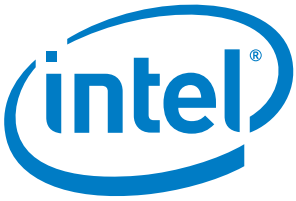


SPDK vhost-scsi Performance Report
Release 17.07

Test Date: Sep 14, 2017

Revision History

Date	Revision	Comment
Aug 15, 2017	1.0	Initial document for release
Aug 21, 2017	2.0	Reviews incorporated
Sept 7, 2017	3.0	Updated Test Case # 3

Contents

Audience and Purpose

This report is intended for people who are interested in looking at SPDK vhost-scsi performance and its comparison to Linux kernel vhost-scsi. It provides performance and efficiency information between SPDK vhost-scsi and Linux Kernel vhost-scsi software stacks under various test cases.

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 with repeatable and reproducible results. This report can also be viewed as information regarding best known method/practice when performance testing SPDK vhost-scsi stack.

Test setup

Intel® Server board S2600WF populated with 5x PCIe Gen3.0 x8 Switches was used which enabled to hook up to as many as 24x NVMe drives. Following are the configuration details listed regarding the platform:

Hardware Configuration

Item	Description
Server Platform	Intel® Server Board S2600WF
CPU	Intel® Xeon® Platinum 8180 Processor (38.5MB L3, 2.50 GHz) https://ark.intel.com/products/120496/Intel-Xeon-Platinum-8180-Processor-38_5M-Cache-2_50-GHz Number of cores 28, number of threads 56
Memory	Total 192 GBs over 12 channels @ 2667 MHz
Operating System	Ubuntu 16.04.1 LTS
BIOS	SE5C620.86B.01.00.0511.051220170820
Linux kernel version	4.11.0 (built from kernel.org)
Qemu version	2.9 with vhost-user scsi support
Libvirt version	3.0.0
SPDK version	SPDK 17.07 (3bfecec994)
VM OS	CentOS 7 (3.10.0 kernel)
Storage	24x Intel® Optane™ P4800x 375GB (FW:E2010311)

BIOS settings

Item	Description
BIOS	VT-d enabled Hyper threading Disabled CPU Power and Performance Policy <Performance> CPU C-state No Limit CPU P-state Enabled Enhanced Intel® Speedstep® Tech Enabled

	Turbo Boost Enabled

Introduction to SPDK vhost-scsi target

SPDK vhost is a userspace target designed to extend the performance efficiencies of SPDK into QEMU/KVM virtualization environments. This SPDK vhost-scsi target presents a broad range of SPDK-managed block devices into virtual machines. SPDK team has leveraged existing SPDK SCSI layer, DPDK vhost library, QEMU vhost-scsi and vhost-user functionality in order to create this high performance SPDK userspace vhost-scsi target.

SPDK vhost-scsi target working

QEMU setups vhost-scsi target via UNIX domain socket. Vhost-scsi target transfers data to/from quest VM via shared memory. QEMU pre-allocates huge pages for guest VM to enable direct DMA by vhost-scsi target. Guest VM submits I/O directly to vhost-scsi target via virtqueues in shared memory as shown in Figure 1. It should be noted that there is no QEMU intervention during the submission I/O process. Vhost-scsi target then completes I/O to guest VM via virtqueues in shared memory. There is a completion interrupt sent using eventfd which requires system call and guest VM exits.

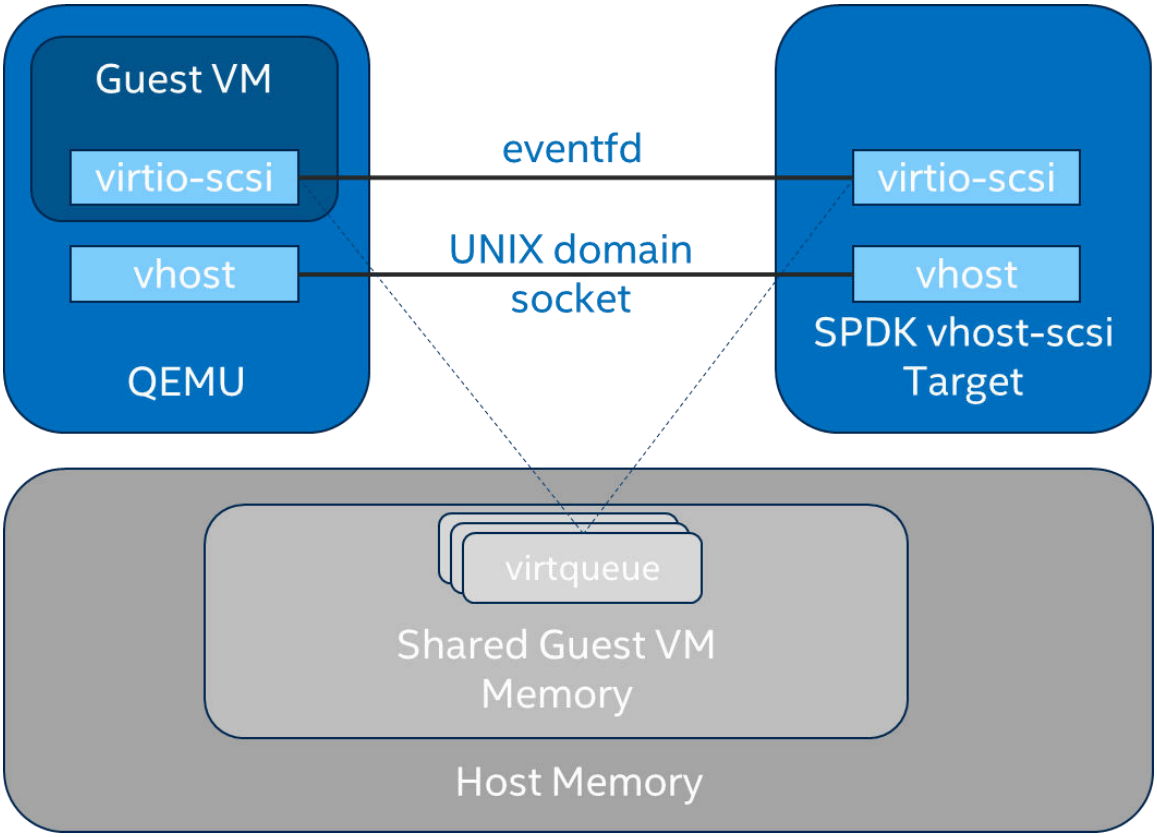
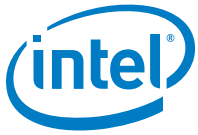


Figure 1: SPDK vhost-scsi Architecture

This report is prepared to uncover performance comparisons between traditional interrupt-driven kernel vhost-scsi vs. accelerated polled-mode driven SPDK vhost-scsi under 4 different test cases using local NVMe storage.



Test Case 1: SPDK vhost-scsi core scaling

This test case was performed in order to understand aggregate VM performance with SPDK vhost-scsi I/O core scaling. We ran 48 Virtual Machines, each running FIO workloads:

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

We increased the number of CPU cores used by the SPDK vhost-scsi target to process I/O from 1 to 2 to 4 and 8 and measured the throughput (in IOPS) and latency. FIO was run in each VM in a client-server mode which helped us kick FIO across all 48 VMs at same time. Results in the table represent aggregate performance (IOPS & avg. latency) seen across all the VMs.

Item	Description
Test Case	Test SPDK vhost-scsi target I/O core scaling performance
Test configuration	**FIO Configuration:** FIO-2.1.10, each VM running block I/O to a single vhost-user-scsi backend device. **VM Configuration:** • Each VM was configured with 2GB Memory, 1 vCPU and CentOS 7 OS. • VM's vCPUs utilized cores which were not running SPDK vhost-user-scsi process. • Total 48 VMs. Each VM has a single vhost-scsi device which is one of two equal partitions of an Optane drive. 24 Optanes x 2 partitions/Optane = 48 partitions. **SPDK vhost-scsi target Configuration:** • 1 core: 0x1 Reactor mask • 2 core: 0x3 Reactor mask • 4 core: 0xF Reactor mask • 8 core: 0xFF Reactor mask **Kernel vhost-scsi Configuration:** • N/A in this test case
FIO config	ioengine=libaio direct=1 rw=randrw rwmixread= 100 (100% reads), 70 (70% reads 30% writes), 0 (100% writes) thread=1 norandommap=1 time_based=1 runtime=300s ramp_time=10s bs=4K iodepth=32 numjobs=1 [test] filename=/dev/sda #sda is vhost-scsi device (Optane drive partition)

Test Result: 4K 100% Random Read IOPS

# of Cores	Throughput (IOPS)	Avg. Latency (usec)
1 core	0.78 M	1920
2 cores	1.82 M	820
4 cores	3.65 M	420
8 cores	7.3 M	210

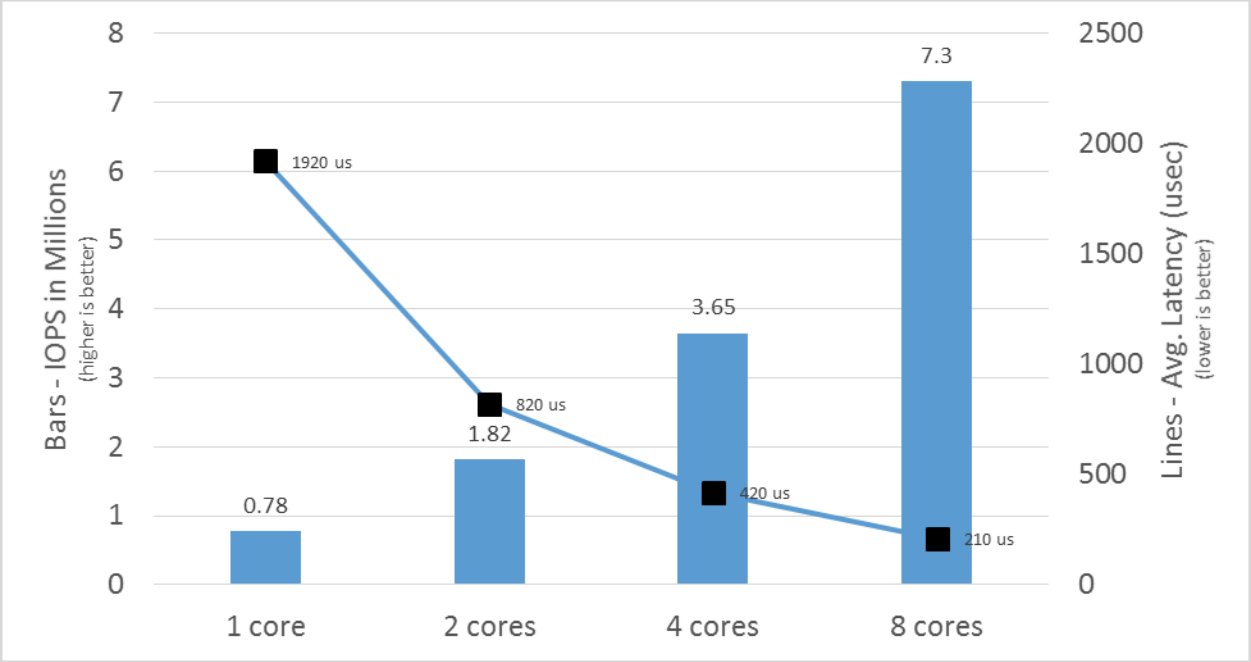


Figure 2: SPDK vhost-scsi target I/O core scaling: IOPS vs. Latency while running 4KB 100% Random read workload

Test Result: 4K 100% Random Writes IOPS

# of Cores	Throughput (IOPS)	Avg. Latency (usec)
1 core	0.79 M	1920
2 cores	1.77 M	830
4 cores	3.6 M	425
8 cores	7.1 M	213

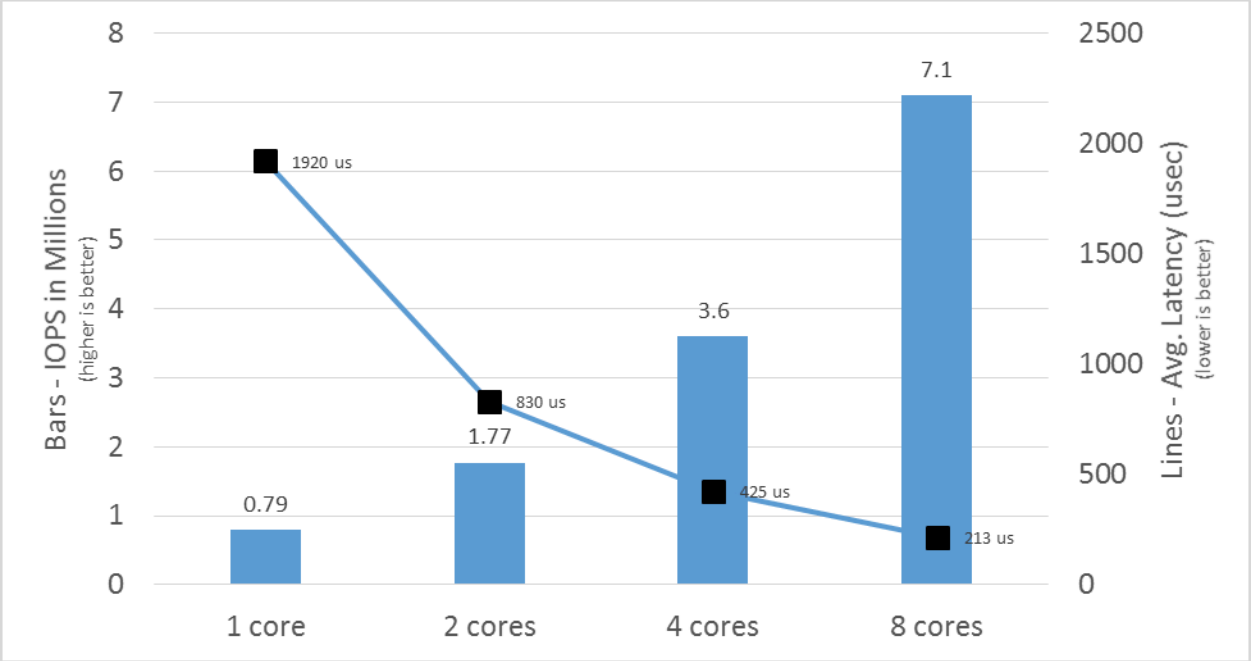


Figure 3: SPDK vhost-scsi target I/O core scaling: IOPS vs. Latency while running 4KB 100% Random write workload

Test Result: 4K 70% Read 30% Write IOPS

# of Cores	Throughput (IOPS)	Avg. Latency (usec)
1 core	0.78 M	1920
2 cores	1.75 M	840
4 cores	3.6 M	425
8 cores	7.1 M	213

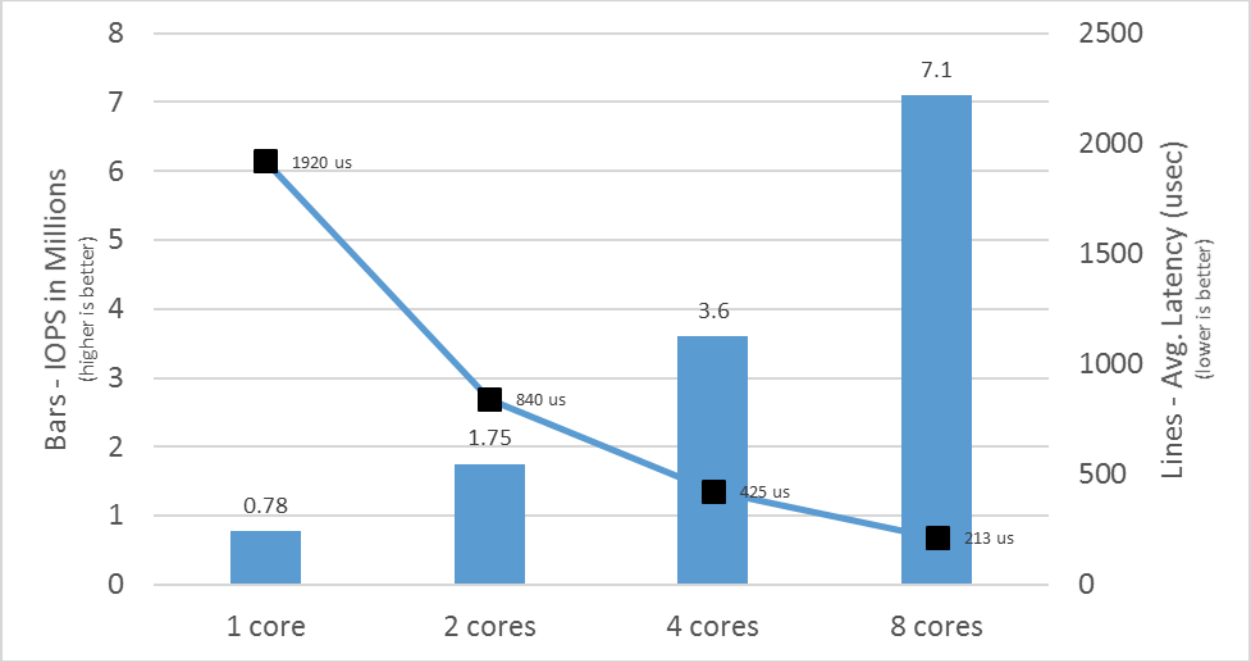


Figure 4: SPDK vhost-scsi tagret I/O core scaling: IOPS vs. Latency while running 4KB Random 70% read 30% write workload

Conclusion:

1. For all the 3 workloads, throughput scales up and latency decreases almost linearly with the scaling of SPDK vhost-scsi target I/O cores.

Test Case 2: Rate Limiting IOPS per VM

This test case was geared towards understanding how many VMs can be supported with a given number of IOPS per vhost-scsi device. Both read and write IOPS were rate limited for each vhost-scsi device on each of the VMs and then VM density was compared between SPDK & Linux Kernel.

20K IOPS were chosen as the rate limiter using linux cgroups. Each individual VM was running FIO with the following workloads:

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

Results in the table represent aggregate performance (IOPS & CPU %) seen across all the VMs.

Item	Description
Test Case	Test Rate Limiting IOPS/VM to 20000 IOPS
Test configuration	**FIO Configuration:** FIO-2.1.10, each VM running block I/O to a single vhost-user-scsi backend device. **VM Configuration:** • 24, 48 and 96 VMs tested • Each VM was configured with 2GB Memory, 1 vCPU and CentOS 7 OS. • VM's vCPUs utilized cores which were not running SPDK vhost-user-scsi process. • Total 48 VMs. Each VM has a single vhost-scsi device which is one of two equal partitions of an Optane drive. 24 Optanes x 2 partitions/Optane = 48 partitions. • For the 48 VMs case, there were total 46 cores available to allocate as 10 cores were running vhost-scsi process. Each of the vCPUs for 44 VMs were run on separate physical cores on the system. The remaining 4 VMs shared each of their vCPUs with 2 physical cores on the system. **Note:** While running 96 VMs, Hyper Threading was enabled to allow each VM to run on each individual logical core on the system. Also for 96 VMs case, 4 partitions were created per Optane drive. 24 Optanes x 4 partitions/Optane = 96 partitions. Each VM with single vhost-scsi device which is one of four equal partition of Optane drive. • vhost-scsi device on each of the VMs was throttled to run 20000 IOPS (read and write) using: *echo "8:0 20000" > /sys/fs/cgroup/blkio/blkio.throttle.read_iops_device* *echo "8:0 20000" > /sys/fs/cgroup/blkio/blkio.throttle.write_iops_device* **SPDK vhost-scsi target Configuration:** • 10 vhost-scsi SPDK cores. • 0x3F000000F. This was chosen a NUMA optimized config which allows the compute and storage I/O to run on the same socket and in-turn avoids cross-socket transactions per I/O. **Kernel vhost-scsi Configuration:** • Used cgroups to let first 10 cores run vhost-scsi process **Note**: We did not explore NUMA optimization with kernel vhost-scsi

Command line	ioengine=libaio direct=1 rw=randrw rwmixread= 100 (100% reads), 0 (100% writes) thread=1 norandommap=1 time_based=1 runtime=300s ramp_time=10s bs=4K iodepth=1 numjobs=1 [test] filename=/dev/sda #sda is vhost-scsi device (Optane drive partition)

Test Result: 4K 100% Random Read IOPS

# of VMs	Throughput Kernel (IOPS)	Throughput SPDK (IOPS)	CPU % util. VM (Kernel)	CPU % util. VM (SPDK)
24 VMs	480000	480000	85	35
48 VMs	730000	960000	90	35
96 VMs	700000	1700000	90	35

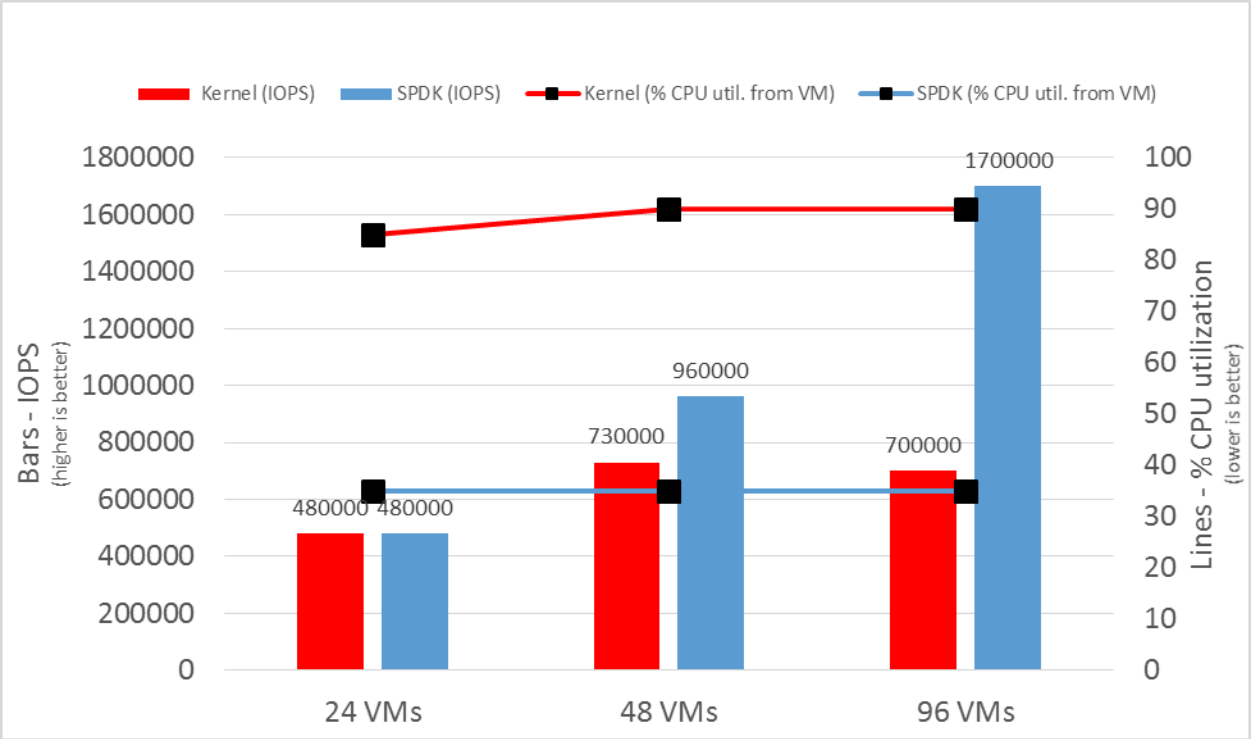


Figure 5: Kernel vhost-scsi vs. SPDK vhost-scsi target : 4KB 100% Random reads, IOPS vs. % CPU Utilization with increasing no. of VMs while rate limiting IOPS

Test Result: 4K 100% Random Writes IOPS

# of VMs	Throughput Kernel (IOPS)	Throughput SPDK (IOPS)	CPU % util. VM (Kernel)	CPU % util. VM (SPDK)
24 VMs	480000	480000	85	35
48 VMs	700000	960000	90	35
96 VMs	700000	1700000	90	35

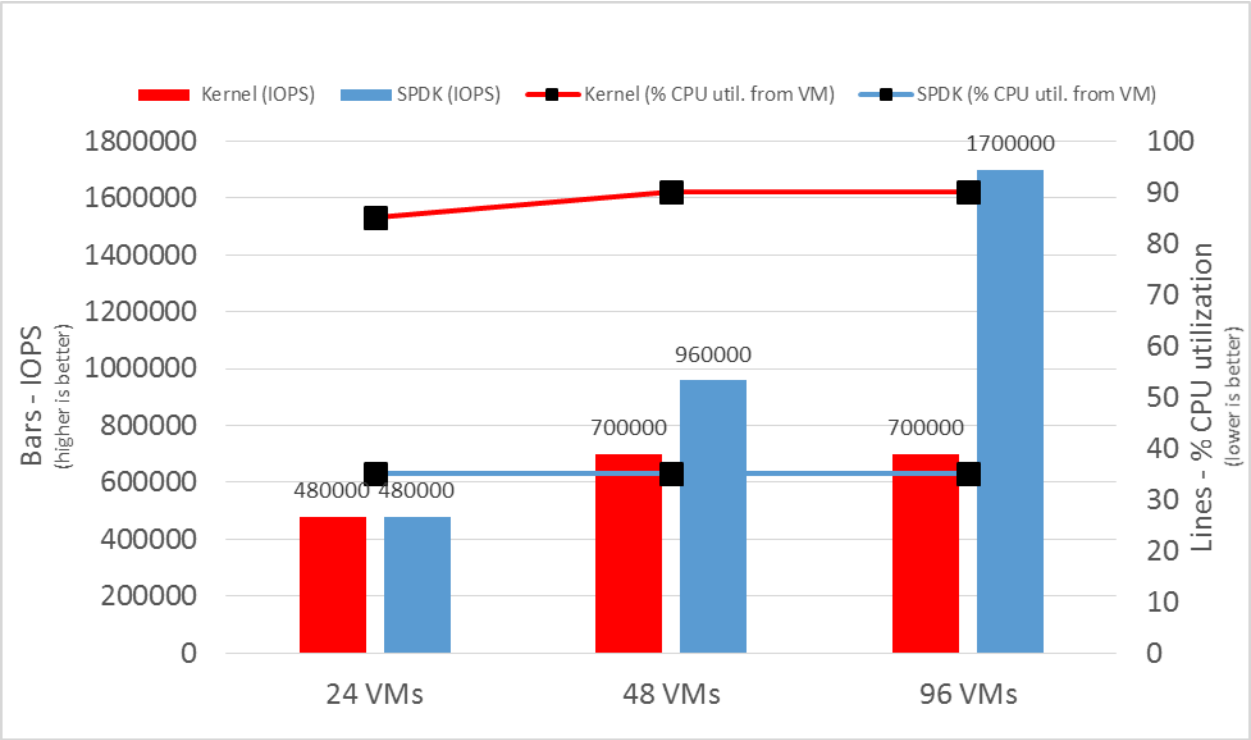
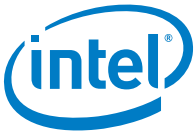


Figure 6: Kernel vhost-scsi vs. SPDK vhost-scsi target: 4KB 100% Random writes, IOPS vs. % CPU Utilization with increasing no. of VMs while rate limiting IOPS

Conclusion:

1. The VM was running a low queue depth (QD=1) workload while running 4KB 100% read or 4KB 100% write to the vhost-scsi device. It was observed that when VMs were using the SPDK vhost-scsi target, the I/O processing consumes approximately 35% of the virtual machines CPU cycles. However, with the kernel vhost-scsi the VMs are using as much as 90% of their CPU cycles for I/O processing. So using vhost-scsi with SPDK, one would get plenty of head room on the VMs for other value add tasks as compared to vhost-scsi with Kernel.



Test Case 3: Performance per NVMe Drive

This test case was performed in order to understand performance and efficiency of the vhost-scsi process using SPDK vs. Linux Kernel with single NVMe drive on 2 VMs. Each VM has a single vhost-scsi device which is one of two equal partitions of an NVMe drive.

Results in the table represent performance (IOPS, avg. latency & CPU %) seen from the VM. The VM was running FIO with following workloads:

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

Item	Description
Test Case	Test storage performance per VM using vhost-scsi
Test configuration	**FIO Configuration:** FIO-2.1.10, each VM running block I/O to a single vhost-user-scsi backend device. **VM Configuration:** • 2 VMs tested • Each VM was configured with 2GB Memory, 1 vCPU and CentOS 7 OS. • Each VM has a single vhost-scsi device which is one of two equal partitions of an Optane drive. **SPDK vhost-scsi target Configuration:** • Vhost-scsi process was run on separate individual physical core. • 0x1 Reactor Mask **Kernel vhost-scsi Configuration:** • Vhost-scsi process was run on separate individual physical core using cgroups.
Command line	ioengine=libaio direct=1 rw=randrw rwmixread= 100 (100% reads), 70 (70% reads, 30% writes) & 0 (100% writes) thread=1 norandommap=1 time_based=1 runtime=300s ramp_time=10s bs=4K iodepth=32 numjobs={1, 8 and 32} [test] filename=/dev/sda #sda is vhost-scsi device (1x Optane drive)

Test Result SPDK

Access Pattern	Queue Depth	Throughput (IOPS)	Avg. Latency (usec)
4K 100% Random Reads	1	163000	12.27
4K 100% Random Reads	8	584000	27.40
4K 100% Random Reads	32	584000	109.59
4K 100% Random Writes	1	149500	13.38
4K 100% Random Writes	8	518000	30.89
4K 100% Random Writes	32	552000	115.94
4K 70% Reads 30% Writes	1	156000	12.82
4K 70% Reads 30% Writes	8	482000	33.20
4K 70% Reads 30% Writes	32	505000	126.73

Test Result Linux Kernel

Access Pattern	Queue Depth	Throughput (IOPS)	Avg. Latency (usec)
4K 100% Random Reads	1	92000	21.74
4K 100% Random Reads	8	305000	52.46
4K 100% Random Reads	32	378000	169.31
4K 100% Random Writes	1	91000	21.98
4K 100% Random Writes	8	291000	54.98
4K 100% Random Writes	32	371000	172.51
4K 70% Reads 30% Writes	1	93000	21.51
4K 70% Reads 30% Writes	8	303000	52.81
4K 70% Reads 30% Writes	32	380000	168.42

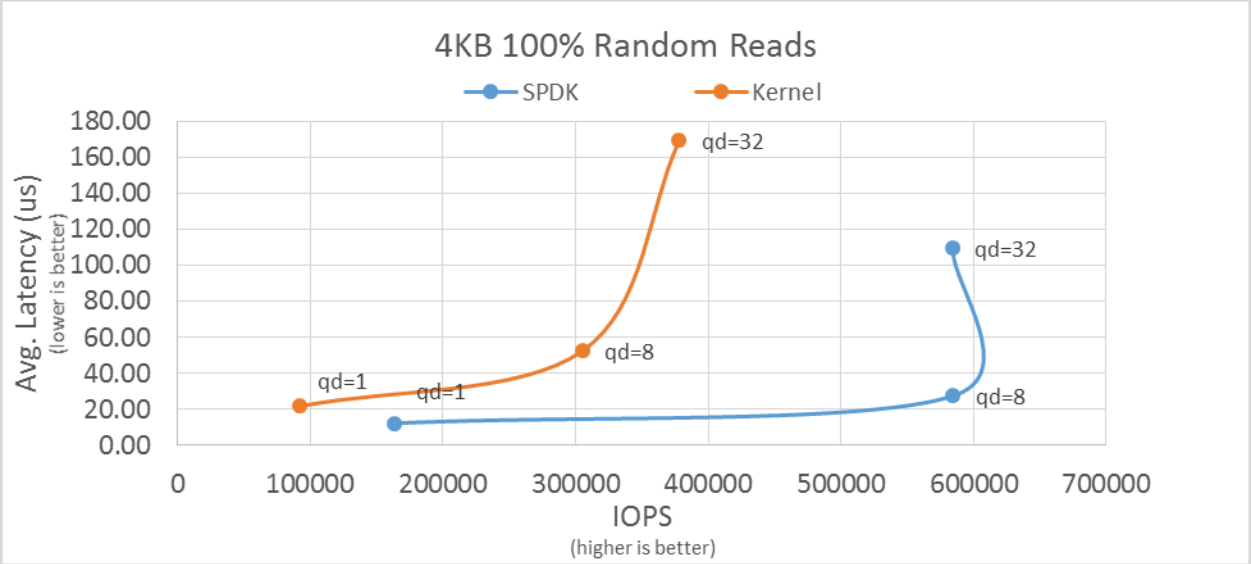


Figure 7: Kernel vhost-scsi vs. SPDK vhost-scsi target: 4KB 100% Random reads, IOPS vs.Latency per VM with increasing queue depth

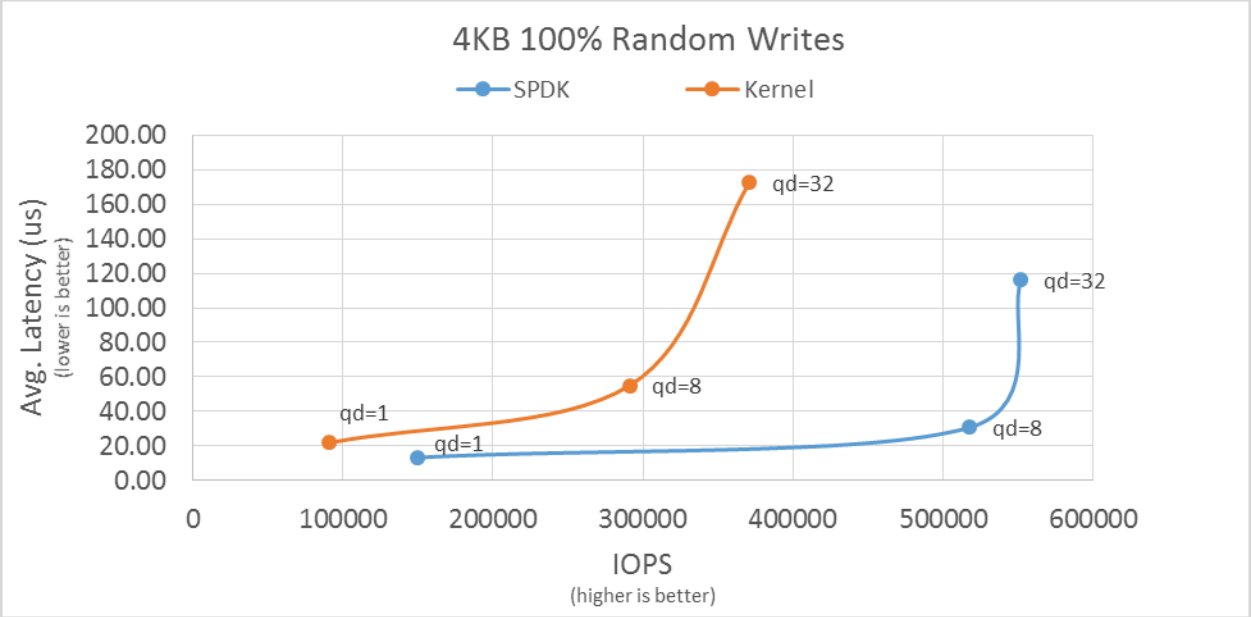


Figure 8: Kernel vhost-scsi vs. SPDK vhost-scsi target: 4KB 100% Random writes, IOPS vs. Latency per VM with increasing queue depth

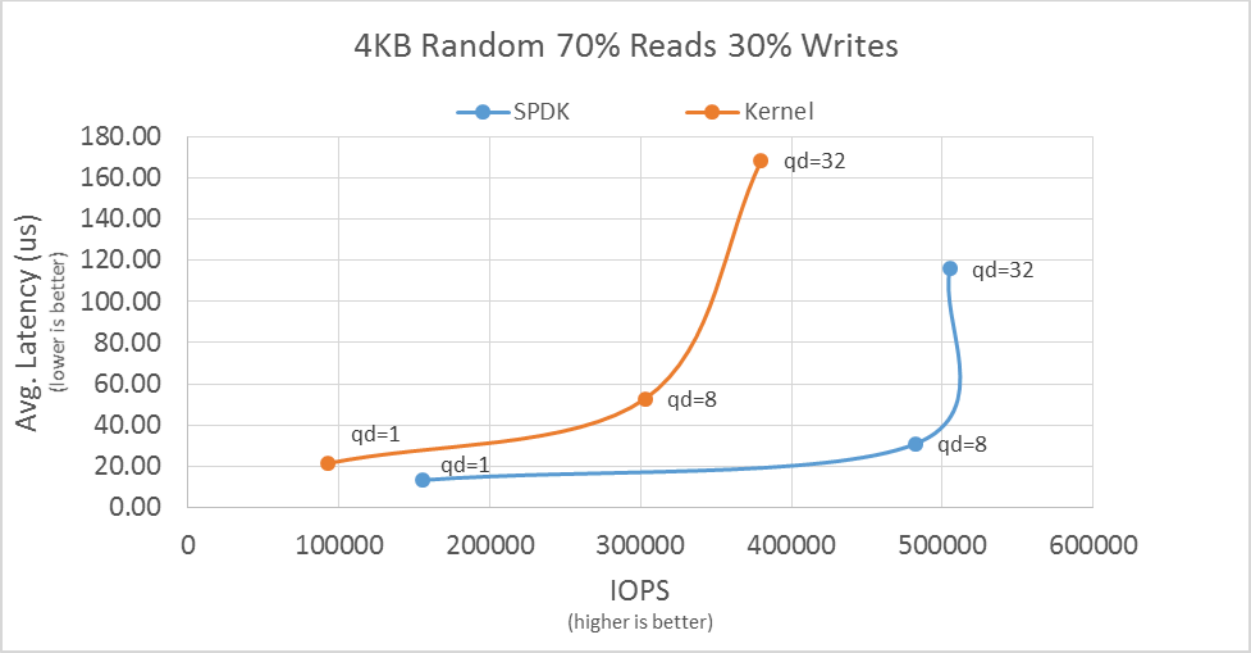


Figure 9: Kernel vhost-scsi vs. SPDK vhost-scsi target: 4KB Random 70% reads 30% writes, IOPS vs.Latency per VM with increasing queue depth

Conclusion:

1. SPDK vhost-scsi has lower latency and higher througput at all the queue depths (1, 8 and 32) against Kernel vhost-scsi

Test Case 4: Maximum IOPS

This test case was performed in order to understand maximum IOPS capability of our underlying system while running VMs using vhost-scsi with SPDK vs. Linux Kernel and at the same utilizing all the cores in the system. In order to do so, 10 physical cores were assigned for vhost-scsi process and remaining 46 cores for the 48 VMs vCPU. Results in the table represent aggregate IOPS across all the VMs.

Item	Description
Test Case	Test Maximum IOPS of underlying system with given number of VMs
Test configuration	• FIO-2.1.10, each VM running block I/O to a single vhost-user-scsi backend device. vhost-user-scsi process using SPDK. • 48 VMs. Both Kernel and SPDK vhost-scsi implementations were run using 10 separate cores, running vhost-scsi process. • SPDK: Reactor mask: 0x3F000000F • Kernel: Used cgroups to let first 10 cores run vhost-scsi process. • Each VM was configured with 2GB Memory, 1 vCPU and CentOS 7 OS. Each vCPU utilized a core which was not running the vhost-scsi process. • There were total 46 cores available to allocate as 10 cores were running vhost-scsi process. Each of the vCPUs for the 44 VMs were ran on separate physical cores on the system. The remaining 4 VMs shared each of their vCPUs with 2 physical cores on the system. **FIO Configuration:** FIO-2.1.10, each VM running block I/O to a single vhost-user-scsi backend device. **VM Configuration:** • 48 VMs tested. Both Kernel and SPDK vhost-scsi implementations were run using 10 separate cores, running vhost-scsi process • Each VM was configured with 2GB Memory, 1 vCPU and CentOS 7 OS. • There were total 46 cores available to allocate as 10 cores were running vhost-scsi process. Each of the vCPUs for the 44 VMs were ran on separate physical cores on the system. The remaining 4 VMs shared each of their vCPUs with 2 physical cores on the system. **SPDK vhost-scsi target Configuration:** • 10 vhost-scsi SPDK cores. • 0x3F000000F. This was chosen a NUMA optimized config which allows the compute and storage I/O to run on the same socket and in-turn avoids cross-socket transactions per I/O. **Kernel vhost-scsi Configuration:** • Used cgroups to let first 10 cores run vhost-scsi process. **Note**: We did not explore NUMA optimization with kernel vhost-scsi

Command line	ioengine=libaio direct=1 rw=randrw rwmixread= 100 (100% reads), 70 (70% reads, 30% writes) & 0 (100% writes) thread=1 norandommap=1 time_based=1 runtime=300s ramp_time=10s bs=4K iodepth=32 numjobs=1 [test] filename=/dev/sda #sda is vhost-scsi device (Optane drive partition)

Test Result SPDK

Access Pattern	Throughput (IOPS)	Avg. Latency (usec)	CPU % util. on VM
4K 100% Random Reads	9.5 M	161	100
4K 100% Random Writes	9.3 M	164	100
4K 70% Reads 30% Writes	9.5 M	159	100

Test Result Linux Kernel

Access Pattern	Throughput (IOPS)	Avg. Latency (usec)	CPU % util. on VM
4K 100% Random Reads	2.47 M	624	50
4K 100% Random Writes	2.47 M	627	50
4K 70% Reads 30% Writes	2.53 M	602	50

Figure 10: Kernel vhost-scsi vs. SPDK vhost-scsi target: IOPS vs. Avg. Latency for various I/O workloads using 10 vhost-scsi cores

Conclusion:

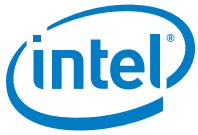
1. SPDK vhost-scsi can achieve up to 3.8x more IOPs and up to 74 % lower latency when compared to Kernel vhost-scsi.

2. It was also observed that the % CPU utilization per VM when using SPDK vhost-scsi target was 100% vs. 50% when using Kernel vhost-scsi. VM's running with kernel vhost-scsi could not do more than 50% of the CPU because the vhost-scsi I/O cores were seen running at 100% CPU which indicated CPU bottleneck with 10 kernel vhost-scsi I/O cores.

Note: Performance comparisons that were performed in this test case were done in such a way to be apples to apples while using 10 vhost-scsi cores (both with SPDK vs. Kernel)

Summary

This report compared performance results while running vhost-scsi using traditional interrupt-driven kernel vhost-scsi against the accelerated polled-mode driven SPDK implementation. Various local ephemeral configurations were demonstrated, including rate limiting IOPS, performance per VM, and maximum performance from underlying system when comparing kernel vs. SPDK vhost-scsi target implementations. This report provides information regarding methodologies and practices while benchmarking vhost-scsi 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.



DISCLAIMERS

§