SPDK vhost target: A practical solution to accelerate storage IOs inside VMs

Changpeng Liu, Ziye Yang

Cloud Storage Software Engineer
Intel Data Center Group

Notices and 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.

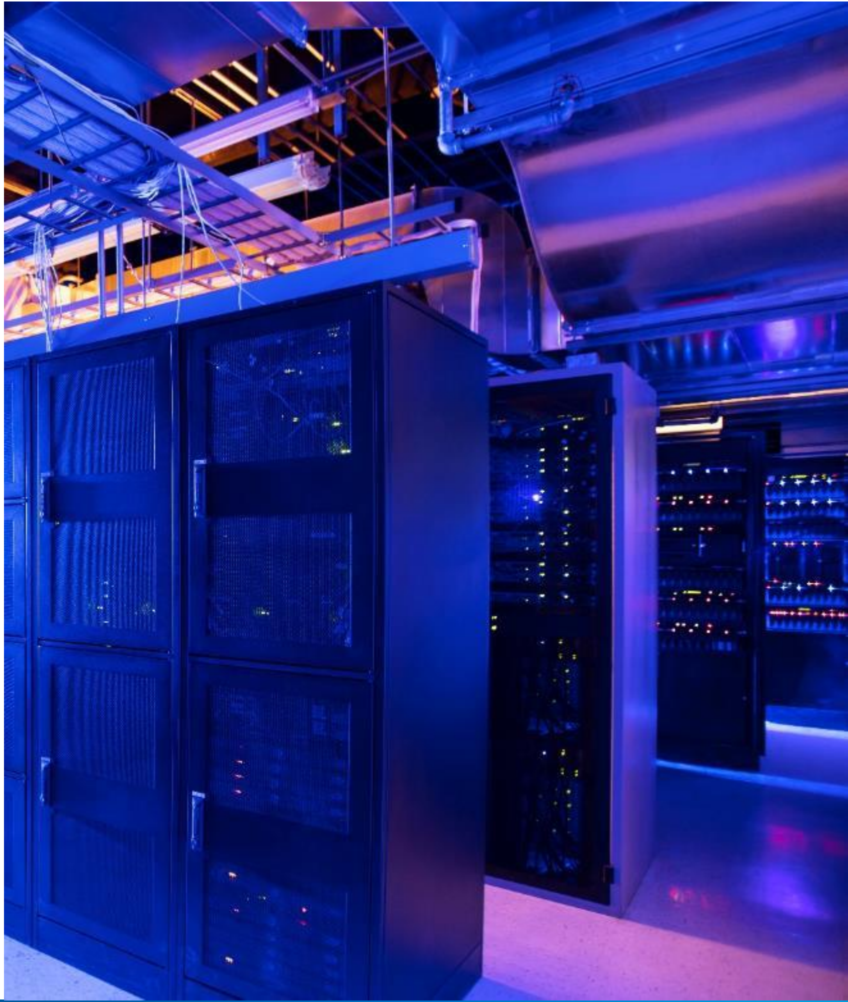
Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, 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.

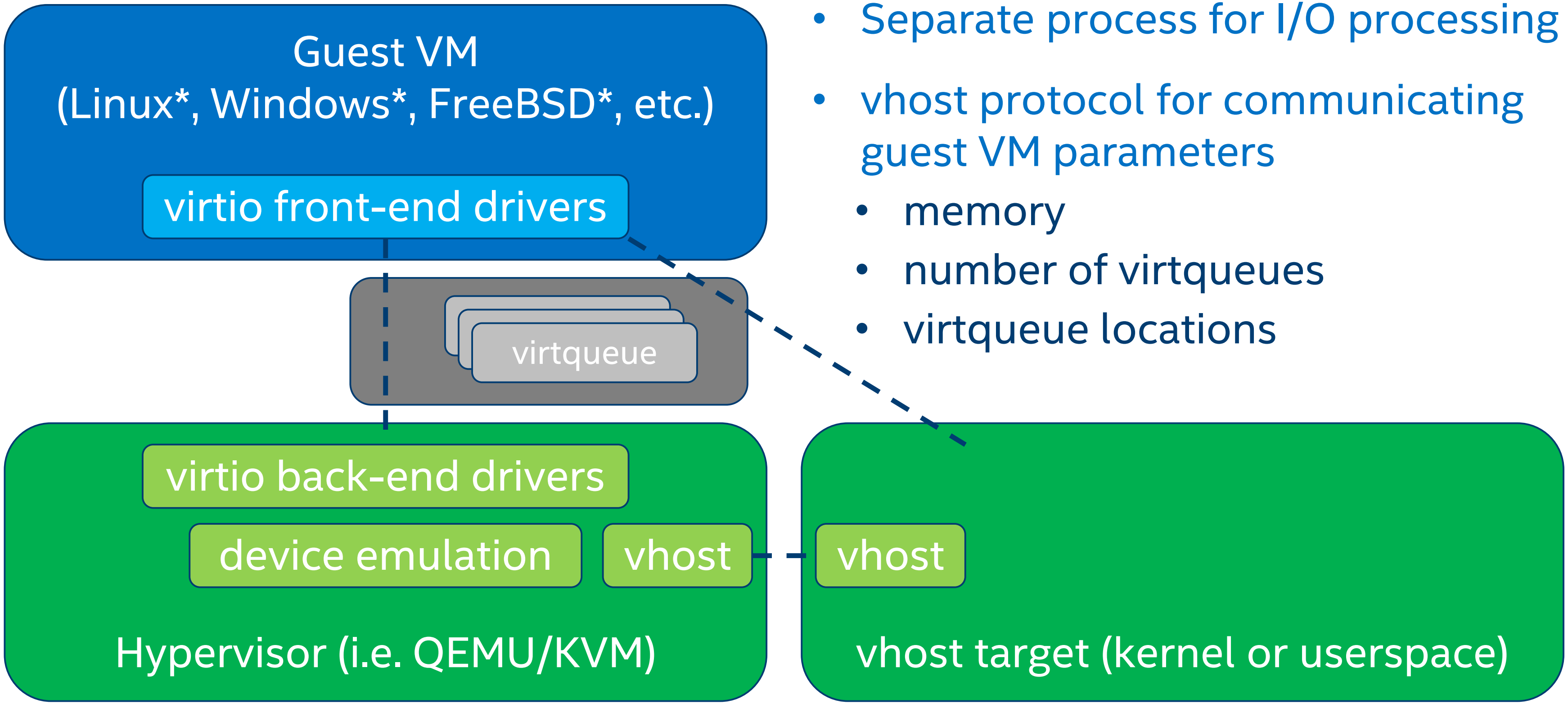
Agenda

- Introduction
- Implementation Details
- Benchmarks
- Future work



Introduction

Accelerate virtio with vhost target



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

Storage Performance Development Kit

Available via spdk.io

@SPDKProject

Scalable and Efficient Software Ingredients

- User space, lockless, polled-mode
- Up to millions of IOPS per core
- Minimize average and tail latencies
- Designed for non-volatile media

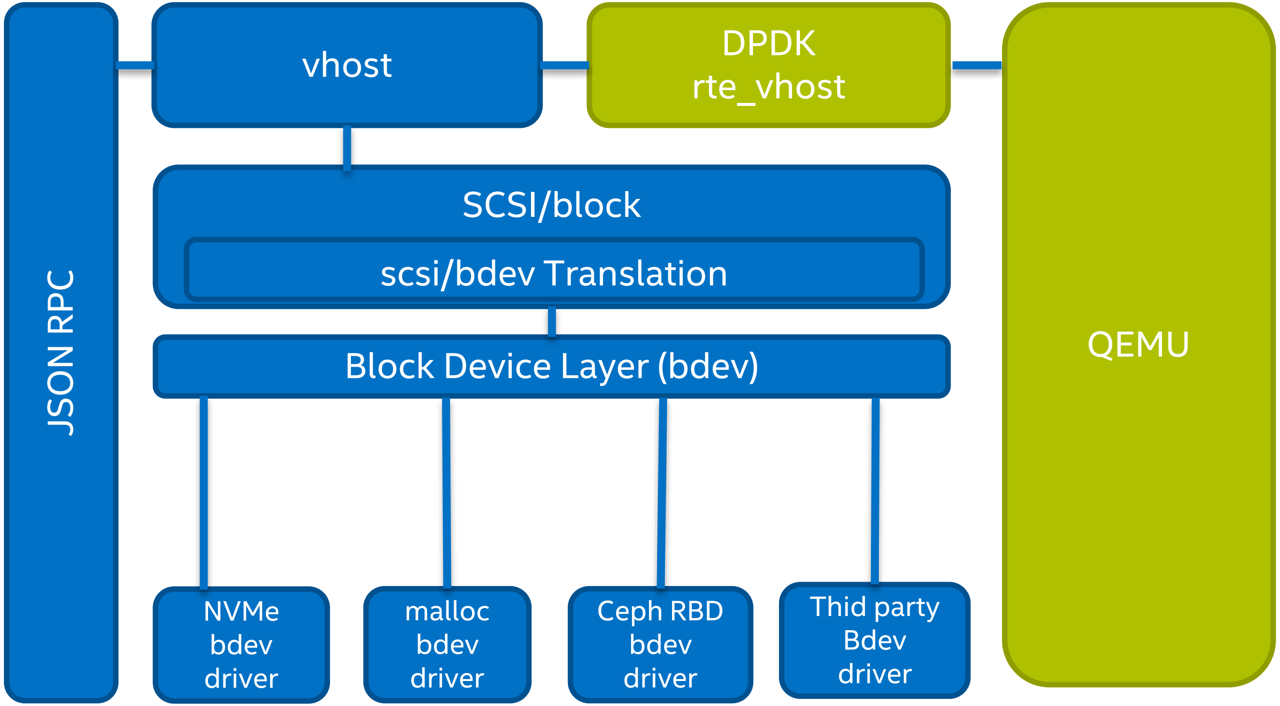
Storage Reference Software

- Optimized for latest generation CPUs and SSDs
- Provides Future Proofing
- Extends to Storage Virtualization and Networking

Open Source community

- Open source building blocks (BSD licensed)
- Faster TTM, fewer resources required

SPDK vhost target for accelerating virtio SCSI/BLK



WILL SPDK VHOST FOR SCSI/BLK BE ENOUGH?

Non-Volatile Memory Express

NVMe protocol

- Parallel and high performance interface designed for non-volatile memory based backend
- Admin commands with Admin queue, slow path
- I/O commands with I/O queues, fast path
- Multiple submission queues and completion queues
- No SCSI middle layer involved in IO submission path compared with SCSI interface, which can decrease latency for each IO submission

Block devices interface used in Guest VM

- Virtio SCSI/block Controllers
- NVMe Controllers

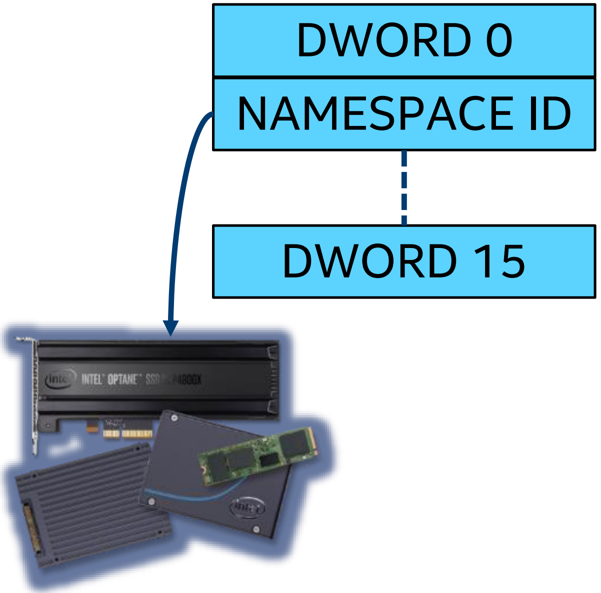
Comparison of Several Known Solutions

Solution / Usage	SPDK Vhost-SCSI	SPDK Vhost-BLK	SPDK Vhost-NVMe	QEMU Emulated NVMe	QEMU VFIO based NVMe	QEMU PCI-Passthrough	Mediated-NVMe VFIO	Scalable I/O Virtualization for NVMe
Guest OS Interface	VIRITO-SCSI	VIRTIO-BLK	NVMe	NVMe	NVMe	NVMe	NVMe	NVMe
Backend Device sharing	Y	Y	Y	Y	N	N	(*)	(*)
Live Migration support	Y	Y	Y	N	N	N	(*)	(*)
QEMU Support	Y	Y	N	Y	Y	Y	Y	(*)
NVMe Hardware Required	N	N	N	N	Y	Y	Y	Y

(*) - the features can be supported or depend on future detailed implementation

Issues for hardware assistant solutions

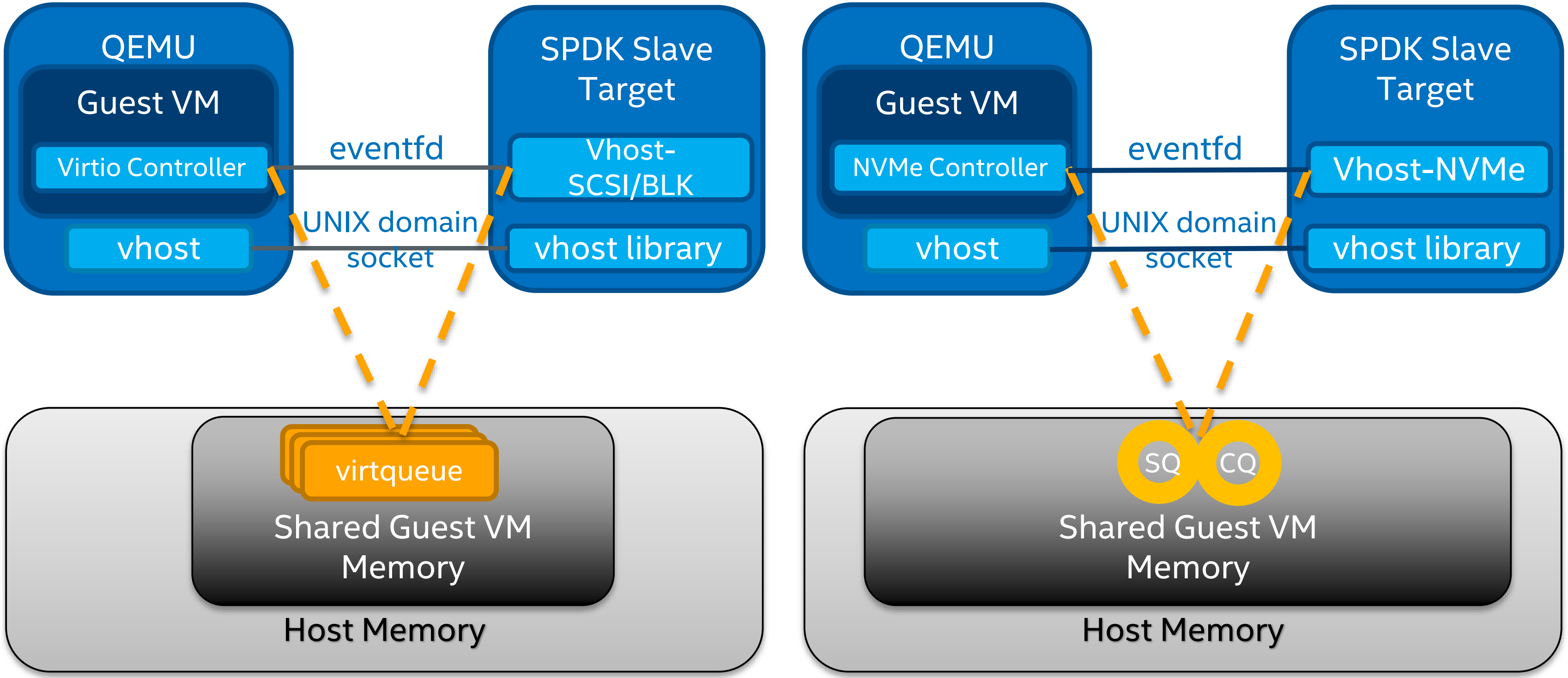
IO Submission Queue 1 Entry



Same Namespace ID can be used at any NVMe IO Submission Queues

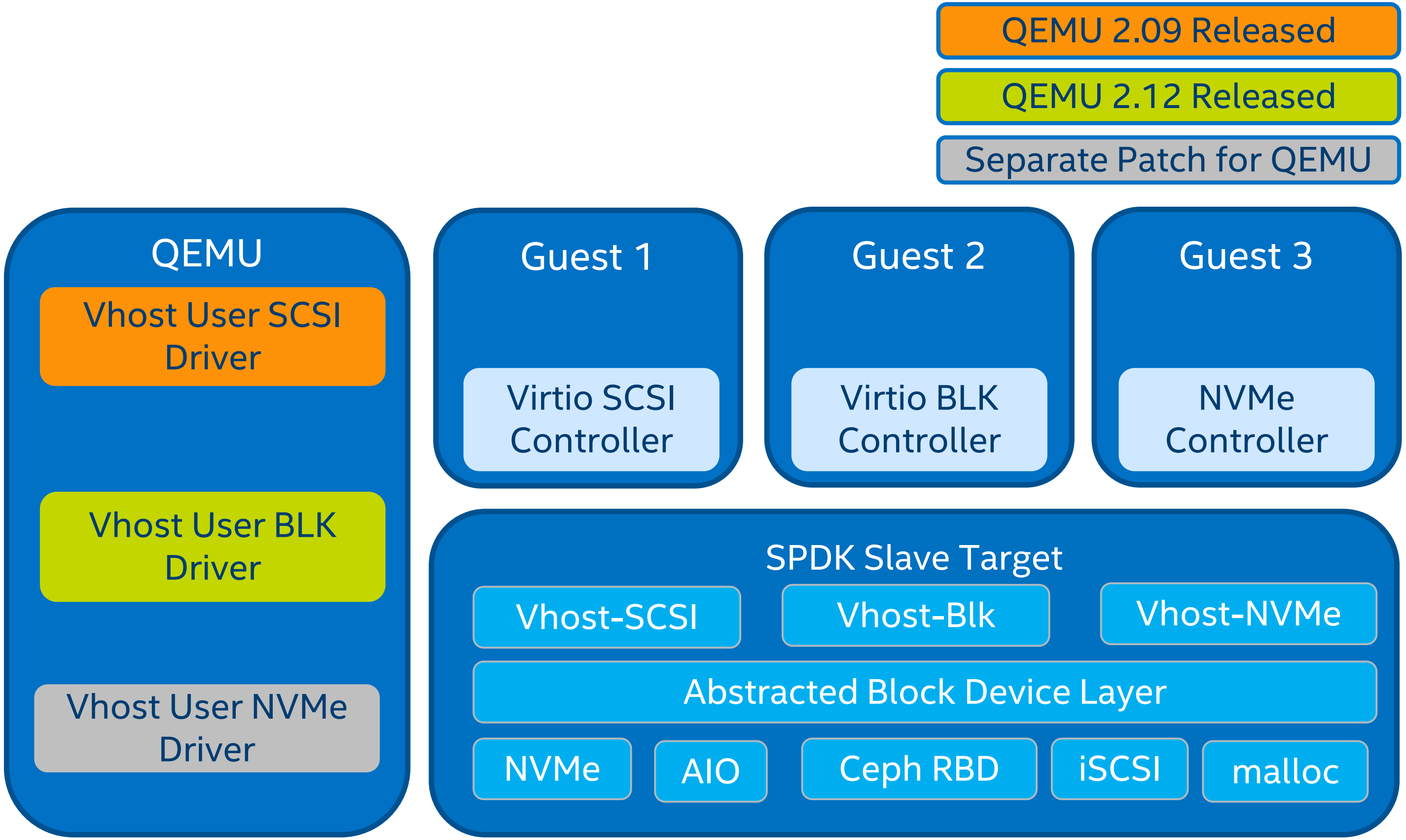
- Hardware assistant accelerator solutions based on the splicing of IO Queues are not suitable for NVMe controllers, because Namespace ID can be used at any IO Submission Queues.
- Difficult to add live migration support for hardware assistant accelerators.
- Hard to share one NVMe controller among different VMs, and advanced features such as QoS is hard to add.

Combine NVMe with Vhost-User



Implementation Details

SPDK Vhost Block Diagram



Socket Messages

Socket Message Protocol
Get/Set Controller Configuration
Admin Pass-through
Set Memory Table
Set Guest Notifier
Set Event Notifier

Table 1: socket messages

Admin Commands
Identify/Identify NS
Create/Delete Submission Queue
Create/Delete Completion Queue
Abort
Asynchronous Event Request
Doorbell Buffer Config

Table 2: Mandatory Admin commands in slave target

Get/Set Controller Configuration and Admin Pass-through messages can be dropped based on different implementation.

Common Socket Messages Benefit from Existing QEMU Vhost Library

- SET_MEMORY_TABLE: Sets the memory map regions on the slave target so it can translate the vring addresses
- SET_GUEST_NOTIFIER: Set the event file descriptor for the purpose to interrupt the Guest when I/O is completed. It can be same with existing SET_VRING_CALL message
- SET/GET_CONFIG: Set/Get PCI BAR space registers

Proposal: Extend existing QEMU vhost library and make it compatible with non-virtio devices such as NVMe

Create IO Queue

Guest: Create IO Queue

QSIZE	QID	CQID	QPRIO	PC
PRP1				

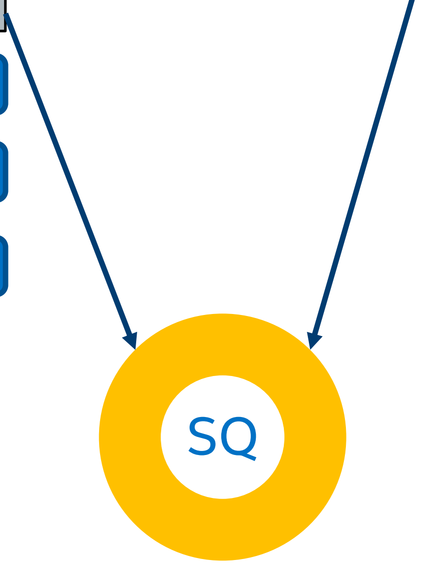
Guest: Submit to Admin, Write DB

QEMU: Pick up Admin Command

QEMU: Send via Domain Socket

SPDK: Start to Create IO Queue

SPDK: Memory Translation



Data Path Optimization for Commands Submission

MMIO Write for IO Submission

- NVMe 1.3 introduced a new feature: *Shadow Doorbell Buffer Config* command which will write to the shadow memory instead of PCI registers

Old Guest Kernel Support

- For those old Linux kernels which don't support this feature, MMIO writes will be performed when submitting new commands

SPDK Vhost Target will poll both shadow doorbell buffer memory and IO submission queue doorbell in PCI BAR0 space.

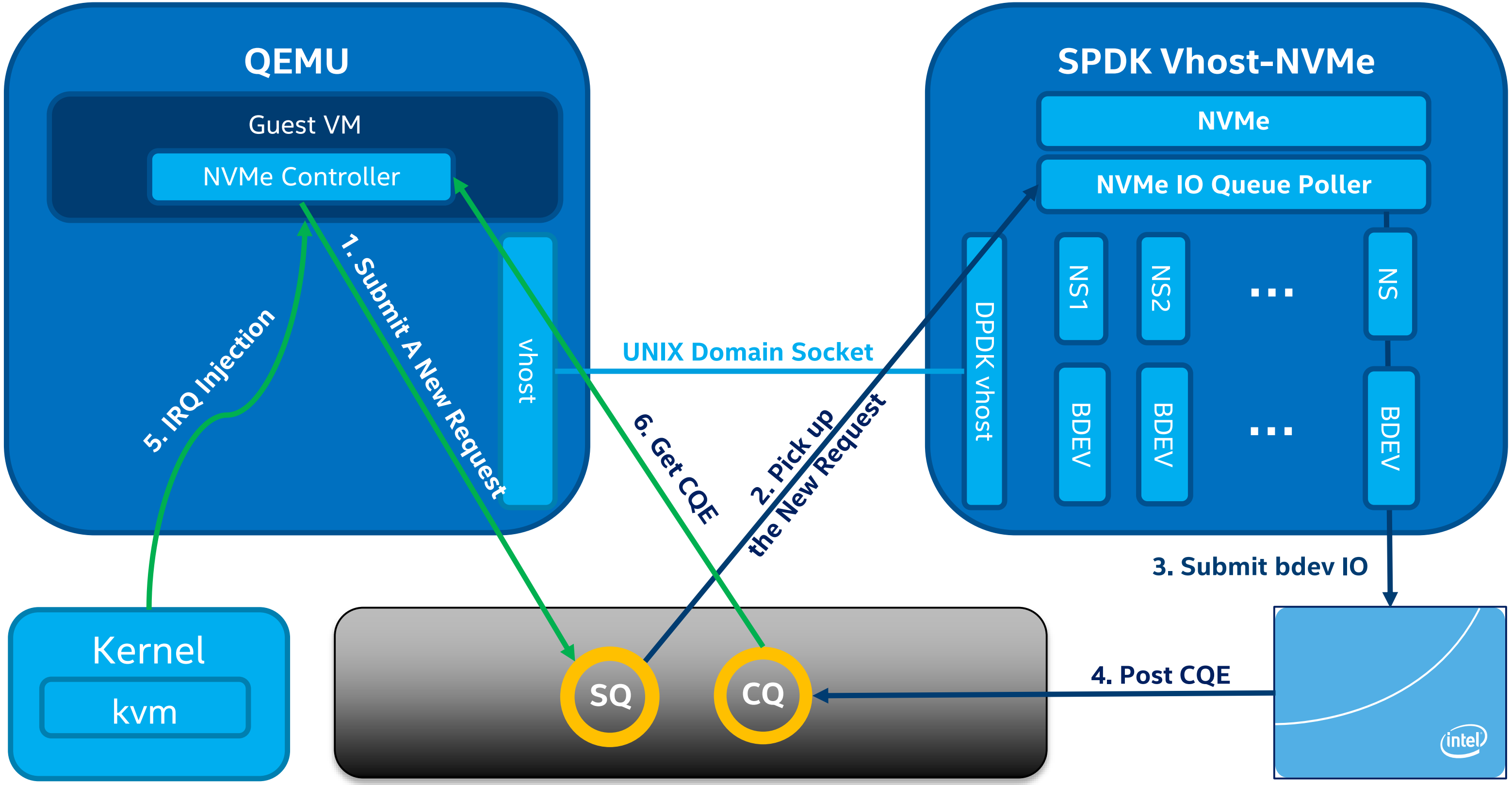
Performance is improved when shadow doorbell is enabled.

NVMe Becomes a Great Para-Virtualized Protocol

NVMe 1.3 New Feature: Optional Admin Command support for Doorbell Buffer Config, only used for emulated NVMe controllers

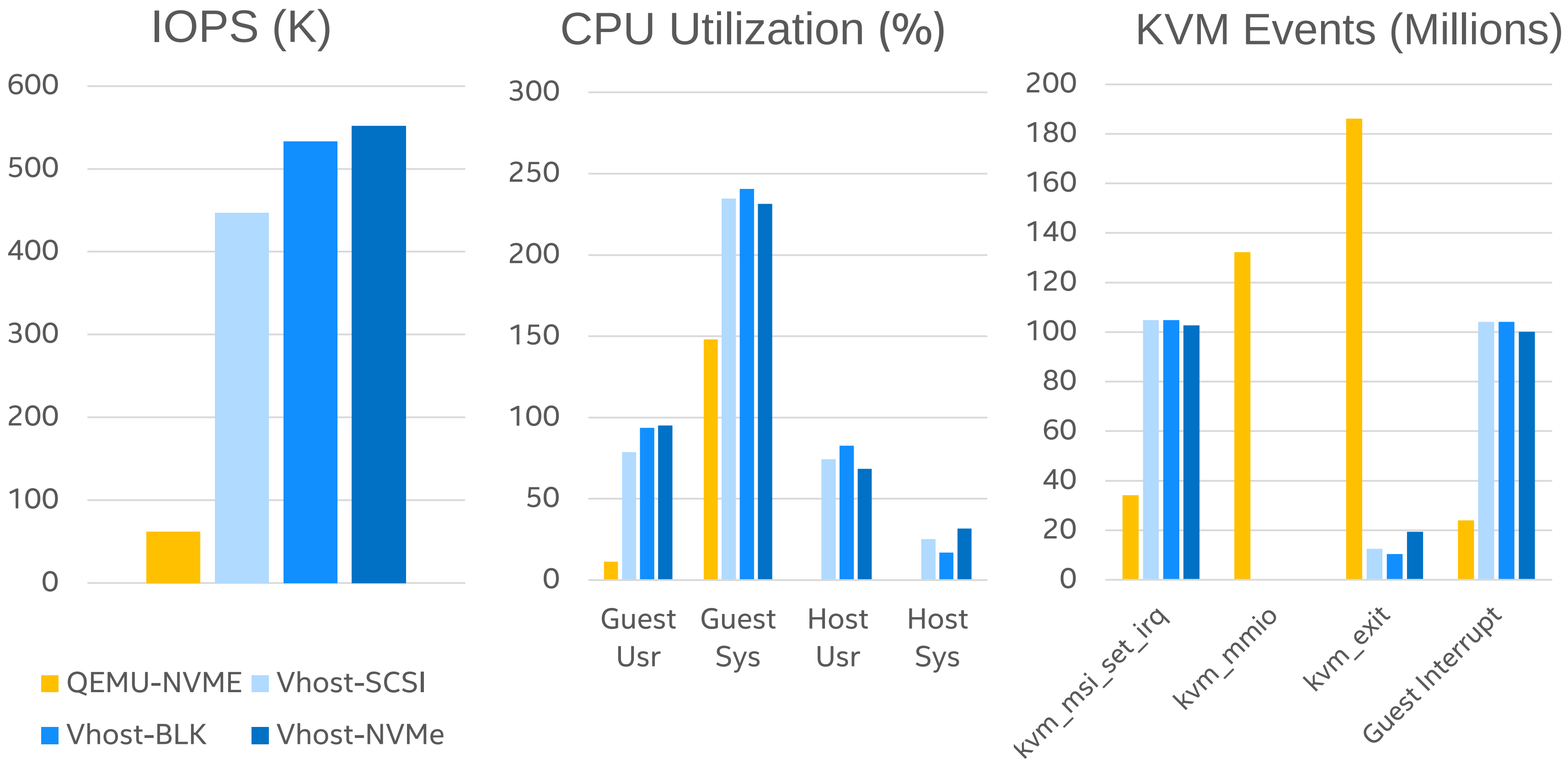
~~MMIO write causes VM_EXIT~~

IO Execution



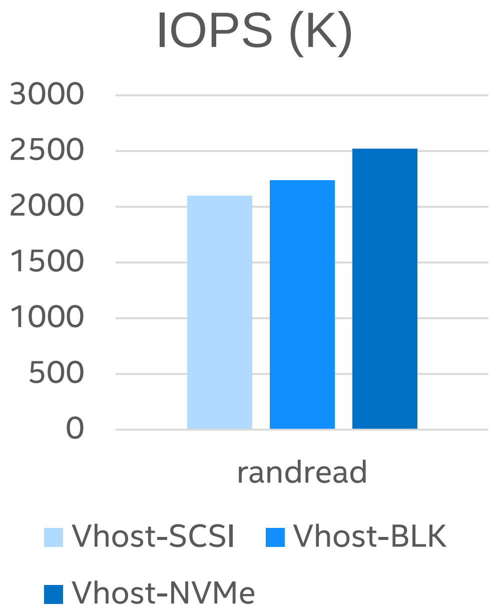
Benchmarks

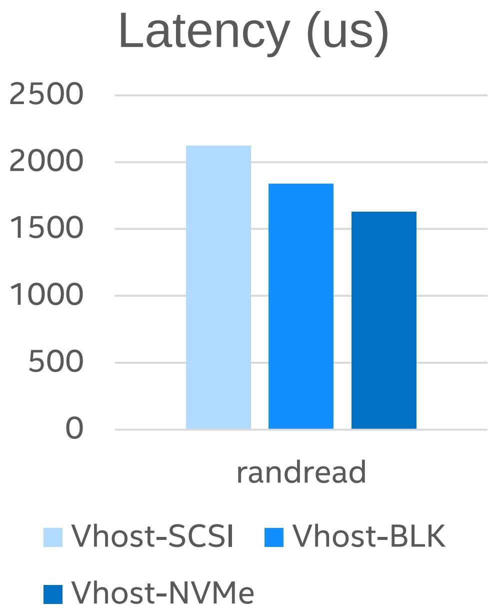
1 VM with 1 NVMe SSD to Get KVM Events



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.

Summary

- Native NVMe driver used inside guest kernel, no extra para-virtualization driver required
- No VM_EXIT for IO submission, user/kernel context switching for IRQ completion
- Zero copy for IO commands
- Benefit from Linux block driver multi-queues feature and Guest NVMe driver
- Fixed 64 Bytes for commands and 16 Bytes for response, more efficient that virtio-scsi protocol
- Hugetlbfs is required

Future Work

- Migration support
- Upstreaming with QEMU driver support
- Container support

Q&A?