transport -t TCP
	(creates TCP transport layer with default values:
	trtype: "TCP"
	max_queue_depth: 128
	max_qpairs_per_ctrlr: 64
	in_capsule_data_size: 4096
	max_io_size: 131072
	"io_unit_size": 131072,
	max_aq_depth: 128
	num_shared_buffers: 4096
	buf_cache_size: 32,
	"c2h_success": true,
	"dif_insert_or_strip": false,
	"sock_priority": 0
	)
	
	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
	
	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** [Nvme] TransportId "trtype:TCP adrfam:IPv4 traddr:20.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode1" Nvme0 TransportId "trtype:TCP adrfam:IPv4 traddr:20.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode2" Nvme1 TransportId "trtype:TCP adrfam:IPv4 traddr:20.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode3" Nvme2 TransportId "trtype:TCP adrfam:IPv4 traddr:20.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode4" Nvme3 TransportId "trtype:TCP adrfam:IPv4 traddr:20.0.1.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode5" Nvme4 TransportId "trtype:TCP adrfam:IPv4 traddr:20.0.1.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode6" Nvme5 TransportId "trtype:TCP adrfam:IPv4 traddr:20.0.1.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode7" Nvme6 TransportId "trtype:TCP adrfam:IPv4 traddr:20.0.1.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode8" Nvme7 **FIO.conf** [global] ioengine=/tmp/spdk/examples/bdev/fio_plugin/fio_plugin spdk_conf=/tmp/spdk/bdev.conf thread=1 group_reporting=1 direct=1 norandommap=1 rw=randrw rwmixread={100, 70, 0}

	bs=4k iodepth={1, 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

Test Result: 4KB 100% Random Read IOPS, QD=128

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	1769.42	453.0	4552.3
4 cores	7389.58	1891.7	1088.3
8 cores	14352.49	3674.2	558.7
12 cores	18237.04	4668.7	439.3
16 cores	19529.59	4999.6	409.7
24 cores	20474.43	5241.4	390.7
32 cores	20296.34	5195.9	394.5
40 cores	20731.32	5307.2	386.8

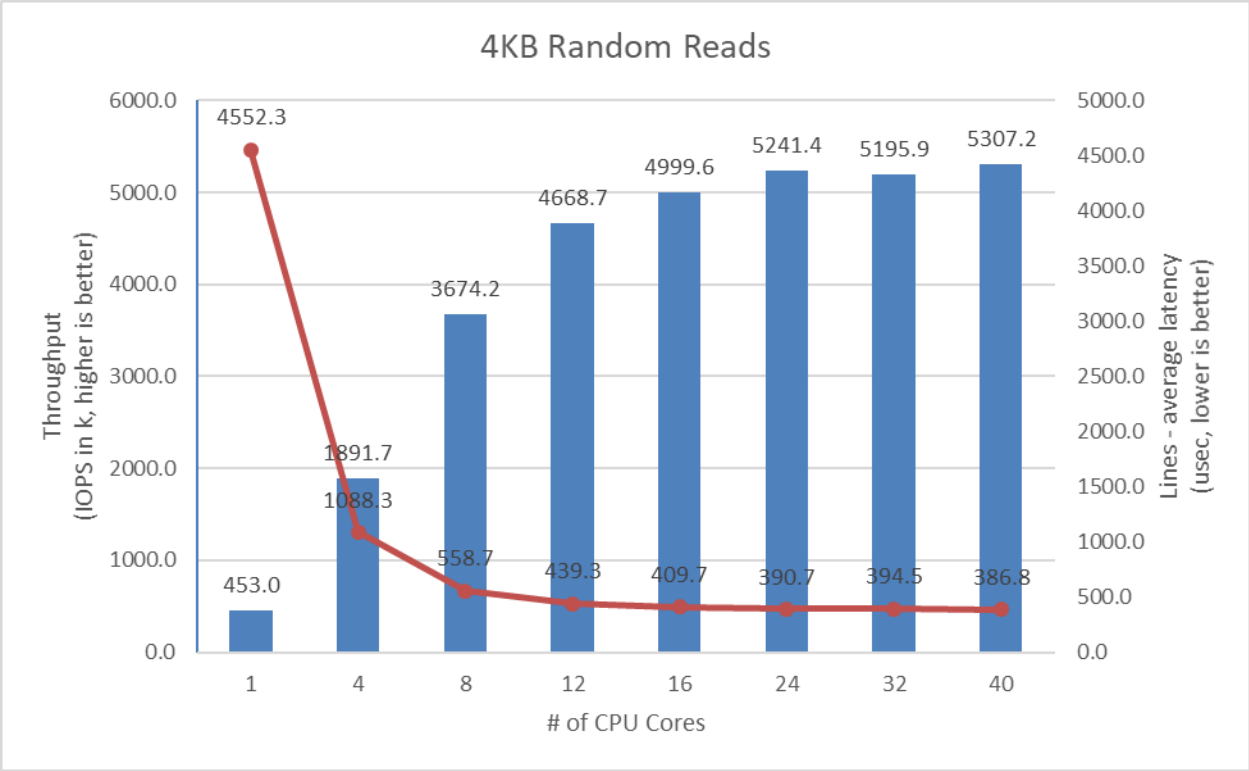

Figure 2: SPDK NVMe-oF TCP Target I/O core scaling: IOPS vs. Latency while running 4KB 100% Random Read workload at QD = 128

4KB Random Write Results

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

Test Result: 4KB 100% Random Writes IOPS, QD=128

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	1006.35	257.6	8036.4
4 cores	4255.58	1089.4	1883.6
8 cores	7620.17	1950.8	1046.3
12 cores	10588.19	2710.6	754.9
16 cores	12340.15	3159.1	648.7
24 cores	14427.97	3693.6	553.9
32 cores	15335.30	3925.8	524.0
40 cores	15720.55	4024.5	509.5

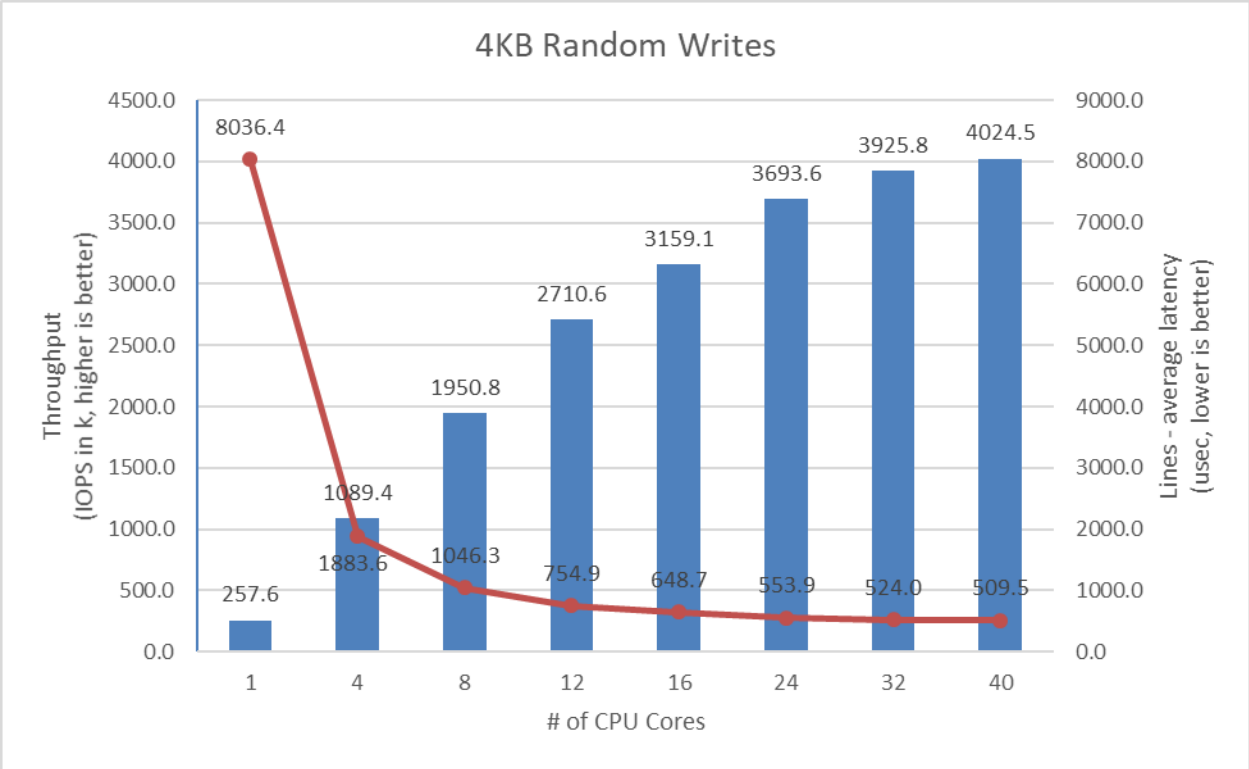

Figure 3: SPDK NVMe-oF TCP Target I/O core scaling: IOPS vs. Latency while running 4KB 100% Random Write Workload at QD=128

4KB Random Read-Write Results

Test Result: 4KB Random Read/Write 70%/30% IOPS, QD=192

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	1355.09	346.9	8850.7
4 cores	5550.17	1420.8	2158.1
8 cores	11097.99	2841.1	1078.3
12 cores	15214.74	3895.0	787.0
16 cores	17998.52	4607.6	663.8
24 cores	19538.26	5001.8	611.3
32 cores	18240.25	4669.5	655.3
40 cores	19248.25	4927.5	620.3

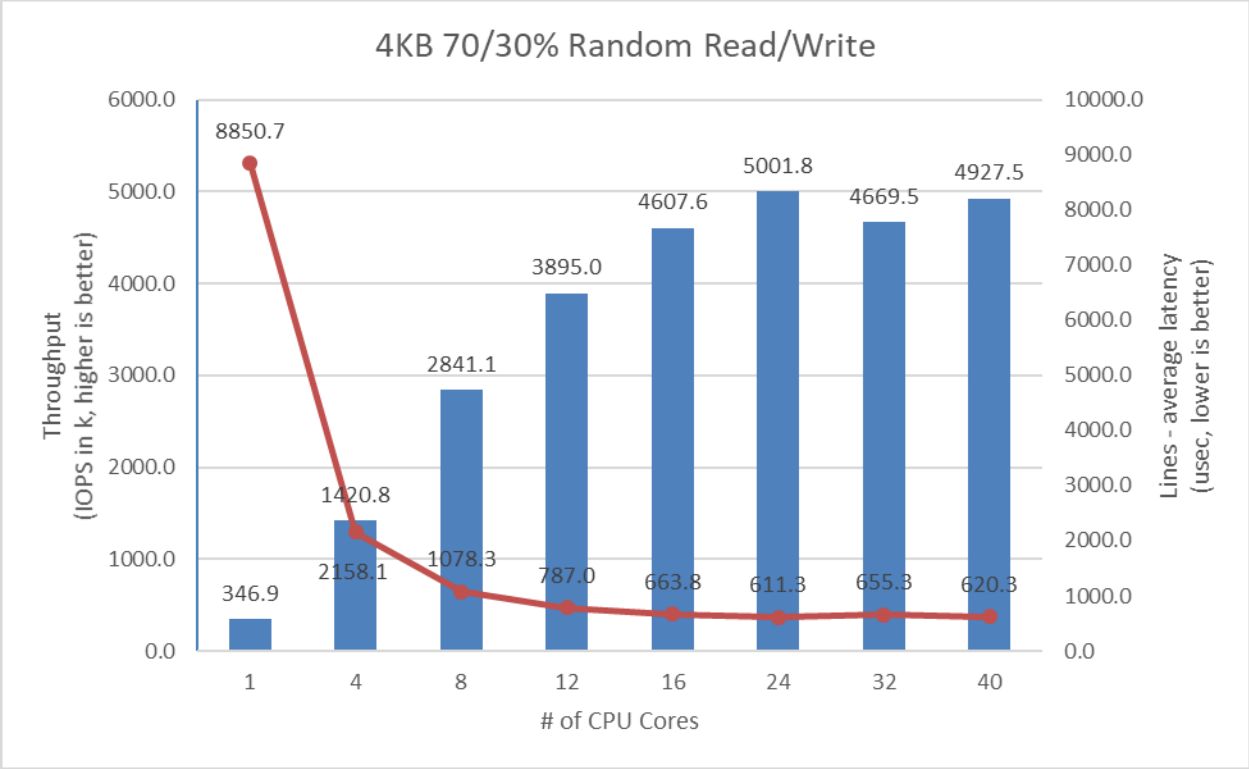

Figure 4: SPDK NVMe-oF TCP Target I/O core scaling: IOPS vs. Latency while running 4KB Random 70/30 Read/Write workload at QD=192

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.

Test Result: 128KB 100% Sequential Reads, QD=8

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	11778.24	94.2	1366.1
4 cores	23330.64	186.6	685.5
8 cores	23563.36	188.5	678.7
12 cores	23632.86	189.1	676.7
16 cores	23518.38	188.1	680.0
24 cores	23525.37	188.2	679.8
32 cores	23615.23	188.9	677.2
40 cores	23440.11	187.5	682.3

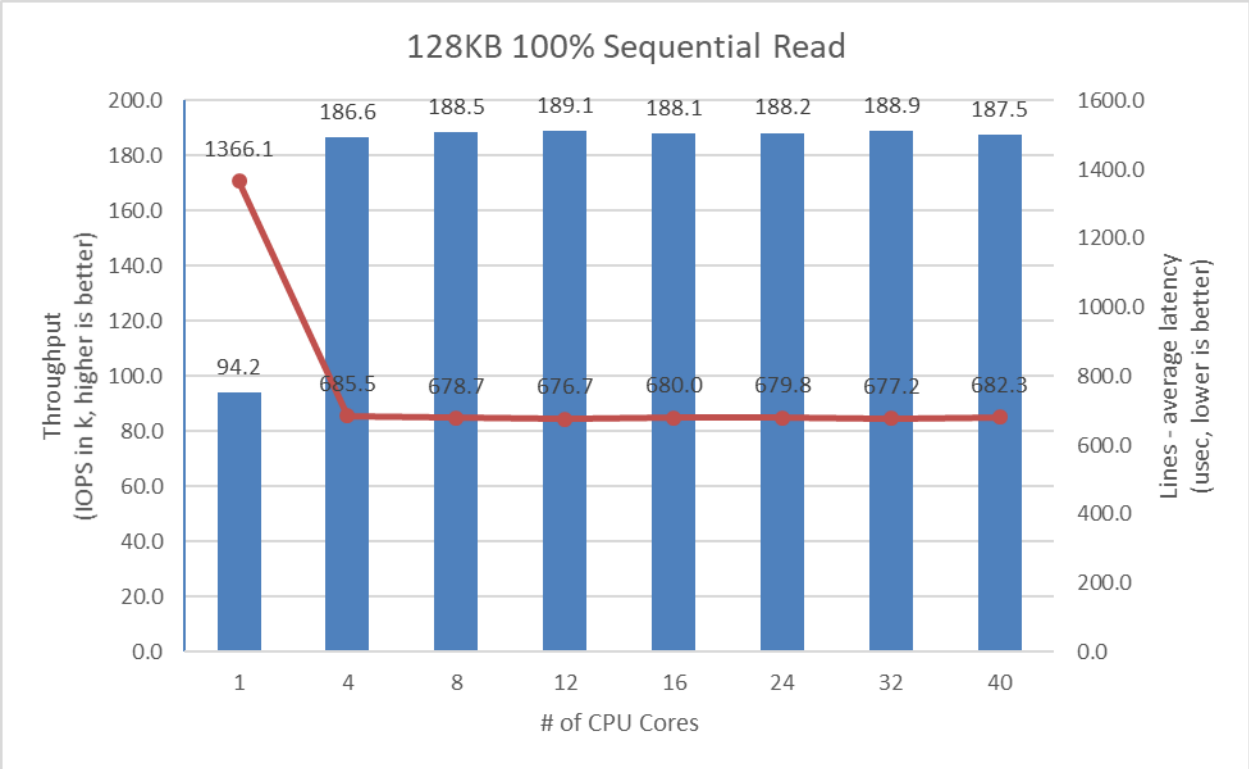

Figure 5: SPDK NVMe-oF TCP Target I/O core scaling: IOPS vs. Latency while running 128KB 100% Sequential Read Workload at QD=8 and initiator FIO numjobs=2

Test Result: 128KB 100% Sequential Writes, QD=8

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	1900.26	15.2	8421.2
4 cores	8293.24	66.3	1929.8
8 cores	13937.74	111.5	1149.5
12 cores	18906.19	151.2	846.5
16 cores	20810.93	166.5	769.5
24 cores	19502.42	156.0	827.3
32 cores	22000.24	176.0	728.4
40 cores	21561.80	172.5	743.5

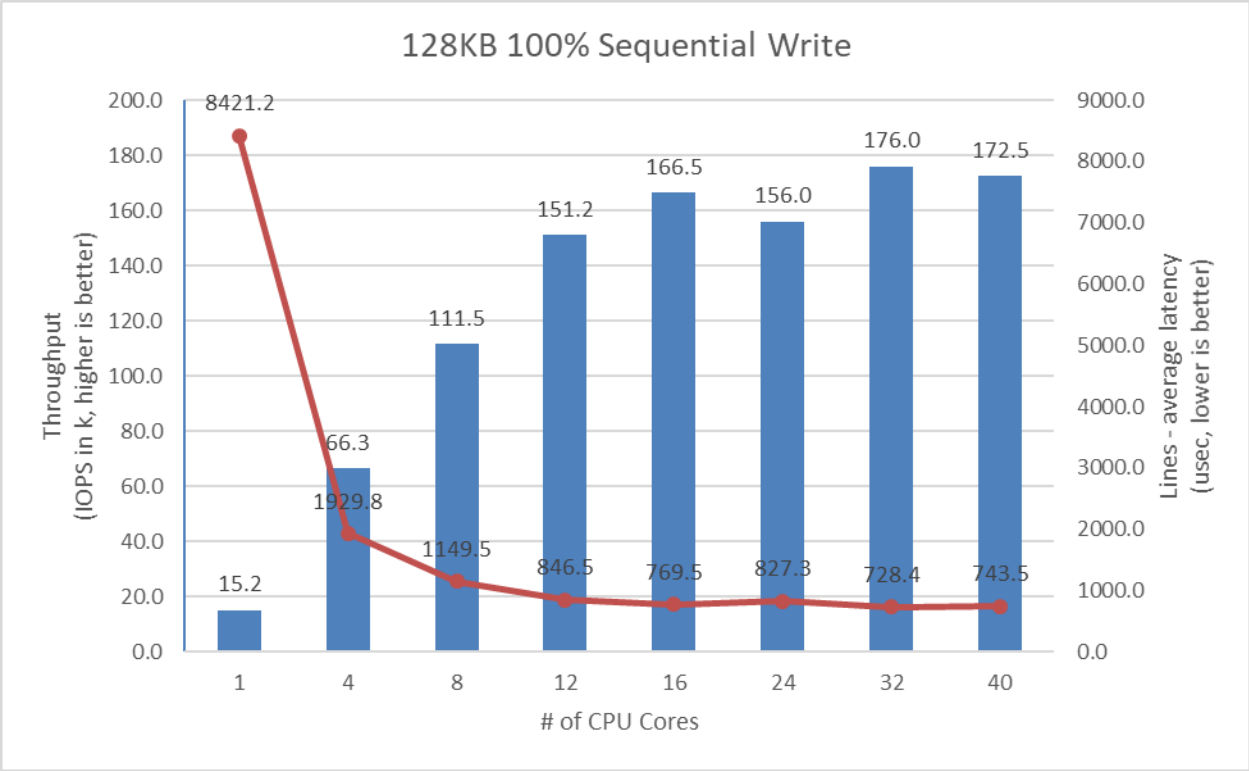

Figure 6: SPDK NVMe-oF TCP Target I/O core scaling: IOPS vs. Latency while running 128KB 100% Sequential Write Workload at QD=8 and Initiator FIO numjobs=2

Test Result: 128KB Sequential 70% Reads 30% Writes, QD=8

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	4478.45	35.8	1800.4
4 cores	14973.54	119.8	534.3
8 cores	21416.43	171.3	373.7
12 cores	23850.45	190.8	336.2
16 cores	23717.24	189.7	337.2
24 cores	24872.64	199.0	322.0
32 cores	24172.65	193.4	333.5
40 cores	25722.90	205.8	310.8

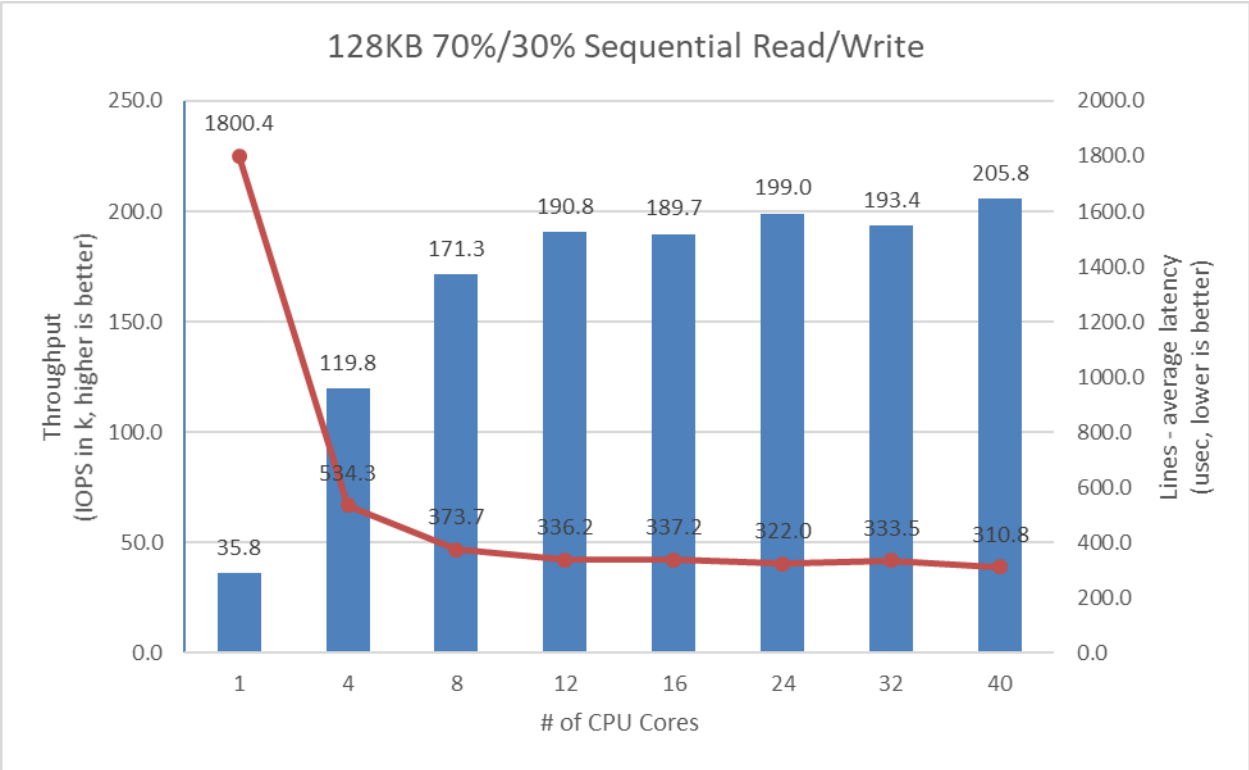

Figure 7: SPDK NVMe-oF TCP Target I/O core scaling: IOPS vs. Latency while running 128KB Sequential 70% Read 30% Write Workload at QD=8 and Initiator FIO numjobs=2

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 over 100GbE network speeds. Adding more CPUs to Target configuration results in non-linear performance gains peaking at about 5.3 million IOPs at 40 CPU cores, almost completely saturating 200GbE network link.

2. For Random Write workload performance scales linearly up 12 CPU cores, reaching 2.7 million IOPS, saturating a 100GbE link. Peak performance is at 40CPU cores with 4.0 million IOPS. 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 12 CPU cores with 3.9M IOPS and reaching peak performance of 5 million IOPS at 24 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 NVMe drives, which if preconditioned, 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 182 Gbps was observed at 4 CPU cores. For the 128K Sequential Writes, the throughput scaled linearly to 111 Gbps at 8 cores, however, beyond 8 cores the scaling was non-linear with a peak throughput of 172 Gbps observed at 32 CPU cores. For the 128K Sequential 70/30 Read/Write workload the scaling was non-linear, we observed 120 Gbps with just 4 cores and 171 Gbps at 8 cores, the peak throughput of 200 Gbps was observed at 40 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.

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

	TransportId "trtype:TCP adrfam:IPv4 traddr:10.0.1.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode6" Nvme13 TransportId "trtype:TCP adrfam:IPv4 traddr:10.0.1.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode7" Nvme14 TransportId "trtype:TCP adrfam:IPv4 traddr:10.0.1.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode8" Nvme15 **FIO.conf** **For 1 CPU initiator configuration:** [global] ioengine=/tmp/spdk/examples/bdev/fio_plugin/fio_plugin spdk_conf=/tmp/spdk/bdev.conf thread=1 group_reporting=1 direct=1 norandommap=1 rw=randrw rwmixread={100, 70, 0} bs=4k iodepth={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

Test Result: 4KB 100% Random Read, QD=128

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	1171.84	300.0	6779.1
4 cores	4697.35	1202.5	1693.5
8 cores	8626.60	2208.4	912.6
12 cores	11276.56	2886.8	694.4
16 cores	10169.76	2603.5	777.5
24 cores	9212.92	2358.5	860.0
32 cores	8282.96	2120.4	962.9
40 cores	7627.35	1952.6	1041.8

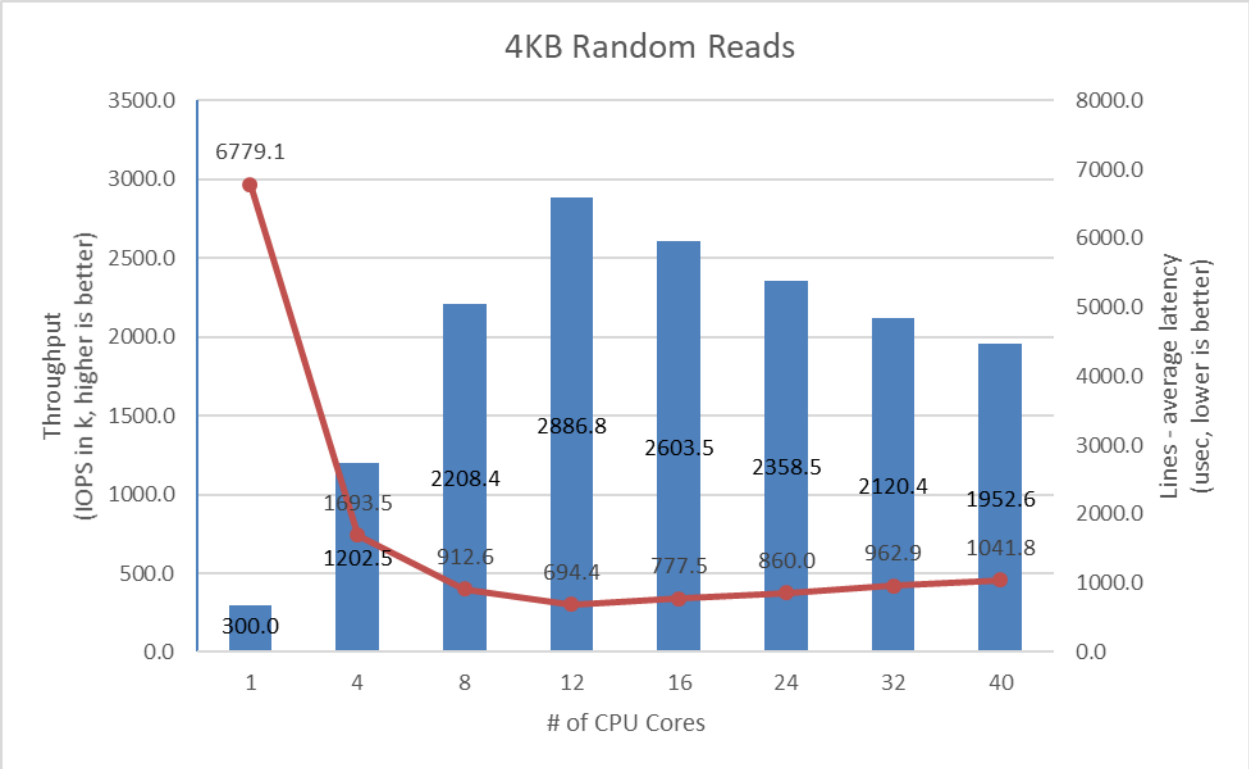

Figure 8: SPDK NVMe-oF TCP Initiator I/O core scaling: IOPS vs. Latency while running 4KB 100% Random Read QD=128 workload

4KB Random Write Results

Test Result: 4KB 100% Random Write, QD=128

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	2081.39	532.8	2366.0
4 cores	7708.16	1973.3	909.9
8 cores	8423.35	2156.4	961.8
12 cores	8851.19	2265.9	902.1
16 cores	8591.30	2199.4	933.4
24 cores	8201.17	2099.5	967.8
32 cores	7932.80	2030.8	1007.4
40 cores	7939.36	2032.5	1000.7

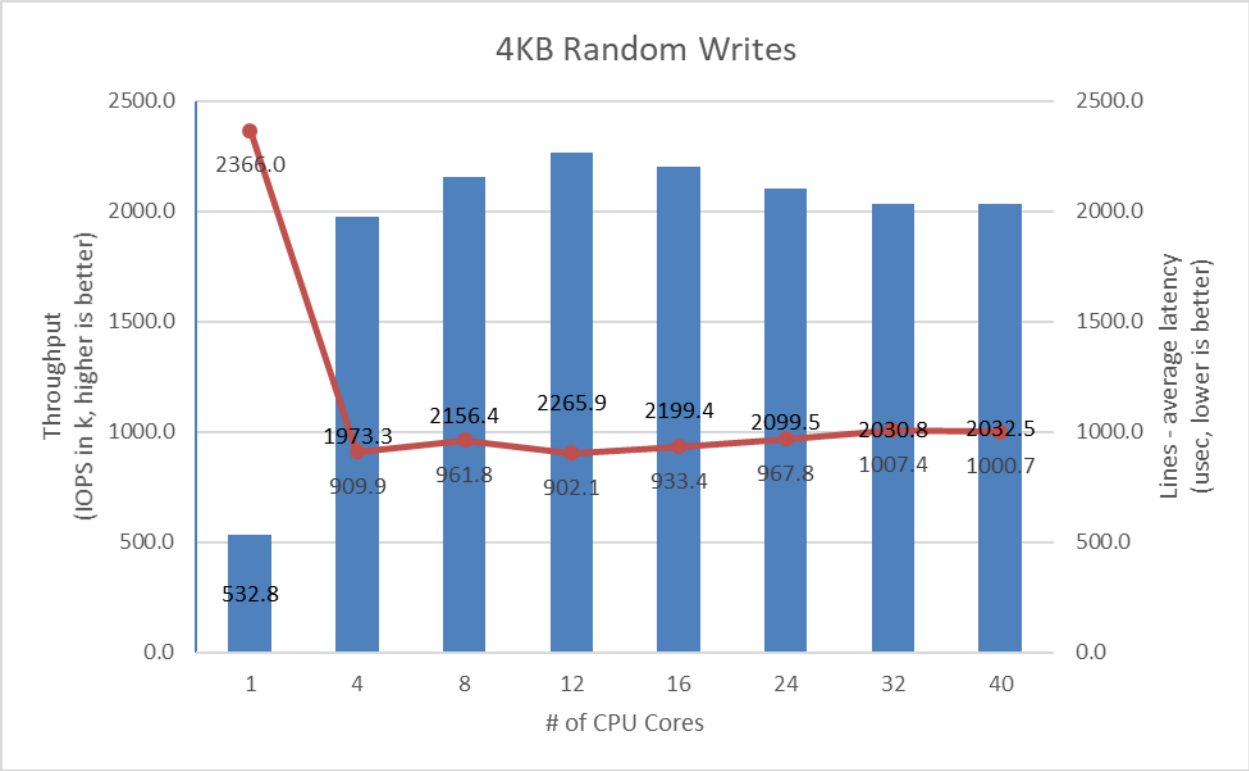

Figure 9: SPDK NVMe-oF TCP Initiator I/O core scaling: IOPS vs. Latency while running 4KB 100% Random Write Workload at QD=128

4KB Random Read-Write Results

Test Result: 4KB 70% Random Read 30% Random Write, QD=128

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	1300.82	333.0	6083.3
4 cores	5513.58	1411.5	1435.0
8 cores	9668.92	2475.2	811.7
12 cores	11899.42	3046.2	661.3
16 cores	12859.14	3291.9	615.1
24 cores	11809.25	3023.2	669.8
32 cores	11139.03	2851.6	713.9
40 cores	10379.52	2657.1	764.1

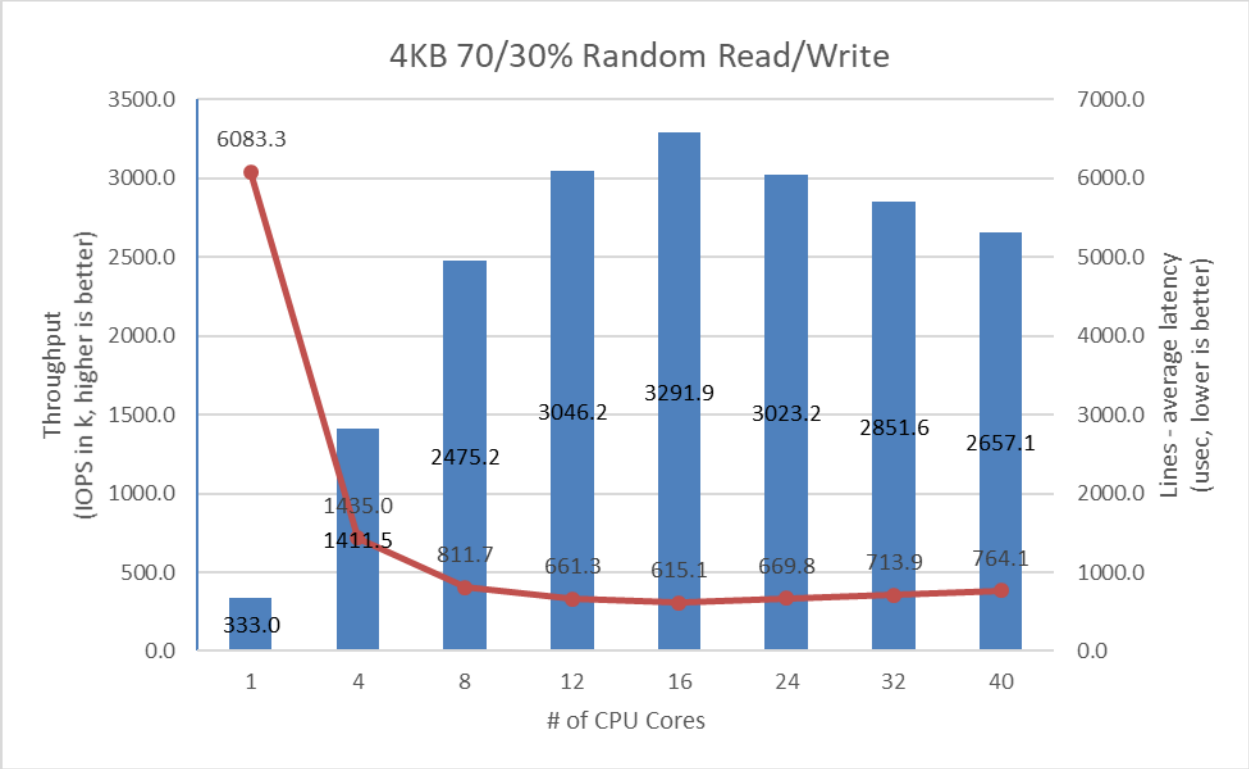

Figure 10: SPDK NVMe-oF TCP Initiator I/O core scaling: IOPS vs. Latency while running 4KB Random 70% Read 30% Write Workload at QD=128

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 2.8 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 2.25M IOPS at 12 cores. Increasing the number of Initiator cores beyond 12 cores does not improve results.

3. Mixed Random Read-Write workload performance scales linearly up to 8 CPU cores, reaching 2.5M IOPS and reaches peak performance of 3.3M IOPS at 16 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.

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_qpairs_per_ctrlr: 64 in_capsule_data_size: 4096 max_io_size: 131072 io_unit_size: 8192 max_aq_depth: 128 num_shared_buffers: 4096 buf_cache_size: 32) construct_null_bdev Nvme0n1 10240 4096 nvmf_subsystem_create nqn.2018-09.io.spdk:cnode1 -s SPDK001 -a -m 8 nvmf_subsystem_add_ns nqn.2018-09.io.spdk:cnode1 Nvme0n1 nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode1 -t tcp -f ipv4 -s 4420 -a 20.0.0.1

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

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

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

	[filename0] filename=/dev/nvme0n1

SPDK vs Kernel NVMe-oF Target Latency Results

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

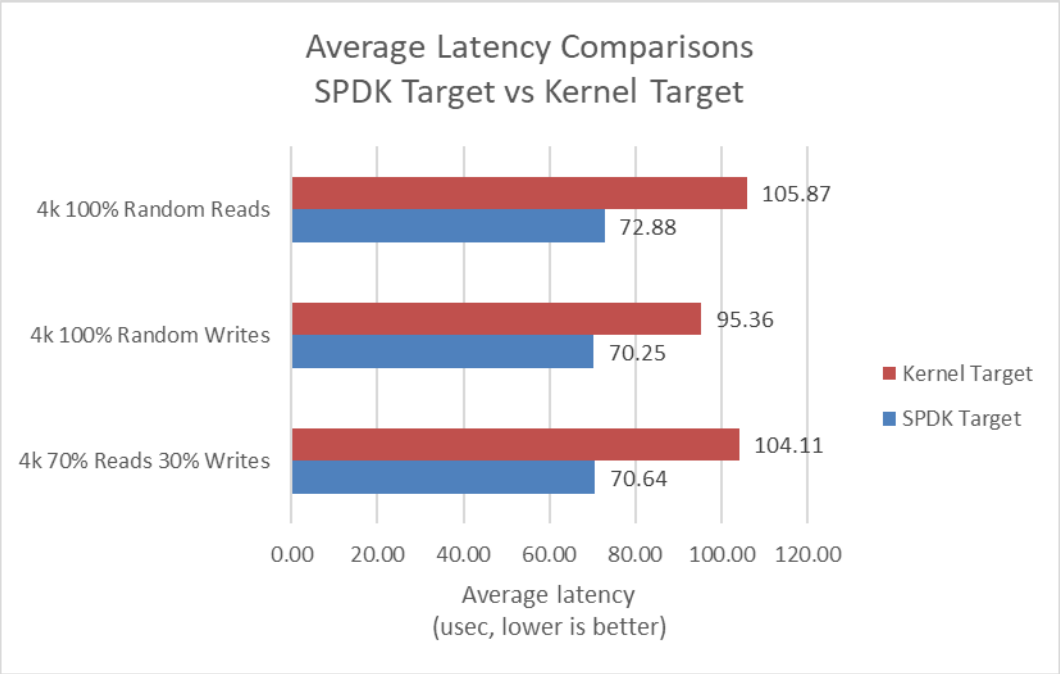

Figure 11: SPDK vs. Kernel NVMe-oF TCP Average I/O Latency for various workloads run using the Kernel Initiator

SPDK NVMe-oF 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	72.88	13382	78.3	174.4	237.9	398.0
4KB 100% Random Writes IOPS	70.25	13869	60.5	178.5	227.0	346.1
4KB 100% Random 70% Reads 30% Writes IOPS	70.64	13787	67.3	120.6	182.1	295.8

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	104.11	9430	133.8	171.1	225.8	1034.9
4KB 100% Random Writes IOPS	95.36	10293	92.0	157.9	218.8	305.2
4KB 100% Random 70% Reads 30% Writes IOPS	104.11	9430	133.8	171.1	225.8	760.5

SPDK vs Kernel NVMe-oF TCP Initiator results

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

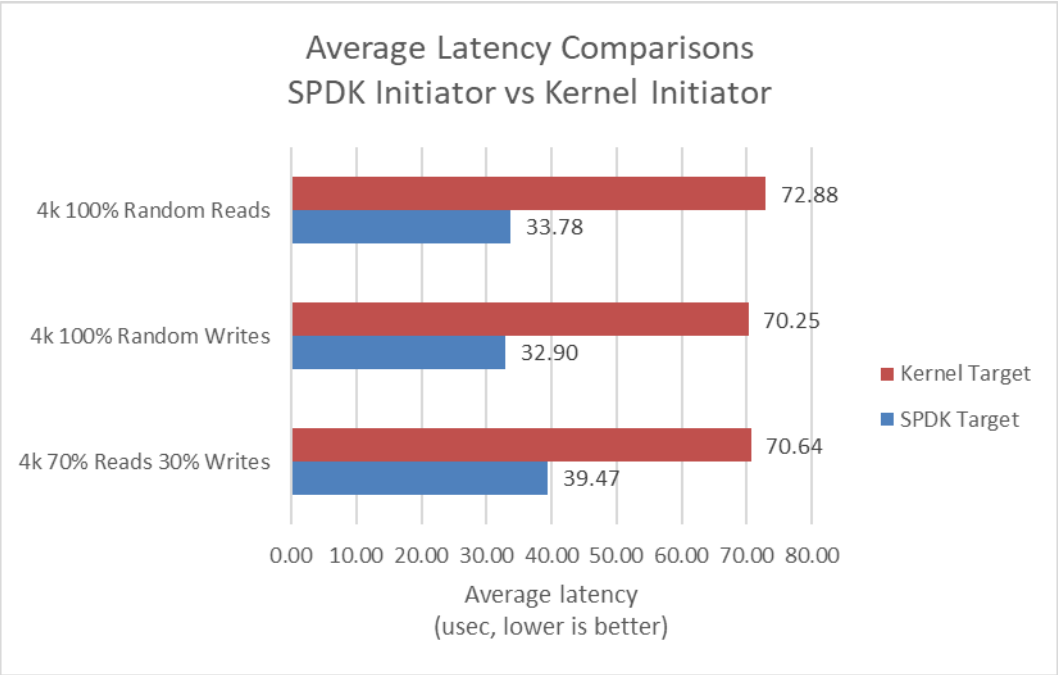

Figure 12: SPDK vs. Kernel NVMe-oF TCP Average I/O Latency for various workloads against SPDK Target

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	33.78	29572	47.2	59.8	72.9	192.2
4KB 100% Random Writes IOPS	32.90	30198	45.1	111.4	72.2	171.7
4KB 100% Random 70% Reads 30% Writes IOPS	39.47	25252	78.7	111.4	127.3	179.1

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	72.88	13382	78.3	174.4	237.9	398.0
4KB 100% Random Writes IOPS	70.25	13869	60.5	178.5	227.0	346.1
4KB 100% Random 70% Reads 30% Writes IOPS	70.64	13787	67.3	120.6	182.1	295.8

SPDK vs Kernel NVMe-oF Latency Results

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

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	33.78	29572	47.2	59.8	72.9	192.2
4KB 100% Random Writes IOPS	32.90	30198	45.1	111.4	72.2	171.7
4KB 100% Random 70% Reads 30% Writes IOPS	39.47	25252	78.7	111.4	127.3	179.1

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	104.11	9430	133.8	171.1	225.8	1034.9
4KB 100% Random Writes IOPS	95.36	10293	92.0	157.9	218.8	305.2
4KB 100% Random 70% Reads 30% Writes IOPS	104.11	9430	133.8	171.1	225.8	760.5

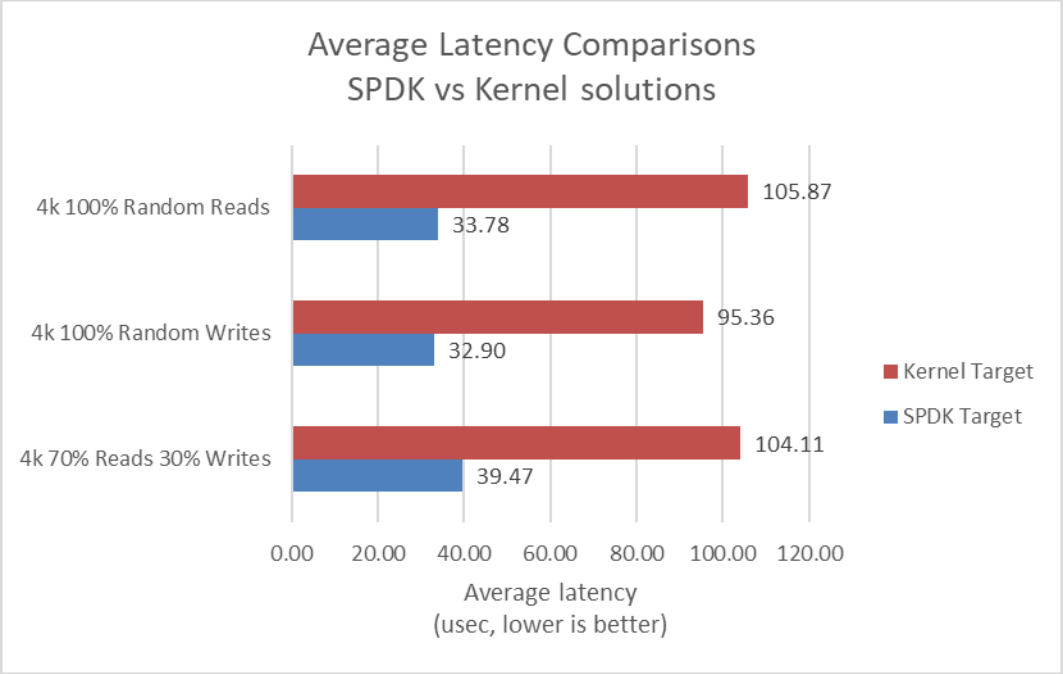

Figure 13: SPDK vs. Kernel NVMe-oF TCP solutions Average I/O Latency for various workloads

Conclusions

1. SPDK NVMe-oF Initiator reduces the average latency by up to 39 usec vs. the Linux Kernel NVMe-oF Initiator, which eliminates up to 53% NVMe-oF software overhead.

2. The SPDK NVMe-oF Target reduces the NVMe-oF average round trip I/O latency (reads/writes) by up to 33 usec vs. the Linux Kernel NVMe-oF target, which eliminates up to 31% NVMe-oF software overhead.

3. The SPDK NVMe-oF TCP target and initiator reduced the average latency by up to 68% vs. the Linux Kernel NVMe-oF target and initiator.

4. The SPDK NVMe-oF TCP target reduces tail latencies (99th percentile) by about 50-60% for the 4KB Random Read and 4KB Random Read/Write 70/30 workloads.

5. The SPDK NVMe-oF Initiator reduces the p99 latency by 40% and 25% for the 4KB random reads and write workloads respectively.

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

This test case was performed in order to understand throughput and latency capabilities of SPDK 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 P4600) 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

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

	ioengine=libaio thread=1 group_reporting=1 direct=1 norandommap=1 rw=randrw rwmixread={100, 70, 0} bs=4k iodepth={8, 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 performance in IOPS/core which was calculated based on total aggregate IOPS divided by total CPU cores used while running that specific workload. For the 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

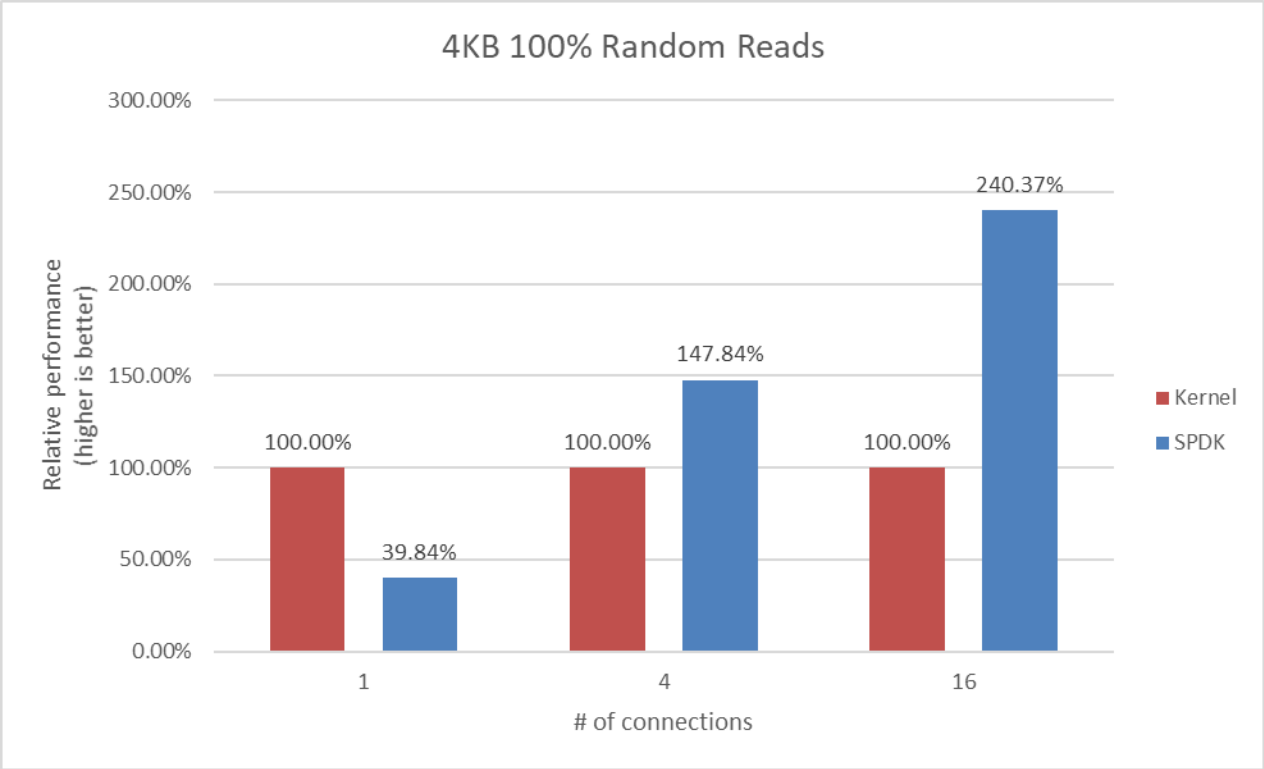

Figure 14: Relative Performance Comparison of Linux Kernel vs. SPDK NVMe-oF Target IOPS/Core for 4KB 100% Random Reads using the Kernel Initiator

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

Connections per subsystem	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)	# CPU Cores
1	2786.13	713.2	1436.2	11.3
4	7221.91	1848.8	552.2	38.3
16	7481.39	1915.2	532.8	55.8

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

Connections per subsystem	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)	# CPU Cores
1	2943.59	753.6	1358.6	30.0
4	8358.96	2139.9	478.1	30.0
16	9668.39	2475.1	412.3	30.0

4KB Random Write Results

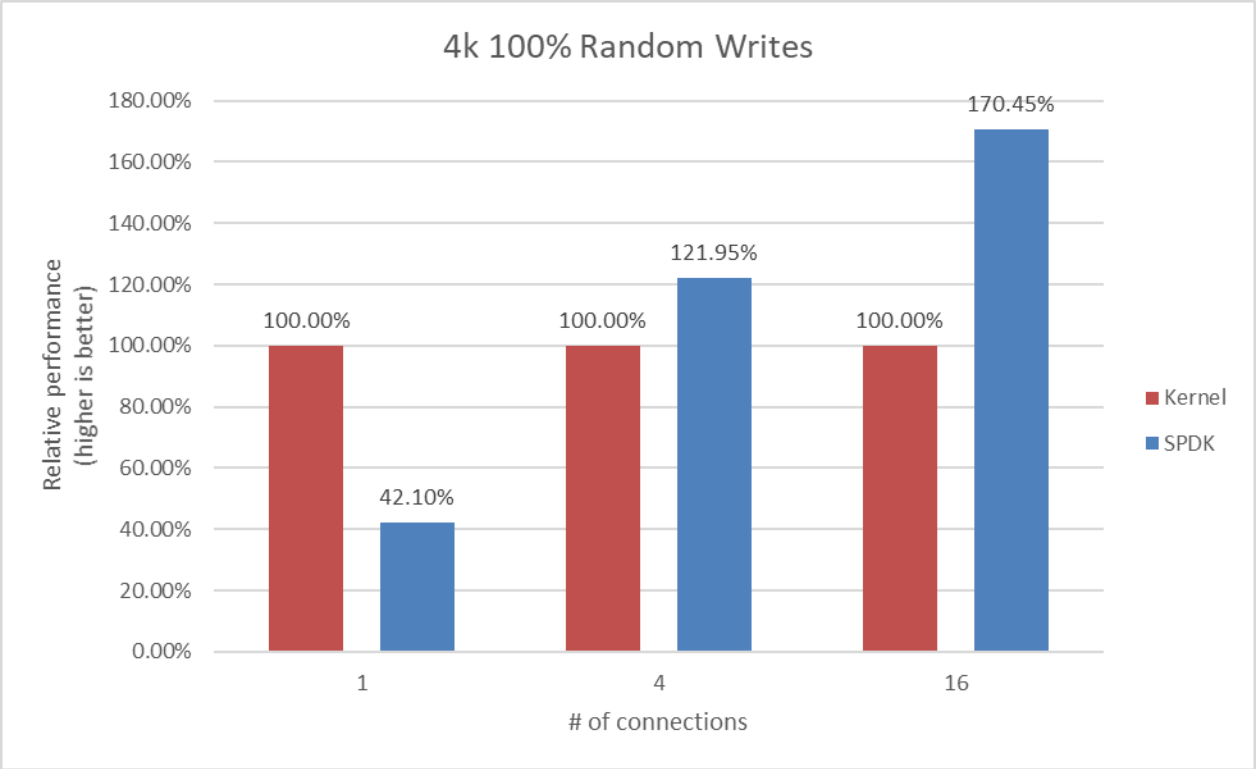

Figure 15: Relative Performance Comparison of Linux Kernel vs. SPDK NVMe-oF Target IOPS/Core for 4KB 100% Random Writes Workload

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

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	3234.27	828.0	1236.5	12.0
4	7856.74	2011.3	507.7	39.7
16	9732.51	2491.5	409.5	62.6

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	3402.06	870.9	1176.7	30.0
4	7244.69	1854.6	550.4	30.0
16	7946.64	2034.3	501.4	30.0

4KB Random Read-Write Results

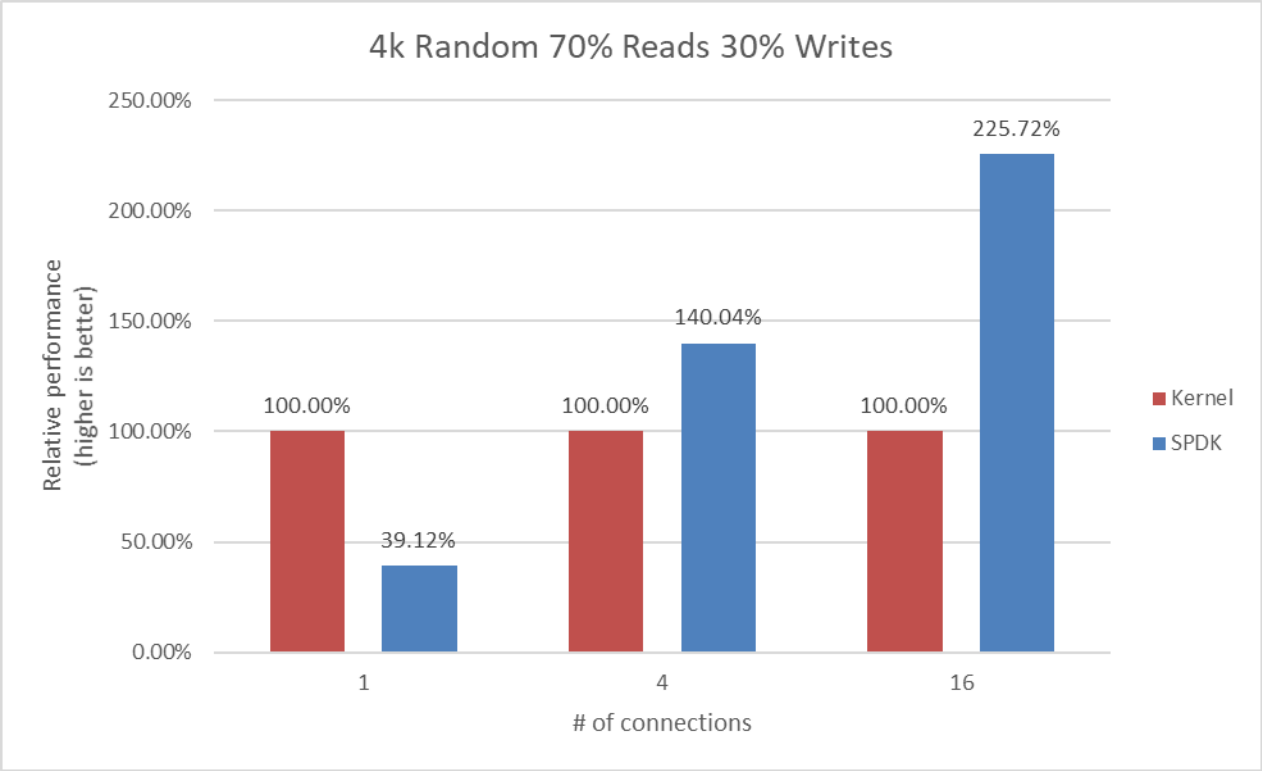

Figure 16: Relative Performance Comparison of Linux Kernel vs. SPDK NVMe-oF Target IOPS/Core for 4KB Random 70% Reads 30% Writes Workload

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

Connections per subsystem	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)	# CPU Cores
1	2845.29	728.4	1406.4	11.5
4	7129.48	1825.1	559.2	39.4
16	8075.96	2067.4	493.7	59.7

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

Connections per subsystem	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)	# CPU Cores
1	2910.33	745.0	1374.4	30.0
4	7599.66	1945.5	524.6	30.0
16	9166.77	2346.7	434.8	30.0

Low Connections Results

During testing it was observed that relative performance of SPDK Target is about 60-70% of Kernel Target performance 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.

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

Workload	Target	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)	# CPU Cores
Random Read	Linux	2786.13	713.2	1436.2	11.3
	SPDK	2926.27	749.1	1366.8	4.0
Random Write	Linux	3234.27	828.0	1236.5	12.0
	SPDK	2662.33	681.6	1501.7	4.0
Random Read/Write	Linux	2845.29	728.4	1406.4	11.5
	SPDK	2854.90	730.9	1400.7	4.0

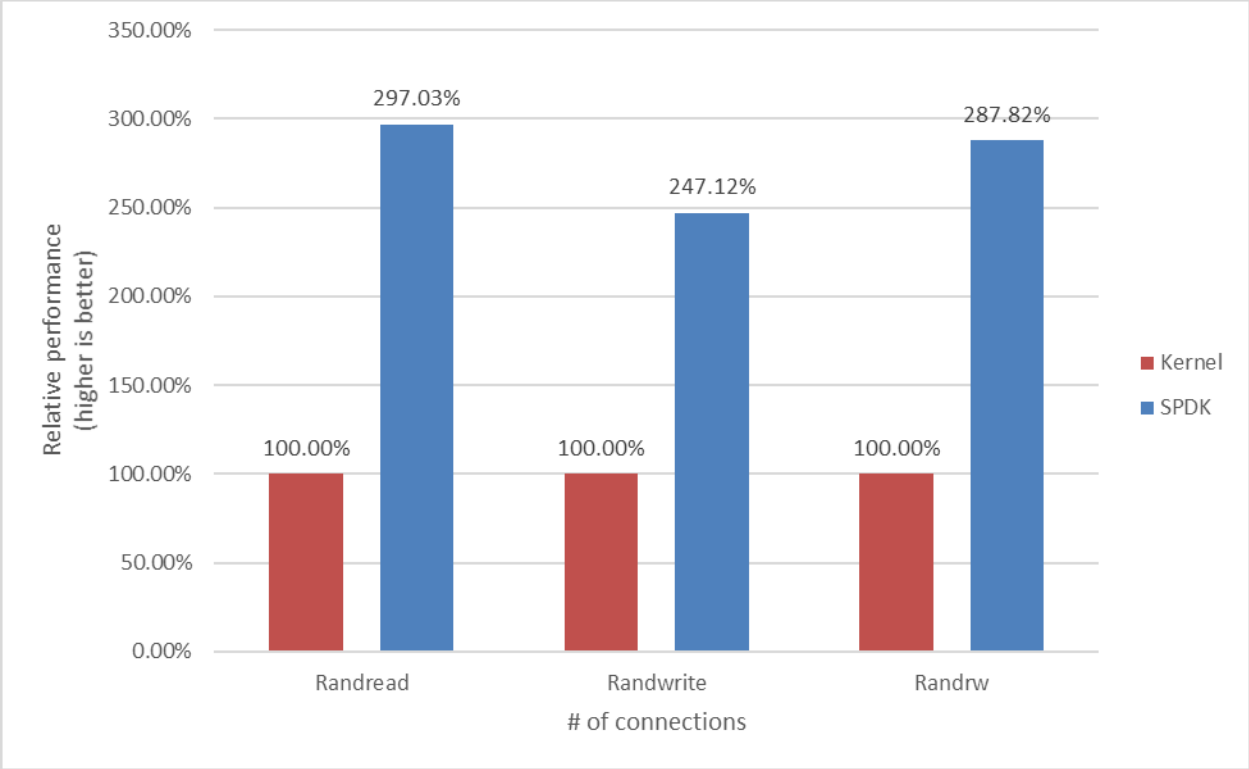

Figure 17: Relative Performance Comparison of Linux Kernel vs. SPDK NVMe-oF Target IOPS/Core for various workloads, 1 connection per subsystem and reduced number of SPDK Target CPU Cores (4)

Conclusions

1. The Linux Kernel NVMe-oF TCP target relative performance 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 of about 3 times better than Linux Kernel NVMe-oF TCP target.

2. The SPDK NVMe-oF TCP target relative performance in IOPS/Core improved as we increased the number of connections per subsystem. It was between 1.2 - 1.45 times better than the Linux Kernel NVMe-oF target at 4 connections per NVMe-oF subsystem and up to 2.4 time 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. Beyond 100 Gbps, the IOPS scalability is non-linear for 4KB Random Read and 4KB Random Read/Write 70/30 workloads.

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

For the NVMe-oF TCP latency comparison, the SPDK NVMe-oF Target and Initiator average latency is up to 53% and 31% lower than their Linux Kernel counterparts respectively when testing against null block device-based backend.

The SPDK NVMe-oF TCP Target performed up to 2.4 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.

Appendix A – 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

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors.

Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details. No product or component can be absolutely secure.

Your costs and results may vary.

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

§