SPDK NVMe-oF RDMA (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:

Contents

Audience and Purpose

This report is intended for people who are interested in evaluating SPDK NVMe-oF (Target & Initiator) performance as compared to the Linux Kernel NVMe-oF (Target & Initiator). This report contains performance and efficiency of the SPDK vs. Linux Kernel NVMe-oF Target and Initiator for the RDMA transport only.

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

Test setup

Target Configuration

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: 0x500002c
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) Microcode: 0x500002c
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) Microcode: 0x500002c
Memory	6 x 32GB Micron M393A1G40EB1-CRC, DDR4, 2933MHz Total 192GBs
Operating System	Fedora 30~~02.01.0008 03/19/2019~~
BIOS	02.01.0008 03/19/2019~~3.1 06/08/2018~~
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

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 as close to the CPU cache as possible.

The SPDK NVMe-oF target and initiator uses the Infiniband/RDMA verbs API to access an RDMA-capable NIC. These should work on all flavors of RDMA transports, but are currently tested against RoCEv2, and iWARP. Similar to the SPDK NVMe driver, SPDK provides a user-space, lockless, polled-mode NVMe-oF initiator. The host system uses the initiator to establish a connection and submit I/O requests to an NVMe subsystem within an NVMe-oF target. NVMe subsystems contain namespaces, each of which maps to a single block device exposed via SPDK's bdev layer. SPDK's bdev layer is a block device abstraction layer and general-purpose block storage stack akin to what is found in many operating systems. Using the bdev interface completely decouples the storage media from the front-end protocol used to access storage. Users can build their own virtual bdevs that provide complex storage services and integrate them with the SPDK NVMe-oF target with no additional code changes. There can be many subsystems within an NVMe-oF target and each subsystem may hold many namespaces. Subsystems and namespaces can be configured dynamically via a JSON-RPC interface.

Figure 1 shows a high-level schematic of the systems used for testing in the rest of this report. The set up consists of three individual systems (two used as initiators and one used as the target). The NVMe-oF target is connected to both initiator systems point-to-point using QSFP28 cables without any switches. The target system has sixteen Intel 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 Ex 100GbE NIC connected directly to the target without any switch.

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

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

1) Linux Kernel libaio engine
2) SPDK bdev engine

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

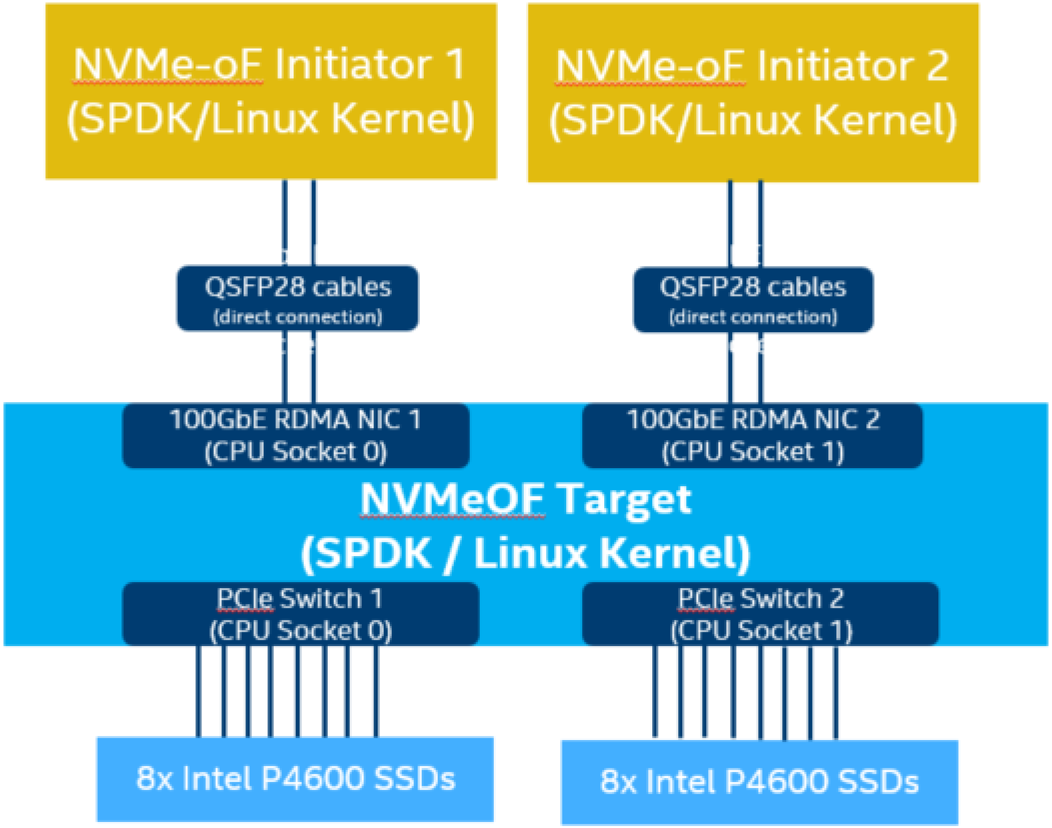

Figure 1: High-Level NVMe-oF performance testing setup

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

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

The SPDK NVMe-oF RDMA target was configured to run with 16 NVMe-oF subsystems. Each NVMe-oF subsystem ran on top of an individual bdev backed by a single Intel® SSD DC P4610 device. Each of the 2 initiators were connected to 8 individual NVMe-oF subsystems which were exposed via SPDK NVMe-oF Target over 1x 100GbE NIC. SPDK bdev FIO plugin was used to target 8 individual NVMe-oF bdevs on each of the initiators. SPDK Target Reactor Mask was configured to use 1, 2, 3, 4, 5 and 6 cores tests while running following workloads on each initiator:

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

The table below contains more information the test configuration. The SPDK NVMe-oF Target was configured using JSON-RPC; the table contains a 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. We preconditioned the SSDs once before running the 4KB Rand Read and 4KB Rand 70/30 Read/Write workloads to ensure that the SSDs reached their steady state where we get repeatable results. However, for the 4KB Rand Write workload we didn't precondition the NVMe devices to ensure workload saturated the network rather than being limited to the steady state performance of the SSDs which is much lower than the available network bandwidth.

Item	Description
Test Case	SPDK NVMe-oF Target I/O core scaling
SPDK NVMe-oF Target configuration	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 RDMA
	(creates RDMA transport layer with default values:
	trtype: "RDMA"
	max_queue_depth: 128
	max_qpairs_per_ctrlr: 64
	in_capsule_data_size: 4096
	max_io_size: 131072
	io_unit_size: 8192
	max_aq_depth: 128
	num_shared_buffers: 4096
	buf_cache_size: 32)
	
	for i in $(seq 1 16); do
	    nvmf_subsystem_create nqn.2018-09.io.spdk:cnode${i} -s SPDK00${i} -a -m 8
	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 rdma \\
	                -f ipv4 -s 4420 -a ${ip}
	        ((i++))
	    done
	done

SPDK NVMe-oF Initiator - FIO plugin configuration	**BDEV.conf** [Nvme] TransportId "trtype:RDMA adrfam:IPv4 traddr:20.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode1" Nvme0 TransportId "trtype:RDMA adrfam:IPv4 traddr:20.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode2" Nvme1 TransportId "trtype:RDMA adrfam:IPv4 traddr:20.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode3" Nvme2 TransportId "trtype:RDMA adrfam:IPv4 traddr:20.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode4" Nvme3 TransportId "trtype:RDMA adrfam:IPv4 traddr:20.0.1.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode5" Nvme4 TransportId "trtype:RDMA adrfam:IPv4 traddr:20.0.1.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode6" Nvme5 TransportId "trtype:RDMA adrfam:IPv4 traddr:20.0.1.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode7" Nvme6 TransportId "trtype:RDMA adrfam:IPv4 traddr:20.0.1.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode8" Nvme7 **FIO.conf** [global] ioengine=/tmp/spdk/examples/bdev/fio_plugin/fio_plugin spdk_conf=/tmp/spdk/bdev.conf thread=1 group_reporting=1 direct=1 norandommap=1 rw=randrw rwmixread={100, 70, 0} bs=4k iodepth={1, 8, 16, 32} time_based=1 ramp_time=60 runtime=300

	[filename0] filename=Nvme0n1 [filename1] filename=Nvme1n1 [filename2] filename=Nvme2n1 [filename3] filename=Nvme3n1 [filename4] filename=Nvme4n1 [filename5] filename=Nvme5n1 [filename6] filename=Nvme6n1 [filename7] filename=Nvme7n1

4KB Random Read Results

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

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	3952.67	1011.9	1011.6
2 cores	9151.72	2342.8	436.8
3 cores	15523.36	3974.0	257.0
4 cores	21043.94	5387.2	189.9
5 cores	21609.59	5532.1	185.1
6 cores	22026.52	5638.8	182.1

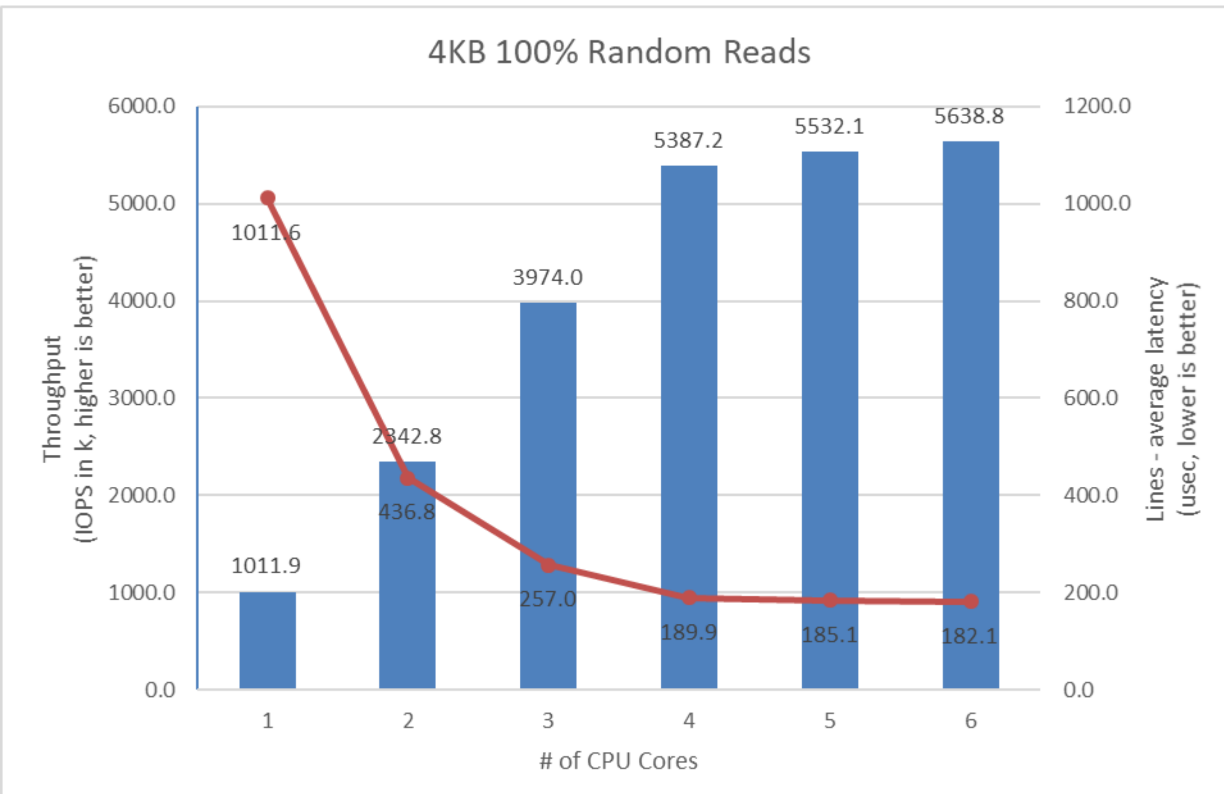

Figure 2: SPDK NVMe-oF RDMA Target I/O core scaling: IOPS vs. Latency while running 4KB 100% Random Read workload at QD=64

4KB Random Write Results

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

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	5569.08	1425.7	357.0
2 cores	12383.21	3170.1	160.7
3 cores	18883.67	4834.2	103.9
4 cores	22976.34	5881.9	85.5
5 cores	22848.22	5849.1	86.6
6 cores	22952.29	5875.8	86.5

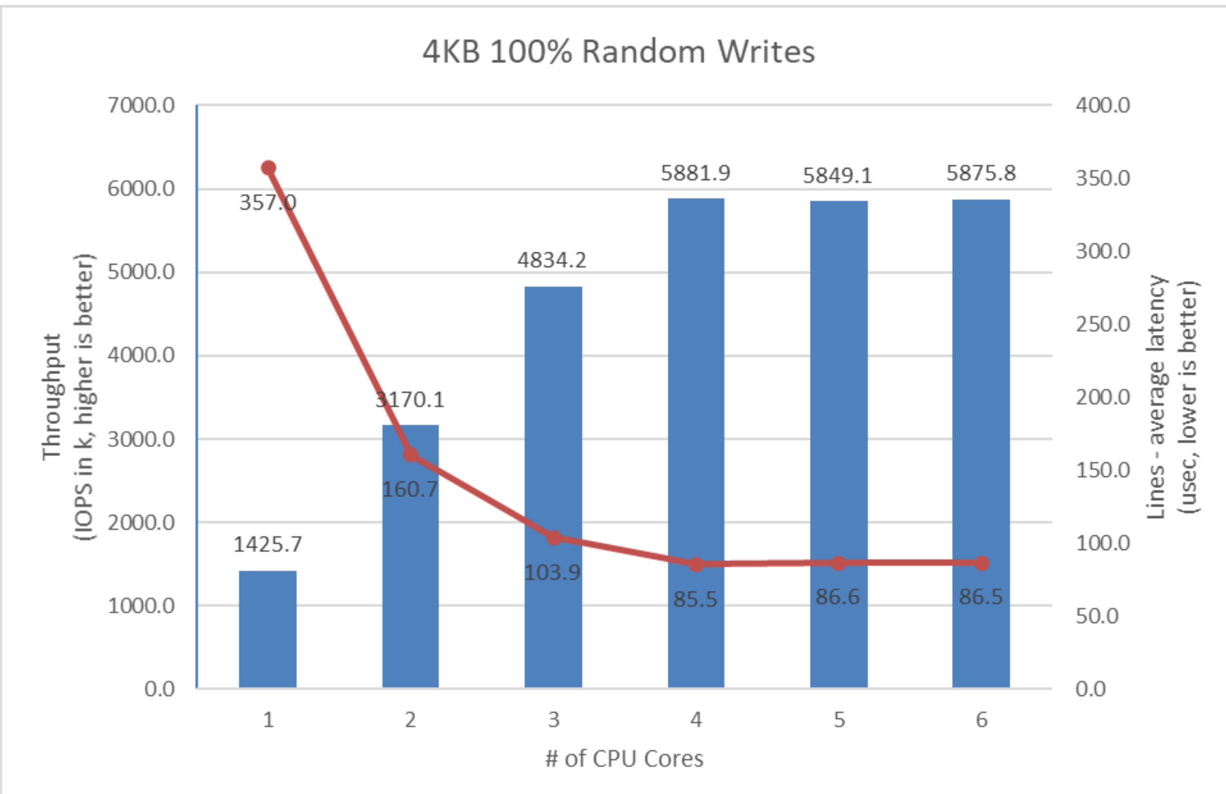

Figure 3: SPDK NVMe-oF RDMA Target I/O core scaling: IOPS vs. Latency while running 4KB 100% Random Write workload at QD=32

4KB Random Read-Write Results

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

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	3964.99	1015.0	2017.1
2 cores	9059.51	2319.2	881.9
3 cores	14731.65	3771.3	557.1
4 cores	20609.63	5276.1	390.5
5 cores	22404.87	5735.6	356.6
6 cores	24785.04	6345.0	323.9

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

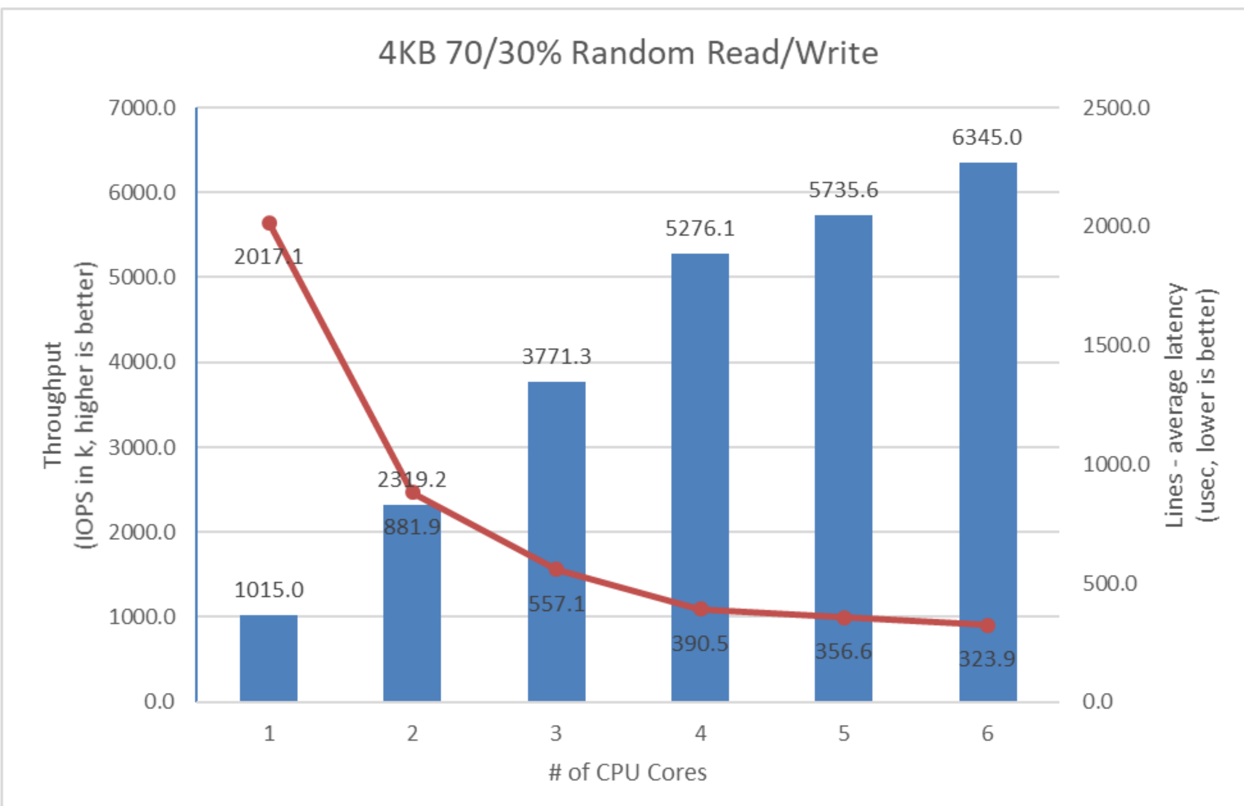

Figure 4: SPDK NVMe-oF RDMA Target I/O core scaling: IOPS vs. Latency while running 4KB Random 70% Read 30% Write workload at QD=128

Conclusions

1. For all tested workloads throughput scales up and latency decreases almost linearly with the addition of I/O cores up to 4 cores. Adding more CPU cores results in small performance gain as network is almost saturated.

Large Sequential I/O Performance

128KB block size I/O tests were performed with sequential I/O workloads at queue depth 8. The rest of the FIO configuration is similar to the 4KB test case in the previous part of this document. We used iodepth=8 (iodepth=1 for write workload) because higher queue depth resulted in negligible bandwidth gain and a significant increase in the latency.

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

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	24739.41	197.9	646.7
2 cores	24687.37	197.5	648.1
3 cores	24796.48	198.4	645.1
4 cores	24863.67	198.9	643.3

Test Result: 128KB 100% Sequential Writes QD=1

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	22967.66	183.7	86.9
2 cores	23184.38	185.5	86.1
3 cores	23187.66	185.5	86.1
4 cores	23200.40	185.6	86.0

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

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	25614.07	204.9	627.9
2 cores	26796.34	214.4	601.0
3 cores	25763.89	206.1	626.4
4 cores	29917.93	239.3	537.8

Conclusions

1. For large sequential I/Os, a single CPU core saturated the network bandwidth. The SPDK NVMe-oF target running on 1 core does close to 24-25 GBps of bandwidth in all tested workloads. Therefore, adding more CPU cores did not result in increased performance for these workloads because the network was the bottleneck.

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

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

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

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

Depending on the number of initiators and initiator cores, we varied the number of NVMe-oF subsystems exported by the target as shown below, instead of exposing all 16 NVMe-oF subsystems.

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

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

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

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

This was done to avoid a situation where single initiator would connect to all 16 target subsystems, which resulted in unnatural latency spike in the 1 CPU core test case.

Item	Description
Test Case	SPDK NVMe-oF RDMA Initiator I/O core scaling
SPDK NVMe-oF Target configuration	Same as in Test Case #1, using 6 CPU cores.
SPDK NVMe-oF Initiator 1 - FIO plugin configuration	**BDEV.conf** [Nvme] TransportId "trtype:RDMA adrfam:IPv4 traddr:20.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode1" Nvme0 TransportId "trtype:RDMA adrfam:IPv4 traddr:20.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode2" Nvme1 TransportId "trtype:RDMA adrfam:IPv4 traddr:20.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode3" Nvme2 TransportId "trtype:RDMA adrfam:IPv4 traddr:20.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode4" Nvme3 TransportId "trtype:RDMA adrfam:IPv4 traddr:20.0.1.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode5" Nvme4 TransportId "trtype:RDMA adrfam:IPv4 traddr:20.0.1.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode6" Nvme5 TransportId "trtype:RDMA adrfam:IPv4 traddr:20.0.1.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode7" Nvme6 TransportId "trtype:RDMA adrfam:IPv4 traddr:20.0.1.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode8" Nvme7 **FIO.conf** [global] ioengine=/tmp/spdk/examples/bdev/fio_plugin/fio_plugin spdk_conf=/tmp/spdk/bdev.conf

	thread=1 group_reporting=1 direct=1 norandommap=1 rw=randrw rwmixread={100, 70, 0} bs=4k iodepth={32, 64, 128, 256} time_based=1 ramp_time=60 runtime=300 {filename_section} **FIO.conf filename section for 1 & 2 CPU test run** [filename0] filename=Nvme0n1 filename=Nvme1n1 filename=Nvme2n1 filename=Nvme3n1 **FIO.conf filename section for 3 & 4 CPU test run** [filename0] filename=Nvme0n1 filename=Nvme1n1 filename=Nvme2n1 filename=Nvme3n1 [filename1] filename=Nvme4n1 filename=Nvme5n1 filename=Nvme6n1 filename=Nvme7n1
SPDK NVMe-oF Initiator 2 - FIO plugin configuration	**BDEV.conf** [Nvme] TransportId "trtype:RDMA adrfam:IPv4 traddr:10.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode9" Nvme0 TransportId "trtype:RDMA adrfam:IPv4 traddr:10.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode10" Nvme1 TransportId "trtype:RDMA adrfam:IPv4 traddr:10.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode11" Nvme2 TransportId "trtype:RDMA adrfam:IPv4 traddr:10.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode12" Nvme3 TransportId "trtype:RDMA adrfam:IPv4 traddr:10.0.1.1 trsvcid:4421 subnqn:nqn.2018-09.io.spdk:cnode13" Nvme4 TransportId "trtype:RDMA adrfam:IPv4 traddr:10.0.1.1 trsvcid:4421 subnqn:nqn.2018-09.io.spdk:cnode14" Nvme5 TransportId "trtype:RDMA adrfam:IPv4 traddr:10.0.1.1 trsvcid:4421 subnqn:nqn.2018-09.io.spdk:cnode15" Nvme6 TransportId "trtype:RDMA adrfam:IPv4 traddr:10.0.1.1 trsvcid:4421 subnqn:nqn.2018-09.io.spdk:cnode16" Nvme7 **FIO.conf** Similar as Initiator 1. The only differences are were in the filename section which are shown below. **FIO.conf filename section for 1 CPU test run** N/A (only Initiator 1 is used) **FIO.conf filename section for 2 CPU test run** [filename0] filename=Nvme0n1 filename=Nvme1n1 filename=Nvme2n1 filename=Nvme3n1

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

4KB Random Read Results

Test Result: 4KB 100% Random Read, QD=64, SPDK Target 6 CPU cores

# of Initiator CPU Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	5793.48	1483.13	558.01
2 cores	15514.13	3971.61	225.96
3 cores	18572.74	4754.62	206.21
4 cores	22211.27	5686.08	179.67
5 cores	22358.36	5723.74	178.79
6 cores	22544.87	5771.48	177.78
7 cores	21443.51	5489.54	186.76
8 cores	21433.95	5487.09	186.83

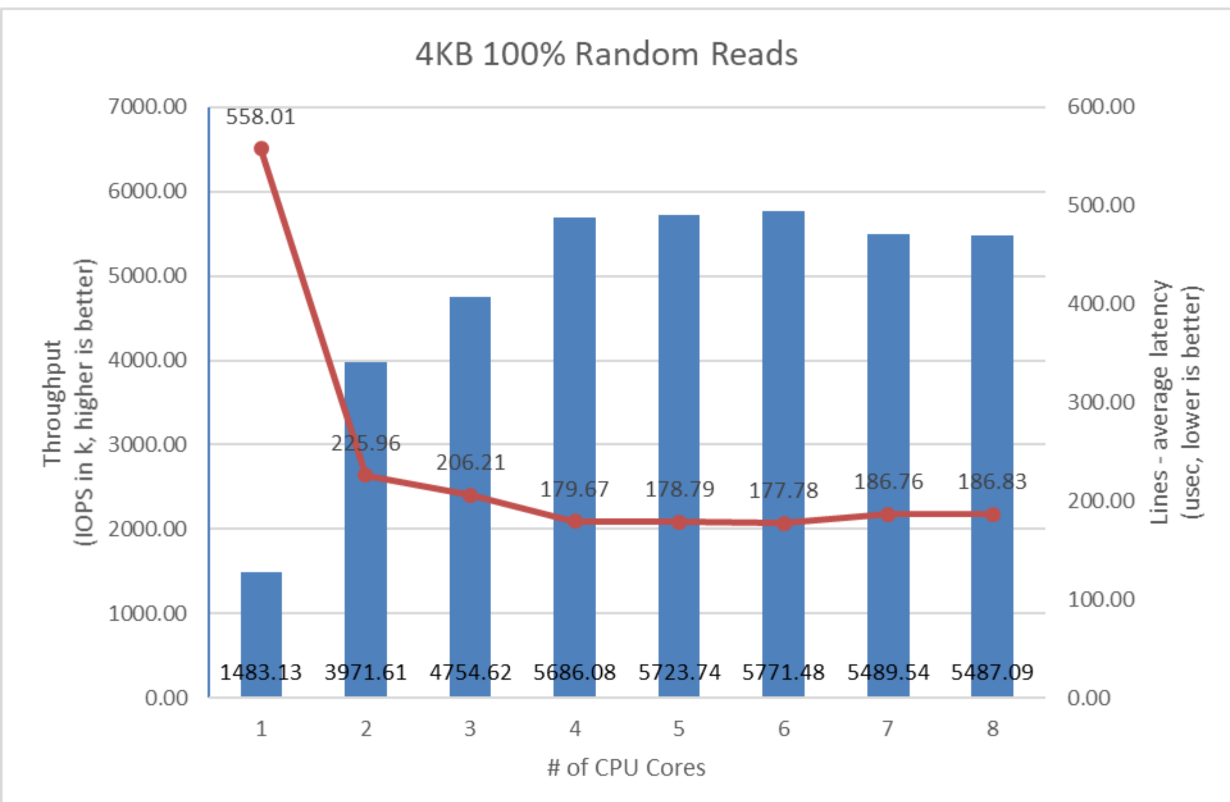

Figure 5: SPDK NVMe-oF RDMA Initiator I/O core scaling: IOPS vs. Latency while running 4KB 100% Random Read workload at QD=64

4KB Random Write Results

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

Test Result: 4KB 100% Random Write, QD=32, SPDK Target 6 CPU Cores

# of Initiator CPU Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	5709.81	1461.71	283.33
2 cores	13410.36	3433.05	119.96
3 cores	18071.06	4626.19	102.85
4 cores	22718.49	5815.93	85.53
5 cores	22638.64	5795.49	86.44
6 cores	22958.68	5877.42	85.99
7 cores	22960.78	5877.96	86.12
8 cores	22866.44	5853.81	86.64

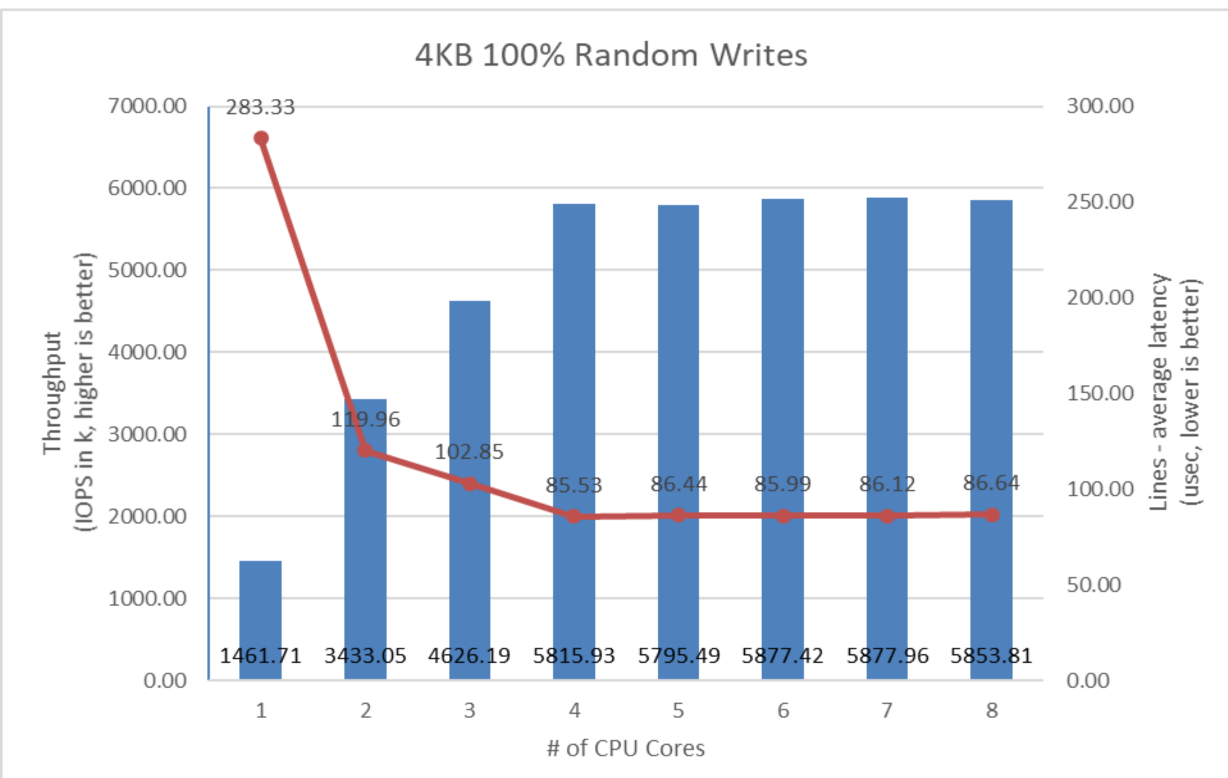

Figure 6: SPDK NVMe-oF RDMA Initiator I/O core scaling: IOPS vs. Latency while running 4KB 100% Random Write workload at QD=32

4KB Random 70/30 Read/Write Results

Test Result: 4KB 70% Random Read 30% Random Write QD=128, SPDK Target 6 CPU Cores

# of Initiator CPU Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	4849.21	1241.39	1331.20
2 cores	11241.22	2877.75	590.86
3 cores	17508.32	4482.12	464.71
4 cores	21419.95	5483.50	372.42
5 cores	21889.43	5603.69	361.96
6 cores	21405.97	5479.92	373.87
7 cores	20961.55	5366.15	383.55
8 cores	21355.74	5467.06	379.39

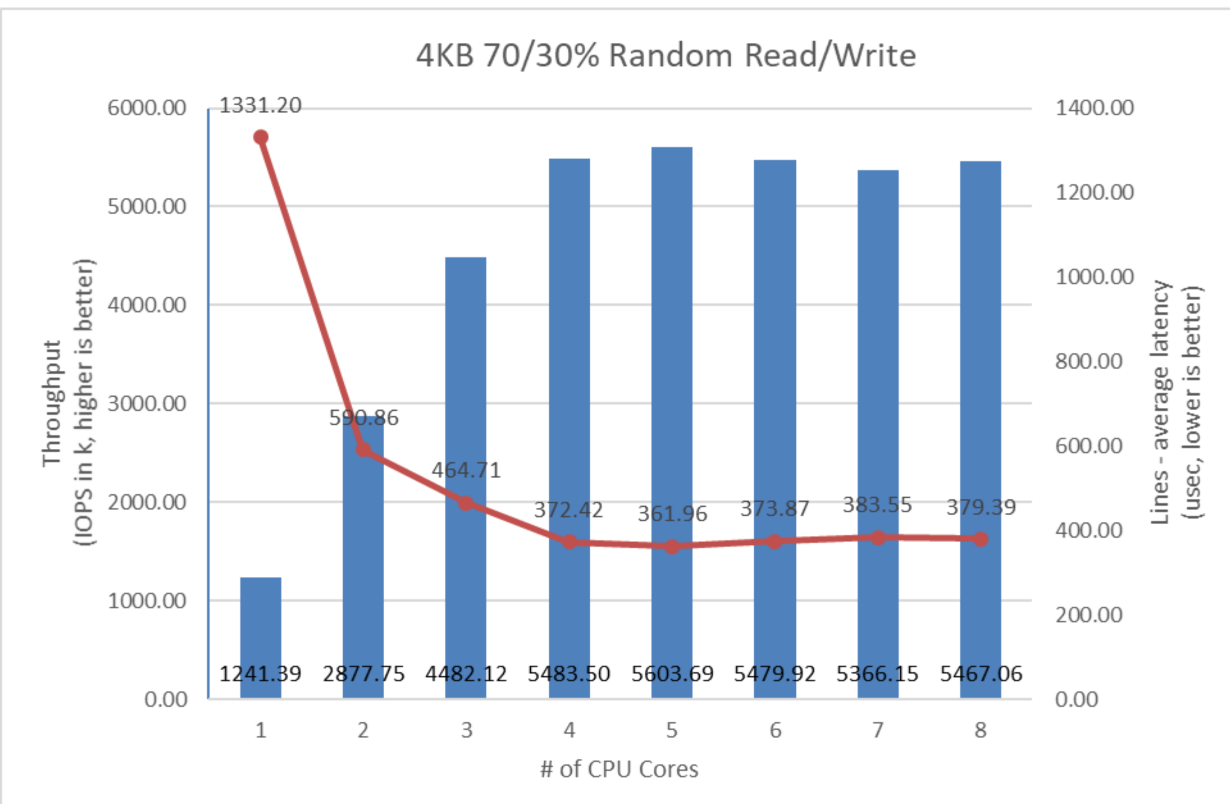

Figure 7: SPDK NVMe-oF RDMA Initiator I/O core scaling: IOPS vs. Latency while running 4KB Random 70% Read 30% Write workload at QD=128

Conclusions

1. IOPS scaling was linear or close to linear for all tested workloads as we increased the number of initiator cores up to 4 CPU cores. Beyond 4 CPU cores there was no increase in IOPS because the 200 Gbps network was saturated.
2. Peak performance of 5.7 million IOPS was reached at 4 Initiator CPU cores for random read workload, 5.8 million at 4 CPU cores for random write workload and 5.6 million at 5 CPU cores for mixed random read/write workload.

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

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

Item	Description
Test Case	Linux Kernel vs. SPDK NVMe-oF RDMA Latency
Test configuration	
SPDK NVMe-oF Target configuration	The following commands are executed with spdk/scripts/rpc.py script to configure the SPDK NVMe-oF target. nvmf_create_transport -t RDMA (creates RDMA transport layer with default values: trtype: "RDMA" max_queue_depth: 128 max_qpairs_per_ctrlr: 64 in_capsule_data_size: 4096 max_io_size: 131072 io_unit_size: 8192 max_aq_depth: 128 num_shared_buffers: 4096 buf_cache_size: 32) construct_null_bdev Nvme0n1 10240 4096 nvmf_subsystem_create nqn.2018-09.io.spdk:cnode1 -s SPDK001 -a -m 8 nvmf_subsystem_add_ns nqn.2018-09.io.spdk:cnode1 Nvme0n1 nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode1 -t rdma -f ipv4 -s 4420 -a 20.0.0.1

Kernel NVMe-oF Target configuration	The following target configuration file loaded using nvmet-cli tool.
	{
	"ports": [
	{
	"addr": {
	"adrfam": "ipv4",
	"traddr": "20.0.0.1",
	"trsvcid": "4420",
	"trtype": "rdma"
	},
	"portid": 1,
	"referrals": [],

<table>
<tr><td></td><td><pre>
    "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"
  }
 ]
}
</pre></td></tr>
<tr><th colspan="2">FIO configuration</th></tr>
<tr><td>SPDK NVMe-oF Initiator FIO plugin configuration</td><td>BDEV.conf<pre>
[Nvme]
 TransportId "trtype:RDMA adrfam:IPv4 traddr:20.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode1" Nvme0
</pre>FIO.conf<pre>
[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
</pre></td></tr>
<tr><td>Kernel initiator configuration</td><td>Device config
The following configuration was performed using nvme-cli tool.
modprobe nvme-fabrics
nvme connect –n nqn.2018-09.io.spdk:cnode1 –t rdma –a 20.0.0.1 –s 4420</td></tr>
</table>

	FIO.conf [global] ioengine=libaio thread=1 group_reporting=1 direct=1 norandommap=1 rw=randrw rwmixread={100, 70, 0} bs=4k iodepth=1 time_based=1 ramp_time=60 runtime=300 [filename0] filename=/dev/nvme0n1

SPDK vs Kernel NVMe-oF RDMA Target Results

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

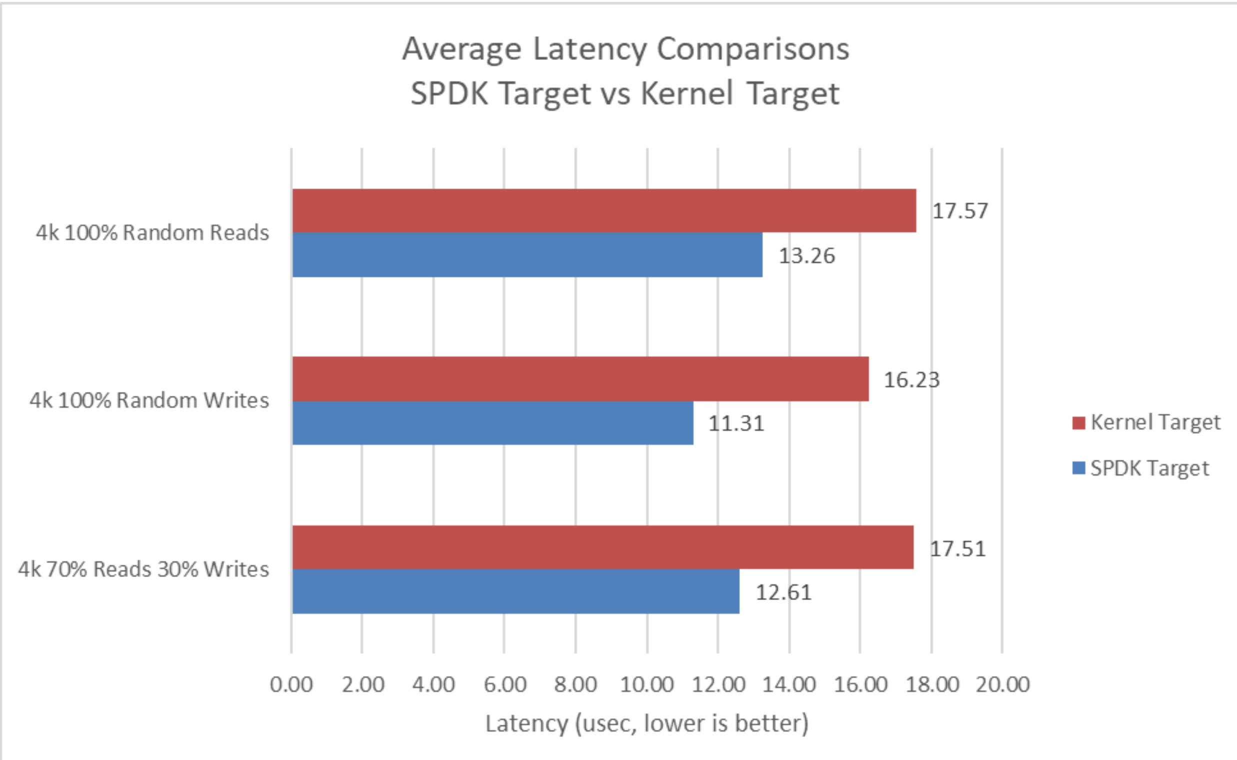

Figure 8: Average I/O Latency comparisons at QD=1 between SPDK and Linux Kernel NVMe-oF RDMA Target for various workloads using the Linux Kernel Initiator

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

Access Pattern	Average Latency (usec)	IOPS	p99 (usec)	p99.9 (usec)	p99.99 (usec)	p99.999 (usec)
4KB 100% Random Read	12.61	77744	17.4	26.0	38.4	143.5
4KB 100% Random Write	11.31	86388	17.6	26.0	35.9	143.7
4KB 70/30% Random Read/Write	13.26	74042	20.4	27.2	38.3	131.1

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

Access Pattern	Average Latency (usec)	IOPS	p99 (usec)	p99.9 (usec)	p99.99 (usec)	p99.999 (usec)
4KB 100% Random Read	17.51	56220	21.5	30.9	35.3	73.3
4KB 100% Random Write	16.23	60649	20.3	29.7	35.8	67.8
4KB 70/30% Random Read/Write	17.57	56099	19.2	29.3	33.0	68.1

Conclusions

1. For the RDMA transport, the SPDK NVMe-oF Target reduces the NVMe-oF average round trip I/O latency (reads/writes) by up to 5 usec vs. the Linux Kernel NVMe-oF target used in Fedora 30 5.4.14 setup. This is entirely software overhead, therefore, using the SPDK NVMe-oF target reduces the NVMe-oF software overhead by approximately 30% vs. the Linux Kernel NVMe-oF target.

SPDK vs Kernel NVMe-oF RDMA Initiator Results

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

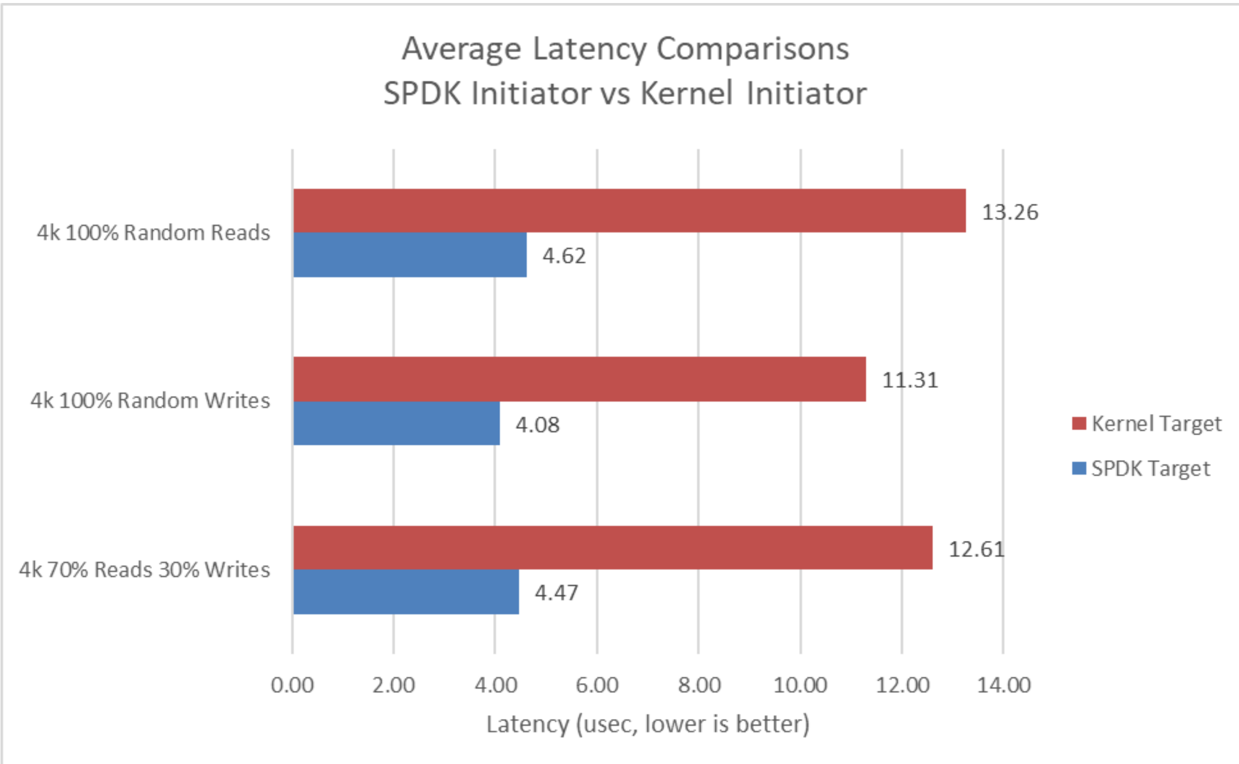

Figure 9: Average I/O Latency Comparisons at QD=1 between SPDK and Linux Kernel NVMe-oF RDMA Initiator for various workloads against SPDK NVMe-OF Target

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

Access Pattern	Average Latency (usec)	IOPS	p99 (usec)	p99.9 (usec)	p99.99 (usec)	p99.999 (usec)
4KB 100% Random Read	4.47	215081	4.6	18.2	20.5	46.5
4KB 100% Random Write	4.08	234765	4.2	18.2	20.0	46.2
4KB 70/30% Random Read/Write	4.62	209226	4.7	18.0	20.6	46.7

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

Access Pattern	Average Latency (usec)	IOPS	p99 (usec)	p99.9 (usec)	p99.99 (usec)	p99.999 (usec)
4KB 100% Random Read	12.61	77744	17.4	26.0	38.4	143.5
4KB 100% Random Write	11.31	86388	17.6	26.0	35.9	143.7
4KB 70/30% Random Read/Write	13.26	74042	20.4	27.2	38.3	131.1

Conclusions

1. The SPDK NVMe-oF initiator reduces the NVMe-oF software overhead by up to 2.9x times vs. the Linux Kernel NVMe-oF Initiator for the RDMA transport.

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

This test case was designed to demonstrate the throughput and latency of the SPDK NVMe-oF RDMA Target vs. Linux Kernel NVMe-oF RDMA Target under increasing number of connections per subsystem. The number of connections (or I/O queue pairs) per NVMe-oF subsystem were varied, we measured the aggregated IOPS and number of CPU cores used by each target. The number of CPU cores metric was calculated from %CPU utilization measured using sar (systat package in Linux). The SPDK NVMe-oF RDMA Target was configured to run on 4 CPU cores, export 16 NVMe-oF subsystems (1 per Intel P4610) and 2 initiators were used both running the I/O workloads below to 8 separate subsystems using Kernel NVMe-oF RDMA initiator.

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

Item	Description
Test Case	NVMe-oF RDMA Target performance with increasing # of connections
SPDK NVMe-oF Target configuration	Same as in Test Case #1, using 4 CPU cores.
Kernel NVMe-oF Target configuration	Target configuration file loaded using nvmet-cli tool. For detailed configuration file contents please see Appendix A.
Kernel NVMe-oF Initiator #1	**Device config** Performed using nvme-cli tool. modprobe nvme-fabrics nvme connect –n nqn.2018-09.io.spdk:cnode1 –t rdma –a 20.0.0.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode2 –t rdma –a 20.0.0.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode3 –t rdma –a 20.0.0.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode4 –t rdma –a 20.0.0.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode5 –t rdma –a 20.0.1.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode6 –t rdma –a 20.0.1.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode7 –t rdma –a 20.0.1.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode8 –t rdma –a 20.0.1.1 –s 4420
Kernel NVMe-oF Initiator #2	**Device config** Performed using nvme-cli tool. modprobe nvme-fabrics nvme connect –n nqn.2018-09.io.spdk:cnode9 –t rdma –a 10.0.0.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode10 –t rdma –a 10.0.0.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode11 –t rdma –a 10.0.0.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode12 –t rdma –a 10.0.0.1 –s 4420

	nvme connect –n nqn.2018-09.io.spdk:cnode13 –t rdma –a 10.0.1.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode14 –t rdma –a 10.0.1.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode15 –t rdma –a 10.0.1.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode16 –t rdma –a 10.0.1.1 –s 4420
FIO configuration (used on both initiators)	**FIO.conf** [global] ioengine=libaio thread=1 group_reporting=1 direct=1 norandommap=1 rw=randrw rwmixread={100, 70, 0} bs=4k iodepth={8, 16, 32, 64, 128} time_based=1 ramp_time=60 runtime=300 numjobs={1, 4, 16, 32} [filename1] filename=/dev/nvme0n1 [filename2] filename=/dev/nvme1n1 [filename3] filename=/dev/nvme2n1 [filename4] filename=/dev/nvme3n1 [filename5] filename=/dev/nvme4n1 [filename6] filename=/dev/nvme5n1 [filename7] filename=/dev/nvme6n1 [filename8] filename=/dev/nvme7n1

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

4KB Random Read Results

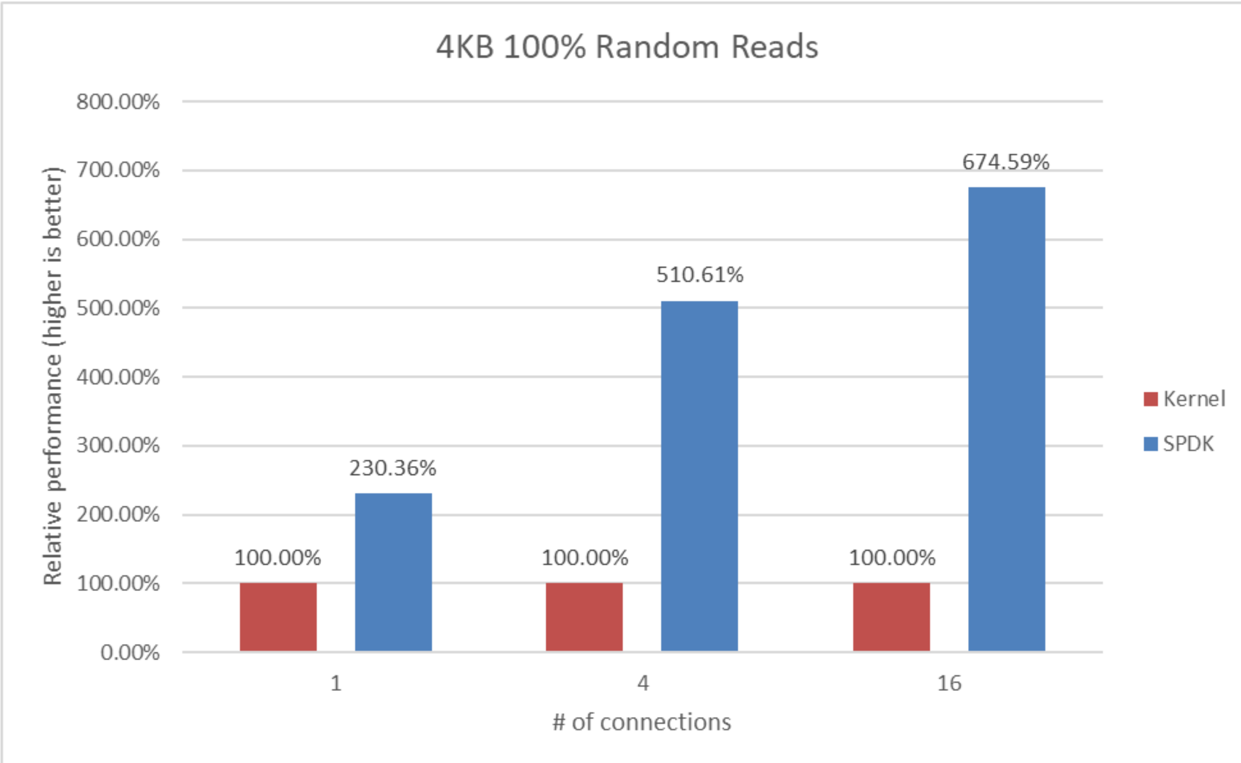

Figure 10: Relative Performance Comparison of Linux Kernel vs. SPDK NVMe-oF RDMA Target for 4KB 100% Random Read QD=64, using Kernel Initiators

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

Connections per subsystem	Bandwidth (MBps)	IOPS (k)	Avg. Latency (usec)	# CPU Cores
1	16862.99	4316.9	237.1	10.6
4	13146.69	3365.5	303.8	20.4
16	11845.44	3032.4	336.5	27.6

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

Connections per subsystem	Bandwidth (MBps)	IOPS (k)	Avg. Latency (usec)	# CPU Cores
1	14654.43	3751.5	272.8	4.0
4	13155.36	3367.8	303.6	4.0
16	11562.23	2959.9	344.7	4.0

4KB Random Write results

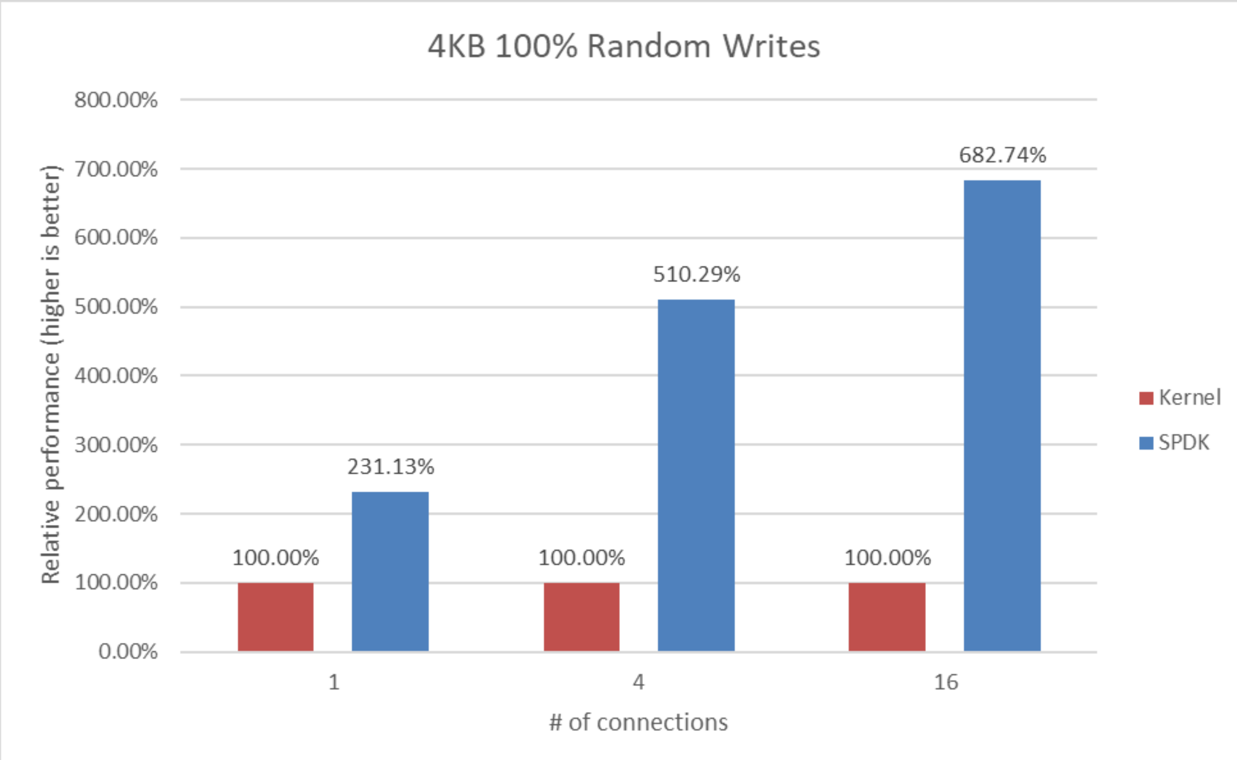

Figure 11: Relative Performance Comparison of Linux Kernel vs. SPDK NVMe-oF RDMA Target for 4KB 100% Random Write QD=64, using Kernel Initiators

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

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

Connections per subsystem	Bandwidth (MBps)	IOPS (k)	Avg. Latency (usec)	# CPU Cores
1	22342.71	5719.7	178.9	11.4
4	15766.84	4036.3	253.9	19.9
16	13428.55	3437.7	296.7	26.7

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

Connections per subsystem	Bandwidth (MBps)	IOPS (k)	Avg. Latency (usec)	# CPU Cores
1	18097.41	4632.9	221.1	4.0
4	16164.64	4138.1	247.1	4.0
16	13737.85	3516.9	290.0	4.0

4KB Random Read-Write Results

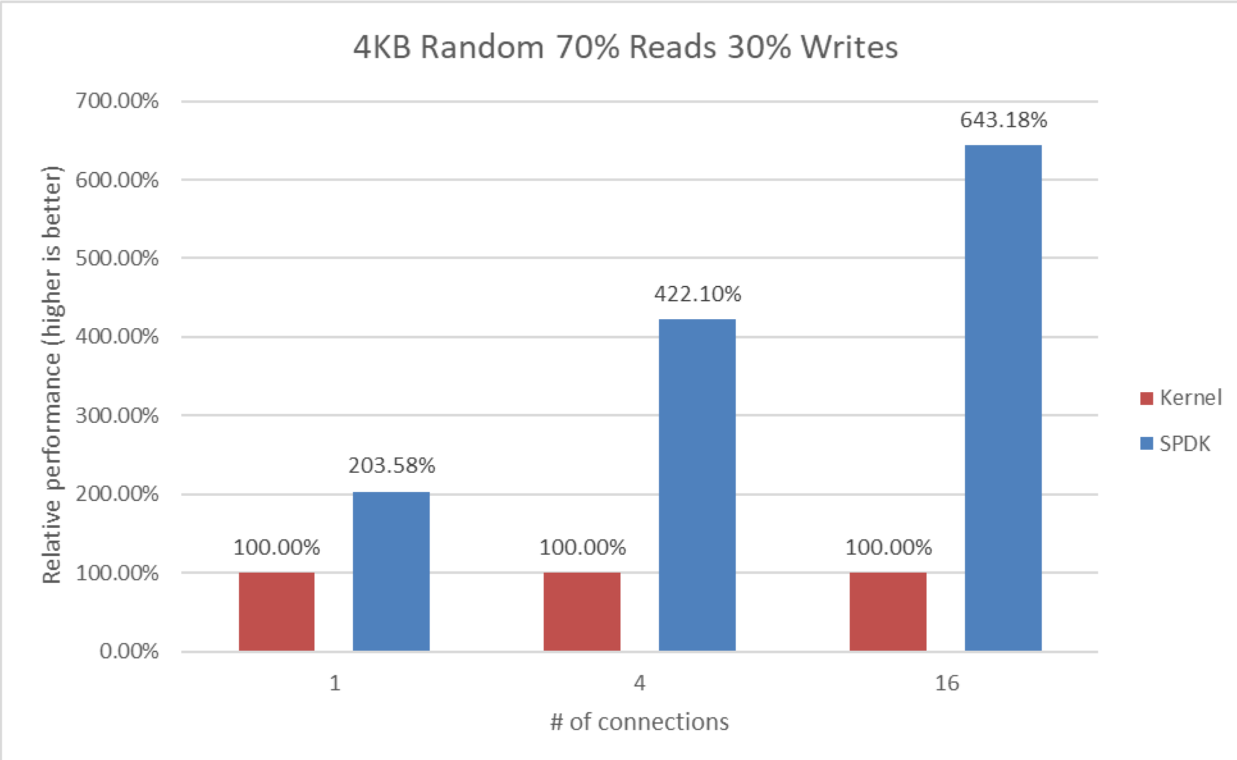

Figure 12: Relative Performance Comparison ofLinux Kernel vs. SPDK NVMe-oF RDMA Target for 4KB Random 70% Read 30% Write QD=64, using Kernel Initiators

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

Connections per subsystem	Bandwidth (MBps)	IOPS (k)	Avg. Latency (usec)	# CPU Cores
1	17672.51	4524.2	226.2	10.6
4	17182.05	4398.6	232.3	21.3
16	14330.68	3668.7	277.9	30.9

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

Connections per subsystem	Bandwidth (MBps)	IOPS (k)	Avg. Latency (usec)	# CPU Cores
1	13549.62	3468.7	295.1	4.0
4	13631.62	3489.7	292.9	4.0
16	11942.88	3057.4	333.6	4.0

Conclusions

1. When the SPDK NVMe-oF Target was configured with 4 CPU cores the performance peaked at 1 connection per subsystem for all workloads. Increasing the number of connections per subsystem beyond these values resulted in higher average latency and lower IOPS.

2. The performance for the Linux Kernel NVMe-oF Target peaked at 1 connection per subsystem for all workloads. Increasing the number of connections only increases the latency and CPU utilization.

3. The SPDK NVMe-oF target shows up to 6.8x more IOPS/Core relative to the Linux Kernel NVMe-oF target as the number of connections per subsystem increased.

Summary

This report showcased performance results with SPDK NVMe-oF RDMA Target and Initiator under various test cases, including:

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

It compared performance results while running the Linux Kernel NVMe-oF RDMA (Target/Initiator) against the accelerated polled-mode driven SPDK NVMe-oF RDMA (Target/Initiator) implementation. Like in the last report, throughput scales up and latency decreases almost linearly with the scaling of SPDK NVMe-oF target and initiator cores.

It was also observed that the SPDK NVMe-oF Target average latency is up to 4.9 usec lower than Kernel when testing against null bdev based backend. The advantage of SPDK is even greater when comparing NVMe-oF Initiators: the SPDK NVMe-oF RDMA average latency is up to 2.9 times lower than Kernel initiator.

The SPDK NVMe-oF Target performed up to 6.7 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 different hardware and software configurations.

Appendix A

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


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

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

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

```
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode13"
      ]
    },
    {
      "addr": {
        "adrfam": "ipv4",
        "traddr": "10.0.1.1",
        "trsvcid": "4433",
        "trtype": "rdma"
      },
      "portid": 14,
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode14"
      ]
    },
    {
      "addr": {
        "adrfam": "ipv4",
        "traddr": "10.0.1.1",
        "trsvcid": "4434",
        "trtype": "rdma"
      },
      "portid": 15,
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode15"
      ]
    },
    {
      "addr": {
        "adrfam": "ipv4",
        "traddr": "10.0.1.1",
        "trsvcid": "4435",
        "trtype": "rdma"
      },
      "portid": 16,
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode16"
      ]
    }
  ],
"hosts": [],
 "subsystems": [
    {
      "allowed_hosts": [],
      "attr": {
        "allow_any_host": "1",
        "version": "1.3"
      },
      "namespaces": [
        {
          "device": {
            "path": "/dev/nvme0n1",
```

```
          "uuid": "b53be81d-6f5c-4768-b3bd-203614d8cf20"
        },
        "enable": 1,
        "nsid": 1
      }
    ],
    "nqn": "nqn.2018-09.io.spdk:cnode1"
  },
  {
    "allowed_hosts": [],
    "attr": {
      "allow_any_host": "1",
      "version": "1.3"
    },
    "namespaces": [
      {
        "device": {
          "path": "/dev/nvme1n1",
          "uuid": "12fcf584-9c45-4b6b-abc9-63a763455cf7"
        },
        "enable": 1,
        "nsid": 2
      }
    ],
    "nqn": "nqn.2018-09.io.spdk:cnode2"
  },
  {
    "allowed_hosts": [],
    "attr": {
      "allow_any_host": "1",
      "version": "1.3"
    },
    "namespaces": [
      {
        "device": {
          "path": "/dev/nvme2n1",
          "uuid": "ceae8569-69e9-4831-8661-90725bdf768d"
        },
        "enable": 1,
        "nsid": 3
      }
    ],
    "nqn": "nqn.2018-09.io.spdk:cnode3"
  },
 {
   "allowed_hosts": [],
    "attr": {
      "allow_any_host": "1",
      "version": "1.3"
    },
    "namespaces": [
      {
        "device": {
          "path": "/dev/nvme3n1",
          "uuid": "39f36db4-2cd5-4f69-b37d-1192111d52a6"
        },
        "enable": 1,
```

```
          "nsid": 4
        }
      ],
      "nqn": "nqn.2018-09.io.spdk:cnode4"
    },
    {
      "allowed_hosts": [],
      "attr": {
        "allow_any_host": "1",
        "version": "1.3"
      },
      "namespaces": [
        {
          "device": {
            "path": "/dev/nvme4n1",
            "uuid": "984aed55-90ed-4517-ae36-d3afb92dd41f"
          },
          "enable": 1,
          "nsid": 5
        }
      ],
      "nqn": "nqn.2018-09.io.spdk:cnode5"
    },
    {
      "allowed_hosts": [],
      "attr": {
        "allow_any_host": "1",
        "version": "1.3"
      },
      "namespaces": [
        {
          "device": {
            "path": "/dev/nvme5n1",
            "uuid": "d6d16e74-378d-40ad-83e7-b8d8af3d06a6"
          },
          "enable": 1,
          "nsid": 6
        }
      ],
      "nqn": "nqn.2018-09.io.spdk:cnode6"
    },
   {
      "allowed_hosts": [],
      "attr": {
        "allow_any_host": "1",
        "version": "1.3"
      },
      "namespaces": [
        {
          "device": {
            "path": "/dev/nvme6n1",
            "uuid": "a65dc00e-d35c-4647-9db6-c2a8d90db5e8"
          },
          "enable": 1,
          "nsid": 7
        }
      ],
```

```
      "nqn": "nqn.2018-09.io.spdk:cnode7"
    },
    {
      "allowed_hosts": [],
      "attr": {
        "allow_any_host": "1",
        "version": "1.3"
      },
      "namespaces": [
        {
          "device": {
            "path": "/dev/nvme7n1",
            "uuid": "1b242cb7-8e47-4079-a233-83e2cd47c86c"
          },
          "enable": 1,
          "nsid": 8
        }
      ],
      "nqn": "nqn.2018-09.io.spdk:cnode8"
    },
    {
      "allowed_hosts": [],
      "attr": {
        "allow_any_host": "1",
        "version": "1.3"
      },
      "namespaces": [
        {
          "device": {
            "path": "/dev/nvme8n1",
            "uuid": "f12bb0c9-a2c6-4eef-a94f-5c4887bbf77f"
          },
          "enable": 1,
          "nsid": 9
        }
      ],
      "nqn": "nqn.2018-09.io.spdk:cnode9"
    },
    {
      "allowed_hosts": [],
      "attr": {
        "allow_any_host": "1",
        "version": "1.3"
      },
      "namespaces": [
        {
          "device": {
            "path": "/dev/nvme9n1",
            "uuid": "40fae536-227b-47d2-bd74-8ab76ec7603b"
          },
          "enable": 1,
          "nsid": 10
        }
      ],
      "nqn": "nqn.2018-09.io.spdk:cnode10"
    },
    {
```

```
      "allowed_hosts": [],
      "attr": {
        "allow_any_host": "1",
        "version": "1.3"
      },
      "namespaces": [
        {
          "device": {
            "path": "/dev/nvme10n1",
            "uuid": "b9756b86-263a-41cf-a68c-5cfb23c7a6eb"
          },
          "enable": 1,
          "nsid": 11
        }
      ],
      "nqn": "nqn.2018-09.io.spdk:cnode11"
    },
    {
      "allowed_hosts": [],
      "attr": {
        "allow_any_host": "1",
        "version": "1.3"
      },
      "namespaces": [
        {
          "device": {
            "path": "/dev/nvme11n1",
            "uuid": "9d7e74cc-97f3-40fb-8e90-f2d02b5fff4c"
          },
          "enable": 1,
          "nsid": 12
        }
      ],
      "nqn": "nqn.2018-09.io.spdk:cnode12"
    },
{
      "allowed_hosts": [],
      "attr": {
        "allow_any_host": "1",
        "version": "1.3"
      },
      "namespaces": [
        {
          "device": {
            "path": "/dev/nvme12n1",
            "uuid": "d3f4017b-4f7d-454d-94a9-ea75ffc7436d"
          },
          "enable": 1,
          "nsid": 13
        }
      ],
      "nqn": "nqn.2018-09.io.spdk:cnode13"
    },
    {
      "allowed_hosts": [],
      "attr": {
        "allow_any_host": "1",
```

```
        "version": "1.3"
      },
      "namespaces": [
        {
          "device": {
            "path": "/dev/nvme13n1",
            "uuid": "6b9a65a3-6557-4713-8bad-57d9c5cb17a9"
          },
          "enable": 1,
          "nsid": 14
        }
      ],
      "nqn": "nqn.2018-09.io.spdk:cnode14"
    },
    {
      "allowed_hosts": [],
      "attr": {
        "allow_any_host": "1",
        "version": "1.3"
      },
      "namespaces": [
        {
          "device": {
            "path": "/dev/nvme14n1",
            "uuid": "ed69ba4d-8727-43bd-894a-7b08ade4f1b1"
          },
          "enable": 1,
          "nsid": 15
        }
      ],
      "nqn": "nqn.2018-09.io.spdk:cnode15"
    },
  {
      "allowed_hosts": [],
      "attr": {
        "allow_any_host": "1",
        "version": "1.3"
      },
      "namespaces": [
        {
          "device": {
            "path": "/dev/nvme15n1",
            "uuid": "5b8e9af4-0ab4-47fb-968f-b13e4b607f4e"
          },
          "enable": 1,
          "nsid": 16
        }
      ],
      "nqn": "nqn.2018-09.io.spdk:cnode16"
    }
  ]
}
```

Notices & Disclaimers

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.

§