SPDK Vhost performance report Release 20.10

Testing Date: August 2020

Performed by: Karol Latecki

Maciej Wawryk

Acknowledgments:

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

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

Contents

Audience and Purpose

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

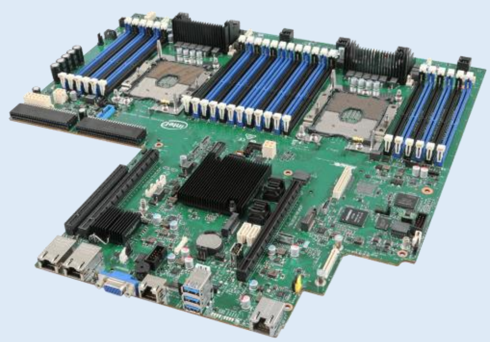
The purpose of this report is not to imply a single correct approach, but rather to provide a baseline of well-tested configurations and procedures that produce repeatable and reproducible results. This report can also be viewed as information regarding best known method when performance testing SPDK Vhost-scsi and Vhost-blk stacks.

Test setup

Hardware configuration

Table 1: Hardware setup configuration

<table>
<tr><th>Item</th><th>Description</th></tr>
<tr><td>Server Platform</td><td>Intel WolfPass R2224WFTZS

Server board S2600WFT
</td></tr>
<tr><td>Motherboard</td><td>S2600WFT</td></tr>
<tr><td>CPU</td><td>2 CPU sockets, Intel(R) Xeon(R) Gold 6230N CPU @ 2.30GHz
Number of cores 20 per socket, number of threads 40 per socket
Both sockets populated
Microcode: 0x4002f01</td></tr>
<tr><td>Memory</td><td>10 x 32GB Micron DDR4 36ASF4G72PZ-2G6H1R
Total 320 GBs
Memory channel population:
<table>
<tr><th>P1</th><th>P2</th></tr>
<tr><td>CPU1_DIMM_A1</td><td>CPU2_DIMM_A1</td></tr>
<tr><td>CPU1_DIMM_B1</td><td>CPU2_DIMM_B1</td></tr>
<tr><td>CPU1_DIMM_C1</td><td>CPU2_DIMM_C1</td></tr>
<tr><td>CPU1_DIMM_D1</td><td>CPU2_DIMM_D1</td></tr>
<tr><td>CPU1_DIMM_E1</td><td>CPU2_DIMM_E1</td></tr>
</table></td></tr>
<tr><td>Operating System</td><td>Fedora 30</td></tr>
<tr><td>BIOS</td><td>02.01.0010 (06.01.2020)</td></tr>
<tr><td>Linux kernel version</td><td>5.4.14-100.fc30.x86_64</td></tr>
</table>

SPDK version	SPDK 20.10
Qemu version	QEMU emulator version 3.1.1 (qemu-3.1.1-2.fc30)
Storage	**OS:** 1x 120GB Intel SSDSC2BB120G4 **Storage**: 24x Intel® P4610™ 1.6TBs (FW: VDV10152) (6 on CPU NUMA Node 0, 18 on CPU NUMA Node 1)

BIOS Settings

Table 2: Test platform BIOS settings

Item	Description
BIOS	VT-d = Enabled CPU Power and Performance Policy = <Performance> CPU C-state = No Limit CPU P-state = Enabled Enhanced Intel® Speedstep® Tech = Enabled Turbo Boost = Enabled Hyper Threading = Enabled

Virtual Machine Settings

Table 3: Guest VM configuration

Item	Description
CPU	2vCPU, pass through from physical host server. Explicit core usage enforced using "taskset –a –c" command. Related QEMU arguments used for starting the VM: -cpu host -smp 1
Memory	4 GB RAM. Memory is pre-allocated for each VM using Hugepages on host system and used from appropriate NUMA node, to match the CPU which was passed to the VM. Related QEMU arguments: -m 4096 -object memory-backend-file,id=mem,size=4096M,mem-path=/dev/hugepages,share=on,prealloc=yes,host-nodes=0,policy=bind
Operating System	Fedora 29
Linux kernel version	5.1.20-200.fc29.x86_64
Additional boot options in /etc/default/grub	• Multi queue enabled: scsi_mod.use_blk_mq=1 • Spectre-meltdown patches disabled: spectre_v2=off nopti

Kernel & BIOS Spectre-Meltdown information

Host server system uses 5.4.14 kernel version which is available from the DNF repository. The default Spectre-Meltdown mitigation patches for this kernel version have been left enabled.

The guest VM systems use 5.1.20 kernel version, which is available from the DNF repository. The default Spectre-Meltdown mitigation patches for this kernel version have been disabled on guest systems by adding the following in their /etc/default/grub file:

spectre_v2=off nopti

Introduction to the SPDK Vhost target

SPDK Vhost is a userspace target designed to extend the performance efficiencies of SPDK into QEMU/KVM virtualization environments. The 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 the high performance SPDK userspace Vhost target.

SPDK Vhost target architecture

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

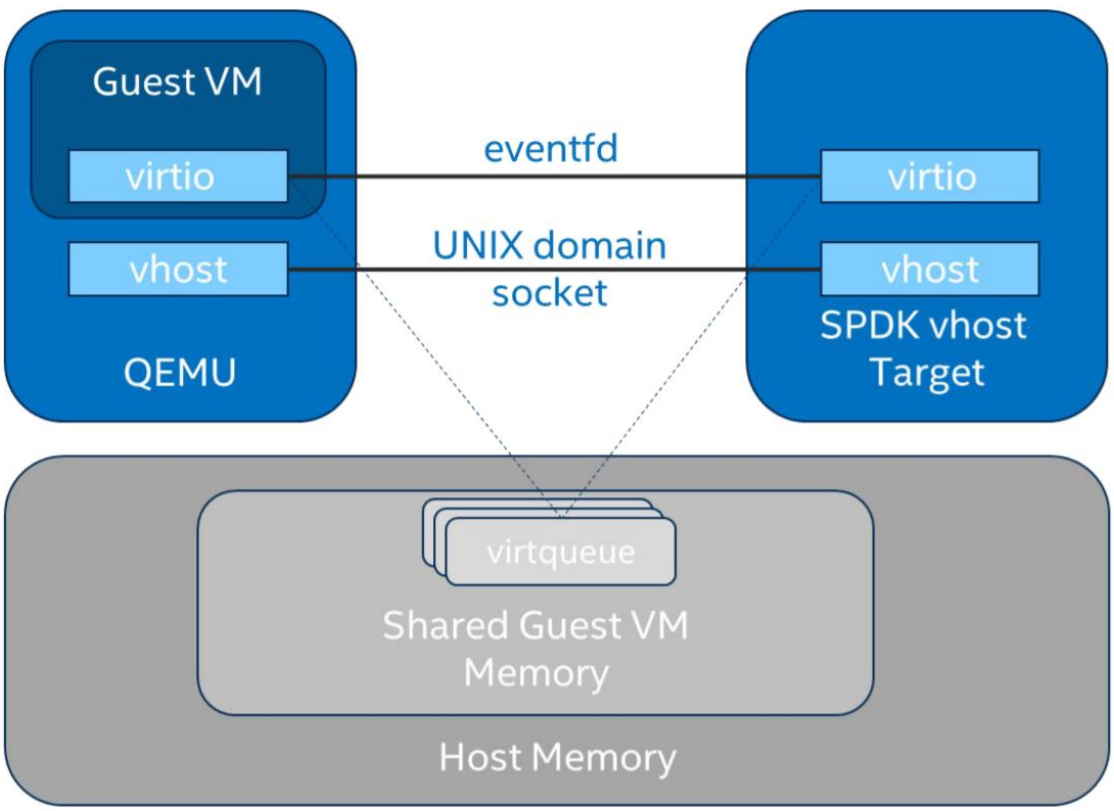


Figure 1: SPDK Vhost-scsi architecture

This report shows the performance comparisons between the traditional interrupt-driven kernel Vhost-scsi and the accelerated polled-mode driven SPDK Vhost-scsi under 4 different test cases using local

NVMe storage. Additionally, the SPDK Vhost-blk stack is included in the report for further comparison with the scsi stack.

Test Case 1: SPDK Vhost Core Scaling

This test case was performed in order to understand aggregate VM performance with SPDK Vhost I/O core scaling. We ran up to 36 virtual machines, each running following 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 SPDK Vhost target to process I/O from 1 up to 12 and measured the throughput (in IOPS) and latency. The number of VMs between test runs was not constant and was increased by 6 for each Vhost CPU added, up to a maximum of 36 VMs. VM number was not increased beyond 36 because of the platform capabilities in terms of available CPU cores.

FIO was run in client-server mode. FIO client was run on the host machine and distributed jobs to FIO servers run on each VM. This allowed us to start the FIO jobs across all VMs at the same time. The gtod_reduce=1 option was used to disable FIO latency measurements which allowed better IOPS and bandwidth results.

Results in the table and charts represent aggregate performance (IOPS and average latency) seen across all the VMs. The results are average of 3 runs.

Table 4: SPDK Vhost Core Scaling test configuration

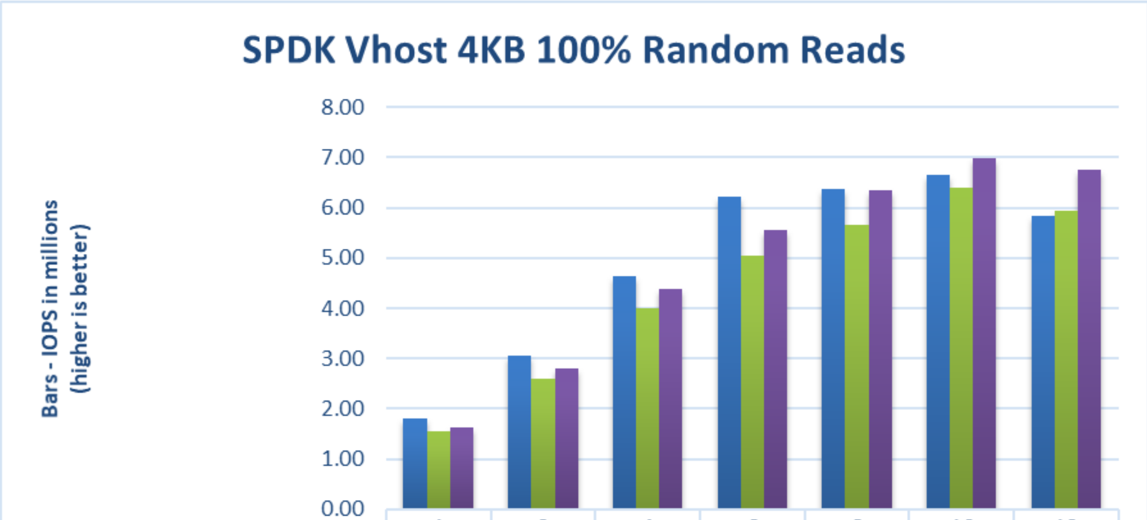
Item	Description
Test case	Test SPDK Vhost target I/O core scaling performance
Test configuration	**FIO Version**: fio-3.19 **VM Configuration**: • Common settings are described in the **Virtual Machine Settings** chapter. • Number of VMs: variable (6 VMs per 1 Vhost CPU core, up to 36 VMs max). • Each VM has a single Vhost device as a target for the FIO workload. This is achieved by sharing SPDK NVMe bdevs by using either a Split NVMe vbdev or Logical Volume bdev configuration. **SPDK Vhost target configuration:** • Test were run with both the Vhost-scsi and Vhost-blk stacks. • The Vhost-scsi stack was run with Split NVMe bdevs and Logical Volume bdevs. • Vhost-blk stack was run with Logical Volume bdevs. • Tests were ran with 1,2,4,6,8,10 and 12 cores for each stack-bdev combination. **Kernel Vhost target configuration:** - N/A

FIO configuration	[global] ioengine=libaio direct=1 thread=1 norandommap=1 time_based=1 gtod_reduce=1 ramp_time=60s runtime=240s numjobs=1 bs=4k rw=randrw rwmixread=100 (100% reads), 70 (70% reads, 30% writes), 0 (100% writes) iodepth={1, 32, 64}

4KB Random Read Results

Table 5: SPDK Vhost core scaling results, 4KB 100% Random Reads IOPS, QD=64

# of CPU cores	# of VMs	Stack / Backend	IOPS (millions)
1	6	SCSI / Split NVMe Bdev	1.81
		SCSI / Lvol Bdev	1.55
		BLK / Lvol Bdev	1.63
2	12	SCSI / Split NVMe Bdev	3.05
		SCSI / Lvol Bdev	2.61
		BLK / Lvol Bdev	2.79
4	24	SCSI / Split NVMe Bdev	4.63
		SCSI / Lvol Bdev	4.00
		BLK / Lvol Bdev	4.38
6	36	SCSI / Split NVMe Bdev	6.23
		SCSI / Lvol Bdev	5.04
		BLK / Lvol Bdev	5.56
8	36	SCSI / Split NVMe Bdev	6.37
		SCSI / Lvol Bdev	5.65
		BLK / Lvol Bdev	6.35
10	36	SCSI / Split NVMe Bdev	6.64
		SCSI / Lvol Bdev	6.40
		BLK / Lvol Bdev	6.99
12	36	SCSI / Split NVMe Bdev	5.83
		SCSI / Lvol Bdev	5.95
		BLK / Lvol Bdev	6.75



	1	2	4	6	8	10	12
Scsi + Split NVMe IOPS (million)	1.81	3.05	4.63	6.23	6.37	6.64	5.83
Scsi + Lvol Bdev IOPS (million)	1.55	2.61	4.00	5.04	5.65	6.40	5.95
BLK + Lvol Bdev IOPS (million)	1.63	2.79	4.38	5.56	6.35	6.99	6.75

Figure 2: Comparison of performance between various SPDK Vhost stack-bdev combinations for 4KB Random Read QD=64 workload

4KB Random Write Results

Table 6: SPDK Vhost core scaling results, 4KB 100% Random Write IOPS, QD=32

# of CPU cores	# of VMs	Stack / Backend	IOPS (millions)
1	6	SCSI / Split NVMe Bdev	1.57
		SCSI / Lvol Bdev	1.42
		BLK / Lvol Bdev	1.53
2	12	SCSI / Split NVMe Bdev	2.92
		SCSI / Lvol Bdev	2.62
		BLK / Lvol Bdev	2.90
4	24	SCSI / Split NVMe Bdev	4.82
		SCSI / Lvol Bdev	4.28
		BLK / Lvol Bdev	4.57
6	36	SCSI / Split NVMe Bdev	6.29
		SCSI / Lvol Bdev	5.62
		BLK / Lvol Bdev	6.15
8	36	SCSI / Split NVMe Bdev	6.30
		SCSI / Lvol Bdev	5.84
		BLK / Lvol Bdev	6.62
10	36	SCSI / Split NVMe Bdev	5.55
		SCSI / Lvol Bdev	5.46
		BLK / Lvol Bdev	5.98
12	36	SCSI / Split NVMe Bdev	5.63
		SCSI / Lvol Bdev	5.83
		BLK / Lvol Bdev	6.59

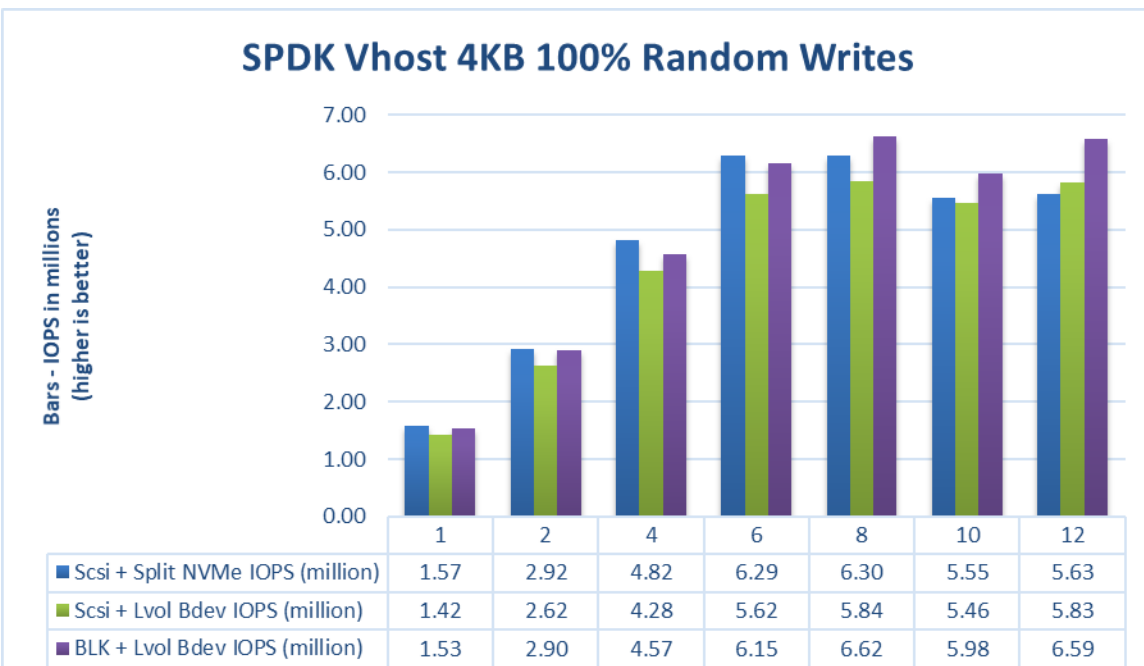
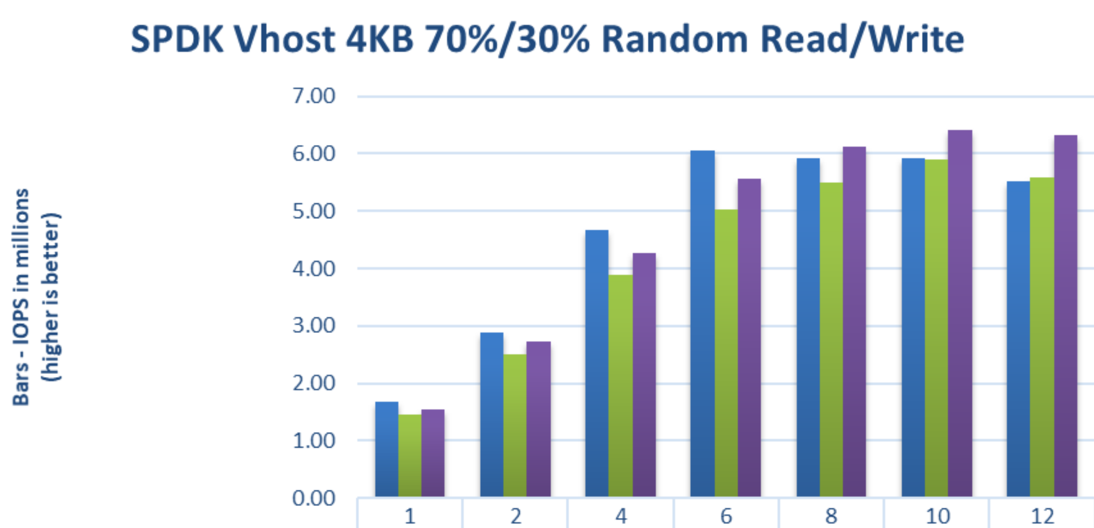


Figure 3: Comparison of performance between various SPDK Vhost stack-bdev combinations for 4KB Random Write QD=32 workload

4KB Random Read-Write Results

Table 7: SPDK Vhost core scaling results, 4KB Random 70% Read 30% Write IOPS, QD=64

# of CPU cores	# of VMs	Stack / Backend	IOPS (millions)
1	6	SCSI / Split NVMe Bdev	1.68
		SCSI / Lvol Bdev	1.47
		BLK / Lvol Bdev	1.56
2	12	SCSI / Split NVMe Bdev	2.89
		SCSI / Lvol Bdev	2.51
		BLK / Lvol Bdev	2.72
4	24	SCSI / Split NVMe Bdev	4.68
		SCSI / Lvol Bdev	3.90
		BLK / Lvol Bdev	4.27
6	36	SCSI / Split NVMe Bdev	6.05
		SCSI / Lvol Bdev	5.03
		BLK / Lvol Bdev	5.57
8	36	SCSI / Split NVMe Bdev	5.92
		SCSI / Lvol Bdev	5.50
		BLK / Lvol Bdev	6.13
10	36	SCSI / Split NVMe Bdev	5.92
		SCSI / Lvol Bdev	5.89
		BLK / Lvol Bdev	6.40
12	36	SCSI / Split NVMe Bdev	5.51
		SCSI / Lvol Bdev	5.59
		BLK / Lvol Bdev	6.31



	1	2	4	6	8	10	12
Scsi + Split NVMe IOPS (million)	1.68	2.89	4.68	6.05	5.92	5.92	5.51
Scsi + Lvol Bdev IOPS (million)	1.47	2.51	3.90	5.03	5.50	5.89	5.59
BLK + Lvol Bdev IOPS (million)	1.56	2.72	4.27	5.57	6.13	6.40	6.31

Figure 4: Comparison of performance between various SPDK Vhost stack-bdev combinations for 4KB Random 70% Read 30% Write QD=64 workload

Logical Volumes performance impact

The SPDK Vhost SCSI tests were run using two bdev backends – Split NVMes and Logical Volumes. Both "Split NVMe Bdevs" and "Logical Volume Bdevs" allow to logically partition NVMe SSDs, the latter being more flexible in configuration. Here we measure the overhead of extra flexibility afforded by Logical Volumes.

Table 8: Logical Volumes performance impact for SPDK Vhost SCSI

Workload	# of CPU cores	# of VMs	Vhost SCSI + Split NVMe IOPS (millions)	Vhost SCSI + Lvol IOPS (millions)	Lvol Impact (%)
4KB 100% Random Read	1	6	1.81	1.55	-14.33%
	2	12	3.05	2.61	-14.65%
	4	24	4.63	4.00	-13.53%
	6	36	6.23	5.04	-19.13%
	8	36	6.37	5.65	-11.33%
	10	36	6.64	6.40	-3.63%
	12	36	5.83	5.95	1.94%
4KB 100% Random Write	1	6	1.57	1.42	-9.40%
	2	12	2.92	2.62	-10.31%
	4	24	4.82	4.28	-11.21%
	6	36	6.29	5.62	-10.72%
	8	36	6.30	5.84	-7.34%
	10	36	5.55	5.46	-1.58%
	12	36	5.63	5.83	3.54%
4KB 70% Random Read 30% Random Write	1	6	1.68	1.47	-12.89%
	2	12	2.89	2.51	-13.12%
	4	24	4.68	3.90	-16.73%
	6	36	6.05	5.03	-16.87%
	8	36	5.92	5.50	-7.08%
	10	36	5.92	5.89	-0.59%
	12	36	5.51	5.59	1.42%

LTO performance impact

Selected test cases were re-run with LTO (Link Time Optimization) enabled for SPDK compilation. This should positively impact overall SPDK performance. The following comparison was done using SPDK Vhost SCSI with Logical Volume bdevs.

Table 9: LTO performance SPDK Vhost SCSI with Logical Volume bdevs

Workload	# of CPU cores	# of VMs	IOPS (millions) LTO Disabled	IOPS (millions) LTO Enabled	LTO Impact (%)
4KB 100% Random Read	1	6	1.55	1.66	6.63%
	2	12	2.61	2.78	6.71%
	4	24	4.00	4.19	4.65%
	6	36	5.04	5.37	6.54%
	8	36	5.65	5.85	3.52%
	10	36	6.40	6.52	1.81%
	12	36	5.95	5.98	0.52%
4KB 100% Random Write	1	6	1.42	1.51	5.84%
	2	12	2.62	2.80	6.85%
	4	24	4.28	4.40	2.62%
	6	36	5.62	5.98	6.50%
	8	36	5.84	6.13	5.08%
	10	36	5.46	5.34	-2.25%
	12	36	5.83	5.78	-0.82%
4KB 70% Random Read 30% Random Write	1	6	1.47	1.58	8.16%
	2	12	2.51	2.66	5.79%
	4	24	3.90	4.12	5.68%
	6	36	5.03	5.33	5.91%
	8	36	5.50	5.66	2.92%
	10	36	5.89	5.82	-1.21%
	12	36	5.59	5.61	0.33%

Packed Ring performance impact

Selected test cases were re-run to show benefits of using Packed Rings as an option when configuring SPDK Vhost BLK controllers. For this, an optional parameter "—packed_ring" must be used when creating a SPDK Vhost BLK controller.

For this test, unlike described in "Test setup" chapter, a different version of Qemu emulator was used. Packed Ring feature requires that at lest Qemu 4.2.0 version is used.

Following results show comparison of running SPDK Vhost-Blk with Packed Ring enabled with fio latency measurements both enabled and disabled. Because other Qemu version was used, base results (Split Ring) were run again to produce a fresh base to compare to.

Table 10: Packed Ring performance impact on SPDK Vhost BLK controllers. Fio gtod_reduce=disabled

Workload	# of CPU cores	# of VMs	IOPS (millions) Split Ring	IOPS (millions) Packed Ring	Avg. Latency (usec) Split Ring	Avg. Latency (usec) Packed Ring	Packed Ring IOPS impact (%)	Packed Ring Avg. Latency impact (%)
4KB 100% Random Read QD=64	1	6	1.55	1.67	247.65	229.68	7.80%	-7.26%
	2	12	2.68	2.95	286.52	259.72	10.30%	-9.35%
	4	24	4.34	4.59	350.90	333.10	5.56%	-5.07%
	6	36	5.52	5.95	418.15	385.33	7.86%	-7.85%
	8	36	6.45	6.75	355.96	341.19	4.63%	-4.15%
	10	36	6.93	7.25	332.12	316.35	4.71%	-4.75%
	12	36	6.53	6.43	354.57	358.46	-1.48%	1.10%
4KB 100% Random Write QD=32	1	6	1.49	1.53	130.43	127.83	2.38%	-1.99%
	2	12	2.80	2.98	137.46	127.98	6.56%	-6.90%
	4	24	4.71	4.88	163.67	157.85	3.62%	-3.56%
	6	36	6.11	6.41	187.61	178.78	4.88%	-4.71%
	8	36	6.70	6.77	171.27	172.10	1.13%	0.48%
	10	36	5.67	5.79	207.18	202.18	2.02%	-2.41%
	12	36	6.42	6.38	179.17	180.10	-0.62%	0.52%
4KB 70% Random Read 30% Random Write QD=64	1	6	1.52	1.62	247.52	239.44	6.44%	-3.26%
	2	12	2.63	2.86	291.58	268.19	8.62%	-8.02%
	4	24	4.32	4.64	353.93	326.42	7.52%	-7.77%
	6	36	5.59	5.94	412.65	386.82	6.43%	-6.26%
	8	36	6.00	6.33	384.75	362.89	5.49%	-5.68%
	10	36	6.16	6.40	376.81	361.54	3.92%	-4.05%
	12	36	5.90	6.08	393.42	380.50	2.98%	-3.29%

Conclusions

1. SPDK Vhost SCSI performance when using Split NVMe bdevs for backend is noticeably better than the same setup with Logical Volume bdevs. It scales near linearly up to 6 CPU cores and achieves peak performance at this point for all workloads. Further increasing the number of cores does not result in performance improvement or it is not significant.

2. SPDK Vhost SCSI using Logical Volume backend devices performance scales near linearly up to 6 CPU cores, reaching around 5.0-5.5 million IOPS. Increasing the number of cores improves performance further, but the gains are not linear. Peak performance is reached at 10 CPU cores for Random Read and Random Read/Write workloads and at 8 CPU cores for Random Write workloads.

3. SPDK Vhost BLK using Logical Volume backend devices performance scales near linearly up to 6 CPU cores, reaching around 5.5-6.0 million IOPS. Increasing the number of cores improves performance further, but the gains are not linear and max out at about 6.5-7.0 million IOPS.

4. Using Logical Volumes as part of testing setup has a noticeable impact on the overall performance. For Vhost tests using 6 or less CPUs (when Vhost is saturated with IO traffic from VMs) performance impact of Logical Volumes is between 10-20%. Further increasing SPDK Vhost CPU cores allow Logical Volumes to perform better and their performance impact is on par with Split NVMe Bdevs (10% difference or less).

5. LTO compilation option increased SPDK Vhost performance by about 5-10% percent in the scaling phase (6 Vhost CPU cores or less). With increasing number of cores LTO benefit vanishes and performance is only up by 1-2% percent or slightly drops. The reason for this behavior is described in point 6.

6. For some workloads there is a slight performance drop when Vhost is run with 10 or 12 CPU cores. The platform has 80 CPU threads available, and when 10 or 12 are used for the Vhost process there is not enough left to accommodate all the VMs. Some of the VMs need to share CPU threads, thus becoming less efficient.

7. Using Packed Ring option instead of default Split Ring mode for SPDK Vhost BLK controllers results in up to 10% performance improvement.

Test Case 2: Rate Limiting IOPS per VM

This test case was geared towards understanding how many VMs can be supported at a pre-defined Quality of Service of IOPS per Vhost device. Both read and write IOPS were rate limited for each Vhost device on each of the VMs and then VM density was compared between SPDK & the Linux Kernel. 10K 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

The results in tables are average of 3 runs.

Table 11: Rate Limiting IOPS per VM test case configuration

Item	Description
Test case	Test rate limiting IOPS/VM to 10000 IOPS

Test configuration	**FIO Version:** fio-3.19
	
	VM Configuration:
	• Common settings are described in the **Virtual Machine Settings** chapter.
	• Total of 24 / 48 / 72 VMs
	• Each VM has a single Vhost device which is one of equal partitions of NVMe drive. Total number of partitions depends on run test case.
	  ○ For 24 VMs: 24xNVMe * 1 partition per NVMe = 24 partitions
	  ○ For 48 VMs: 24xNVMe * 2 partitions per NVMe = 48 partitions
	  ○ For 72 VMs: 24xNVMe * 3 partitions per NVMe = 72 partitions
	• Devices on VMs were throttled to run at a maximum of 10k IOPS (read and write)
	
	SPDK Vhost target configuration:
	• Test were run with both Vhost-scsi and Vhost-blk stacks.
	• The Vhost-scsi stack was run with Split NVMe bdevs and Logical Volume bdevs.
	• The Vhost-blk stack was run with Logical Volume bdevs.
	• Test were run with 4 CPU cores (NUMA optimized).
	
	Kernel Vhost-scsi configuration:
	• Cgroups were used to limit the Vhost process to 4 cores.
	• NUMA optimization were not explored.

FIO configuration run on each VM	[global] 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 Case 2 Results

Table 12: 4KB 100% Random Reads QD=1 rate limiting test results

# of VMs	Stack	Backend bdev	IOPS (k)	Avg Lat. (usec)
24 VMs	SPDK-SCSI	Split NVMe	239.80	98.56
	SPDK-SCSI	Logical Volume	239.81	98.54
	SPDK-BLK	Logical Volume	239.84	98.55
	Kernel-SCSI	Partitioned NVMe	96.33	249.00
48 VMs	SPDK-SCSI	Split NVMe	476.64	98.67
	SPDK-SCSI	Logical Volume	475.43	98.93
	SPDK-BLK	Logical Volume	478.38	98.35
	Kernel-SCSI	Partitioned NVMe	107.16	467.38
72 VMs	SPDK-SCSI	Split NVMe	664.85	106.08
	SPDK-SCSI	Logical Volume	656.68	107.43
	SPDK-BLK	Logical Volume	676.48	104.21
	Kernel-SCSI	Partitioned NVMe	202.21	385.12

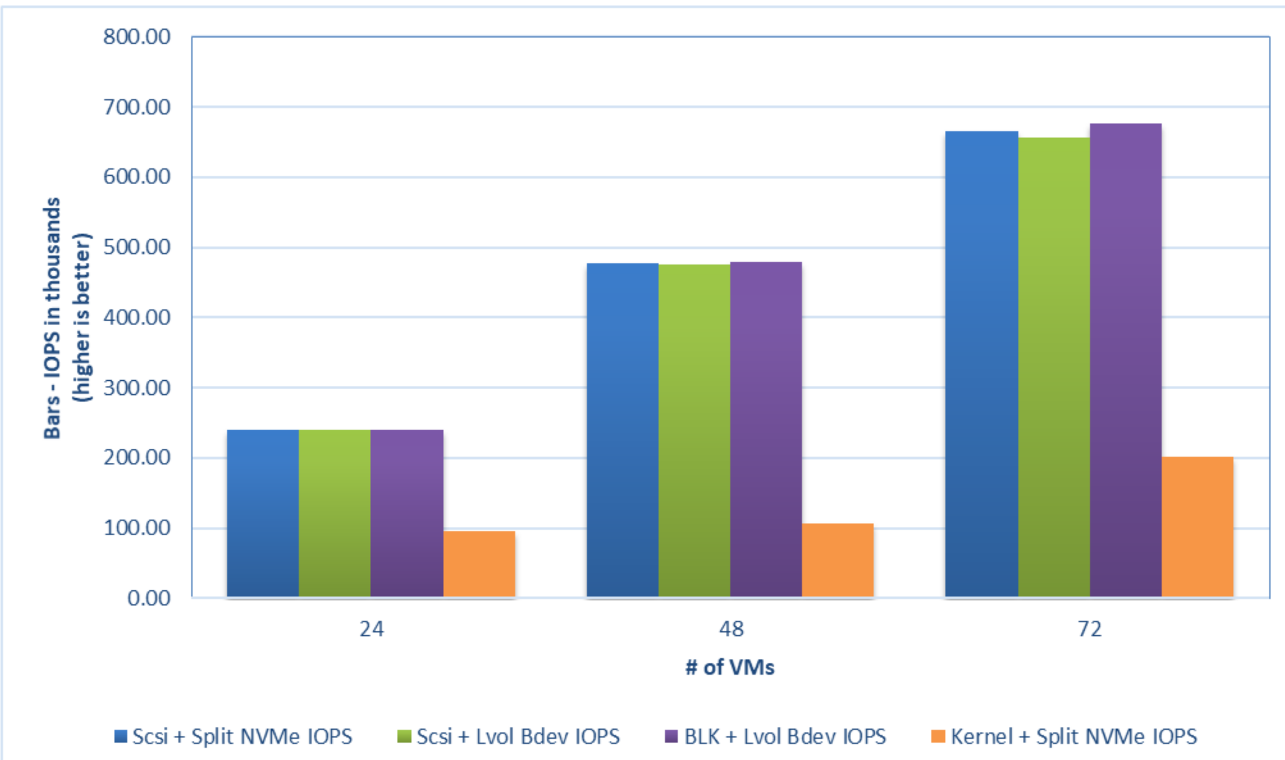


Figure 5: 4KB 100% Random Reads IOPS, QD=1, throttling = 10k IOPS

Table 13: 4KB 100% Random Writes QD=1 rate limiting test results

# of VMs	Stack	Backend bdev	IOPS (k)	Avg Lat. (usec)
24 VMs	SPDK-SCSI	Split NVMe	239.99	96.72
	SPDK-SCSI	Logical Volume	239.99	96.76

	SPDK-BLK	Logical Volume	239.99	96.71
	Kernel-SCSI	Partitioned NVMe	93.55	251.56
48 VMs	SPDK-SCSI	Split NVMe	479.97	96.87
	SPDK-SCSI	Logical Volume	479.96	97.04
	SPDK-BLK	Logical Volume	479.97	96.84
	Kernel-SCSI	Partitioned NVMe	91.17	522.34
72 VMs	SPDK-SCSI	Split NVMe	719.86	97.15
	SPDK-SCSI	Logical Volume	719.85	97.15
	SPDK-BLK	Logical Volume	719.88	97.04
	Kernel-SCSI	Partitioned NVMe	228.47	319.11

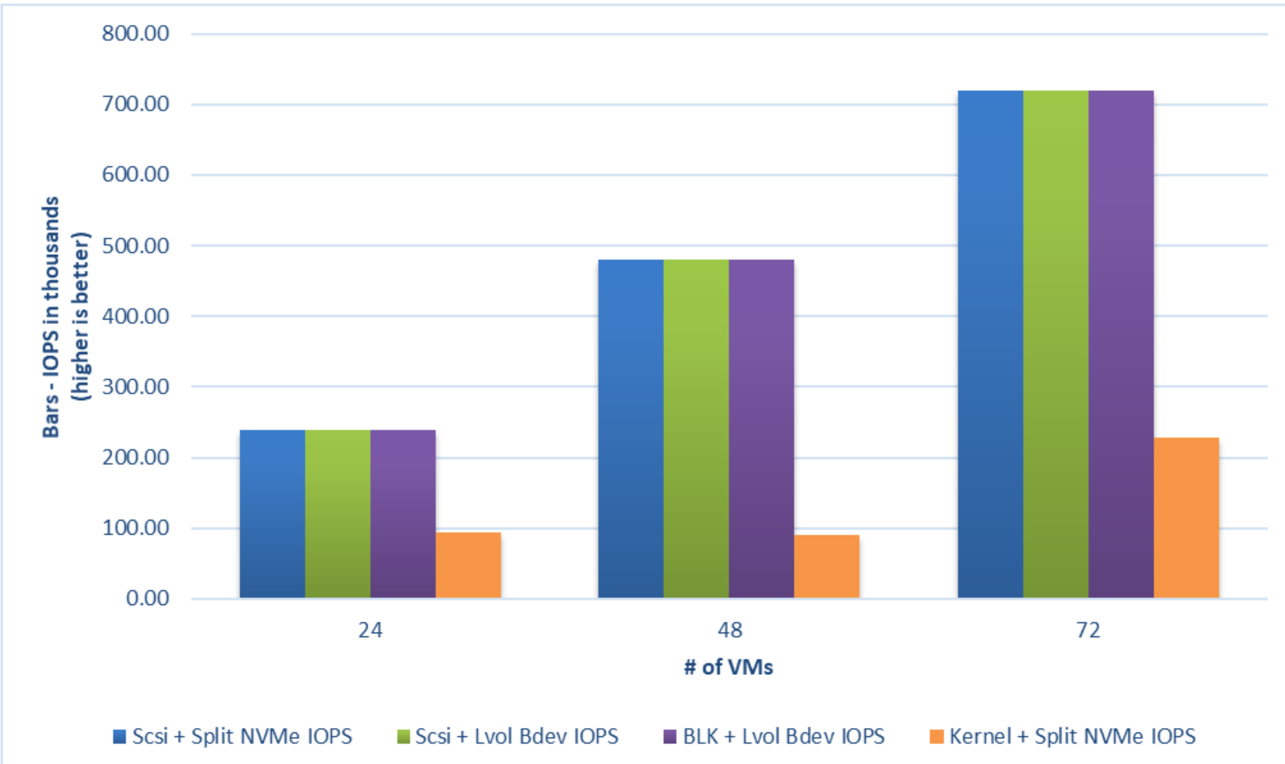


Figure 6: 4KB 100% Random Writes IOPS, QD=1, throttling = 10k IOPS

Conclusions

1. VMs using SPDK Vhost exposed devices were able to achieve the expected IOPS result.

2. SPDK Vhost was able to serve IO at the desired level for an increasing number of VMs.

3. Average latencies were up to 4.7x times better for Random Read and up to 5.4x times better for Random Write workloads in SPDK Vhost when compared to Kernel Vhost.

Note: The Kernel-Vhost process was not NUMA-optimized for this scenario.

Test Case 3: Performance per NVMe drive

This test case was performed in order to understand performance and efficiency of the Vhost scsi and blk process using SPDK vs. Linux Kernel with a single NVMe drive on 2 VMs. Each VM had a single Vhost 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 the following workloads:

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

The results in tables are average of 3 runs.

Table 14: Performance per NVMe drive test case configuration

Item	Description
Test case	Test SPDK Vhost target I/O core scaling performance
Test configuration	**FIO Version:** fio-3.19 **VM Configuration**: • Common settings are described in the **Virtual Machine Settings** chapter. • 2 VMs were tested • Each VM had a single Vhost device which was one of two equal partitions of a single NVMe drive. **SPDK Vhost target configuration:** • The SPDK Vhost process was run on a single, physical CPU core. • The Vhost-scsi stack was run with Split NVMe bdevs and Logical Volume bdevs. • The Vhost-blk stack was run with Logical Volume bdevs. **Kernel Vhost target configuration**: • The Vhost process was run on a single, physical CPU core using cgroups.
FIO configuration	[global] 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=240s ramp_time=60s bs=4k iodepth=1 / 8 / 32 / 64 numjobs=1

Test Case 3 results

SPDK Vhost-Scsi

Table 15:Performance per NVMe drive IOPS and latency results, SCSI stack

Access pattern	Backend	QD	Throughput (IOPS)	Avg. latency (usec)
4k 100% Random Reads	Split NVMe	1	23970.402	81.904
4k 100% Random Reads	Split NVMe	8	164091.882	96.24
4k 100% Random Reads	Split NVMe	32	388379.528	163.761
4k 100% Random Reads	Split NVMe	64	389791.156	328.22
4k 100% Random Reads	Lvol	1	23892.458	82.133
4k 100% Random Reads	Lvol	8	163398.067	96.57
4k 100% Random Reads	Lvol	32	385654.306	164.136
4k 100% Random Reads	Lvol	64	390579.733	328.743
4k 100% Random Writes	Split NVMe	1	108198.284	17.1
4k 100% Random Writes	Split NVMe	8	370134.248	42.021
4k 100% Random Writes	Split NVMe	32	359414.298	179.376
4k 100% Random Writes	Split NVMe	64	376701.85	338.69
4k 100% Random Writes	Lvol	1	110388.541	16.896
4k 100% Random Writes	Lvol	8	370767.462	41.8
4k 100% Random Writes	Lvol	32	363231.867	178.085
4k 100% Random Writes	Lvol	64	381265.472	333.857
4k 70%/30% Random Read Writes	Split NVMe	1	28446.278	68.753
4k 70%/30% Random Read Writes	Split NVMe	8	161757.844	97.969
4k 70%/30% Random Read Writes	Split NVMe	32	320863.718	201.306
4k 70%/30% Random Read Writes	Split NVMe	64	349598.398	368.073
4k 70%/30% Random Read Writes	Lvol	1	29891.321	66.195
4k 70%/30% Random Read Writes	Lvol	8	153636.919	103.423
4k 70%/30% Random Read Writes	Lvol	32	295702.033	216.124
4k 70%/30% Random Read Writes	Lvol	64	371361.727	342.374

SPDK Vhost-Blk

Table 16: Performance per NVMe drive IOPS and latency results, BLK stack

Access pattern	Backend	QD	Throughput (IOPS)	Avg. latency (usec)
4k 100% Random Reads	Lvol	1	23860.305	82.505
4k 100% Random Reads	Lvol	8	165829.605	95.252
4k 100% Random Reads	Lvol	32	415323.215	153.366
4k 100% Random Reads	Lvol	64	426704.878	296.901
4k 100% Random Writes	Lvol	1	99011.247	19.147
4k 100% Random Writes	Lvol	8	402109.274	38.555
4k 100% Random Writes	Lvol	32	391430.081	165.746
4k 100% Random Writes	Lvol	64	413309.782	307.919
4k 70%/30% Random Read Writes	Lvol	1	30308.263	64.762
4k 70%/30% Random Read Writes	Lvol	8	171004.622	91.973
4k 70%/30% Random Read Writes	Lvol	32	314395.725	205.601
4k 70%/30% Random Read Writes	Lvol	64	384327.873	332.607

Kernel Vhost-Scsi

Table 17: Performance per NVMe drive IOPS and latency results, Kernel Vhost-Scsi

Access pattern	Backend	QD	Throughput (IOPS)	Avg. latency (usec)
4k 100% Random Reads	NVMe	1	16277.853	121.691
4k 100% Random Reads	NVMe	8	88450.207	179.927
4k 100% Random Reads	NVMe	32	218528.515	292.133
4k 100% Random Reads	NVMe	64	272401.541	469.548
4k 100% Random Writes	NVMe	1	36903.155	52.295
4k 100% Random Writes	NVMe	8	97855.564	162.271
4k 100% Random Writes	NVMe	32	163162.164	394.869
4k 100% Random Writes	NVMe	64	208487.017	610.345
4k 70%/30% Random Read Writes	NVMe	1	19140.798	103.409
4k 70%/30% Random Read Writes	NVMe	8	89371.149	177.87
4k 70%/30% Random Read Writes	NVMe	32	210807.103	304.636
4k 70%/30% Random Read Writes	NVMe	64	265885.626	481.441

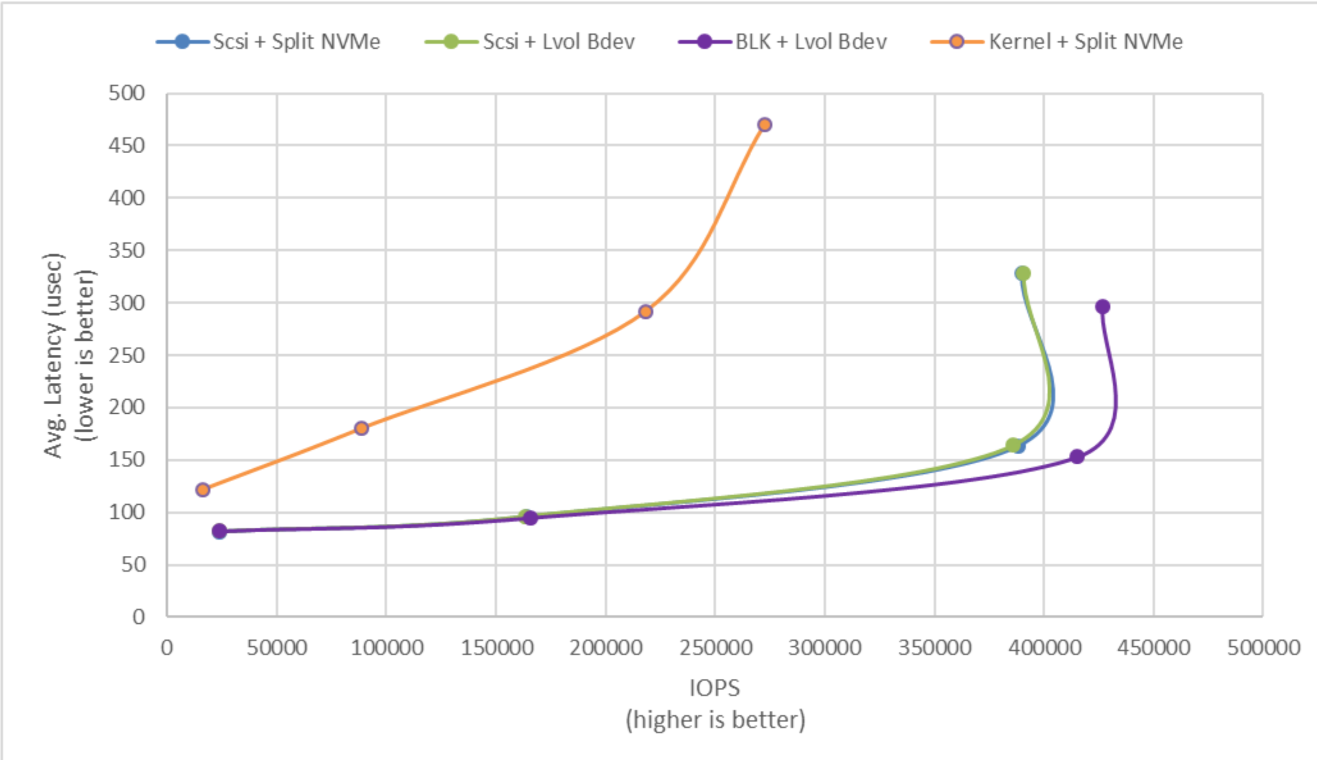


Figure 7: 4KB 100% Random Reads IOPS and latency

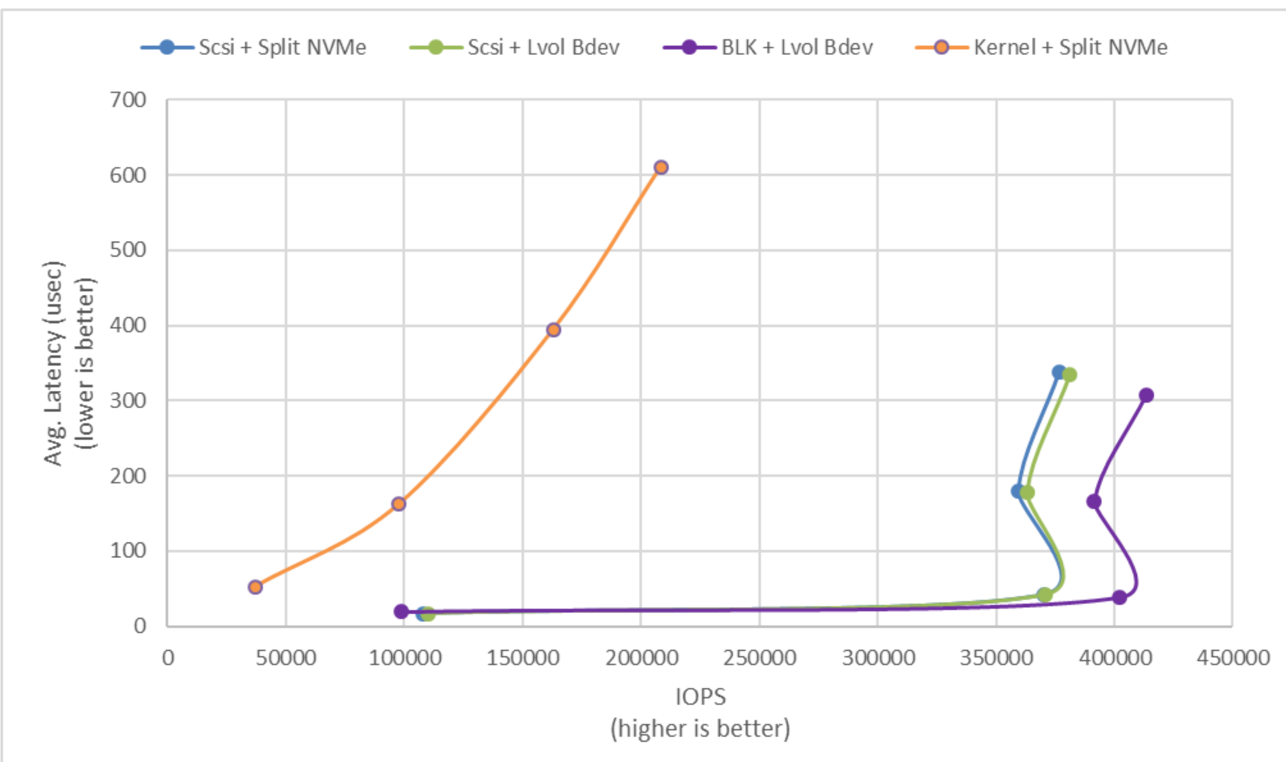


Figure 8: 4KB 100% Random Writes IOPS and latency

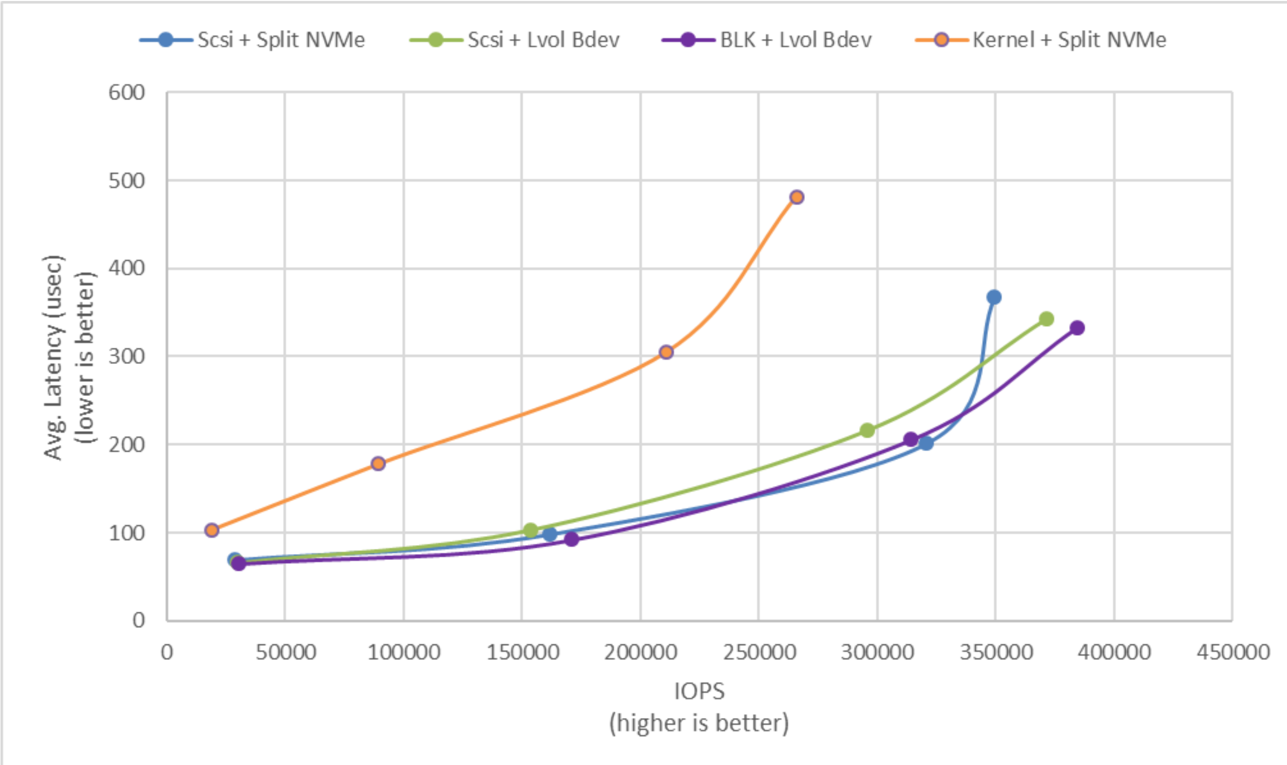


Figure 9: 4KB 70%/30% Random Read/Write IOPS and latency

Conclusions

1. SPDK Vhost-scsi with NVMe Split bdevs has lower latency and higher throughput than Kernel Vhost-scsi in all workload / queue depth combinations.

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 an underlying system when comparing kernel vs. SPDK Vhost-scsi target implementations.

In addition, performance impacts of using SPDK Logical Volume Bdevs and the SPDK Vhost-blk stack were presented.

This report provided information regarding methodologies and practices while benchmarking Vhost-scsi and Vhost-blk using both SPDK and the Linux Kernel. It should be noted that the performance data showcased in this report is based on specific hardware and software configurations and that performance results may vary depending on different hardware and software configurations.

List of Tables

List of Figures

Notices & Disclaimers

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

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

Your costs and results may vary.

No product or component can be absolutely secure.

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

§