SPDK Vhost Performance Report Release 24.01

Testing Date: February 2024

Performed by:

Jaroslaw Chachulski (jaroslawx.chachulski@intel.com)

Acknowledgments:

Karol Latecki (karol.latecki@intel.com)

Krzysztof Karaś (krzysztof.karas@intel.com)

Michał Berger (michal.berger@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><b>Server Platform</b></td><td>Ultra SuperServer SYS-220U-TNR
 </td></tr>
<tr><td><b>Motherboard</b></td><td>Server board <b>X12DPU-6</b></td></tr>
<tr><td><b>CPU</b></td><td>2 CPU sockets, Intel(R) Xeon(R) Gold 6348 CPU @ 2.60GHz
Number of cores 28 per socket, number of threads 56 per socket
Both sockets populated.
Microcode: 0xd000389</td></tr>
<tr><td><b>Memory</b></td><td>16 x 32GB SK Hynix DDR4 HMA84GR7DJR4N-XN; Total 512 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>
<tr><td>CPU1_DIMM_F1</td><td>CPU2_DIMM_F1</td></tr>
<tr><td>CPU1_DIMM_G1</td><td>CPU2_DIMM_G1</td></tr>
<tr><td>CPU1_DIMM_H1</td><td>CPU2_DIMM_H1</td></tr>
</table></td></tr>
<tr><td><b>Operating System</b></td><td>Fedora 37</td></tr>
<tr><td><b>BIOS</b></td><td>1.4b</td></tr>
<tr><td><b>Linux kernel version</b></td><td>6.1.6-200.fc37.x86_64
Spectre-meltdown mitigations enabled</td></tr>
<tr><td><b>SPDK version</b></td><td>SPDK 24.01</td></tr>
<tr><td><b>Qemu version</b></td><td>7.0.0 (qemu-7.0.0-12.fc37)</td></tr>
<tr><td><b>Storage</b></td><td><b>OS:</b> 1x 250GB Crucial CT250MX500SSD1
<b>Storage</b>:
22x Kioxia® KCM61VUL3T20 3.2TBs (FW: 0105) (10 on CPU NUMA Node 0, 12 on CPU NUMA Node 1)</td></tr>
</table>

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

Table 3: Test System NVMe storage setup

Item	Description	
PCIe Riser cards	**"Ultra" Riser Card:** AOC-2UR68G4-i2XT • PCIe Slot 1 – x16, CPU2 • PCIe Slot 2 – x8, CPU2 • PCIe Slot 3 – x8, CPU2 **Right-facing riser card:** RSC-WR-6 • PCIe Slot 4 – x16, CPU1 **Left-facing riser card:** RSC-W2-66G4 • PCIe Slot 5 – x16, CPU2 • PCIe Slot 7 – x16, CPU1 More information can be found in SYS-220U-TNR manual document.	
PCIe Retimer cards	3 x AOC-SLG4-4E4T Installed in: ○ PCIe Retimer 1: RSC-WR-6, PCIe Slot 4 (using CPU1 PCIe Lanes) ○ PCIe Retimer 2: AOC-2UR68G4-i2XT, PCIe Slot 1 (using CPU2 PCIe Lanes) ○ PCIe Retimer 3: RSC-W2-66G4, PCIe Slot 5 (using CPU2 PCIe Lanes)	
NVMe Drives distribution across the system	Nvme0 – 5	Motherboard ports (CPU1 PCIe Lanes)
	Nvme6 – 9	Motherboard ports (CPU2 PCIe Lanes)
	Nvme9 – 13	PCIe Retimer 1 (CPU1 PCIe Lanes)
	Nvme14 - 17	PCIe Retimer 2 (CPU2 PCIe Lanes)
	Nvme18 - 21	PCIe Retimer 3 (CPU2 PCIe Lanes)

Virtual Machine Settings

Table 4: Guest VM configuration

Item	Description
CPU	2 vCPU, pass through from physical host server. Explicit core usage enforced using "taskset –a –c" command. QEMU arguments for starting the VM: -cpu host -smp 1

Memory	2 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. QEMU arguments: -m 2048 -object memory-backend-file,id=mem,size=2048M,mem-path=/dev/hugepages,share=on,prealloc=yes,host-nodes=0,policy=bind
Operating System	Fedora 35
Linux kernel version	5.15.7-200.fc35.x86_64
Additional boot options in /etc/default/grub	Multi queue enabled: scsi_mod.use_blk_mq=1

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 community has leveraged existing SPDK SCSI layer, DPDK Vhost library, QEMU Vhost-Scsi and Vhost-Blk functionality to create the high performance SPDK userspace Vhost target.

SPDK Vhost target architecture

QEMU sets up the Vhost target via UNIX domain socket. 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. The Vhost target transfers data to/from the guest VM via shared memory. 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. It should be noted that there is no QEMU intervention during the I/O submission process.

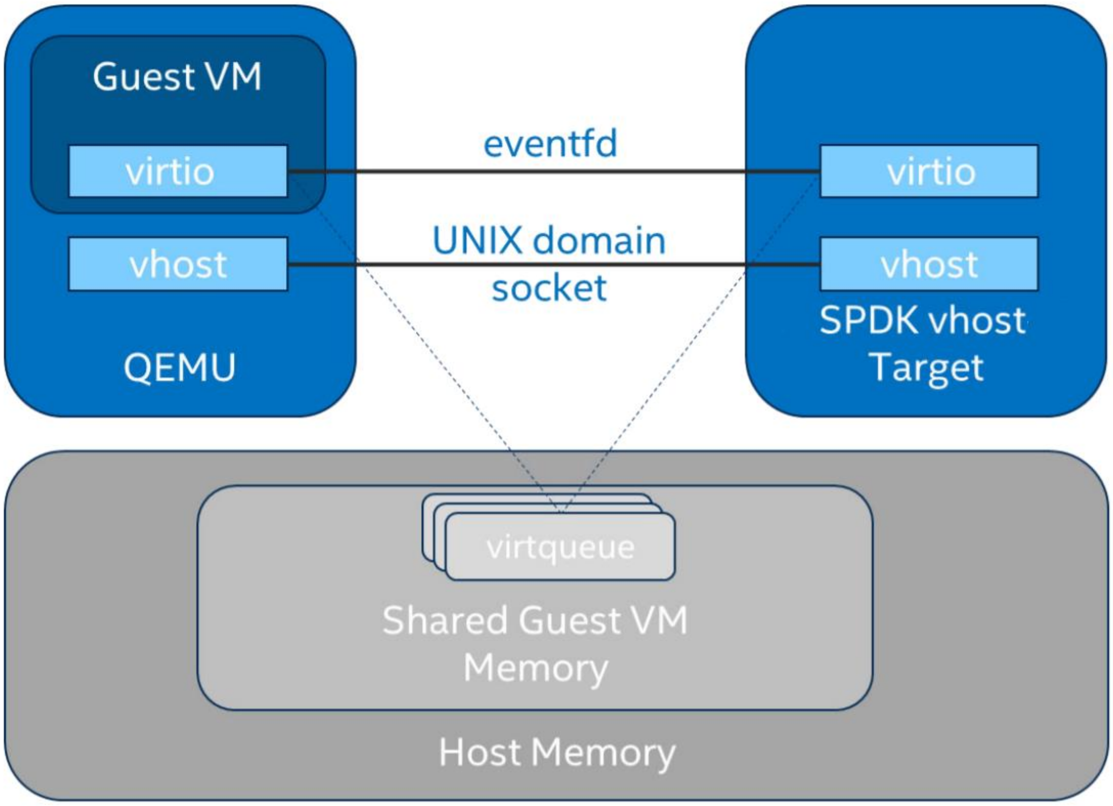


Figure 1: SPDK Vhost-Scsi architecture

This report shows the performance comparisons between the traditional interrupt-driven Linux Kernel Vhost-Scsi and the accelerated polled-mode SPDK Vhost-Scsi under 3 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 to understand aggregate VM performance with SPDK Vhost I/O core scaling. We ran up to 36 virtual machines, each running following FIO workloads:

- 4KiB 100% Random Read
- 4KiB 100% Random Write
- 4KiB 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 22 and measured the throughput (in IOPS) and latency. The number of VMs between test runs was not constant and was increased by 2 for each Vhost CPU added, up to a maximum of 44 VMs. VM number was not increased beyond 44 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.

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 5: SPDK Vhost Core Scaling test configuration

Item	Description
Test case	Test SPDK Vhost target I/O core scaling performance
Test configuration	**FIO Version**: fio-3.28 **VM Configuration**: • Common settings are described in the Virtual Machine Settings chapter. • Number of VMs: variable (2 VMs per 1 Vhost CPU core, up to 44 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 performed with 1, 2, 6, 10, 14, 18 and 22 Vhost cores for each stack-bdev combination. **Kernel Vhost target configuration:** • N/A

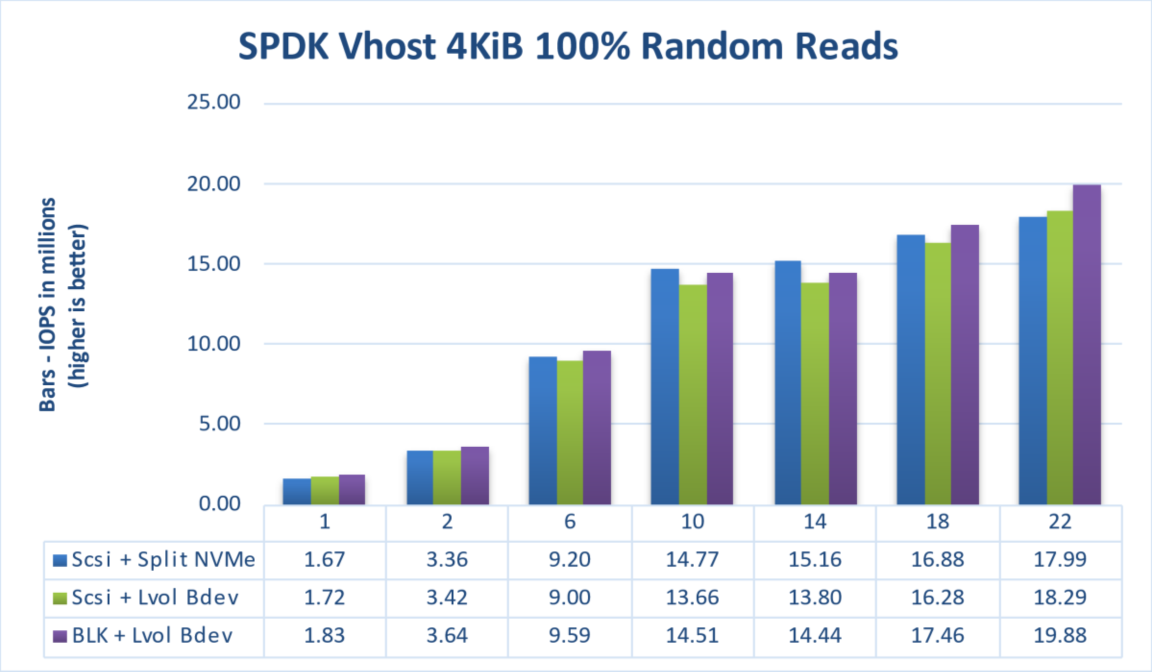
FIO configuration	[global]
	ioengine=libaio
	direct=1
	thread=1

	norandommap=1
	time_based=1
	gtod_reduce=0
	ramp_time=60s
	runtime=240s
	numjobs=2
	bs=4k
	rw=randrw
	rwmixread=100 (100% reads), 70 (70% reads, 30% writes), 0 (100% writes)
	iodepth= {1, 64, 128, 192, 384}

4KiB Random Read Results

Table 6: SPDK Vhost core scaling results, 4KiB 100% Random Reads IOPS, QD=64

# of CPU cores	# of VMs	Stack / Backend	IOPS (millions)
1	2	SCSI / Split NVMe Bdev	1.67
		SCSI / Lvol Bdev	1.72
		BLK / Lvol Bdev	1.83
2	4	SCSI / Split NVMe Bdev	3.36
		SCSI / Lvol Bdev	3.42
		BLK / Lvol Bdev	3.64
6	12	SCSI / Split NVMe Bdev	9.20
		SCSI / Lvol Bdev	9.00
		BLK / Lvol Bdev	9.59
10	20	SCSI / Split NVMe Bdev	14.77
		SCSI / Lvol Bdev	13.66
		BLK / Lvol Bdev	14.51
14	28	SCSI / Split NVMe Bdev	15.16
		SCSI / Lvol Bdev	13.80
		BLK / Lvol Bdev	14.44
18	36	SCSI / Split NVMe Bdev	16.88
		SCSI / Lvol Bdev	16.28
		BLK / Lvol Bdev	17.46
22	44	SCSI / Split NVMe Bdev	17.99
		SCSI / Lvol Bdev	18.29
		BLK / Lvol Bdev	19.88



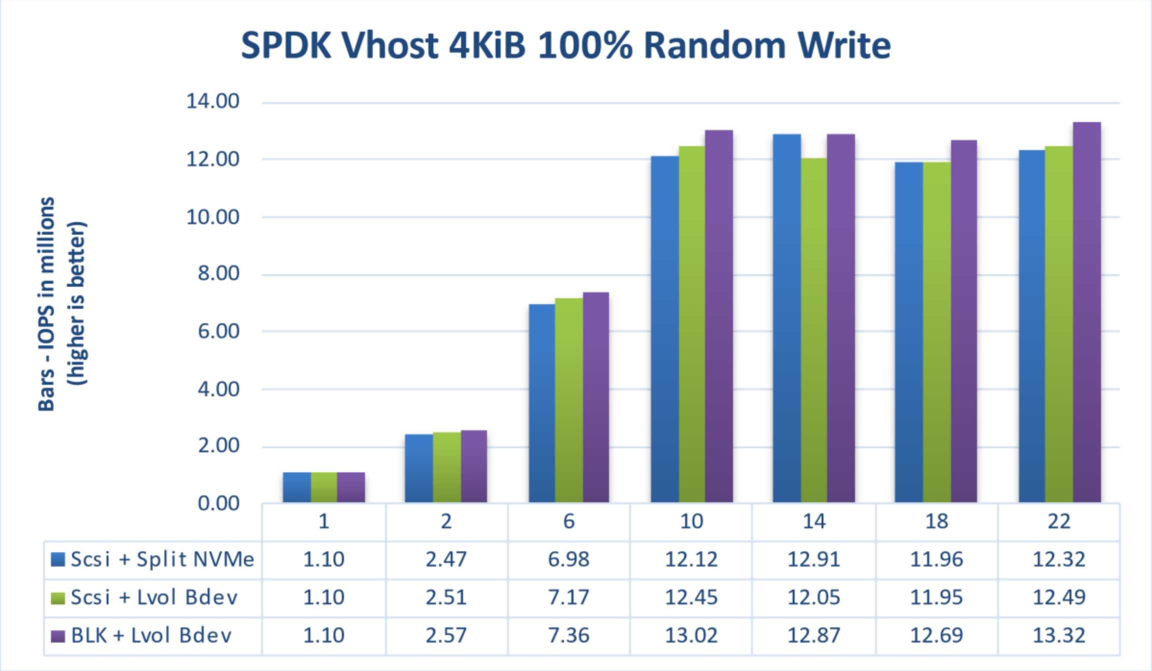
	1	2	6	10	14	18	22
■ Scsi + Split NVMe	1.67	3.36	9.20	14.77	15.16	16.88	17.99
■ Scsi + Lvol Bdev	1.72	3.42	9.00	13.66	13.80	16.28	18.29
■ BLK + Lvol Bdev	1.83	3.64	9.59	14.51	14.44	17.46	19.88

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

4KiB Random Write Results

Table 7: SPDK Vhost core scaling results, 4KiB 100% Random Write IOPS, QD=64

# of CPU cores	# of VMs	Stack / Backend	IOPS (millions)
1	2	SCSI / Split NVMe Bdev	1.10
		SCSI / Lvol Bdev	1.10
		BLK / Lvol Bdev	1.10
2	4	SCSI / Split NVMe Bdev	2.47
		SCSI / Lvol Bdev	2.51
		BLK / Lvol Bdev	2.57
6	12	SCSI / Split NVMe Bdev	6.98
		SCSI / Lvol Bdev	7.17
		BLK / Lvol Bdev	7.36
10	20	SCSI / Split NVMe Bdev	12.12
		SCSI / Lvol Bdev	12.45
		BLK / Lvol Bdev	13.02
14	28	SCSI / Split NVMe Bdev	12.91
		SCSI / Lvol Bdev	12.05
		BLK / Lvol Bdev	12.87
18	36	SCSI / Split NVMe Bdev	11.96
		SCSI / Lvol Bdev	11.95
		BLK / Lvol Bdev	12.69
22	44	SCSI / Split NVMe Bdev	12.32
		SCSI / Lvol Bdev	12.49
		BLK / Lvol Bdev	13.32



	1	2	6	10	14	18	22
🟦 Scsi + Split NVMe	1.10	2.47	6.98	12.12	12.91	11.96	12.32
🟩 Scsi + Lvol Bdev	1.10	2.51	7.17	12.45	12.05	11.95	12.49
🟪 BLK + Lvol Bdev	1.10	2.57	7.36	13.02	12.87	12.69	13.32

Figure 3: Comparison of performance between various SPDK Vhost stack-bdev combinations for 4KiB Random Write QD=64 workload

4KiB Random Read-Write Results

Table 8: SPDK Vhost core scaling results, 4KiB Random 70% Read 30% Write IOPS, QD=64

# of CPU cores	# of VMs	Stack / Backend	IOPS (millions)
1	2	SCSI / Split NVMe Bdev	1.35
		SCSI / Lvol Bdev	1.39
		BLK / Lvol Bdev	1.49
2	4	SCSI / Split NVMe Bdev	2.75
		SCSI / Lvol Bdev	2.76
		BLK / Lvol Bdev	2.99
6	12	SCSI / Split NVMe Bdev	7.92
		SCSI / Lvol Bdev	7.81
		BLK / Lvol Bdev	8.39
10	20	SCSI / Split NVMe Bdev	12.98
		SCSI / Lvol Bdev	12.36
		BLK / Lvol Bdev	13.12
14	28	SCSI / Split NVMe Bdev	13.46
		SCSI / Lvol Bdev	12.50
		BLK / Lvol Bdev	13.20
18	36	SCSI / Split NVMe Bdev	14.59
		SCSI / Lvol Bdev	14.15
		BLK / Lvol Bdev	15.17
22	44	SCSI / Split NVMe Bdev	15.64
		SCSI / Lvol Bdev	15.74
		BLK / Lvol Bdev	16.99

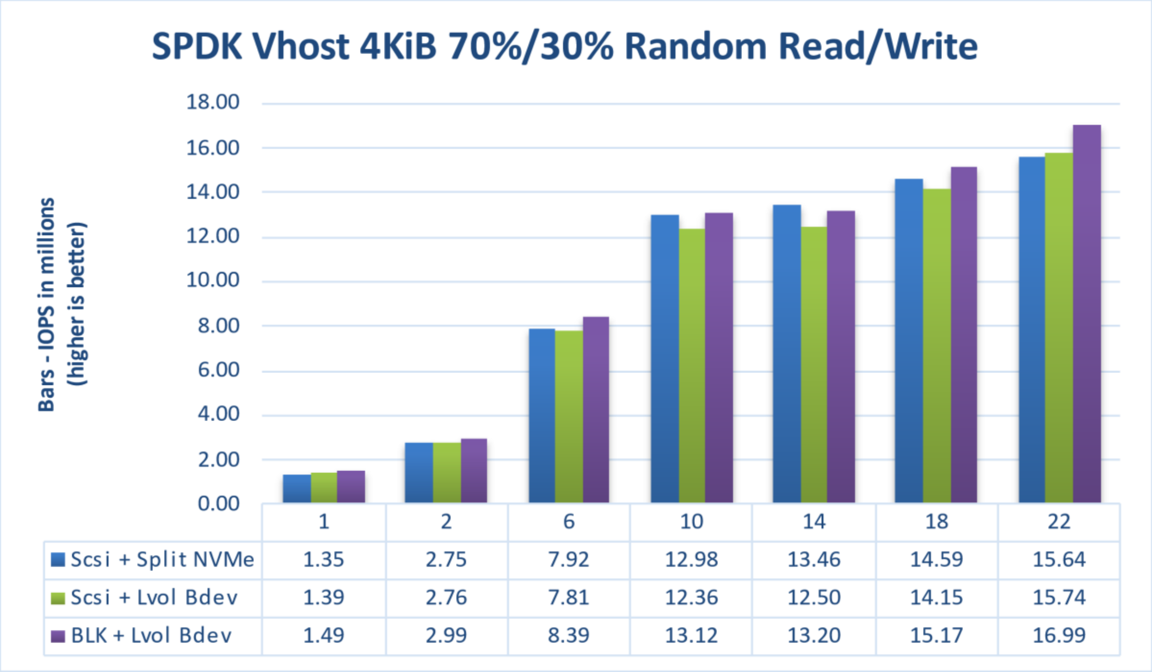


Figure 4: Comparison of performance between various SPDK Vhost stack-bdev combinations for 4KiB Random 70% Read 30% Write QD=64 workload

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 9: 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 (%)
4KiB 100% Random Read	1	2	1.67	1.72	2.72%
	2	4	3.36	3.42	1.87%
	6	12	9.20	9.00	-2.15%
	10	20	14.77	13.66	-7.48%
	14	28	15.16	13.80	-9.00%
	18	36	16.88	16.28	-3.53%
	22	44	17.99	18.29	1.64%
4KiB 100% Random Write	1	2	1.10	1.10	0.14%
	2	4	2.47	2.51	1.71%
	6	12	6.98	7.17	2.67%
	10	20	12.12	12.45	2.73%
	14	28	12.91	12.05	-6.68%
	18	36	11.96	11.95	-0.03%
	22	44	12.32	12.49	1.35%
4KiB 70% Random Read 30% Random Write	1	2	1.35	1.39	2.86%
	2	4	2.75	2.76	0.29%
	6	12	7.92	7.81	-1.37%
	10	20	12.98	12.36	-4.77%
	14	28	13.46	12.50	-7.14%
	18	36	14.59	14.15	-3.08%
	22	44	15.64	15.74	0.63%

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. Packed Ring feature requires QEMU 4.2.0 or later.

Table 10: Packed Ring performance impact on SPDK Vhost BLK controllers

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 (%)
4KiB 100% Random Read QD=64	1	2	3.81	3.87	134.10	131.92	1.56%	-1.62%
	2	4	10.44	10.62	146.56	144.55	1.74%	-1.37%
	6	12	15.94	16.49	160.16	155.03	3.42%	-3.21%
	10	20	16.25	16.75	220.04	213.75	3.11%	-2.86%
	14	28	19.17	19.64	240.06	234.81	2.45%	-2.19%
	18	36	21.82	22.05	257.20	254.62	1.03%	-1.00%
	22	44	1.51	1.59	172.47	179.99	5.27%	-5.78%
4KiB 100% Random Write QD=64	1	2	3.17	3.26	161.75	168.31	2.93%	-3.18%
	2	4	9.14	9.20	166.14	172.09	0.65%	0.06%
	6	12	14.74	15.24	174.10	176.12	3.40%	-3.48%
	10	20	14.58	15.06	245.18	244.98	3.29%	-3.77%
	14	28	14.18	14.62	328.59	285.21	3.10%	-4.41%
	18	36	14.06	14.51	399.70	321.30	3.23%	-2.81%
	22	44	1.91	1.45	173.01	179.99	-1.71%	4.04%
4KiB 70% Random Read 30% Random Write QD=64	1	2	3.81	3.10	168.55	168.31	3.42%	-0.14%
	2	4	10.44	8.95	174.15	172.09	1.93%	-1.18%
	6	12	15.94	14.56	180.47	176.12	3.02%	-2.41%
	10	20	16.25	14.65	246.11	244.98	0.89%	-0.46%
	14	28	19.17	16.18	288.78	285.21	1.75%	-1.24%
	18	36	21.82	17.47	322.67	321.30	0.66%	-0.43%
	22	44	3.81	3.87	134.10	131.92	1.56%	-1.62%

Conclusions

1. For SPDK Vhost-Scsi performance with split NVMe bdevs, we measured 1.67 million IOPS on one Vhost core for the 4KiB 100% Random Read workload. The single Vhost core IOPS for 4 KiB Random Write and 4KiB Random 70/30 Read/Write were 1.09 million and 1.35 million IOPS respectively. For all workloads, the IOPS scaled near linearly with addition of I/O processing cores up to 10 CPU cores. Peak performance was achieved at:

   - 22 CPU cores with 17.99 million IOPS for Random Read workload
   - 14 CPU cores with 12.91 million IOPS for Random Write workload
   - 22 CPU cores with 15.64 million IOPS for Random Read/Write workload

2. For SPDK Vhost-Scsi performance with Logical Volume backend devices, we measured 1.72 million IOPS on one Vhost core for the 4KiB 100% Random Read workload. The single Vhost core IOPS for 4 KiB Random Write and 4KiB Random 70/30 Read/Write were 1.1 million and 1.39 million IOPS respectively. For all workloads, the IOPS scaled near linearly with addition of I/O processing cores up to 10 CPU cores.

   Peak performance was achieved at:

   - 22 CPU cores with 18.29 million IOPS for Random Read workload
   - 22 CPU cores with 12.49 million IOPS for Random Write workload
   - 22 CPU cores with 15.74 million IOPS for Random Read/Write workload

3. For SPDK Vhost-Blk with Logical Volume backend devices, we measured 1.83 million IOPS on one Vhost core for the 4KiB 100% Random Read workload. The single Vhost core IOPS for 4 KiB Random Write and 4KiB Random 70/30 Read/Write were 1.1 million and 1.49 million IOPS respectively. For all workloads, the IOPS scaled near linearly with addition of I/O processing cores up to 10 CPU cores. Peak performance was achieved at:

   - 22 CPU cores with 19.88 million IOPS for Random Read workload
   - 22 CPU cores with 13.32 million IOPS for Random Write workload
   - 22 CPU cores with 16.99 million IOPS for Random Read/Write workload

4. Using Logical Volumes has an impact of up to about 7-9% lower IOPS than when using Split NVMe block devices.

5. Using Packed Ring option instead of default Split Ring mode for SPDK Vhost BLK controllers results in minor 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. 25k IOPS per VM were chosen as the rate limiter using Linux cgroups2.

Each individual VM was running FIO with the following workloads:

- 4KiB 100% Random Read
- 4KiB 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 25000 IOPS
Test configuration	**FIO Version:** fio-3.28 **VM Configuration**: • Common settings are described in the **Virtual Machine Settings** chapter. • Number of VMs: 24, 48, and 72 • 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 25k IOPS (read or 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 the Vhost target using 6 CPU cores (NUMA optimized). **Kernel Vhost-Scsi configuration:** • NUMA optimizations were not explored. • The Vhost kernel threads (single thread per vhost target) each is limited to using up to 6 CPU cores via cgroups (NUMA optimized).

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=240s
	ramp_time=60s
	bs=4k
	iodepth=1
	numjobs=1

Test Case 2 Results

Table 12: 4KiB 100% Random Reads QD=1 rate limiting test results, 6 Vhost CPU cores

# of VMs	Stack	Backend bdev	IOPS (k)	Avg Lat. (usec)
24 VMs	SPDK-SCSI	Split NVMe	292.36	81.79
	SPDK-SCSI	Logical Volume	291.04	82.26
	SPDK-BLK	Logical Volume	293.08	81.67
	Kernel-SCSI	Partitioned NVMe	249.53	95.90
48 VMs	SPDK-SCSI	Split NVMe	575.09	83.21
	SPDK-SCSI	Logical Volume	573.26	83.46
	SPDK-BLK	Logical Volume	577.16	82.90
	Kernel-SCSI	Partitioned NVMe	449.33	106.53
72 VMs	SPDK-SCSI	Split NVMe	798.84	89.72
	SPDK-SCSI	Logical Volume	799.04	89.73
	SPDK-BLK	Logical Volume	806.34	88.91
	Kernel-SCSI	Partitioned NVMe	376.13	191.18
96 VMs	SPDK-SCSI	Split NVMe	937.33	101.90
	SPDK-SCSI	Logical Volume	952.94	100.18
	SPDK-BLK	Logical Volume	965.68	0.00
	Kernel-SCSI	Partitioned NVMe	378.90	253.56
106 VMs	SPDK-SCSI	Split NVMe	993.18	106.15
	SPDK-SCSI	Logical Volume	1001.47	105.26
	SPDK-BLK	Logical Volume	1038.87	101.54
	Kernel-SCSI	Partitioned NVMe	371.91	285.28

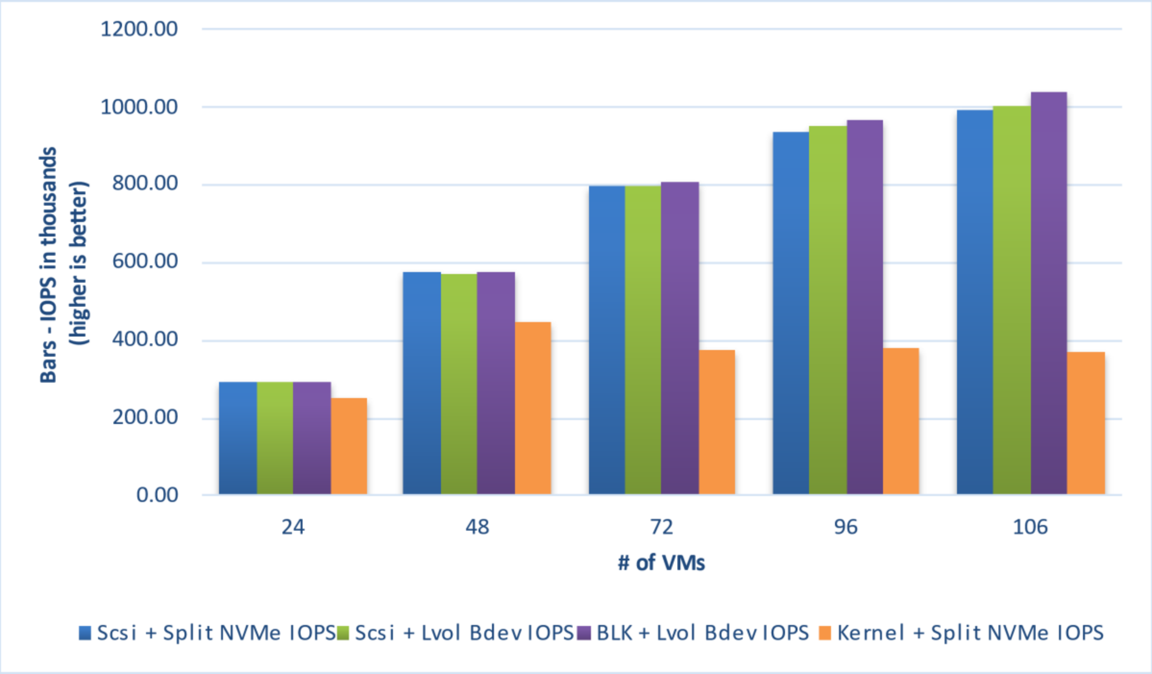


Figure 5: 4KiB 100% Random Reads IOPS, QD=1, throttling = 25k IOPS, 6 Vhost CPU cores

Table 13: 4KiB 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	599.97	39.74
	SPDK-SCSI	Logical Volume	599.95	39.74
	SPDK-BLK	Logical Volume	599.95	39.75
	Kernel-SCSI	Partitioned NVMe	555.01	42.96
48 VMs	SPDK-SCSI	Split NVMe	1199.96	39.73
	SPDK-SCSI	Logical Volume	1199.97	39.73
	SPDK-BLK	Logical Volume	1199.94	39.74
	Kernel-SCSI	Partitioned NVMe	512.40	94.72
72 VMs	SPDK-SCSI	Split NVMe	1799.88	39.59
	SPDK-SCSI	Logical Volume	1799.72	39.59
	SPDK-BLK	Logical Volume	1799.80	39.61
	Kernel-SCSI	Partitioned NVMe	422.84	170.25
96 VMs	SPDK-SCSI	Split NVMe	1596.21	59.60
	SPDK-SCSI	Logical Volume	1516.42	62.73
	SPDK-BLK	Logical Volume	1524.73	62.41
	Kernel-SCSI	Partitioned NVMe	317.31	303.67
106 VMs	SPDK-SCSI	Split NVMe	1500.71	70.04
	SPDK-SCSI	Logical Volume	1475.71	71.32
	SPDK-BLK	Logical Volume	1447.30	72.70
	Kernel-SCSI	Partitioned NVMe	300.93	351.63

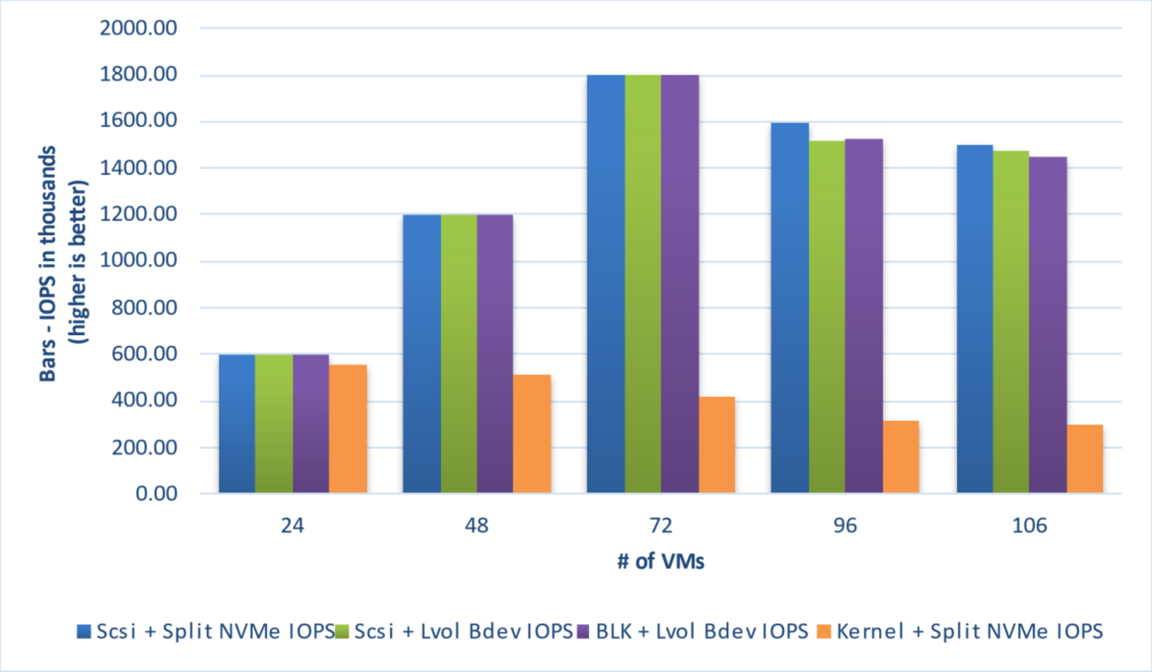


Figure 6: 4KiB 100% Random Writes IOPS, QD=1, throttling = 25k IOPS, 6 Vhost CPU cores

Conclusions

1. None of the tested Vhost solutions was able to serve 25,000 IOPS per VM for 4KiB Random Read workload.

2. Using 6 I/O processing cores, the SPDK Vhost serves 25,000 IOPS per VM to up to 72 VMs for 4 KiB Random Write workload.

3. Throughput and average latencies were up to 1.17x and 4.3x times better for Random Read and Random Write workloads respectively when using SPDK Vhost as compared to Kernel Vhost.

4. In the SPDK Vhost 24.01 performance report, we re-enabled CPU limitation on Kernel-Vhost using Linux cgroups. This rectification addressed a bug in our automation scripts that had previously overlooked such restrictions. By making this adjustment, we ensure a more equitable comparison, enabling us to accurately evaluate SPDK's performance advantages even when Kernel-Vhost operates under CPU constraints.

Test Case 3: Performance per NVMe drive

This test case was performed to understand performance and efficiency of the Vhost-Scsi and Vhost-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:

- 4KiB 100% Random Read
- 4KiB 100% Random Write
- 4KiB 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-3.28 **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 kernel threads (single thread per vhost target) each is limited to using up to 6 CPU cores via cgroups (NUMA optimized).

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, 128, 192}
	numjobs=1

Test Case 3 results

SPDK Vhost-Scsi

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

Access pattern	Backend	QD	Throughput (IOPS k)	Avg. latency (usec)
4KiB 100% Random Reads	Split NVMe	1	24.29	82.08
4KiB 100% Random Reads	Split NVMe	8	185.95	85.82
4KiB 100% Random Reads	Split NVMe	32	619.13	103.03
4KiB 100% Random Reads	Split NVMe	64	735.87	173.53
4KiB 100% Random Reads	Split NVMe	128	736.74	347.80
4KiB 100% Random Reads	Split NVMe	192	734.07	523.78
4KiB 100% Random Reads	Lvol	1	24.20	82.32
4KiB 100% Random Reads	Lvol	8	185.23	86.12
4KiB 100% Random Reads	Lvol	32	619.61	102.99
4KiB 100% Random Reads	Lvol	64	746.67	171.36
4KiB 100% Random Reads	Lvol	128	747.91	342.25
4KiB 100% Random Reads	Lvol	192	749.43	512.66
4KiB 100% Random Writes	Split NVMe	1	139.14	14.06
4KiB 100% Random Writes	Split NVMe	8	523.52	31.68
4KiB 100% Random Writes	Split NVMe	32	653.38	97.69
4KiB 100% Random Writes	Split NVMe	64	660.90	192.93
4KiB 100% Random Writes	Split NVMe	128	665.44	384.33
4KiB 100% Random Writes	Split NVMe	192	653.61	586.57
4KiB 100% Random Writes	Lvol	1	137.68	14.26
4KiB 100% Random Writes	Lvol	8	518.91	31.91
4KiB 100% Random Writes	Lvol	32	646.70	98.77
4KiB 100% Random Writes	Lvol	64	663.08	192.99
4KiB 100% Random Writes	Lvol	128	663.58	386.13
4KiB 100% Random Writes	Lvol	192	668.86	573.04
4KiB 70%/30% Random R/W	Split NVMe	1	32.28	61.83
4KiB 70%/30% Random R/W	Split NVMe	8	219.67	72.57
4KiB 70%/30% Random R/W	Split NVMe	32	571.50	111.66
4KiB 70%/30% Random R/W	Split NVMe	64	660.28	193.64
4KiB 70%/30% Random R/W	Split NVMe	128	675.92	377.90
4KiB 70%/30% Random R/W	Split NVMe	192	687.30	558.74
4KiB 70%/30% Random R/W	Lvol	1	34.22	58.34
4KiB 70%/30% Random R/W	Lvol	8	225.50	70.58
4KiB 70%/30% Random R/W	Lvol	32	582.67	109.72
4KiB 70%/30% Random R/W	Lvol	64	661.65	193.05
4KiB 70%/30% Random R/W	Lvol	128	693.21	368.96
4KiB 70%/30% Random R/W	Lvol	192	703.26	547.32

SPDK Vhost-Blk

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

Access pattern	Backend	QD	Throughput (IOPS k)	Avg. latency (usec)
4KiB 100% Random Reads	Lvol	1	24.32	81.91
4KiB 100% Random Reads	Lvol	8	186.27	85.69
4KiB 100% Random Reads	Lvol	32	630.34	101.27
4KiB 100% Random Reads	Lvol	64	842.20	151.30
4KiB 100% Random Reads	Lvol	128	839.81	304.34
4KiB 100% Random Reads	Lvol	192	836.43	459.56
4KiB 100% Random Writes	Lvol	1	140.38	13.97
4KiB 100% Random Writes	Lvol	8	531.99	31.47
4KiB 100% Random Writes	Lvol	32	684.23	93.10
4KiB 100% Random Writes	Lvol	64	688.71	185.82
4KiB 100% Random Writes	Lvol	128	729.11	350.56
4KiB 100% Random Writes	Lvol	192	694.89	555.31
4KiB 70%/30% Random R/W	Lvol	1	33.21	59.67
4KiB 70%/30% Random R/W	Lvol	8	227.71	70.00
4KiB 70%/30% Random R/W	Lvol	32	580.51	109.98
4KiB 70%/30% Random R/W	Lvol	64	751.15	170.23
4KiB 70%/30% Random R/W	Lvol	128	776.54	330.29
4KiB 70%/30% Random R/W	Lvol	192	760.81	504.90

Kernel Vhost-Scsi

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

Access pattern	Backend	QD	Throughput (IOPS k)	Avg. latency (usec)
4KiB 100% Random Reads	NVMe	1	21.70	91.96
4KiB 100% Random Reads	NVMe	8	150.81	105.76
4KiB 100% Random Reads	NVMe	32	322.43	198.25
4KiB 100% Random Reads	NVMe	64	359.40	355.54
4KiB 100% Random Reads	NVMe	128	415.37	615.08
4KiB 100% Random Reads	NVMe	192	414.97	925.58
4KiB 100% Random Writes	NVMe	1	75.22	26.32
4KiB 100% Random Writes	NVMe	8	263.52	60.61
4KiB 100% Random Writes	NVMe	32	341.23	187.23
4KiB 100% Random Writes	NVMe	64	312.58	409.77
4KiB 100% Random Writes	NVMe	128	289.78	883.27
4KiB 100% Random Writes	NVMe	192	311.46	1232.30
4KiB 70%/30% Random R/W	NVMe	1	27.66	70.95
4KiB 70%/30% Random R/W	NVMe	8	171.92	92.10
4KiB 70%/30% Random R/W	NVMe	32	308.55	206.76
4KiB 70%/30% Random R/W	NVMe	64	313.39	407.66
4KiB 70%/30% Random R/W	NVMe	128	326.31	783.20
4KiB 70%/30% Random R/W	NVMe	192	320.37	1198.84

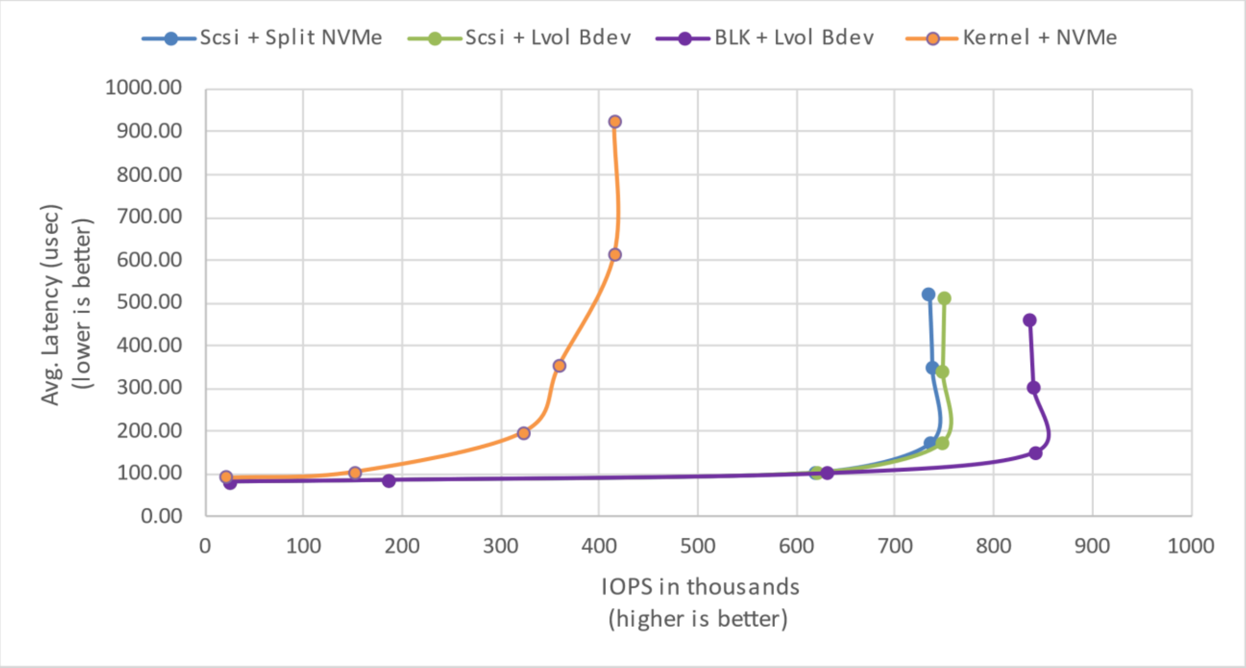


Figure 7: 4KiB 100% Random Reads IOPS and latency

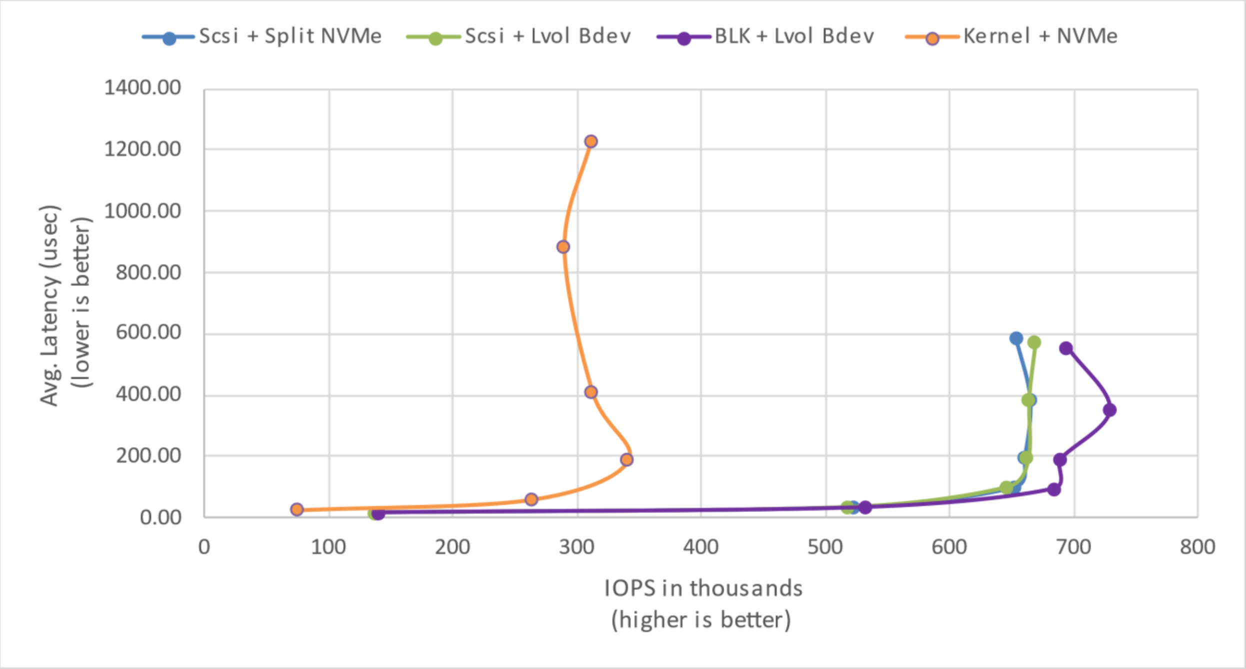


Figure 8: 4KiB 100% Random Writes IOPS and latency

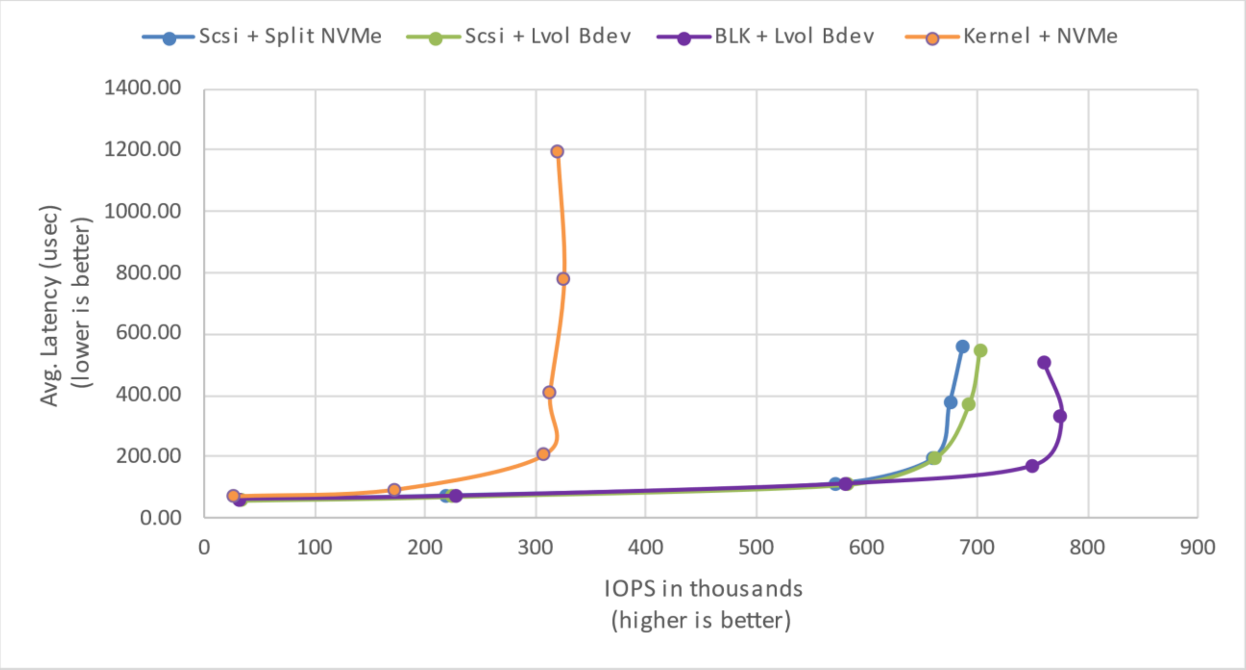


Figure 9: 4KiB 70%/30% Random Read/Write IOPS and latency

Conclusions

1. SPDK Vhost-Scsi with NVMe Split bdevs has lower latency and higher throughput than Kernel Vhost-Scsi in all of 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.

§