SPDK vhost performance report Release 19.07

Testing Date: October 2019

Performed by: Karol Latecki

Acknowledgments:

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

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

Vishal Verma (vishal4.verma@intel.com)

Revision History

Date	Revision	Comment
7/10/19	1.0	Test runs finished
15/10/19	1.0	Draft version of the document created
16/10/19	2.0	Reviews incorporated

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 information between SPDK vhost-scsi and Linux Kernel vhost-scsi software stacks under various test cases.

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 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

Item	Description
Server Platform	Intel WolfPass **R2224WFTZS** 
Motherboard	S2600WFT
CPU	Intel® Xeon® Cascade Lake 6230 Gold (27.5MB L3, 2.10 GHz) Number of cores 20, number of threads 40
Memory	Total 346 GBs
Operating System	Fedora 29
BIOS	02.01.0008 (02.04.2019)
Linux kernel version	5.1.20-200.fc29.x86_64
SPDK version	SPDK 19.07
Qemu version	QEMU emulator version 3.0.1 (qemu-3.0.1-4.fc29)
Storage	**OS:** 1x 120GB Intel SSDSC2BB120G4 **Storage**: 24x Intel® P4610™ 1.6TBs (FW: VDV10140) (6 on CPU NUMA Node 0, 18 on CPU NUMA Node 1)

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

Common settings used for all VMs used in tests.

Item	Description
CPU	2vCPU, pass through from physical host server. Explicit core usage on enforced using "taskset –a –c" command on host. 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 5.1.20 kernel version available from DNF repository with default patches for spectre meltdown issue enabled.

Guest VM systems use 5.1.20 kernel version available from DNF repository, but with spectre-meltdown patches disabled. Following options are added to GRUB options in /etc/default/grub:

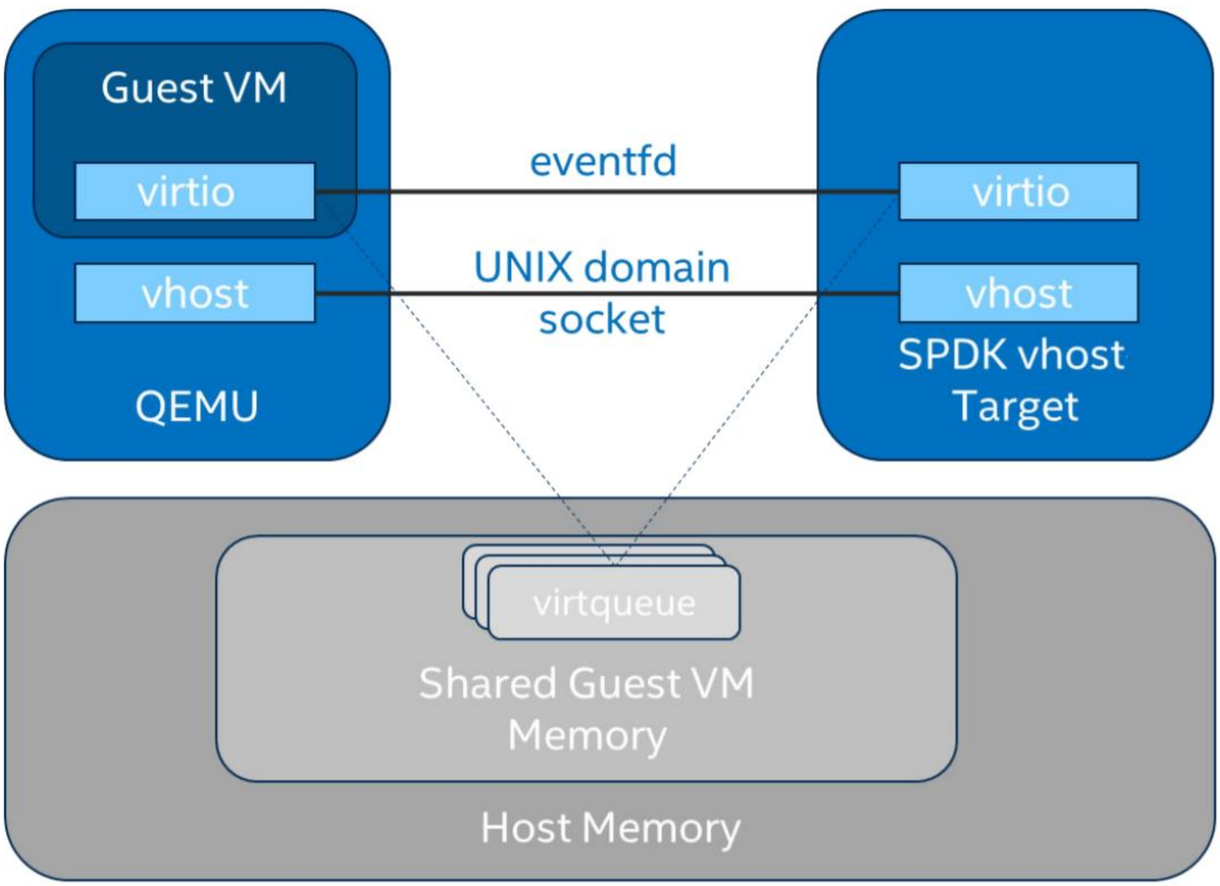
spectre_v2=off nopti

Introduction to SPDK vhost 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 the high performance SPDK userspace vhost target.

SPDK vhost target working

QEMU setups Vhost target via UNIX domain socket. The Vhost target transfers data to/from quest VM via shared memory. QEMU pre-allocates huge pages for guest VM to enable direct DMA by Vhost target. Guest VM submits I/O directly to 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 submission I/O process. Vhost 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.



This report is prepared to show the 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. In addition, SPDK vhost-blk stack is also included in the report for further comparison with 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 48 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 maximum 36 VMs. VM number was not increased beyond 36 because of the platform capabilities in terms of available CPU cores.

FIO was run in a client-server mode. Each VM was running a FIO server and the host server distributed jobs as a client. This allowed us to start FIO jobs across all VMs at the same time. Gtod_reduce=1 option was used to disable FIO latency measurements which allowed better IOPS and bandwidth results.

Results in the table and chart represent aggregate performance (IOPS and average latency) seen across all the VMs.

Item	Description
Test case	Test SPDK vhost target I/O core scaling performance
Test configuration	**FIO Version**: fio-3.3 **VM Configuration**: • Common settings described in **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 target for FIO workload. This is achieved by splitting SPDK NVMe bdevs by using either split vbdevs or lvol bdevs in configuration. **SPDK vhost target configuration:** • Test run with vhost-scsi and vhost-blk stacks • Vhost-scsi stack run with Split NVMe bdevs and Logical Volume bdevs • Vhost-blk stack run with Logical Volume bdevs • Test run with 1,2,3,4,5,6,8,10 and 12 cores for each stack-bdev combination **Kernel vhost target configuration:** - N/A
FIO configuration	[global] ioengine=libaio direct=1

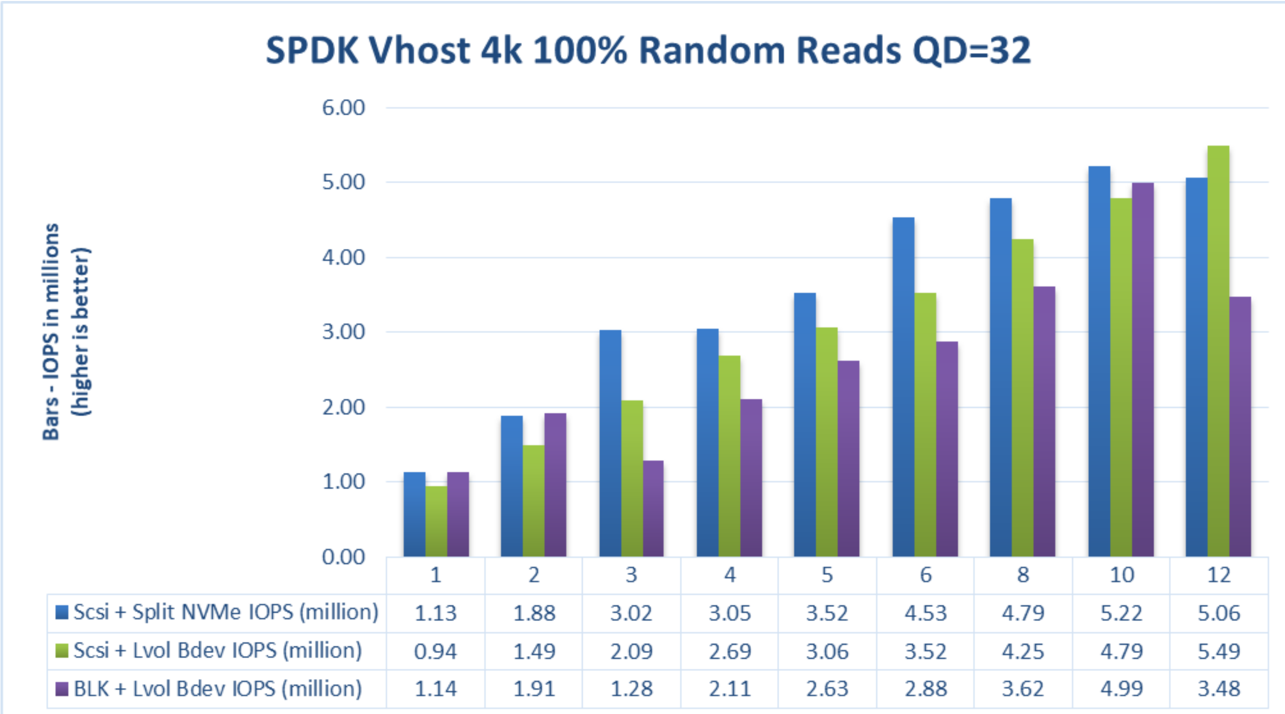
	thread=1 norandommap=1 time_based=1 gtod_reduce=1 ramp_time=60s runtime=240s numjobs=1 bs=4k rw=randrw rwmixread=100 (100% reads), 70 (70% reads, 30% writes), 0 (100% writes) iodepth={1, 8, 32, 64}

4k Random Read Results

Table 1: 4k 100% Random Reads IOPS, QD=32

# of CPU cores	# of VMs	Stack / Backend	IOPS (millions)
1	6	SCSI / Split NVMe Bdev	1.13
		SCSI / Lvol Bdev	0.94
		BLK / Lvol Bdev	1.14
2	12	SCSI / Split NVMe Bdev	1.88
		SCSI / Lvol Bdev	1.49
		BLK / Lvol Bdev	1.91
3	18	SCSI / Split NVMe Bdev	3.02
		SCSI / Lvol Bdev	2.09
		BLK / Lvol Bdev	1.28
4	24	SCSI / Split NVMe Bdev	3.05
		SCSI / Lvol Bdev	2.69
		BLK / Lvol Bdev	2.11
5	30	SCSI / Split NVMe Bdev	3.52
		SCSI / Lvol Bdev	3.06
		BLK / Lvol Bdev	2.63
6	36	SCSI / Split NVMe Bdev	4.53
		SCSI / Lvol Bdev	3.52
		BLK / Lvol Bdev	2.88
8	36	SCSI / Split NVMe Bdev	4.79
		SCSI / Lvol Bdev	4.25
		BLK / Lvol Bdev	3.62
10	36	SCSI / Split NVMe Bdev	5.22
		SCSI / Lvol Bdev	4.79
		BLK / Lvol Bdev	4.99
12	36	SCSI / Split NVMe Bdev	5.06
		SCSI / Lvol Bdev	5.49
		BLK / Lvol Bdev	3.48

Figure 1: Comparison of performance between various SPDK Vhost stack-bdev combinations for 4k Random Read QD=32 workload



4k Random Write Results

Table 2: 4k 100% Random Write IOPS, QD=32

# of CPU cores	# of VMs	Stack / Backend	IOPS (millions)
1	6	SCSI / Split NVMe Bdev	1.76
		SCSI / Lvol Bdev	0.98
		BLK / Lvol Bdev	0.96
2	12	SCSI / Split NVMe Bdev	2.17
		SCSI / Lvol Bdev	1.15
		BLK / Lvol Bdev	1.69
3	18	SCSI / Split NVMe Bdev	2.70
		SCSI / Lvol Bdev	2.17
		BLK / Lvol Bdev	2.12
4	24	SCSI / Split NVMe Bdev	3.15
		SCSI / Lvol Bdev	2.85
		BLK / Lvol Bdev	3.21
5	30	SCSI / Split NVMe Bdev	4.12
		SCSI / Lvol Bdev	3.56
		BLK / Lvol Bdev	3.92
6	36	SCSI / Split NVMe Bdev	4.84
		SCSI / Lvol Bdev	4.02

		BLK / Lvol Bdev	2.55
8	36	SCSI / Split NVMe Bdev	5.70
		SCSI / Lvol Bdev	4.93
		BLK / Lvol Bdev	4.73
10	36	SCSI / Split NVMe Bdev	5.45
		SCSI / Lvol Bdev	4.67
		BLK / Lvol Bdev	4.88
12	36	SCSI / Split NVMe Bdev	5.29
		SCSI / Lvol Bdev	5.52
		BLK / Lvol Bdev	4.41

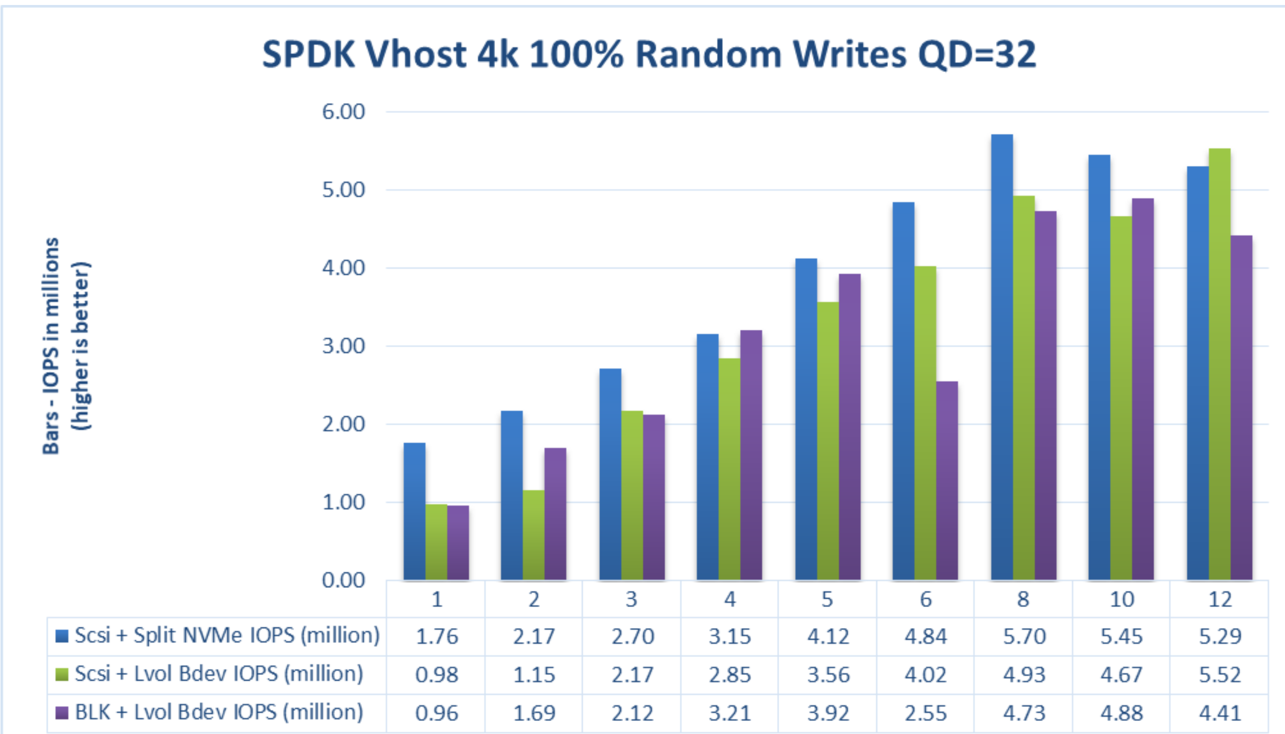


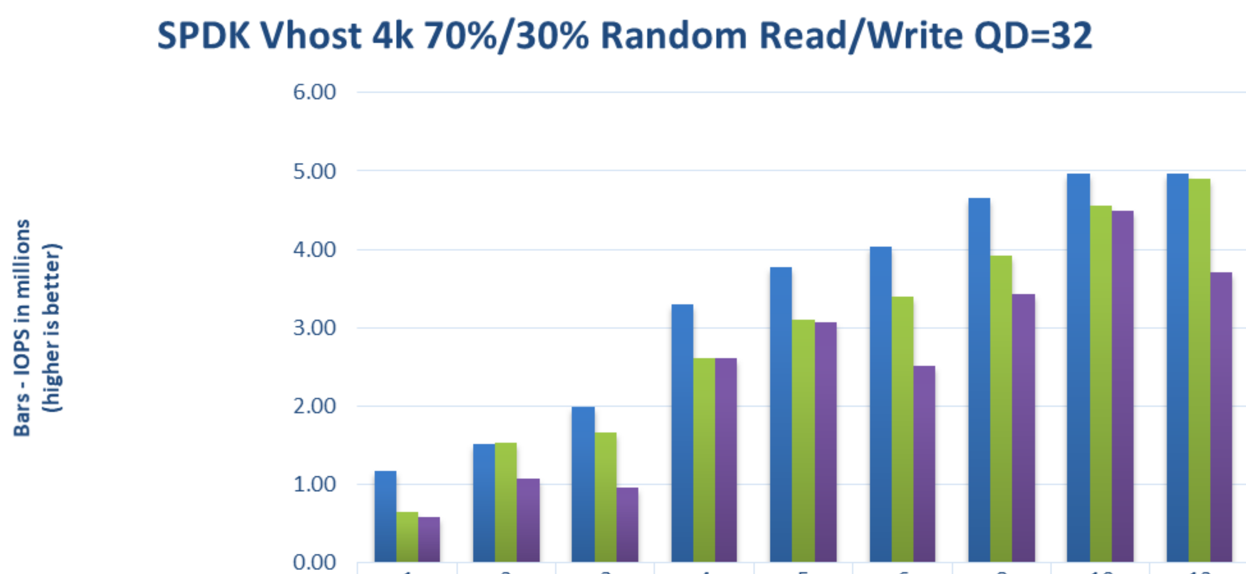
Figure 2: Comparison of performance between various SPDK Vhost stack-bdev combinations for 4k Random Write QD=32 workload

4k Random Read-Write Results

Table 3: 4k Random 70% Read 30% Write IOPS, QD=32

# of CPU cores	# of VMs	Stack / Backend	IOPS (millions)
1	6	SCSI / Split NVMe Bdev	1.17

		SCSI / Lvol Bdev	0.64
		BLK / Lvol Bdev	0.57
2	12	SCSI / Split NVMe Bdev	1.52
		SCSI / Lvol Bdev	1.53
		BLK / Lvol Bdev	1.07
3	18	SCSI / Split NVMe Bdev	1.99
		SCSI / Lvol Bdev	1.66
		BLK / Lvol Bdev	0.95
4	24	SCSI / Split NVMe Bdev	3.30
		SCSI / Lvol Bdev	2.62
		BLK / Lvol Bdev	2.60
5	30	SCSI / Split NVMe Bdev	3.76
		SCSI / Lvol Bdev	3.10
		BLK / Lvol Bdev	3.06
6	36	SCSI / Split NVMe Bdev	4.03
		SCSI / Lvol Bdev	3.39
		BLK / Lvol Bdev	2.52
8	36	SCSI / Split NVMe Bdev	4.65
		SCSI / Lvol Bdev	3.92
		BLK / Lvol Bdev	3.43
10	36	SCSI / Split NVMe Bdev	4.96
		SCSI / Lvol Bdev	4.56
		BLK / Lvol Bdev	4.49
12	36	SCSI / Split NVMe Bdev	4.97
		SCSI / Lvol Bdev	4.91
		BLK / Lvol Bdev	3.71



	1	2	3	4	5	6	8	10	12
Scsi + Split NVMe IOPS (million)	1.17	1.52	1.99	3.30	3.76	4.03	4.65	4.96	4.97
Scsi + Lvol Bdev IOPS (million)	0.64	1.53	1.66	2.62	3.10	3.39	3.92	4.56	4.91
BLK + Lvol Bdev IOPS (million)	0.57	1.07	0.95	2.60	3.06	2.52	3.43	4.49	3.71

Figure 3: Comparison of performance between various SPDK Vhost stack-bdev combinations for 4k Random 70% Read 30% Write QD=32 workload

Conclusions

1. SCSI stack with both Split NVMe and Logical Volume bdev backend scales near linearly up to 10 CPU cores. There is none or small improvement when increasing to 12 CPUs, which might be because there is no enough CPU resources on the platform.
   (With 12 CPU cores for Vhost and 36VMs some of the VMs need to share CPU resources or be cross-NUMA configured in respect to NVMe hardware)

2. BLK stack scaling is not linear. There are visible and reproducible performance drops, especially at 12 CPU cores used for Vhost process. This might suggest some software problem.

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 & Linux Kernel. 10K IOPS were chosen as the rate limiter using linux cgroups.

Note: For those comparing the results with 17.07 Vhost Performance Report - the rate limiter value was lowered to 10k IOPS because of the change in the hardware setup. P4610 1.6TB disks for workloads running with QD=1 are able to reach about 12-13k IOPS at most, so previous 20k IOPS limiter would never be reached.

Each individual VM was running FIO with the following workloads:

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

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

Test configuration	**FIO Version:** fio-3.3
	
	VM Configuration:
	• Common settings described in **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 throttled to run at maximum of 10k IOPS (read and write)
	
	SPDK vhost target configuration:
	• Test run with vhost-scsi and vhost-blk stacks
	• Vhost-scsi stack run with Split NVMe bdevs and Logical Volume bdevs
	• Vhost-blk stack run with Logical Volume bdevs
	• Test run with 4 CPU cores (NUMA optimized)
	
	Kernel vhost-scsi configuration:
	• Used cgroups to limit vhost process to 4 cores
	• NUMA optimization 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

Test result: 4K 100% Random Reads QD=1

# of VMs	Stack	Backend bdev	IOPS (k)	Avg Lat. (usec)	Host CPU utilization
24 VMs	SPDK-SCSI	Split NVMe	237.75	99.72	26.4
	SPDK-SCSI	Logical Volume	236.93	100.19	26.5
	SPDK-BLK	Logical Volume	236.07	100.37	26.4
	Kernel-SCSI	Partitioned NVMe	103.34	230.58	18.6
48 VMs	SPDK-SCSI	Split NVMe	452.85	101.44	50.5
	SPDK-SCSI	Logical Volume	466.46	100.60	50.6
	SPDK-BLK	Logical Volume	462.78	102.16	51.2
	Kernel-SCSI	Partitioned NVMe	107.89	443.29	31.5
72 VMs	SPDK-SCSI	Split NVMe	681.88	103.41	75.6
	SPDK-SCSI	Logical Volume	677.86	104.03	75.8
	SPDK-BLK	Logical Volume	573.03	124.00	75.8
	Kernel-SCSI	Partitioned NVMe	147.99	494.40	46.1

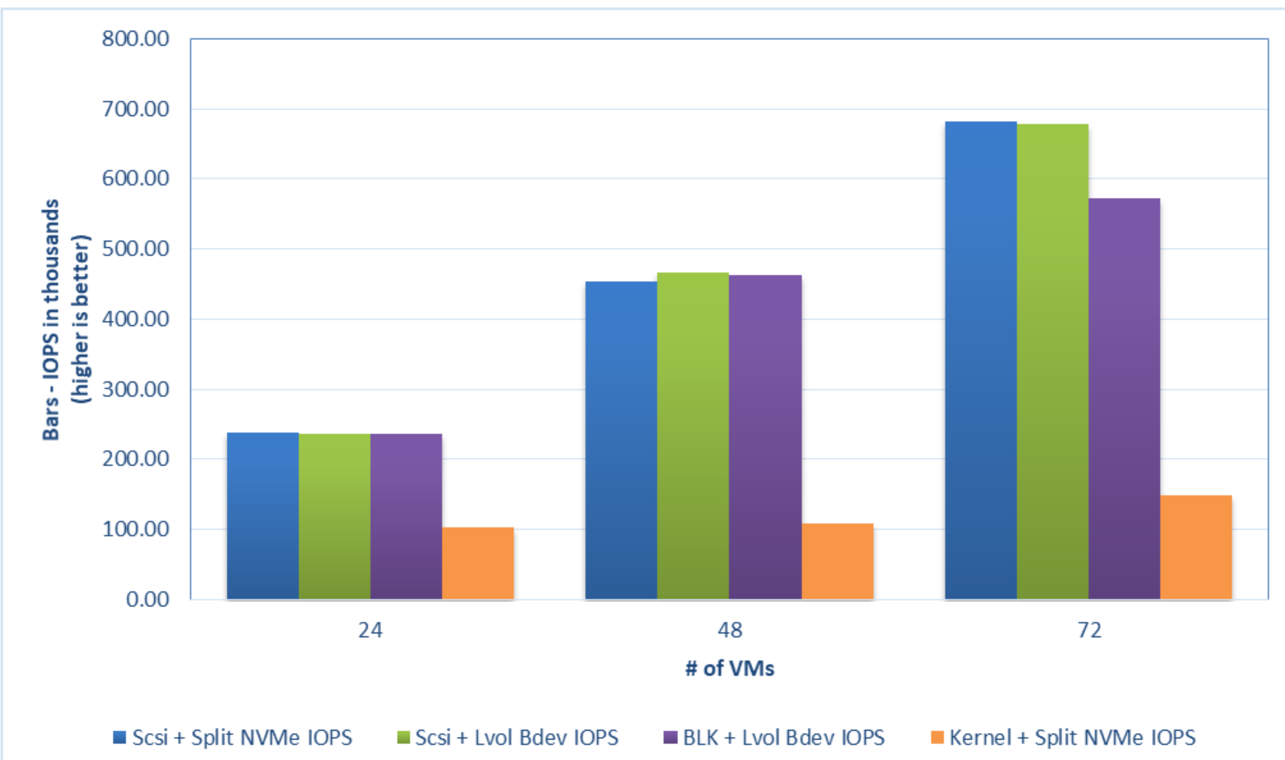


Figure 4: 4k 100% Random Reads IOPS and latency, QD=1, throttling = 10k IOPS

Test result: 4K 100% Random Writes QD=1

# of VMs	Stack	Backend bdev	IOPS (k)	Avg Lat. (usec)	Host CPU utilization
24 VMs	SPDK-SCSI	Split NVMe	240.00	98.19	19.5
	SPDK-SCSI	Logical Volume	240.00	98.20	19.5
	SPDK-BLK	Logical Volume	240.00	98.18	19.6
	Kernel-SCSI	Partitioned NVMe	105.66	225.77	17.4
48 VMs	SPDK-SCSI	Split NVMe	480.00	98.16	35.9
	SPDK-SCSI	Logical Volume	480.00	98.16	36.1
	SPDK-BLK	Logical Volume	479.97	98.19	37.4
	Kernel-SCSI	Partitioned NVMe	131.56	374.32	29.7
72 VMs	SPDK-SCSI	Split NVMe	719.97	98.08	54.7
	SPDK-SCSI	Logical Volume	716.65	98.11	52.7
	SPDK-BLK	Logical Volume	629.14	111.73	72.8
	Kernel-SCSI	Partitioned NVMe	280.43	261.66	59.0

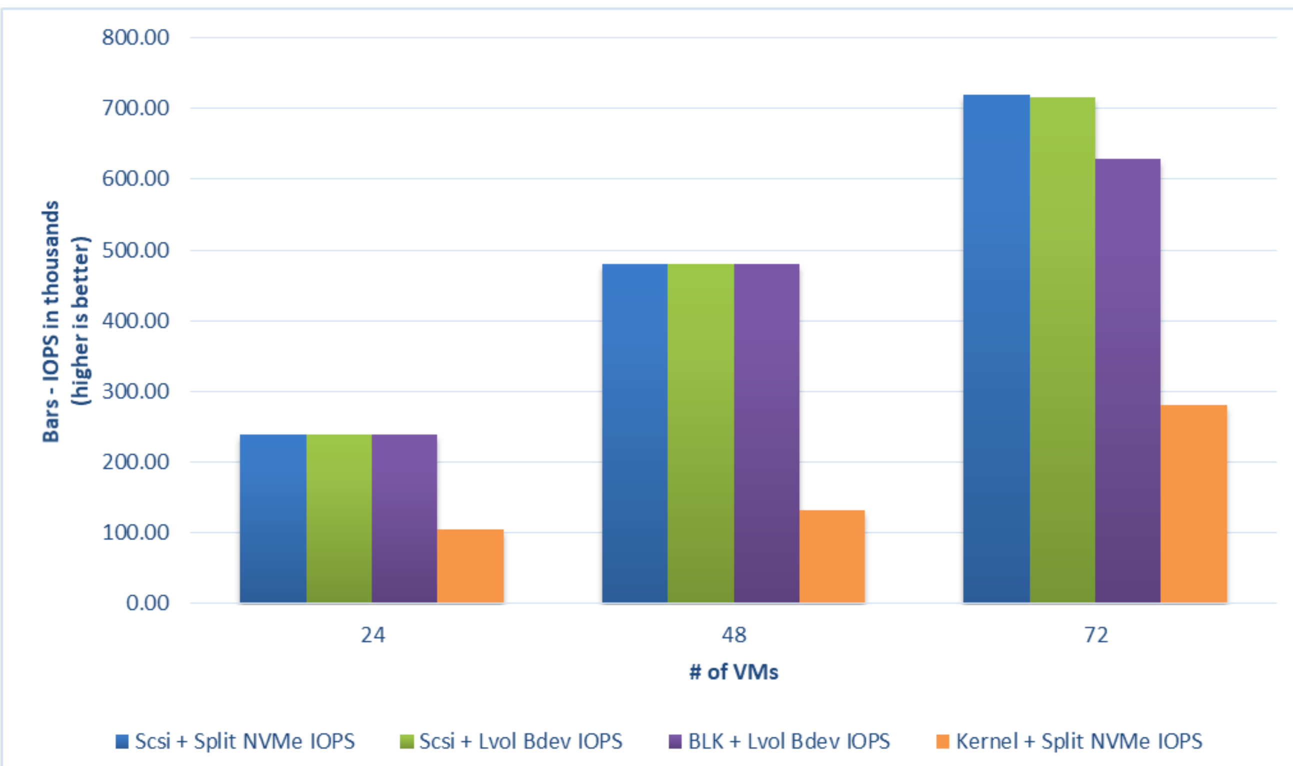


Figure 5: 4k 100% Random Writes IOPS and latency, QD=1, throttling = 10k IOPS

Conclusions

1. In most of the cases VMs using SPDK Vhost exposed devices were able to reach desired IOPS levels, with SPDK Vhost-Blk/Lvol bdev being an exception. The reason for this performance drop was unexplained.

2. SPDK vhost is able to serve IO at desired level for increasing number of VMs.

3. Average latencies are up to 5.0x times better for Random Read workload and up to 3.5x times better for Random Write workload in case of SPDK Vhost when compared to Kernel Vhost.

Note: Kernel-vhost process was not NUMA-optimized for this scenario. We found that using taskset or cgroups to limit kernel vhost to CPUs on 2 NUMA nodes causes the CPUs from the other node to be not used at all.

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 single NVMe drive on 2 VMs. Each VM has 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 following workloads:

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

Item	Description
Test case	Test SPDK vhost target I/O core scaling performance
Test configuration	**FIO Version:** fio-3.3 **VM Configuration**: • Common settings described in **Virtual Machine Settings** chapter • 2 VMs tested • Each VM has a single vhost device which is one of equal partitions of single NVMe drive. **SPDK vhost target configuration:** • SPDK vhost process run on a single, separate individual physical CPU core • Vhost-scsi stack run with Split NVMe bdevs and Logical Volume bdevs • Vhost-blk stack run with Logical Volume bdevs **Kernel vhost target configuration**: • Vhost process was run on separate individual physical 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: IOPS and latency results, SCSI stack

Access pattern	Backend	QD	Throughput (IOPS)	Avg. latency (usec)
4k 100% Random Reads	Split NVMe	1	47828.08	83.12
4k 100% Random Reads	Split NVMe	8	270534.32	107.75
4k 100% Random Reads	Split NVMe	32	522570.35	277.68
4k 100% Random Reads	Split NVMe	64	565386.04	455.88
4k 100% Random Reads	Lvol	1	20542.81	91.2
4k 100% Random Reads	Lvol	8	167691.85	93.73
4k 100% Random Reads	Lvol	32	437854.16	145.29
4k 100% Random Reads	Lvol	64	518683.64	236.21
4k 100% Random Writes	Split NVMe	1	---	---
4k 100% Random Writes	Split NVMe	8	---	---
4k 100% Random Writes	Split NVMe	32	---	---
4k 100% Random Writes	Split NVMe	64	---	---
4k 100% Random Writes	Lvol	1	---	---
4k 100% Random Writes	Lvol	8	---	---
4k 100% Random Writes	Lvol	32	---	---
4k 100% Random Writes	Lvol	64	---	---
4k 70%/30% Random Read Writes	Split NVMe	1	60594.57	64.945
4k 70%/30% Random Read Writes	Split NVMe	8	196838.98	163.89
4k 70%/30% Random Read Writes	Split NVMe	32	436611.4	292.081
4k 70%/30% Random Read Writes	Split NVMe	64	436434.69	589.932
4k 70%/30% Random Read Writes	Lvol	1	28418.11	83.701
4k 70%/30% Random Read Writes	Lvol	8	155719.1	121.154
4k 70%/30% Random Read Writes	Lvol	32	394447.15	161.135
4k 70%/30% Random Read Writes	Lvol	64	398651.849	384.441

SPDK Vhost-Blk

Table: IOPS and latency results, BLK stack

Access pattern	Backend	QD	Throughput (IOPS)	Avg. latency (usec)
4k 100% Random Reads	Lvol	1	24771.41	80.1
4k 100% Random Reads	Lvol	8	169663.98	93.81
4k 100% Random Reads	Lvol	32	420093.79	168.12
4k 100% Random Reads	Lvol	64	576393.34	220.76
4k 100% Random Writes	Lvol	1	---	---
4k 100% Random Writes	Lvol	8	---	---
4k 100% Random Writes	Lvol	32	---	---
4k 100% Random Writes	Lvol	64	---	---
4k 70%/30% Random Read Writes	Lvol	1	31945.54	61.946
4k 70%/30% Random Read Writes	Lvol	8	120744.88	122.033
4k 70%/30% Random Read Writes	Lvol	32	353924.05	239.622
4k 70%/30% Random Read Writes	Lvol	64	362509.49	472.181

Kernel Vhost-Scsi

Table: IOPS and latency results, Kernel Vhost-Scsi

Access pattern	Backend	QD	Throughput (IOPS)	Avg. latency (usec)
4k 100% Random Reads	NVMe	1	16453.27	120.2
4k 100% Random Reads	NVMe	8	91502.1	174.31
4k 100% Random Reads	NVMe	32	184832.11	310.05
4k 100% Random Reads	NVMe	64	167097.56	580.47
4k 100% Random Writes	NVMe	1	---	---
4k 100% Random Writes	NVMe	8	---	---
4k 100% Random Writes	NVMe	32	---	---
4k 100% Random Writes	NVMe	64	---	---
4k 70%/30% Random Read Writes	NVMe	1	16136.46	137.75
4k 70%/30% Random Read Writes	NVMe	8	68559.18	333.453
4k 70%/30% Random Read Writes	NVMe	32	209007.33	304.983
4k 70%/30% Random Read Writes	NVMe	64	130545.83	956.941

Figure 9: *4k 100% Random Reads IOPS and latency*

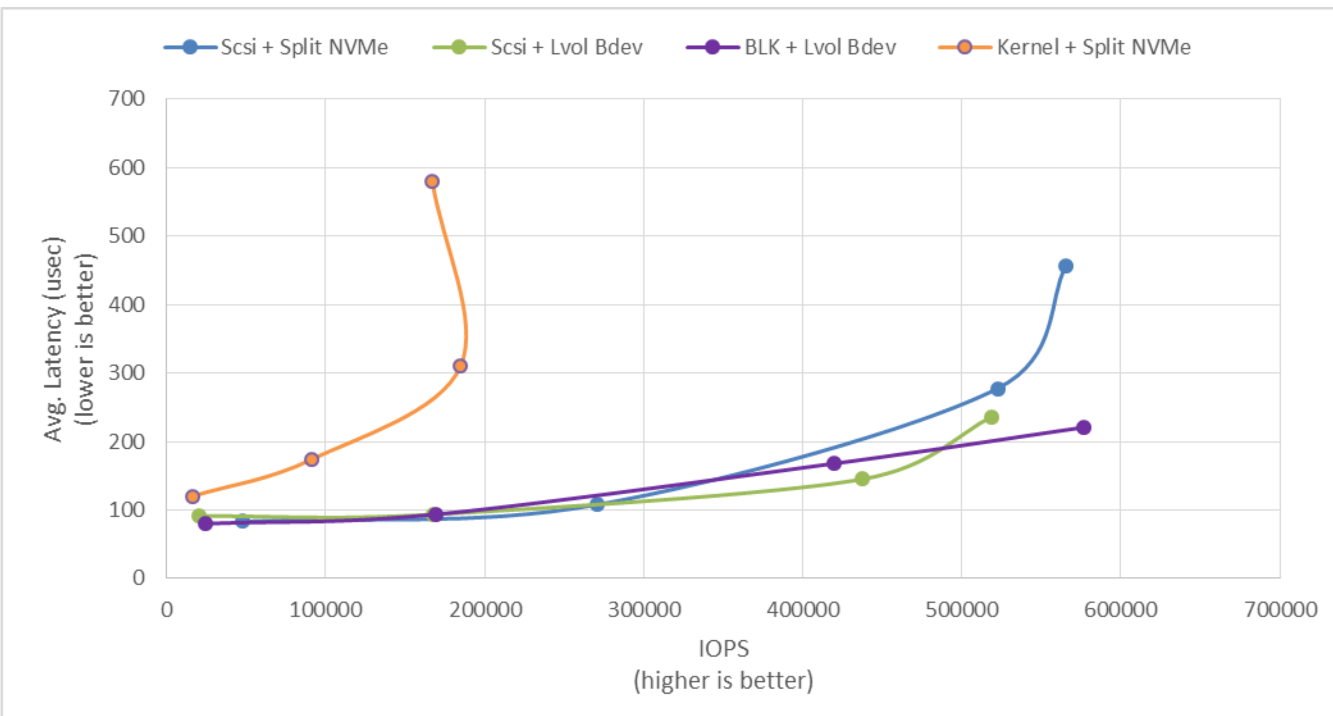
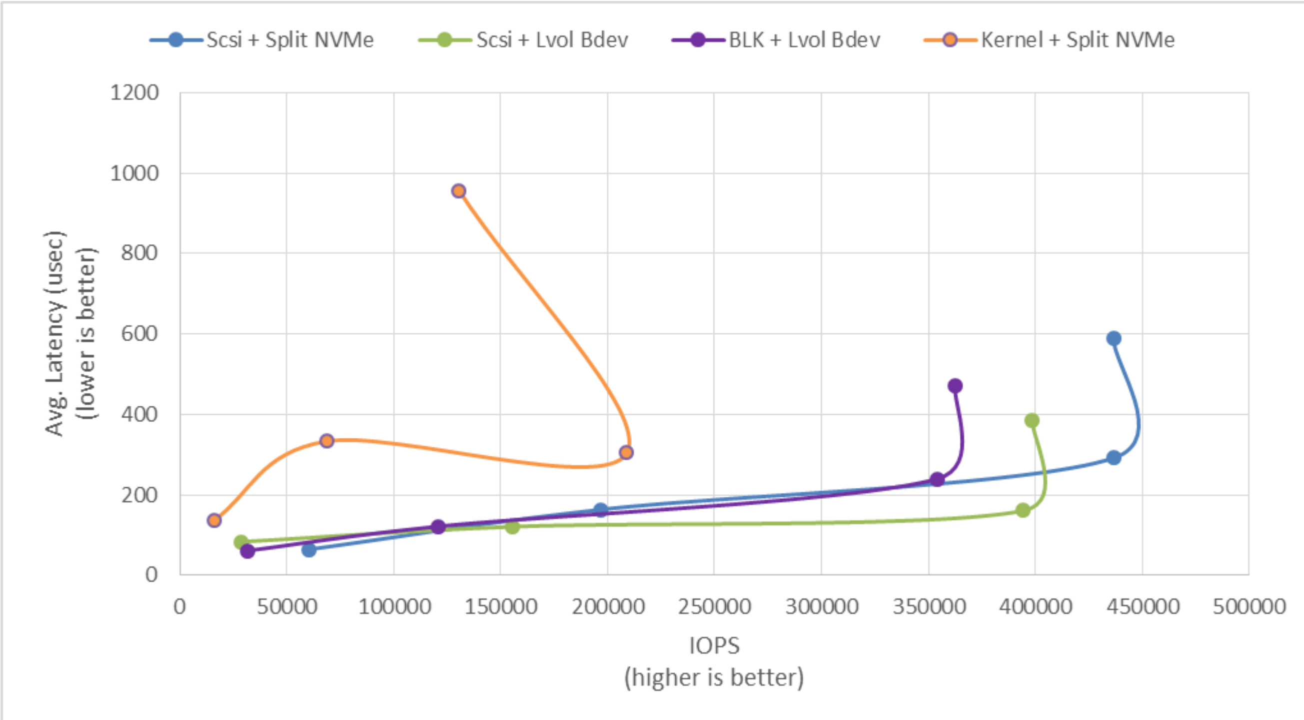


Figure 10: *4k 70%/30% Random Read/Write*



Conclusions

1. Random Write results were not included in the report. The results were not realistic and not conclusive, most probably due to an error when running the test.

2. SPDK vhost-scsi with NVMe Split bdevs has lower latency and higher throughput than Kernel vhost-scsi in case of all run 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 underlying system when comparing kernel vs. SPDK vhost-scsi target implementations.

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

This report provides information regarding methodologies and practices while benchmarking vhost-scsi and vhost-blk 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

§