THE LINUX FOUNDATION
OPEN SOURCE SUMMIT

Accelerating NVMe I/Os in Virtual Machine via SPDK vhost* Solution

Ziye Yang, Changpeng Liu
Senior software Engineer
Intel

@optimistyzy

THE LINUX FOUNDATION

Notices & Disclaimers

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration.

No computer system can be absolutely secure.

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. For more complete information about performance and benchmark results, visit http://www.intel.com/benchmarks .

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 visit http://www.intel.com/benchmarks .

Benchmark results were obtained prior to implementation of recent software patches and firmware updates intended to address exploits referred to as "Spectre" and "Meltdown." Implementation of these updates may make these results inapplicable to your device or system.

Intel® Advanced Vector Extensions (Intel® AVX)* provides higher throughput to certain processor operations. Due to varying processor power characteristics, utilizing AVX instructions may cause a) some parts to operate at less than the rated frequency and b) some parts with Intel® Turbo Boost Technology 2.0 to not achieve any or maximum turbo frequencies. Performance varies depending on hardware, software, and system configuration and you can learn more at http://www.intel.com/go/turbo.

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.

Cost reduction scenarios described are intended as examples of how a given Intel-based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does not guarantee any costs or cost reduction.

Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

THE LINUX FOUNDATION

Outline

- Background
- SPDK vhost solution
- Experiments
- Conclusion

THE LINUX FOUNDATION

Background

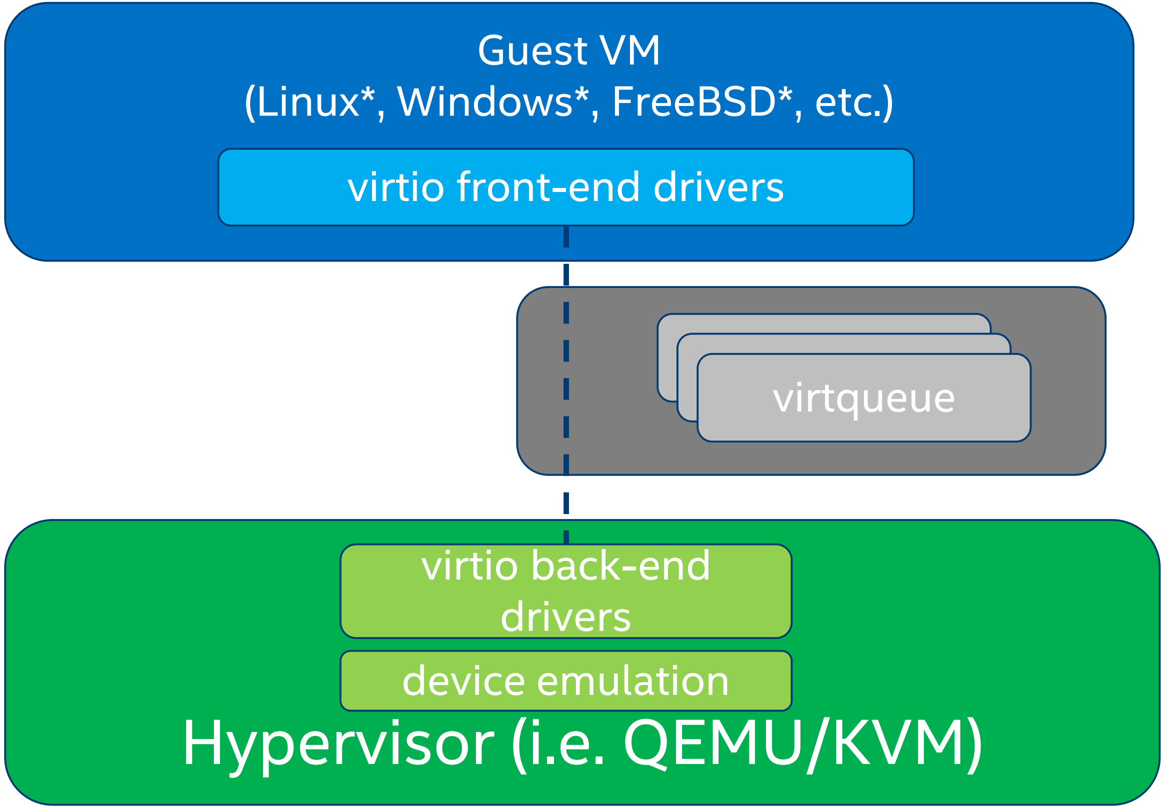
NVMe & virtualization

- NVMe specification enables highly optimized drives (e.g., NVMe SSD)
  - For example, multiple I/O queues allows lockless submission from CPU cores in parallel
- However, even the best kernel mode drivers have non-trivial software overhead
  - Long I/O stack in kernel with resource contention
- Virtualization adds additional overhead
  - Long I/O stack in both guest OS kernel and host OS kernel
  - Context switch overhead (e.g., VM_EXIT caused by I/O interrupt in guest OS)

What is in QEMU's solution?

- The solution in QEMU to virtualize NVMe device:
  - Virtio virtualization
  - NVMe controller virtualization
  - Hardware assisted virtualization
- Virtio virtualization
  - Virtio SCSI/block Controllers
- NVMe controller virtualization
  - QEMU emulated NVMe Device (file based NVMe backend)
  - QEMU NVMe Block Driver based on VFIO (exclusive access by QEMU)

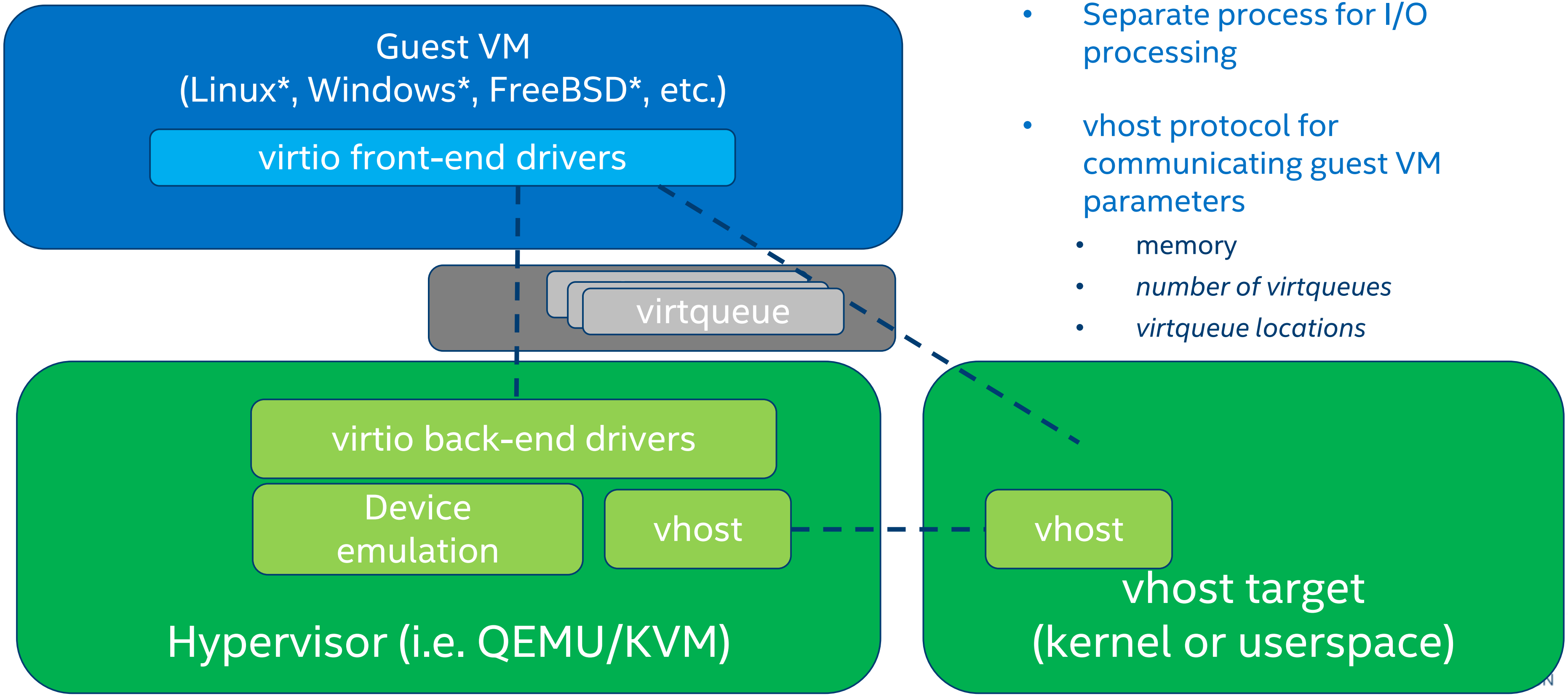
What is in QEMU’s solution?



- Paravirtualized driver specification
- Common mechanisms and layouts for device discovery, I/O queues, etc.
- virtio device types include:
  - virtio-net
  - virtio-blk
  - virtio-scsi
  - virtio-gpu
  - virtio-rng
  - virtio-crypto

THE LINUX FOUNDATION

Accelerate virtio via vhost target



- Separate process for I/O processing
- vhost protocol for communicating guest VM parameters
  - memory
  - *number of virtqueues*
  - *virtqueue locations*

SPDK vhost solution

What is SPDK?

Storage Performance Development Kit

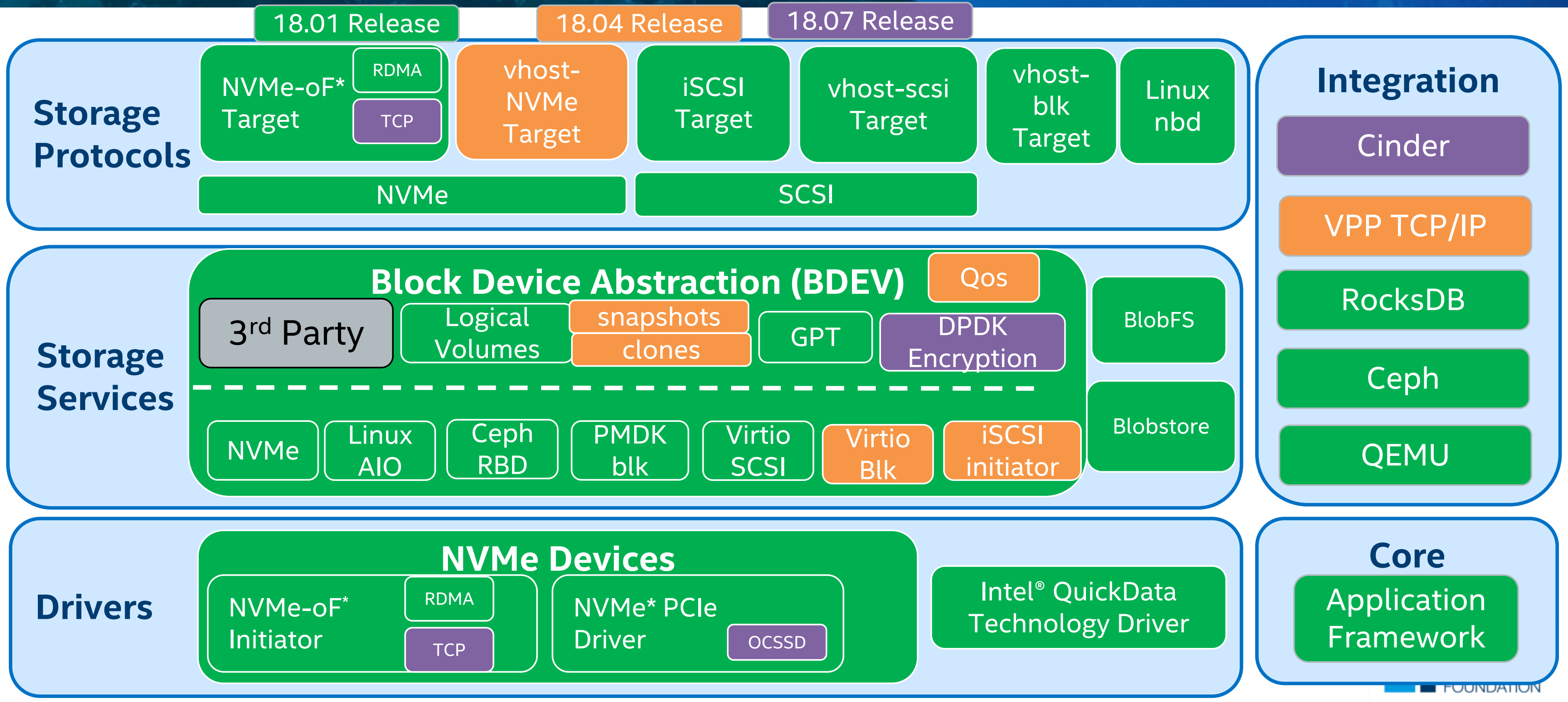
Intel® Platform Storage Reference Architecture

- Optimized for *Intel platform* characteristics
- Open source building blocks (BSD licensed)
- Available via github.com/spdk or spdk.io

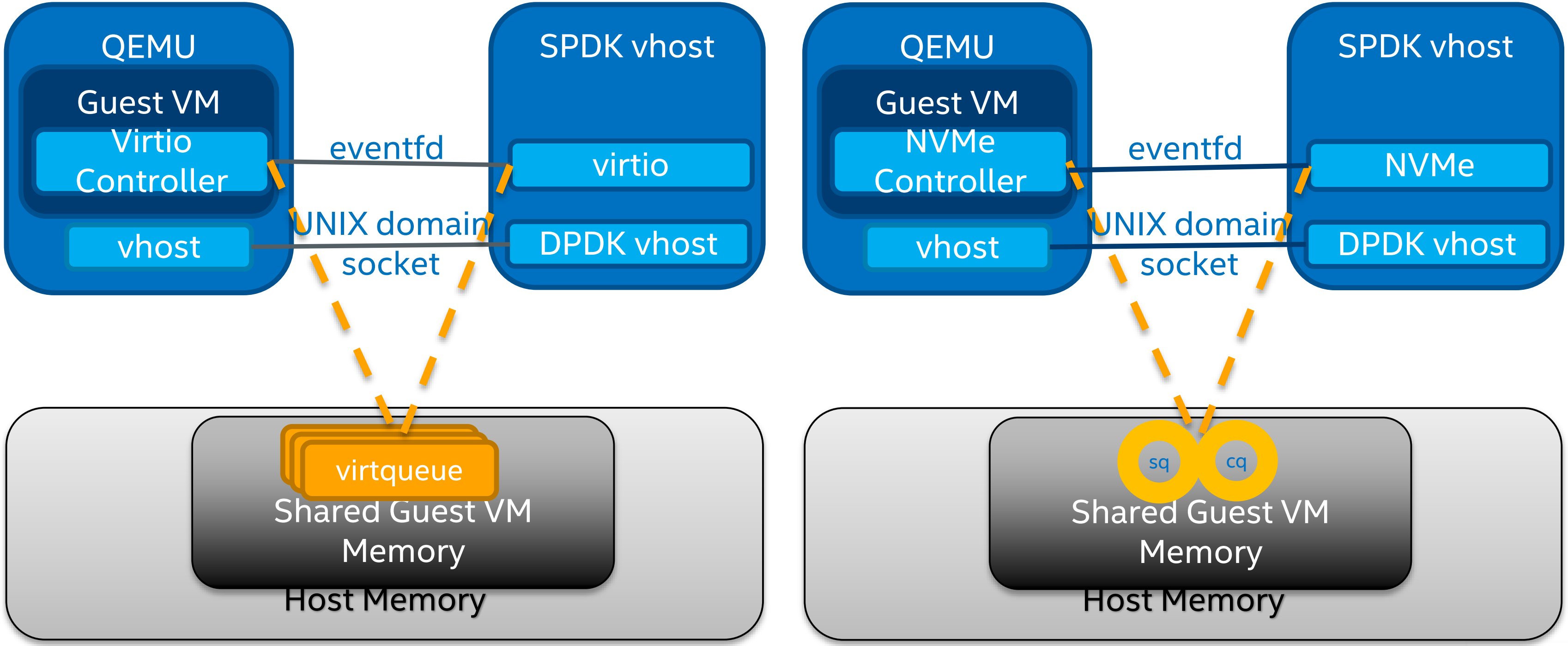
Scalable and Efficient Software Ingredients

- User space, lockless, polled-mode components
- Up to millions of IOPS per core
- Designed for Intel Optane™ technology latencies

SPDK architecture

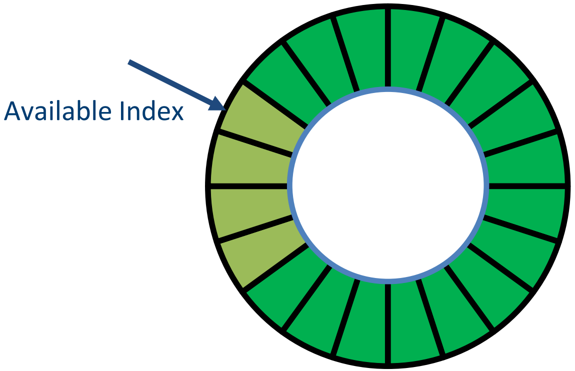


Combine virtio and NVMe to inform a uniform SPDK vhost solution



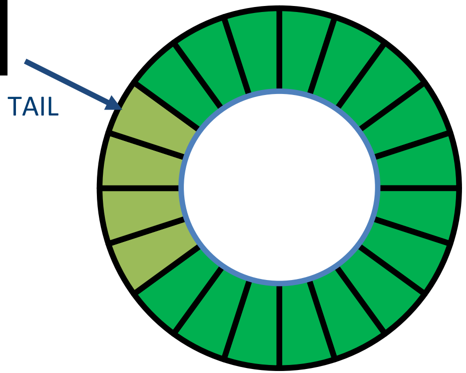
Virtio VS NVMe

Available Ring

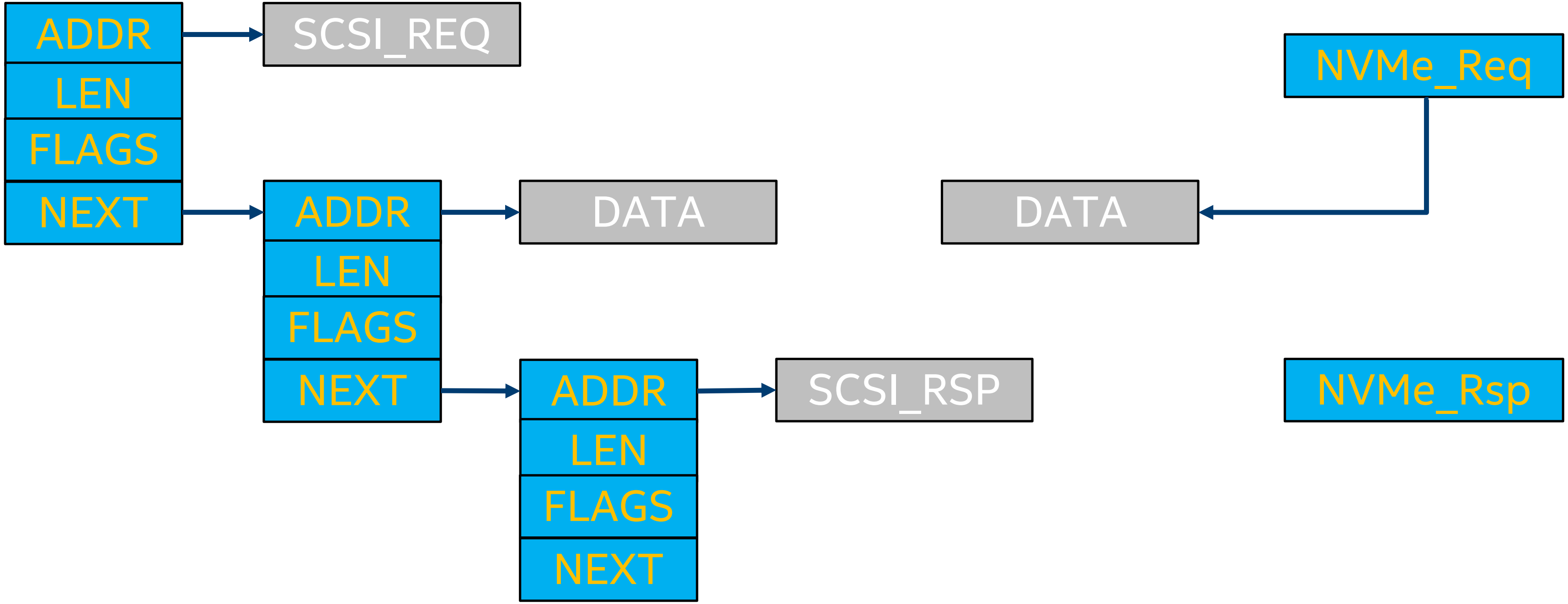


Both Use Ring Data Structures for IO

Submission Queue



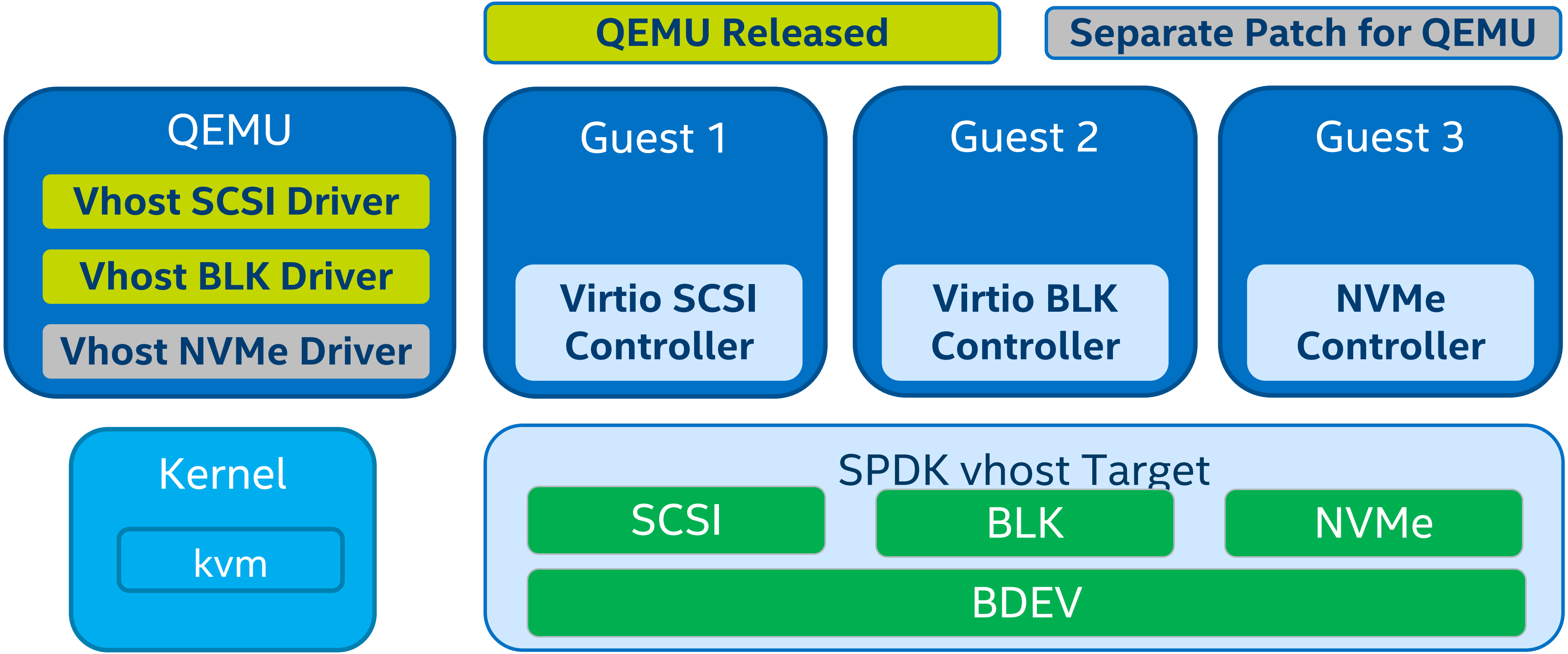
Virtio-SCSI and NVMe protocol format comparison



(16 * 3 + SCSI_Req + SCSI_Rsp + Data) Bytes

(NVMe_Req + Data + NVMe_Rsp) Bytes

SPDK vhost architecture

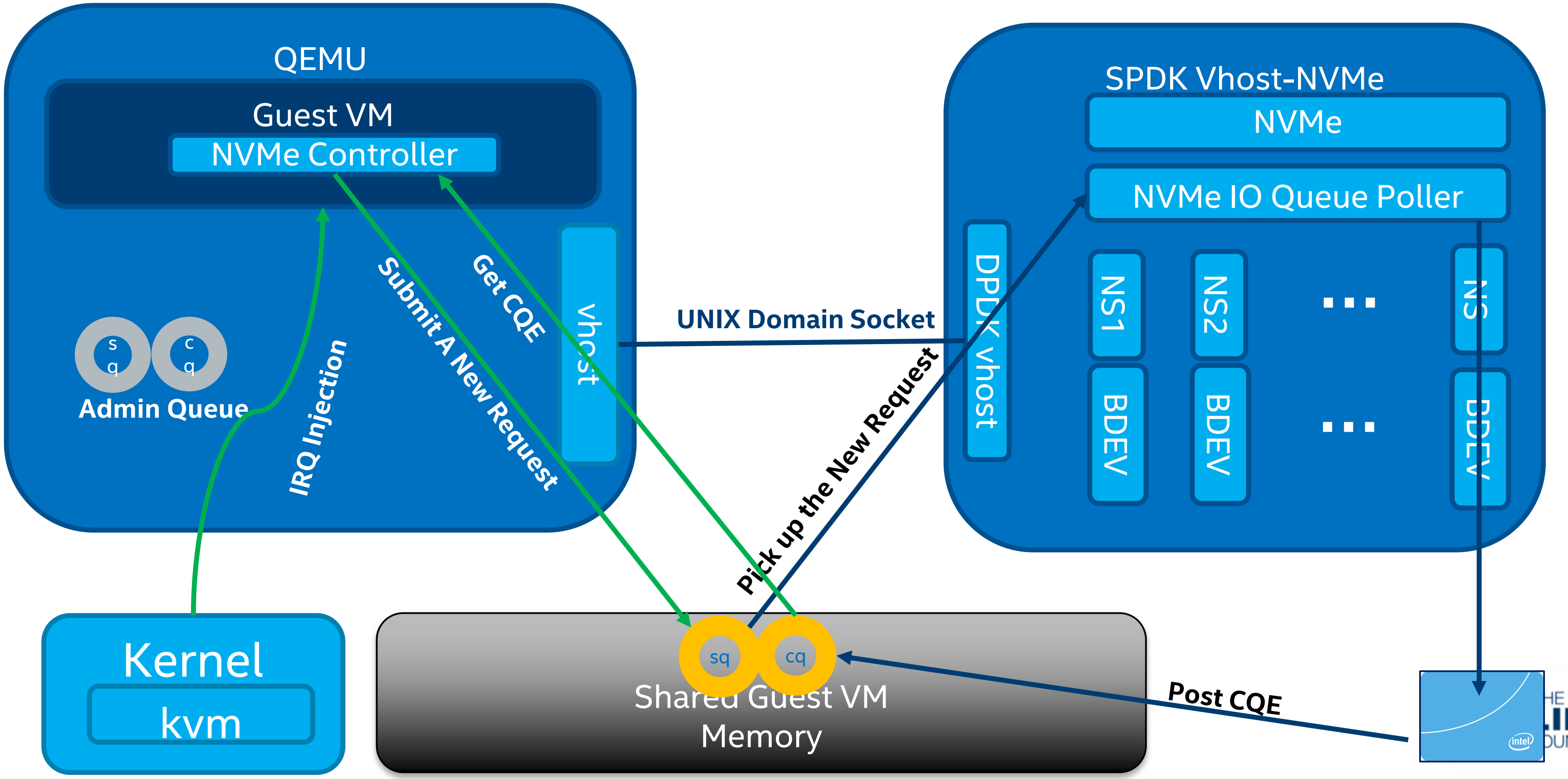


Comparison of known solutions

Solution / Usage	QEMU Emulated NVMe device	QEMU VFIO Based solution	QEMU PCI-Passthrough SR-IOV	SPDK Vhost-SCSI	SPDK Vhost-BLK	SPDK Vhost-NVMe	Mediated-NVMe VFIO
Guest OS Interface	NVMe	NVMe	NVMe	Virtio SCSI	Virtio BLK	NVMe	NVMe
Backend Device sharing	Y	N	N	Y	Y	Y	*Depends on implementation *
Live Migration support	N	N	N	Y	Y	No, Feature is WIP	*Depends on implementation *
VFIO dependency	N	Y	N	Can use UIO or VFIO	Can use UIO or VFIO	Can use UIO or VFIO	Y
QEMU Support	Y	Y	Y	Y	Y	No, Upstream is WIP	Y

SPDK vhost NVMe implementation details

Vhost NVMe implementation details



Create io queue

Guest: Create IO Queue

QSIZE	QID	CQID	QPRIO	PC
PRP1				

Guest: Submit to Admin, Write DB

QEMU: Pick up Admin Command

QEMU: Send via Domain Socket

SPDK: Start to Create IO Queue

SPDK: Memory Translation

SPDK: Both Guest and SPDK see same IO Queue now

sq

THE LINUX FOUNDATION

New feature to address guest NVMe performance issue

MMIO Writes happened, which will cause VM_EXIT

NVMe 1.3 New Feature: Optional Admin Command support for Doorbell Buffer Config, only used for emulated NVMe controllers, Guest can update shadow doorbell buffer instead of submission queue's doorbell registers

Shadow doorbell buffer

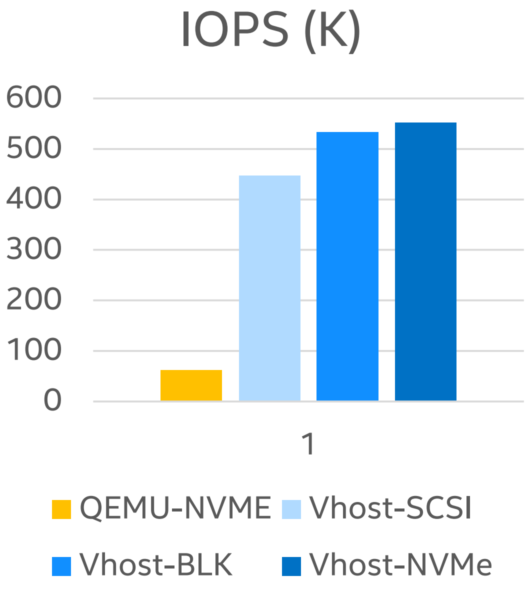
Start	End	Description
00h	03h	Submission Queue 0 Tail Doorbell or Eventidx (Admin)
04h	07h	Completion Queue 0 Head Doorbell or Eventidx (Admin)
08h	0Bh	Submission Queue 1 Tail Doorbell or Eventidx
0Ch	0Fh	Completion Queue 1 Head Doorbell or Eventidx

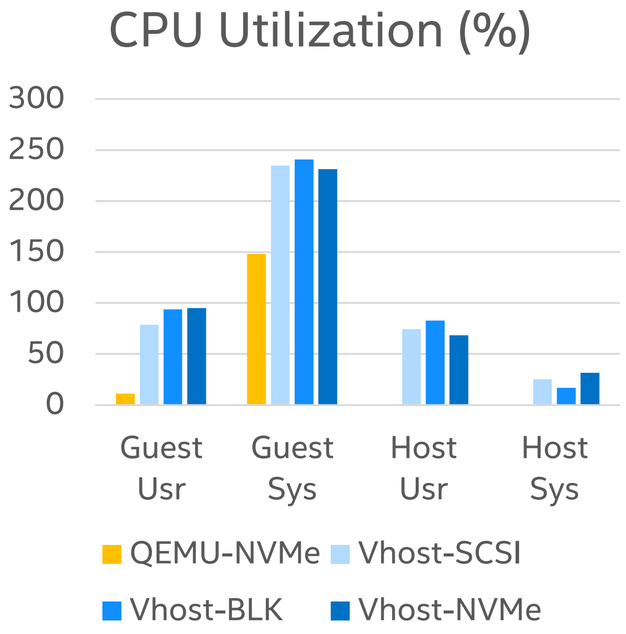
Command	Description
PRP1	Shadow doorbell memory address, updated by Guest NVMe Driver
PRP2	Eventidx memory address, updated by SPDK vhost target

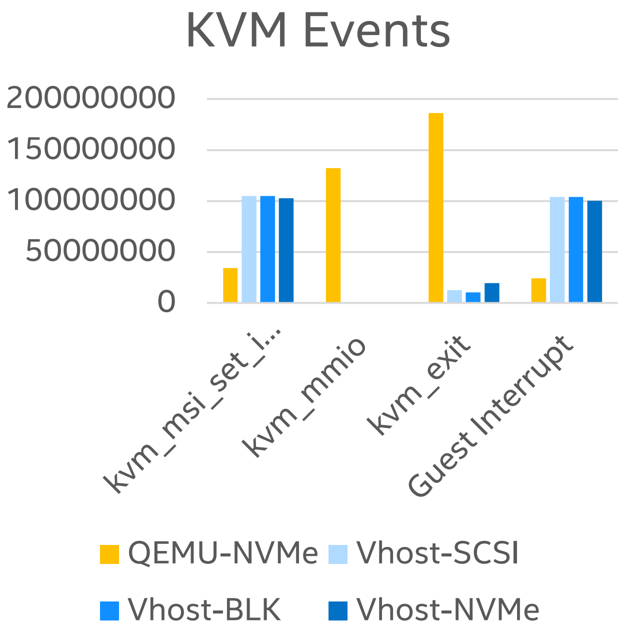
THE LINUX FOUNDATION

Experiments

1 VM with 1 NVMe SSD

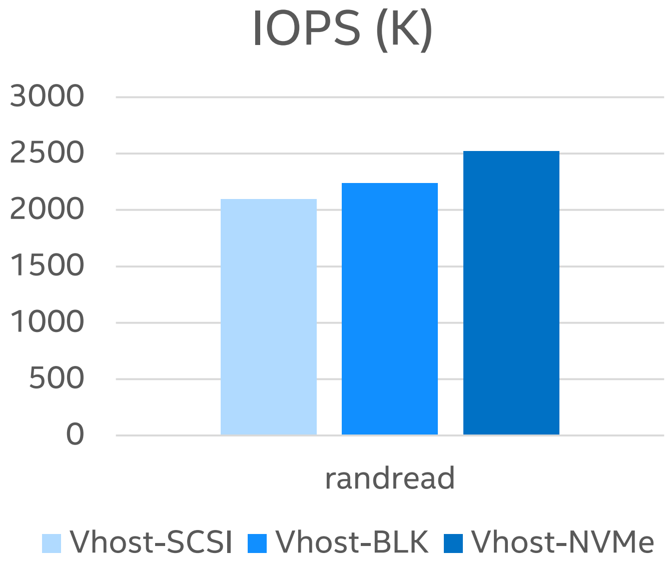


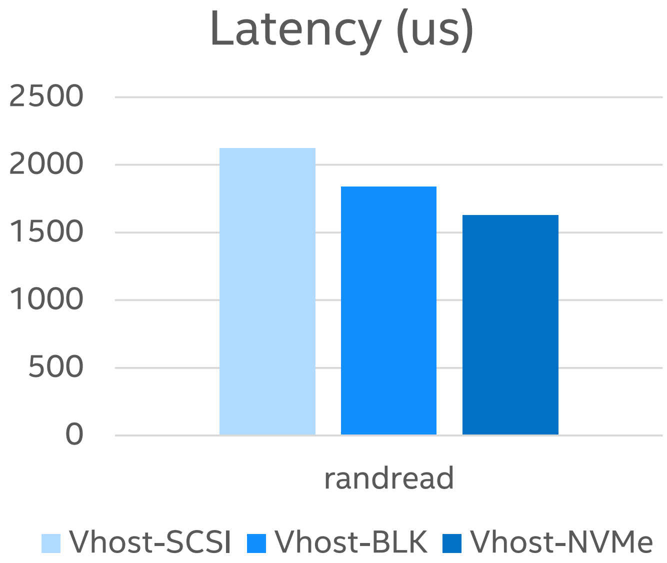




System Configuration: 2 * Intel Xeon E5 2699v4 @ 2.2GHz; 128GB, 2667 DDR4, 6 memory Channels; SSD: Intel Optane™ P4800X, FW: E2010324, 375GiB; Bios: HT disabled, Turbo disabled; OS: Fedora 25, kernel 4.16.0. 1 VM, VM config : 4 vcpu 4GB memory, 4 IO queues; VM OS: Fedora 27, kernel 4.16.5-200, blk-mq enabled; Software: QEMU-2.12.0 with SPDK Vhost-NVMe driver patch, IO distribution: 1 vhost-cores for SPDK, FIO 3.3, io depth=32, numjobs=4, direct=1, block size=4k,total tested data size=400GiB

8 VMs with 4 NVMe SSDs





- Linux kernel NVMe driver will poll completion queue when submitting a new request, which can help to decrease interrupt numbers and vm_exit events.

System Configuration: 2 * Intel Xeon E5 2699v4 @ 2.2GHz; 256GB, 2667 DDR4, 6 memory Channels; SSD: Intel DC P4510, FW: VDV10110, 2TiB; BIOS: HT disabled, Turbo disabled; Host OS: CentOS 7, kernel 4.16.7. 8 VMs, VM config : 4 vcpu 4GB memory, 4 IO queues; Guest OS: Fedora 27, kernel 4.16.5-200, blk-mq enabled; Software: QEMU-2.12.0 with SPDK Vhost-NVMe driver patch, IO distribution: 2 vhost-cores for SPDK, FIO 3.3, io depth=128, numjobs=4, direct=1, block size=4k,runtime=300s,ramp_time=60s; SSDs well preconditioned with 2 hours randwrites before randread tests.

Conclusion

Conclusion

- Conclusion
  - In this presentation, we introduce SPDK vhost solution(i.e., SCSI/Blk/NVMe) to accelerate NVMe I/Os in virtual machines
- Future work
  - VM live migration support for the whole SPDK vhost solution(i.e., vhost SCSI/BLK/NVMe)
  - Upstream QEMU vhost driver.
- Promotion
  - Welcome to evaluate & use SPDK vhost target !
  - Welcome to contribute to SPDK community !

Q & A

THE LINUX FOUNDATION

OPEN SOURCE SUMMIT