

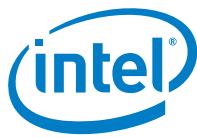
SPDK Vhost performance report

Release 20.07

Testing Date: August 2020

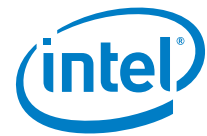
Performed by Karol Latecki

Acknowledgments:



Contents

Contents	2
Audience and Purpose.....	3
Test setup	4
Hardware configuration	4
BIOS Settings	5
Virtual Machine Settings.....	5
Kernel & BIOS Spectre-Meltdown information	6
Introduction to the SPDK Vhost target	7
SPDK Vhost target architecture	7
Test Case 1: SPDK Vhost core scaling	9
4KB Random Read Results	11
4KB Random Write Results	12
4KB Random Read-Write Results.....	13
Logical Volumes performance impact	14
LTO performance impact	15
Packed Ring performance impact.....	16
Test Case 2: Rate Limiting IOPS per VM.....	18
Test Case 2 Results	20
Conclusions	22
Test Case 3: Performance per NVMe drive	23
Test Case 3 results.....	24
Conclusions	27
Summary	28
Appendix – Packed Ring Performance	29




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

Item	Description												
Server Platform	Intel WolfPass R2224WFTZS 												
Motherboard	S2600WFT												
CPU	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												
Memory	10 x 32GB Micron DDR4 36ASF4G72PZ-2G6H1R Total 320 GBs Memory channel population: <table border="1" data-bbox="479 1312 1429 1564"> <thead> <tr> <th>P1</th> <th>P2</th> </tr> </thead> <tbody> <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> </tbody> </table>	P1	P2	CPU1_DIMM_A1	CPU2_DIMM_A1	CPU1_DIMM_B1	CPU2_DIMM_B1	CPU1_DIMM_C1	CPU2_DIMM_C1	CPU1_DIMM_D1	CPU2_DIMM_D1	CPU1_DIMM_E1	CPU2_DIMM_E1
P1	P2												
CPU1_DIMM_A1	CPU2_DIMM_A1												
CPU1_DIMM_B1	CPU2_DIMM_B1												
CPU1_DIMM_C1	CPU2_DIMM_C1												
CPU1_DIMM_D1	CPU2_DIMM_D1												
CPU1_DIMM_E1	CPU2_DIMM_E1												
Operating System	Fedora 30												
BIOS	02.01.0010 (06.01.2020)												
Linux kernel version	5.4.14-100.fc30.x86_64												
SPDK version	SPDK 20.07												
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

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 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	<ul style="list-style-type: none"> 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.

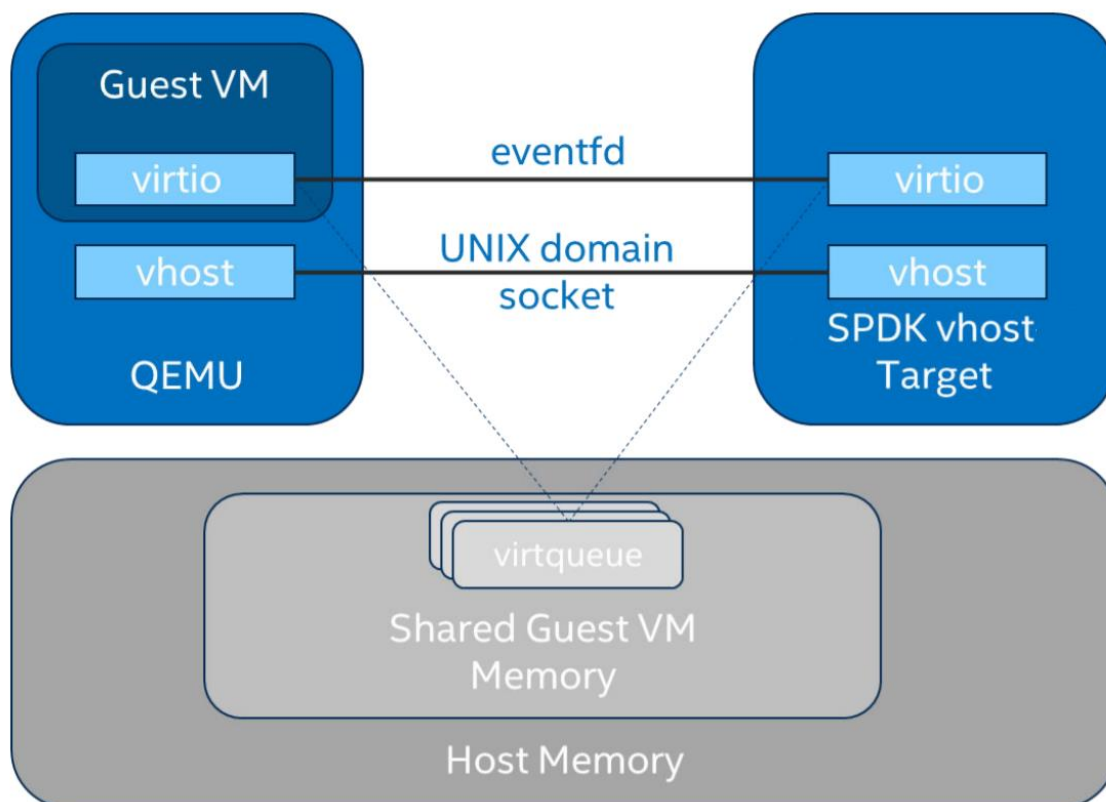
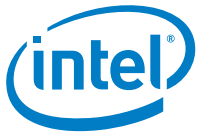


Figure 1: SPDK Vhost-scsi architecture



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

Item	Description
Test case	Test SPDK Vhost target I/O core scaling performance
Test configuration	<p>FIO Version: fio-3.19</p> <p>VM Configuration:</p> <ul style="list-style-type: none">• 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. <p>SPDK Vhost target configuration:</p> <ul style="list-style-type: none">• 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. <p>Kernel Vhost target configuration:</p> <ul style="list-style-type: none">- N/A



FIO configuration	<pre>[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}</pre>
--------------------------	---



4KB Random Read Results

Table 1: 4KB 100% Random Reads IOPS, QD=64

# of CPU cores	# of VMs	Stack / Backend	IOPS (millions)
1	6	SCSI / Split NVMe Bdev	1.80
		SCSI / Lvol Bdev	1.51
		BLK / Lvol Bdev	1.60
2	12	SCSI / Split NVMe Bdev	3.04
		SCSI / Lvol Bdev	2.56
		BLK / Lvol Bdev	2.79
4	24	SCSI / Split NVMe Bdev	4.82
		SCSI / Lvol Bdev	3.80
		BLK / Lvol Bdev	4.37
6	36	SCSI / Split NVMe Bdev	6.20
		SCSI / Lvol Bdev	4.99
		BLK / Lvol Bdev	5.52
8	36	SCSI / Split NVMe Bdev	6.43
		SCSI / Lvol Bdev	5.62
		BLK / Lvol Bdev	6.29
10	36	SCSI / Split NVMe Bdev	6.60
		SCSI / Lvol Bdev	6.41
		BLK / Lvol Bdev	7.05
12	36	SCSI / Split NVMe Bdev	5.88
		SCSI / Lvol Bdev	5.99
		BLK / Lvol Bdev	6.96

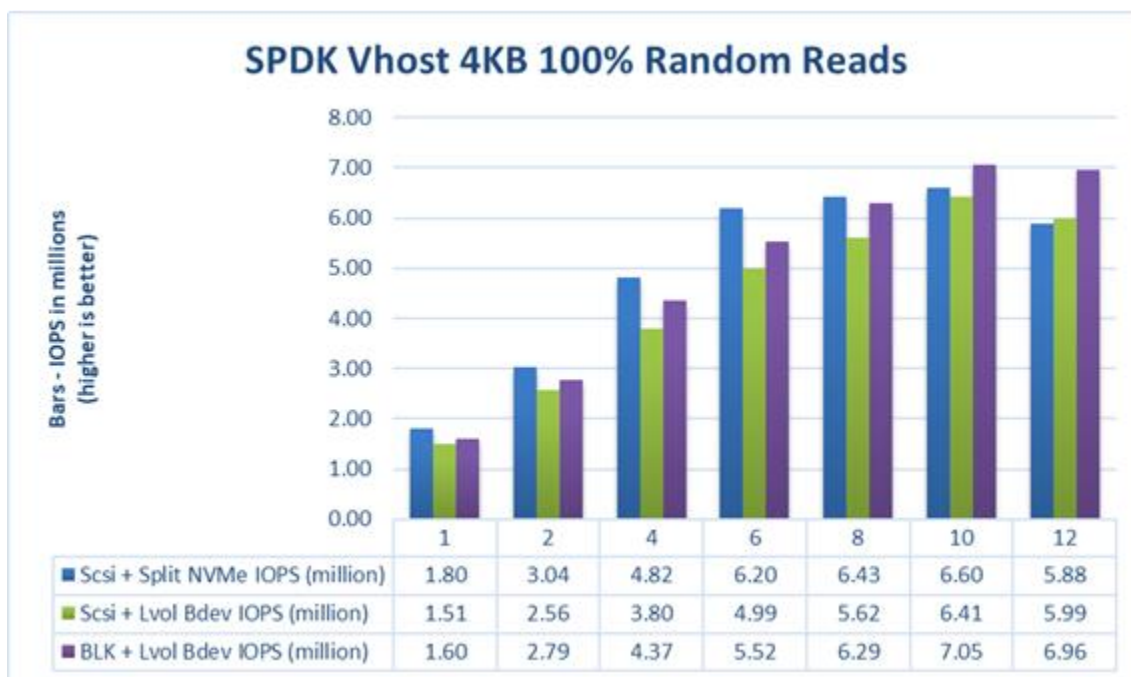


Figure 2: Comparison of performance between various SPDK Vhost stack-bdev combinations for 4KB Random Read QD=64 workload

4KB Random Write Results

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

# of CPU cores	# of VMs	Stack / Backend	IOPS (millions)
1	6	SCSI / Split NVMe Bdev	1.58
		SCSI / Lvol Bdev	1.44
		BLK / Lvol Bdev	1.50
2	12	SCSI / Split NVMe Bdev	2.94
		SCSI / Lvol Bdev	2.59
		BLK / Lvol Bdev	2.86
4	24	SCSI / Split NVMe Bdev	4.74
		SCSI / Lvol Bdev	4.03
		BLK / Lvol Bdev	4.58
6	36	SCSI / Split NVMe Bdev	6.24
		SCSI / Lvol Bdev	5.57
		BLK / Lvol Bdev	6.04
8	36	SCSI / Split NVMe Bdev	6.20
		SCSI / Lvol Bdev	5.77
		BLK / Lvol Bdev	6.55
10	36	SCSI / Split NVMe Bdev	5.50
		SCSI / Lvol Bdev	5.51
		BLK / Lvol Bdev	5.92
12	36	SCSI / Split NVMe Bdev	5.03
		SCSI / Lvol Bdev	5.86
		BLK / Lvol Bdev	6.56

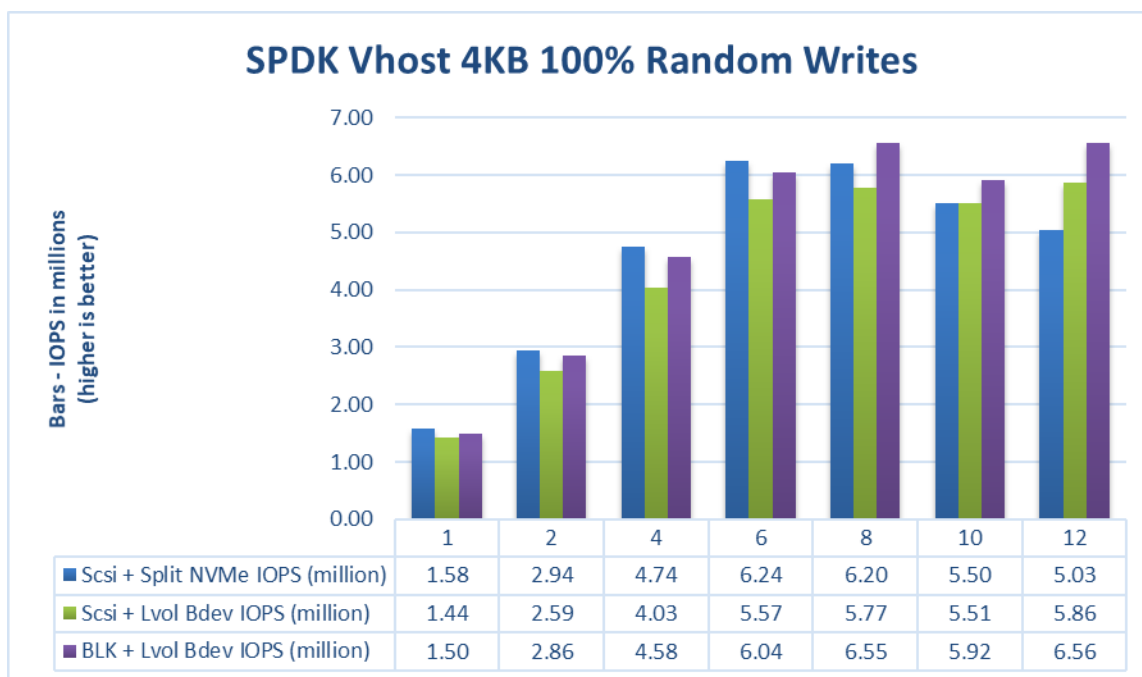
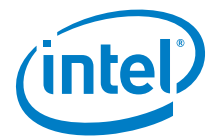


Figure 3: Comparison of performance between various SPDK Vhost stack-bdev combinations for 4KB Random Write QD=32 workload



4KB Random Read-Write Results

Table 3: 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	1.45
		BLK / Lvol Bdev	1.52
2	12	SCSI / Split NVMe Bdev	2.88
		SCSI / Lvol Bdev	2.47
		BLK / Lvol Bdev	2.69
4	24	SCSI / Split NVMe Bdev	4.48
		SCSI / Lvol Bdev	3.85
		BLK / Lvol Bdev	4.20
6	36	SCSI / Split NVMe Bdev	6.07
		SCSI / Lvol Bdev	4.96
		BLK / Lvol Bdev	5.47
8	36	SCSI / Split NVMe Bdev	5.88
		SCSI / Lvol Bdev	5.47
		BLK / Lvol Bdev	6.04
10	36	SCSI / Split NVMe Bdev	5.95
		SCSI / Lvol Bdev	5.92
		BLK / Lvol Bdev	6.36
12	36	SCSI / Split NVMe Bdev	5.55
		SCSI / Lvol Bdev	5.74
		BLK / Lvol Bdev	6.39

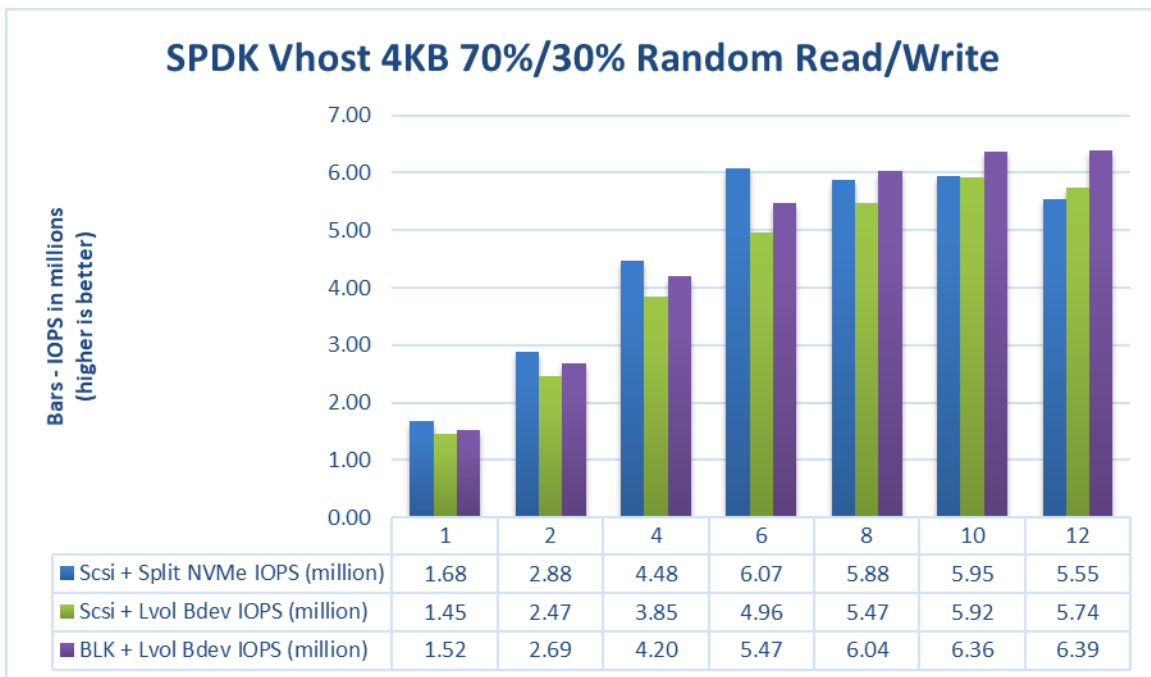


Figure 4: Comparison of performance between various SPDK Vhost stack-bdev combinations for 4KB Random 70% Read 30% Write QD=64 workload

Logical Volumes performance impact

The SPDK Vhost SCSI tests were run using two bdev backends – Split NVMe 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 4: 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	1.80	1.51	-16.02%
	2	12	3.04	2.56	-15.64%
	4	24	4.82	3.80	-21.23%
	6	36	6.20	4.99	-19.56%
	8	36	6.43	5.62	-12.72%
	10	36	6.60	6.41	-2.82%
4KB 100% Random Write	12	36	5.88	5.99	1.89%
	1	6	1.58	1.44	-9.01%
	2	12	2.94	2.59	-12.15%
	4	24	4.74	4.03	-14.91%
	6	36	6.24	5.57	-10.71%
	8	36	6.20	5.77	-6.92%
4KB 70% Random Read 30% Random Write	10	36	5.50	5.51	0.12%
	12	36	5.03	5.86	16.45%
	1	6	1.68	1.45	-13.55%
	2	12	2.88	2.47	-14.44%
	4	24	4.48	3.85	-14.03%
	6	36	6.07	4.96	-18.18%
4KB 70% Random Read 30% Random Write	8	36	5.88	5.47	-7.05%
	10	36	5.95	5.92	-0.51%
	12	36	5.55	5.74	3.37%



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 5: 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	1.51	1.63	7.68%
	2	12	2.56	2.77	7.96%
	4	24	3.80	4.03	6.09%
	6	36	4.99	5.27	5.61%
	8	36	5.62	5.81	3.50%
	10	36	6.41	6.45	0.61%
	12	36	5.99	5.98	-0.26%
4KB 100% Random Write	1	6	1.44	1.48	2.98%
	2	12	2.59	2.73	5.61%
	4	24	4.03	4.34	7.64%
	6	36	5.57	5.78	3.78%
	8	36	5.77	5.95	3.14%
	10	36	5.51	5.37	-2.45%
	12	36	5.86	5.80	-1.06%
4KB 70% Random Read 30% Random Write	1	6	1.45	1.54	6.03%
	2	12	2.47	2.65	7.32%
	4	24	3.85	4.24	10.20%
	6	36	4.96	5.22	5.09%
	8	36	5.47	5.49	0.43%
	10	36	5.92	5.82	-1.73%
	12	36	5.74	5.60	-2.28%

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. This option is disabled by default in SPDK 20.07 release and must be explicitly enabled when creating SPDK Vhost BLK controllers.

For this test, unlike described in [“Test setup”](#) chapter, a different version of Qemu emulator was used. Packed Ring feature requires that at least 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 6: Packed Ring performance impact on SPDK Vhost BLK controllers. Fio qtod_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	1.53	1.61	250.00	238.05	5.05%	-4.78%
	2	12	2.65	2.87	289.15	267.39	8.31%	-7.53%
	4	24	4.24	4.48	361.67	344.35	5.70%	-4.79%
	6	36	5.48	5.79	418.49	398.34	5.54%	-4.81%
	8	36	6.38	6.69	360.95	340.60	4.93%	-5.64%
	10	36	6.97	7.41	330.39	311.00	6.32%	-5.87%
	12	36	6.57	6.68	350.61	345.68	1.66%	-1.41%
4KB 100% Random Write QD=32	1	6	1.45	1.50	134.79	129.58	4.08%	-3.87%
	2	12	2.77	2.89	137.34	132.07	4.47%	-3.83%
	4	24	4.58	4.76	169.06	163.44	3.92%	-3.32%
	6	36	6.09	6.28	188.18	183.62	3.08%	-2.42%
	8	36	6.54	6.74	175.38	169.40	3.06%	-3.41%
	10	36	5.69	5.81	205.30	201.14	2.18%	-2.03%
	12	36	6.37	6.43	181.47	178.57	0.81%	-1.60%
4KB 70% Random Read 30% Random Write QD=64	1	6	1.49	1.57	260.53	243.67	4.79%	-6.47%
	2	12	2.60	2.77	292.21	276.91	6.35%	-5.24%
	4	24	4.15	4.43	374.13	344.98	6.83%	-7.79%
	6	36	5.52	5.83	415.27	394.37	5.55%	-5.03%
	8	36	6.03	6.27	383.12	365.91	4.05%	-4.49%
	10	36	6.18	6.36	374.96	363.20	2.89%	-3.14%
	12	36	6.08	6.12	376.44	377.04	0.60%	0.16%



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.
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 3% percent or slightly drops.
6. For some workloads there is a slight performance drop when Vhost is run with 10 and 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 7-8% 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.

Item	Description
Test case	Test rate limiting IOPS/VM to 10000 IOPS
Test configuration	<p>FIO Version: fio-3.19</p> <p>VM Configuration:</p> <ul style="list-style-type: none"> • 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. <ul style="list-style-type: none"> ○ 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) <p>SPDK Vhost target configuration:</p> <ul style="list-style-type: none"> • Test were run with both Vhost-scsi and Vhost-blk stacks. • The Vhost-scsi stack was run with Split fvNVMe bdevs and Logical Volume bdevs. • The Vhost-blk stack was run with Logical Volume bdevs. • Test were run with 4 CPU cores (NUMA optimized). <p>Kernel Vhost-scsi configuration:</p> <ul style="list-style-type: none"> • Cgroups were used to limit the Vhost process to 4 cores. • NUMA optimization were not explored.
FIO configuration run on each VM	<pre>[global] ioengine=libaio direct=1 rw=randrw rwmixread=100 (100% reads), 0 (100% writes) thread=1</pre>



	<pre>norandommap=1 time_based=1 runtime=300s ramp_time=10s bs=4k iodepth=1 numjobs=1</pre>
--	--

Test Case 2 Results

Table 7: 4KB 100% Random Reads QD=1

# of VMs	Stack	Backend bdev	IOPS (k)	Avg Lat. (usec)
24 VMs	SPDK-SCSI	Split NVMe	239.83	98.59
	SPDK-SCSI	Logical Volume	239.84	98.59
	SPDK-BLK	Logical Volume	239.84	98.60
	Kernel-SCSI	Partitioned NVMe	99.57	237.54
48 VMs	SPDK-SCSI	Split NVMe	477.26	98.58
	SPDK-SCSI	Logical Volume	476.18	98.79
	SPDK-BLK	Logical Volume	478.25	98.39
	Kernel-SCSI	Partitioned NVMe	105.74	460.26
72 VMs	SPDK-SCSI	Split NVMe	668.10	105.52
	SPDK-SCSI	Logical Volume	657.09	107.32
	SPDK-BLK	Logical Volume	676.01	104.32
	Kernel-SCSI	Partitioned NVMe	191.95	395.86

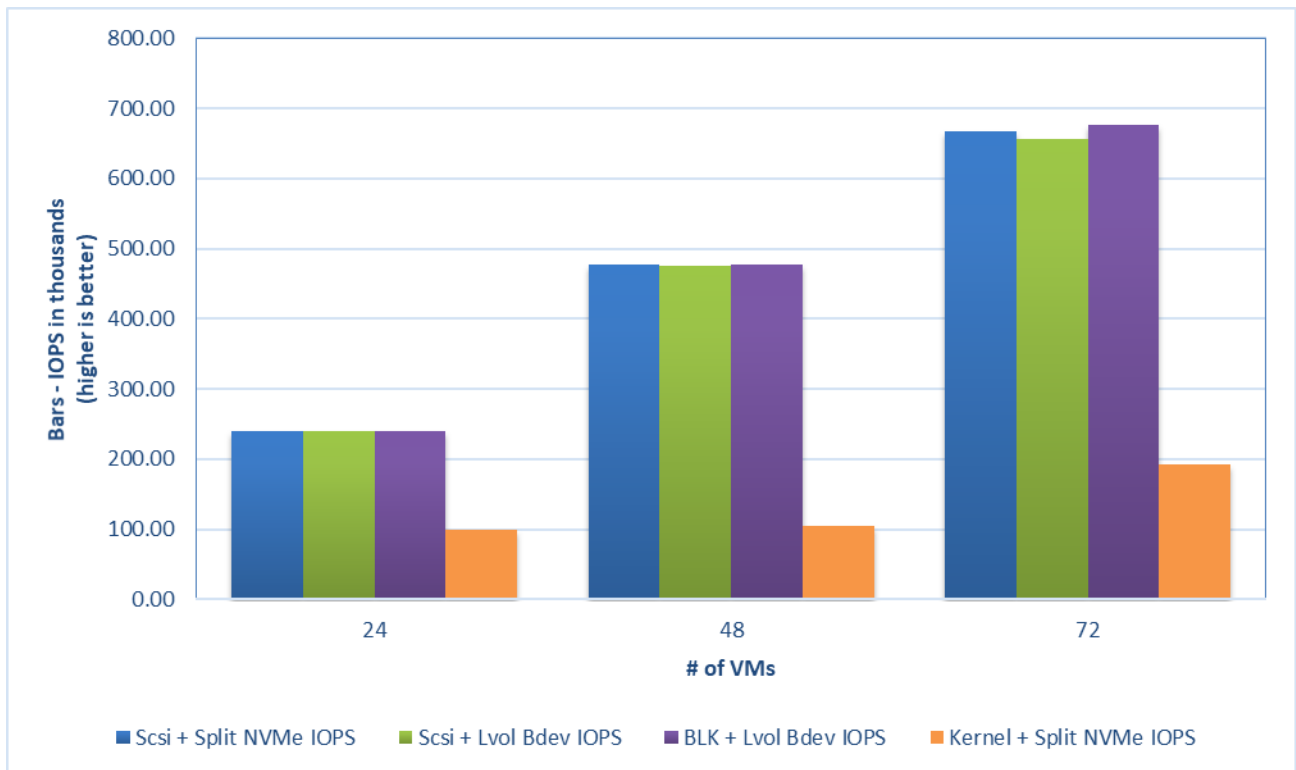


Figure 5: 4KB 100% Random Reads IOPS and latency, QD=1, throttling = 10k IOPS

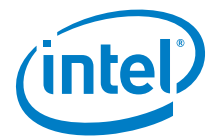


Table 8: 4KB 100% Random Writes QD=1

# of VMs	Stack	Backend bdev	IOPS (k)	Avg Lat. (usec)
24 VMs	SPDK-SCSI	Split NVMe	240.00	97.57
	SPDK-SCSI	Logical Volume	240.00	97.60
	SPDK-BLK	Logical Volume	240.00	97.55
	Kernel-SCSI	Partitioned NVMe	127.56	189.60
48 VMs	SPDK-SCSI	Split NVMe	479.97	97.55
	SPDK-SCSI	Logical Volume	479.96	97.50
	SPDK-BLK	Logical Volume	479.97	97.49
	Kernel-SCSI	Partitioned NVMe	149.90	319.88
72 VMs	SPDK-SCSI	Split NVMe	719.87	97.67
	SPDK-SCSI	Logical Volume	719.85	97.64
	SPDK-BLK	Logical Volume	719.87	97.59
	Kernel-SCSI	Partitioned NVMe	261.75	283.60

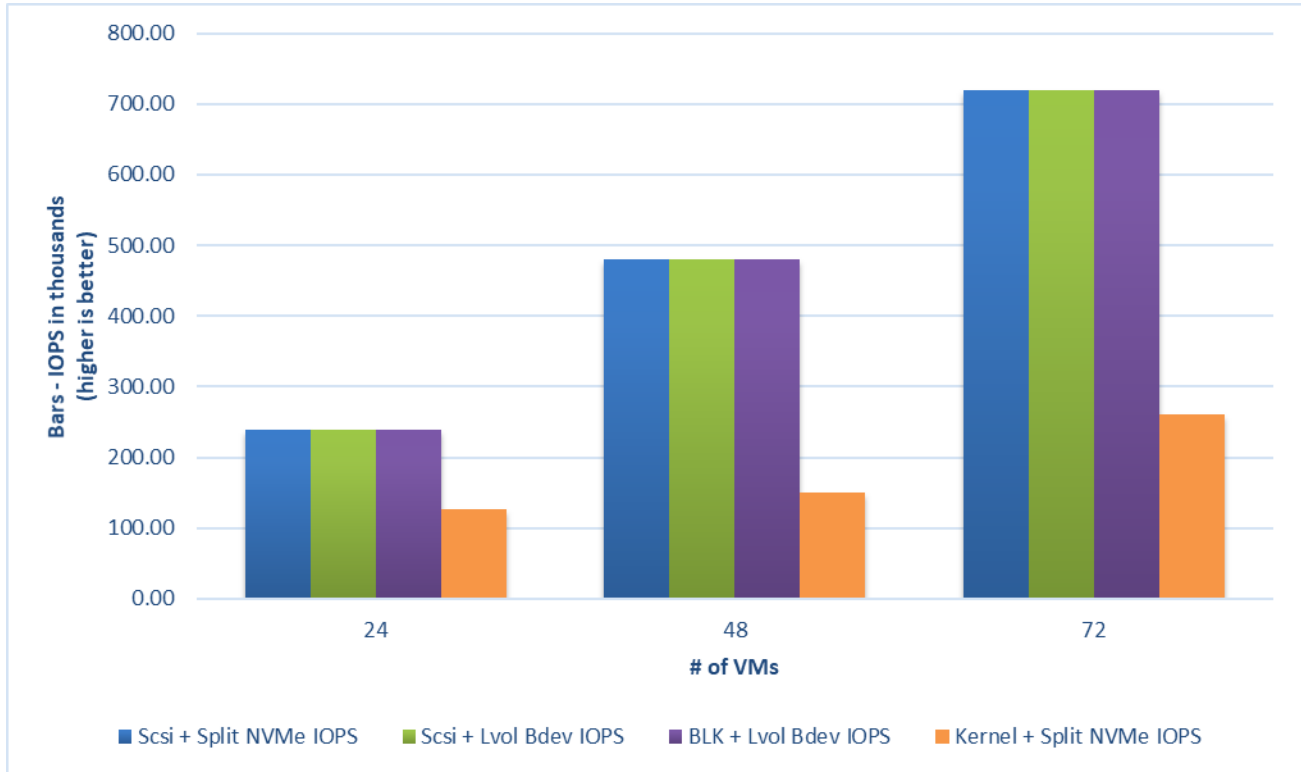


Figure 6: 4KB 100% Random Writes IOPS and latency, QD=1, throttling = 10k IOPS



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 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.

Item	Description
Test case	Test SPDK Vhost target I/O core scaling performance
Test configuration	<p>FIO Version: fio-3.19</p> <p>VM Configuration:</p> <ul style="list-style-type: none"> • 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. <p>SPDK Vhost target configuration:</p> <ul style="list-style-type: none"> • 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. <p>Kernel Vhost target configuration:</p> <ul style="list-style-type: none"> • The Vhost process was run on a single, physical CPU core using cgroups.
FIO configuration	<pre>[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</pre>



Test Case 3 results

SPDK Vhost-Scsi

Table 9: IOPS and latency results, SCSI stack

Access pattern	Backend	QD	Throughput (IOPS)	Avg. latency (usec)
4k 100% Random Reads	Split NVMe	1	23610.679	83.449
4k 100% Random Reads	Split NVMe	8	164796.466	95.987
4k 100% Random Reads	Split NVMe	32	388062.385	162.572
4k 100% Random Reads	Split NVMe	64	385620.001	331.421
4k 100% Random Reads	Lvol	1	23629.19	83.599
4k 100% Random Reads	Lvol	8	164234.68	96.022
4k 100% Random Reads	Lvol	32	391197.239	161.995
4k 100% Random Reads	Lvol	64	389014.245	327.32
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	30389.369	64.661
4k 70%/30% Random Read Writes	Split NVMe	8	166004.456	95.445
4k 70%/30% Random Read Writes	Split NVMe	32	323153.632	198.825
4k 70%/30% Random Read Writes	Split NVMe	64	364390.978	350.504
4k 70%/30% Random Read Writes	Lvol	1	30138.871	64.719
4k 70%/30% Random Read Writes	Lvol	8	176916.496	89.121
4k 70%/30% Random Read Writes	Lvol	32	315604.355	200.546
4k 70%/30% Random Read Writes	Lvol	64	358803.266	353.224



SPDK Vhost-Blk

Table 10: IOPS and latency results, BLK stack

Access pattern	Backend	QD	Throughput (IOPS)	Avg. latency (usec)
4k 100% Random Reads	Lvol	1	23913.744	82.239
4k 100% Random Reads	Lvol	8	165984.061	95.288
4k 100% Random Reads	Lvol	32	414282.141	152.578
4k 100% Random Reads	Lvol	64	423264.385	300.712
4k 100% Random Writes	Lvol	1	78618.926	24.876
4k 100% Random Writes	Lvol	8	403371.342	38.318
4k 100% Random Writes	Lvol	32	414658.988	153.038
4k 100% Random Writes	Lvol	64	413750.531	307.58
4k 70%/30% Random Read Writes	Lvol	1	30212.813	64.954
4k 70%/30% Random Read Writes	Lvol	8	151541.189	104.694
4k 70%/30% Random Read Writes	Lvol	32	301106.706	212.949
4k 70%/30% Random Read Writes	Lvol	64	374483.251	340.597

Kernel Vhost-Scsi

Table 11: IOPS and latency results, Kernel Vhost-Scsi

Access pattern	Backend	QD	Throughput (IOPS)	Avg. latency (usec)
4k 100% Random Reads	NVMe	1	16194.489	122.744
4k 100% Random Reads	NVMe	8	88445.692	179.925
4k 100% Random Reads	NVMe	32	221477.263	287.58
4k 100% Random Reads	NVMe	64	275451.706	463.395
4k 100% Random Writes	NVMe	1	50399.53	37.805
4k 100% Random Writes	NVMe	8	98640.016	160.834
4k 100% Random Writes	NVMe	32	197576.168	322.244
4k 100% Random Writes	NVMe	64	238646.543	538.48
4k 70%/30% Random Read Writes	NVMe	1	19382.23	102.451
4k 70%/30% Random Read Writes	NVMe	8	91713.191	173.427
4k 70%/30% Random Read Writes	NVMe	32	220109.778	290.798
4k 70%/30% Random Read Writes	NVMe	64	271502.886	471.138

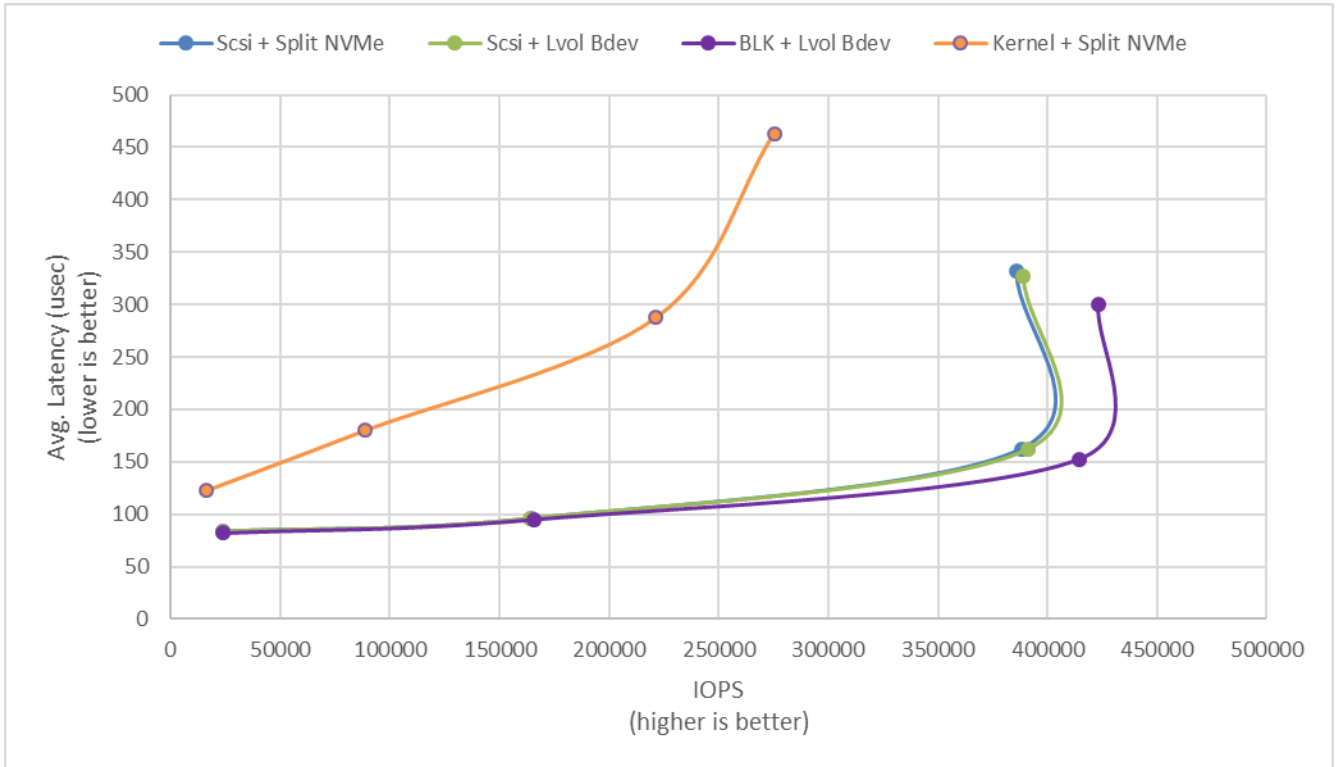


Figure 7: 4KB 100% Random Reads IOPS and latency

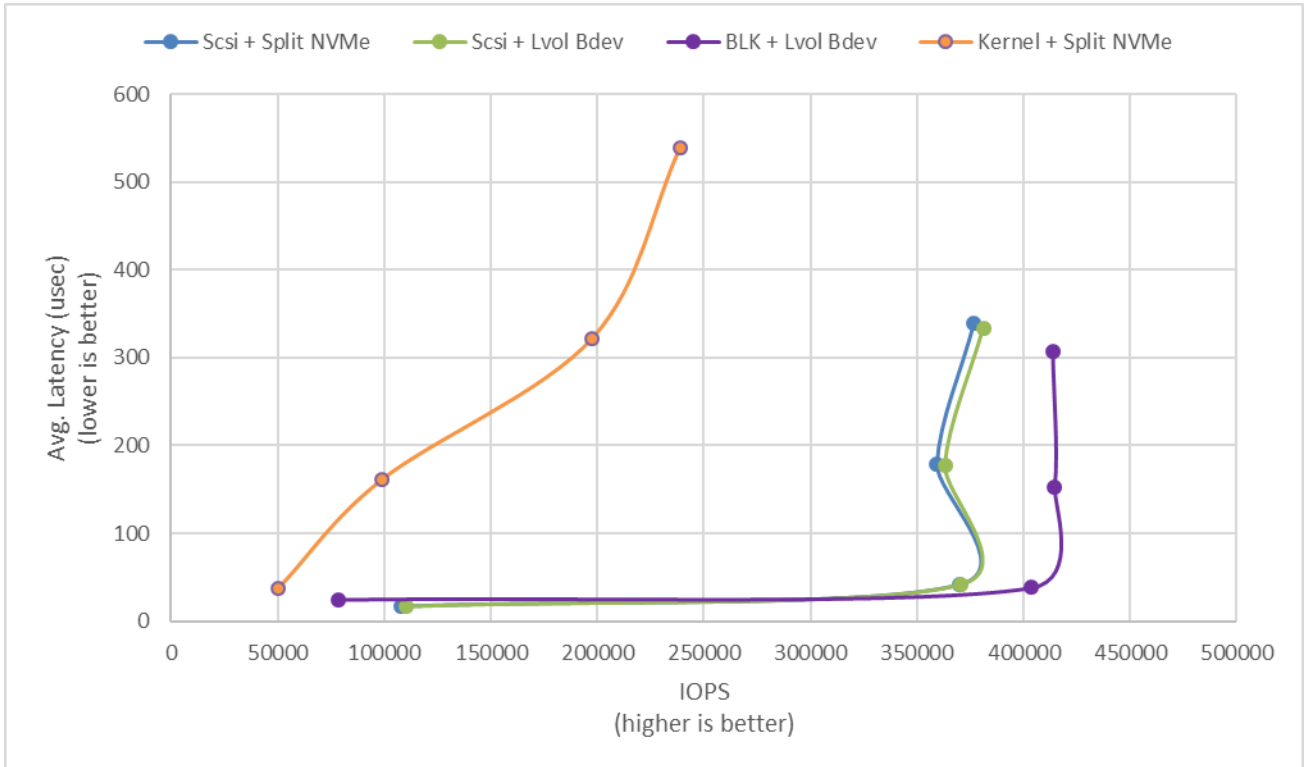


Figure 8: 4KB 100% Random Writes IOPS and latency

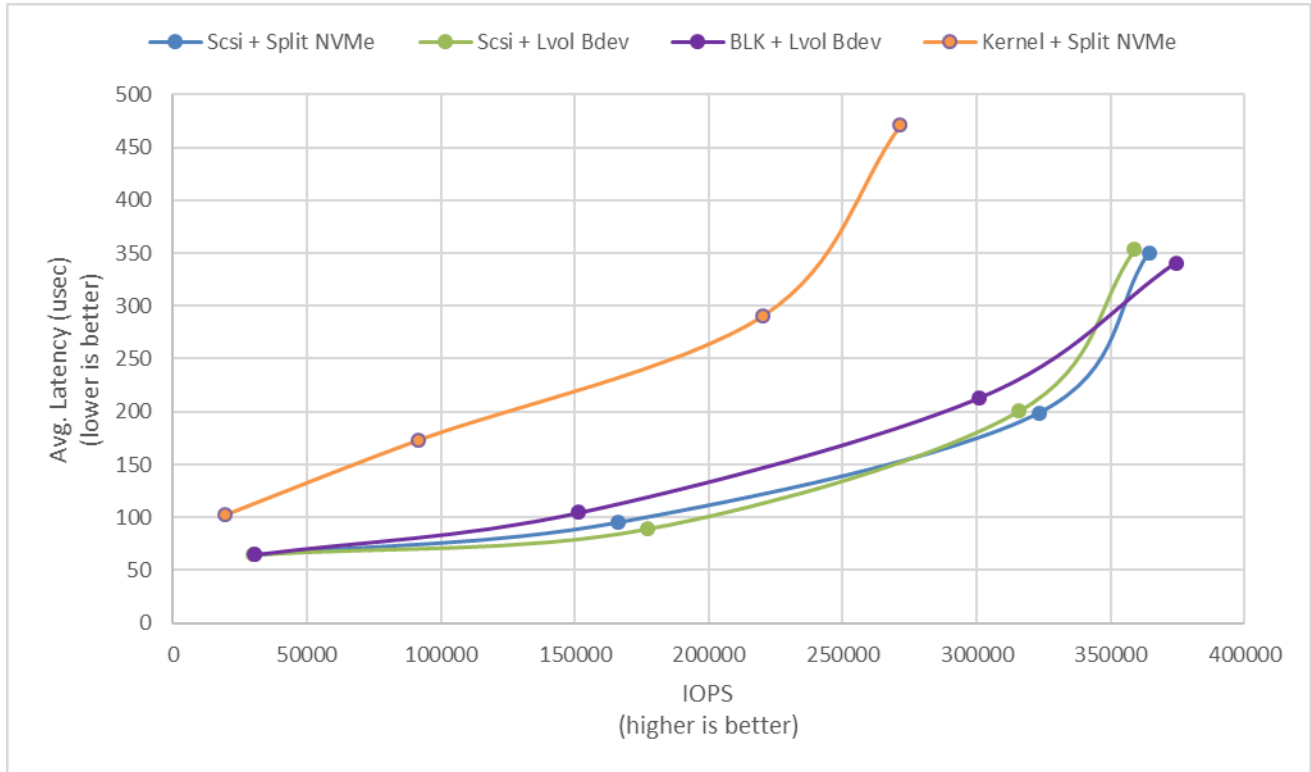
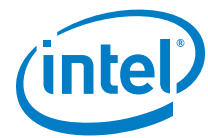


Figure 9: 4KB 70%/30% Random Read/Write IOPS and latency

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.



Appendix – Packed Ring Performance

Additional tests were performed as part of this report to check the performance impact of using Packed Rings in SPDK Vhost configuration instead of default Split Rings configuration. These tests were executed in a similar way as Test Case 1 (SPDK Vhost Core Scaling), but several changes had to be done in order to use Packed Rings:

- Qemu updated to version 4.2.0 (3.1.1 for other test cases in this report)
- SPDK Vhost controllers configured with appropriate “-p” option to enable Packed Rings
- Additional “packed=on” parameter must be passed as part of “-device vhost-user-blk-pci” definition in VM’s Qemu arguments

At the moment SPDK supports Packed Ring feature only for BLK controllers, so it was the only stack tested.

In general, using Packed Rings allowed for up to 10% performance increase.

Item	Description
Test configuration	<p>FIO Version: fio-3.19</p> <p>VM Configuration:</p> <ul style="list-style-type: none"> • Common settings are described in the Virtual Machine Settings chapter. • Qemu version 4.2.0 • 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 Split NVMe vbdev configuration. <p>SPDK Vhost target configuration:</p> <ul style="list-style-type: none"> • Vhost-blk stack with Split NVMe vbdevs used. • Vhost was run with 1,2,4,6,8,10 and 12 cores.
FIO configuration	<pre>[global] ioengine=libaio direct=1 thread=1 norandommap=1 time_based=1 gtod_reduce={0, 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}</pre>



Table 12: Packed Ring performance; SPDK Vhost BLK with Split NVMe bdevs

Workload	# of CPU cores	# of VMs	IOPS (millions) Split Ring	IOPS (millions) Packed Ring	IOPS Packed to Split Ring (%)	Avg. Latency (usec) Split Ring	Avg. Latency (usec) Packed Ring	Avg. Latency Packed to Split Ring (%)
4KB 100% Random Read, QD=64	1	6	1.86	1.98	6.56%	205.97	193.18	-6.21%
	2	12	3.21	3.44	7.34%	238.75	222.45	-6.83%
	4	24	5.24	5.60	6.74%	294.05	273.57	-6.96%
	6	36	6.92	7.32	5.81%	332.21	314.70	-5.27%
	8	36	6.85	7.49	9.41%	336.65	307.82	-8.56%
	10	36	6.97	7.11	1.93%	330.04	323.57	-1.96%
	12	36	6.44	6.48	0.74%	357.08	359.88	0.78%
4KB 100% Random Write, QD=32	1	6	1.61	1.65	2.34%	122.82	120.23	-2.11%
	2	12	3.17	3.30	4.08%	120.50	115.25	-4.36%
	4	24	5.33	5.58	4.73%	143.91	137.51	-4.45%
	6	36	6.94	7.11	2.46%	165.50	161.55	-2.39%
	8	36	6.92	7.10	2.48%	165.22	170.13	2.97%
	10	36	5.64	5.72	1.43%	206.64	207.15	0.25%
	12	36	6.34	6.34	0.00%	180.92	181.40	0.27%
4KB 70%/30% Random Read/Write, QD=64	1	6	1.78	1.78	0.04%	215.62	214.73	2.72%
	2	12	3.04	3.21	5.51%	251.87	239.14	-2.66%
	4	24	5.01	5.30	5.88%	306.83	288.76	-3.71%
	6	36	6.58	6.81	3.50%	348.84	337.69	-1.66%
	8	36	6.33	6.64	5.01%	362.49	344.97	-3.50%
	10	36	6.04	6.31	4.43%	380.86	365.25	-2.50%
	12	36	6.04	6.03	-0.14%	381.31	381.47	0.29%

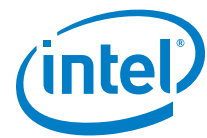


Table 13: Packed Ring performance; SPDK Vhost BLK with Split NVMe bdevs; fio gtod_reduce=0

Workload	# of CPU cores	# of VMs	IOPS (millions) Split Ring	IOPS (millions) Packed Ring	IOPS Packed to Split Ring (%)
4KB 100% Random Read, QD=64	1	6	1.86	1.99	6.92%
	2	12	3.22	3.44	6.73%
	4	24	5.46	5.73	4.97%
	6	36	7.05	7.41	5.03%
	8	36	7.54	7.80	3.44%
	10	36	7.89	8.15	3.33%
	12	36	7.35	7.62	3.66%
4KB 100% Random Write, QD=32	1	6	1.62	1.68	4.14%
	2	12	3.27	3.45	5.53%
	4	24	5.54	5.65	2.07%
	6	36	7.31	7.37	0.84%
	8	36	7.12	7.32	2.86%
	10	36	6.24	6.30	0.88%
	12	36	7.10	7.23	1.87%
4KB 70%/30% Random Read/Write, QD=64	1	6	1.69	1.73	2.01%
	2	12	3.05	3.16	3.72%
	4	24	5.23	5.45	4.07%
	6	36	6.80	7.00	2.89%
	8	36	6.73	6.80	0.98%
	10	36	6.69	6.78	1.27%
	12	36	6.78	6.92	1.98%



DISCLAIMERS

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors.

Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information go to <http://www.intel.com/performance>

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

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

Your costs and results may vary.

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

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.