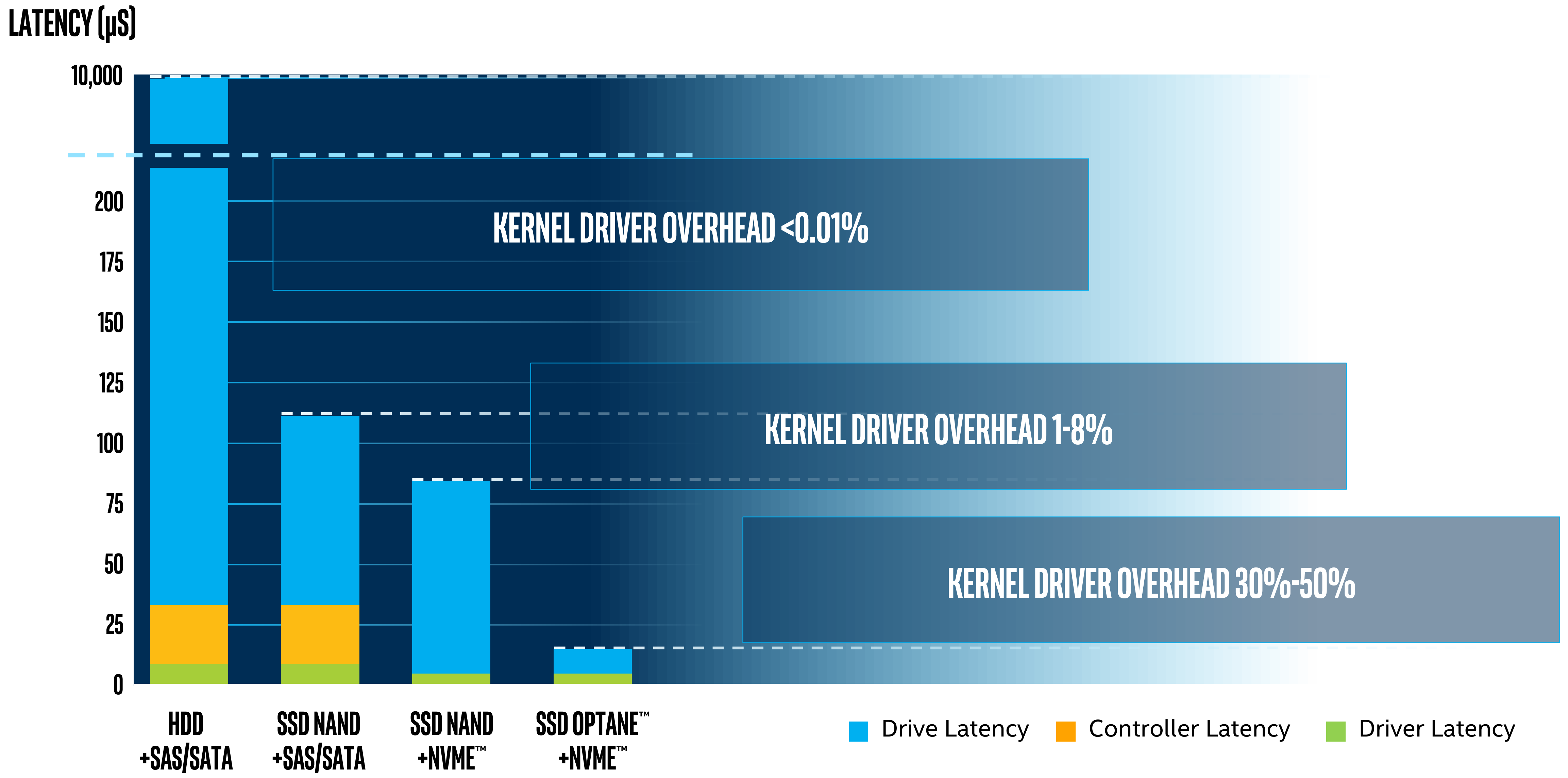
ACCELERATING VIRTUAL MACHINE ACCESS WITH STORAGE PERFORMANCE DEVELOPMENT KIT

Anu H Rao
Storage Software Product line Manager
Datacenter Group, Intel® Corp

Notices & Disclaimers

INTRODUCTION

The Challenge: Media Latency



Storage Performance Development Kit

Scalable and Efficient Software Ingredients

- User space, lockless, polled-mode components
- Up to millions of IOPS per core
- Designed to extract maximum performance from non-volatile media

Storage Reference Architecture

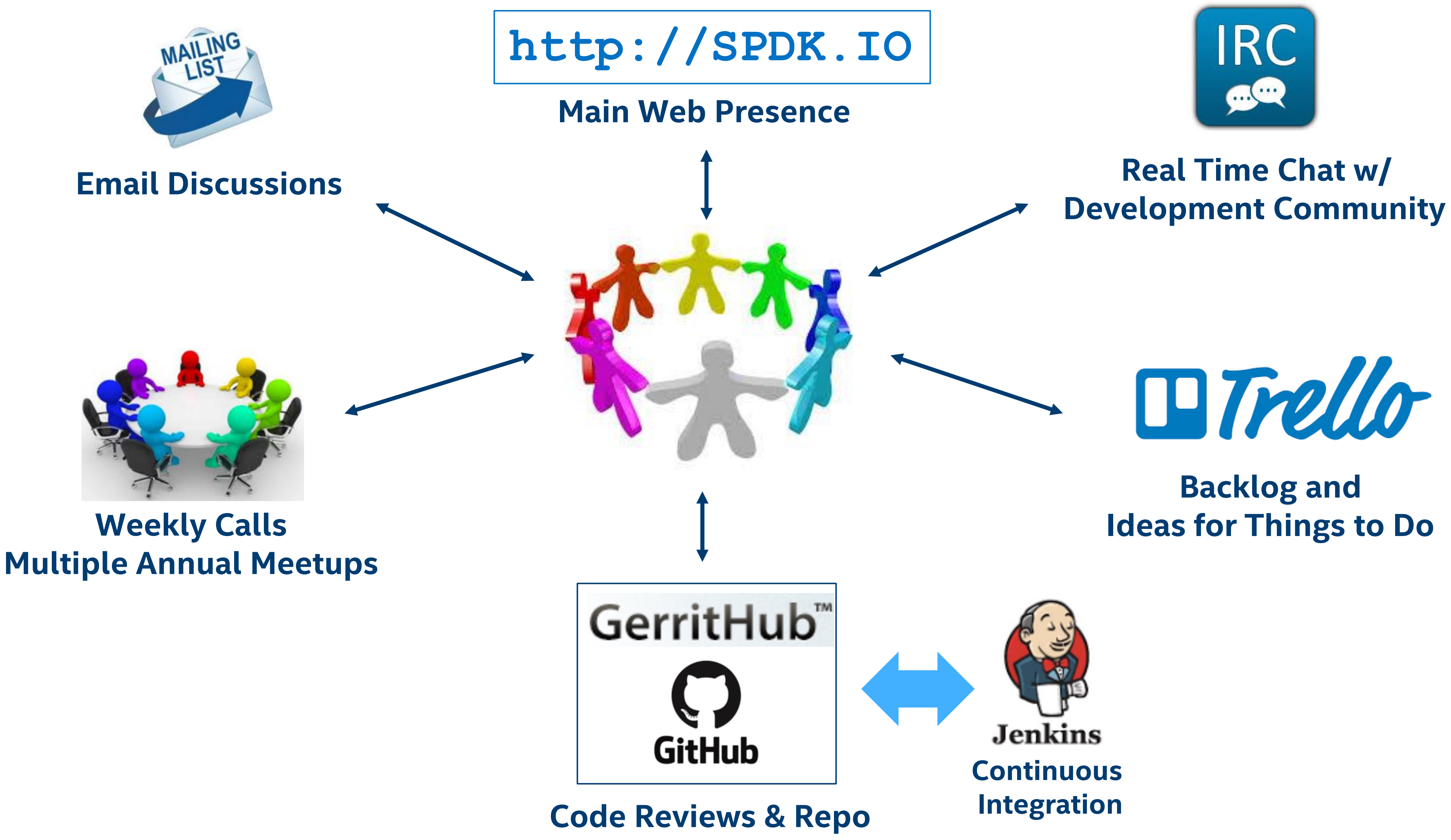
- Optimized for *latest generation CPUs and SSDs*
- Open source composable building blocks (BSD licensed)
- Available via SPDK.io
- Follow @SPDKProject on twitter for latest events and activities

Benefits of using **SPDK**

SPDK
more performance from CPUs, non-volatile media, and networking

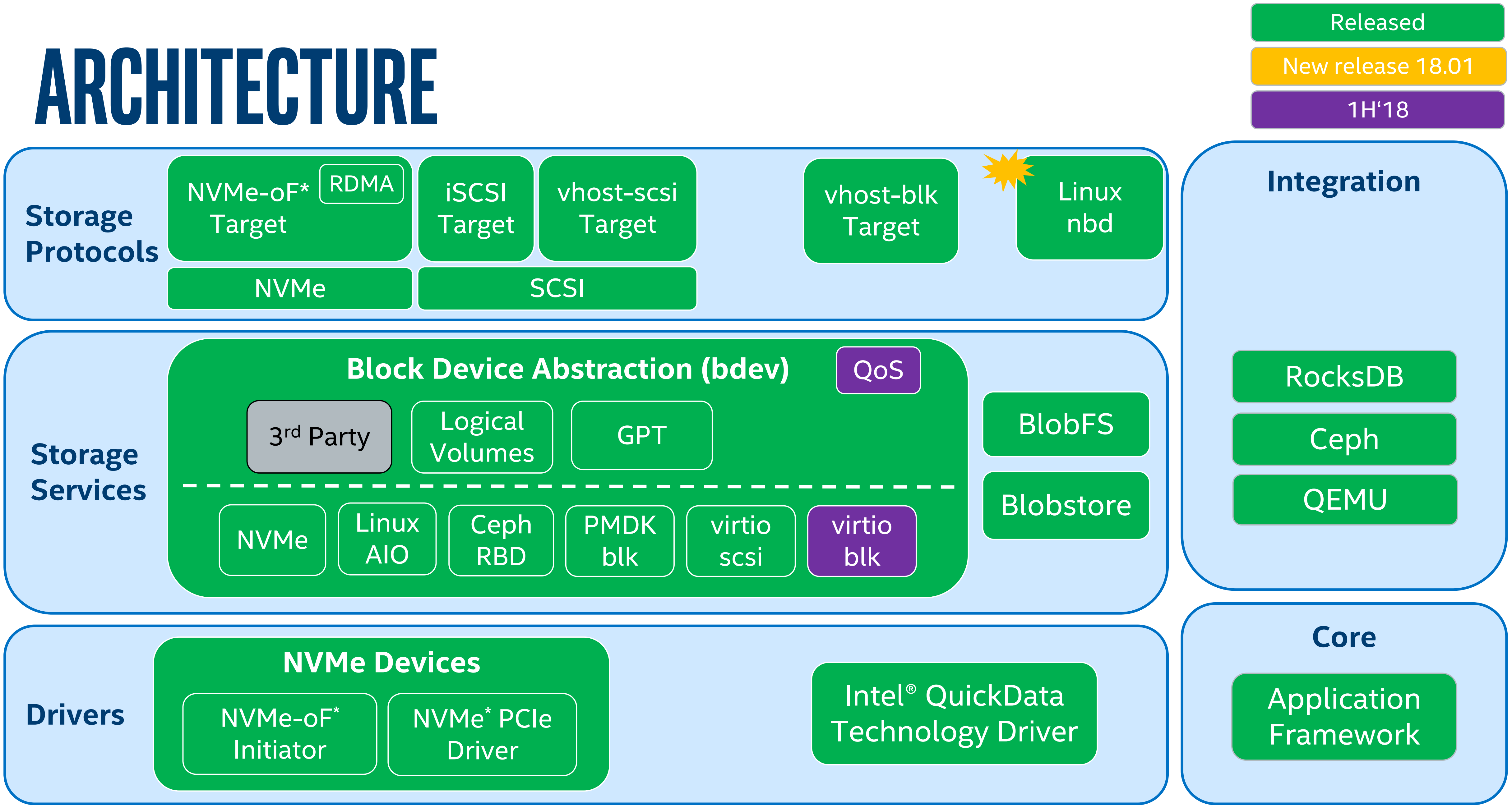
- Up to **10X MORE** IOPS/core for NVMe-oF* vs. Linux kernel
- Up to **8X MORE** IOPS/core for NVMe vs. Linux kernel
- Up to **50% BETTER** Tail Latency for RocksDB workload
- Up to **3X BETTER** IOPS/core & Latency for Virtualized Storage
- Up to **1.5X BETTER** Latency for NVMEoF vs Kernel for Optane SSD
- Up to **10.8 Million** IOPS with Intel® Xeon Scalable Family and 24 Intel® Optane™ SSD DC P4800X

SPDK Community



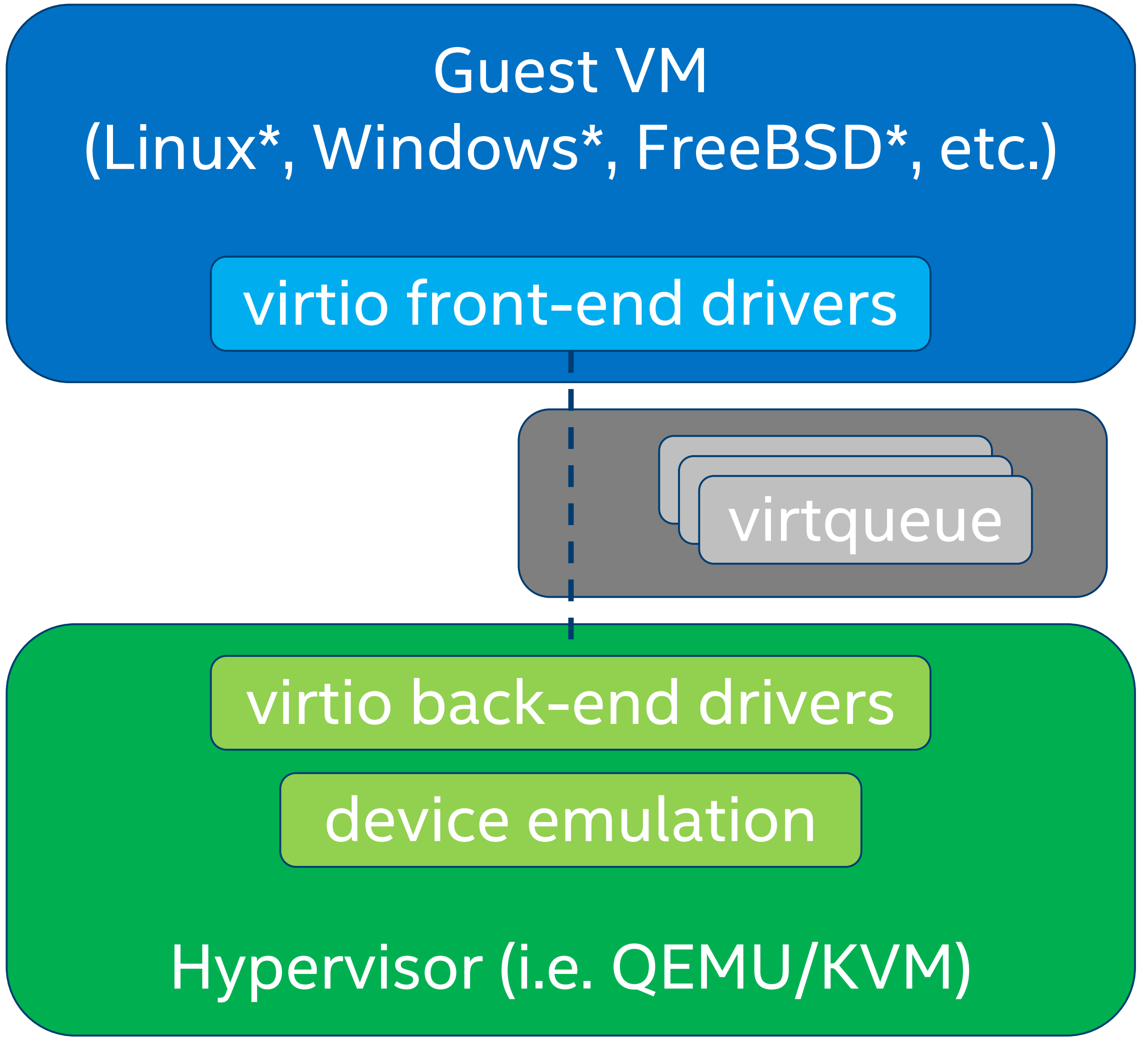
ARCHITECTURE

ARCHITECTURE



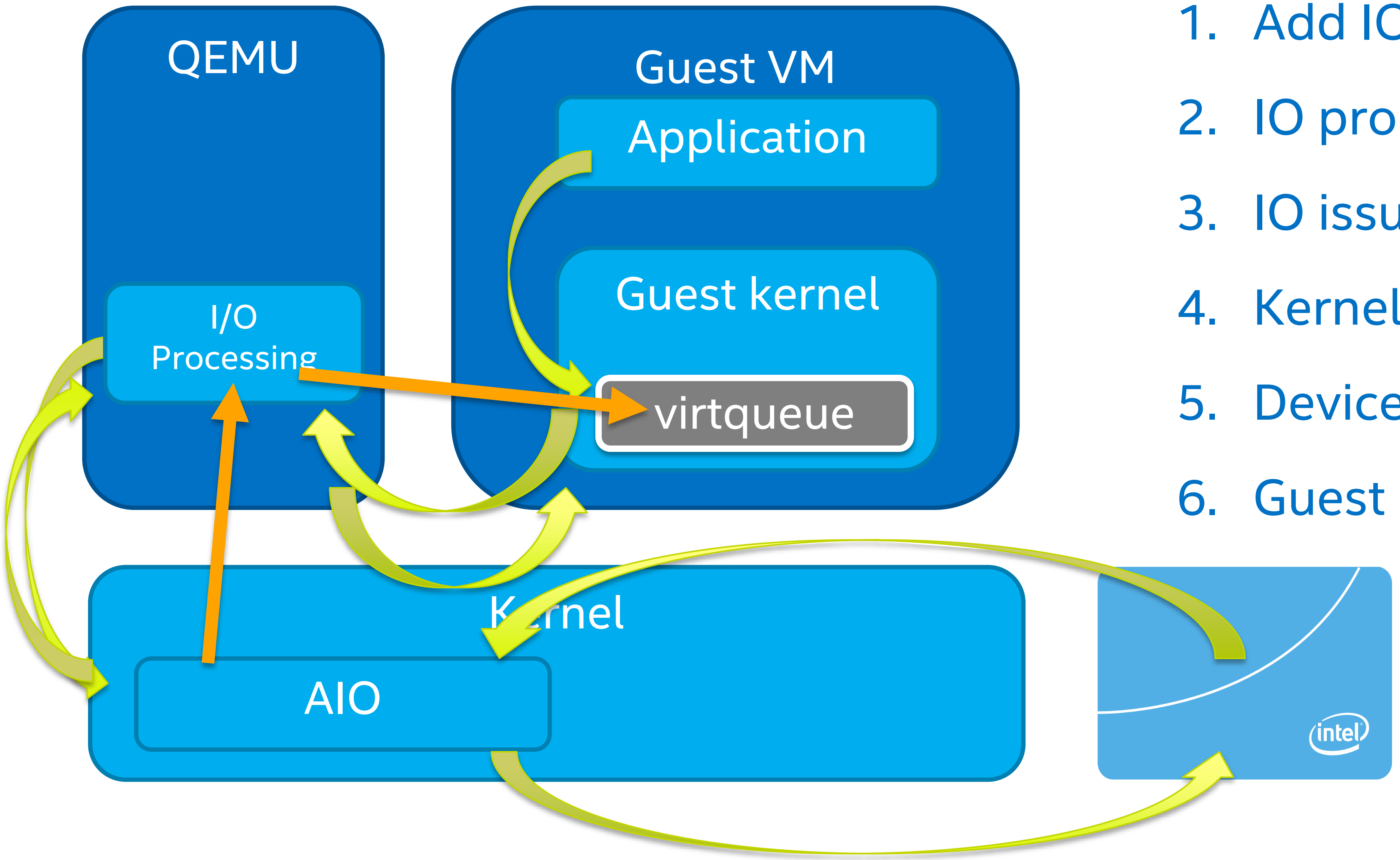
VHOST DEEP DIVE

VIRTIO



- 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

QEMU VIRTIO SCSI



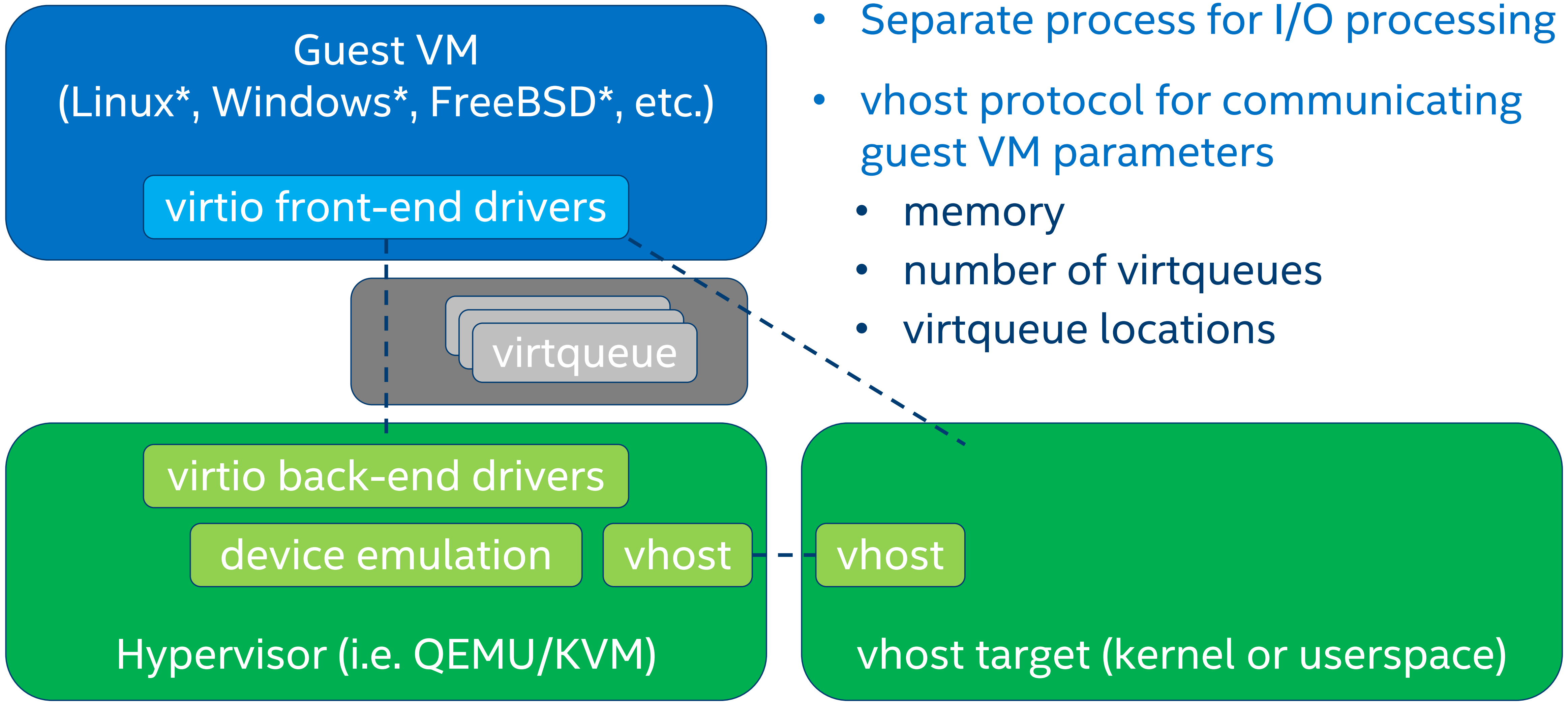
1. Add IO to virtqueue
2. IO processed by QEMU
3. IO issued to kernel
4. Kernel pins memory
5. Device executes IO
6. Guest completion interrupt

QEMU VIRTIO

⬇

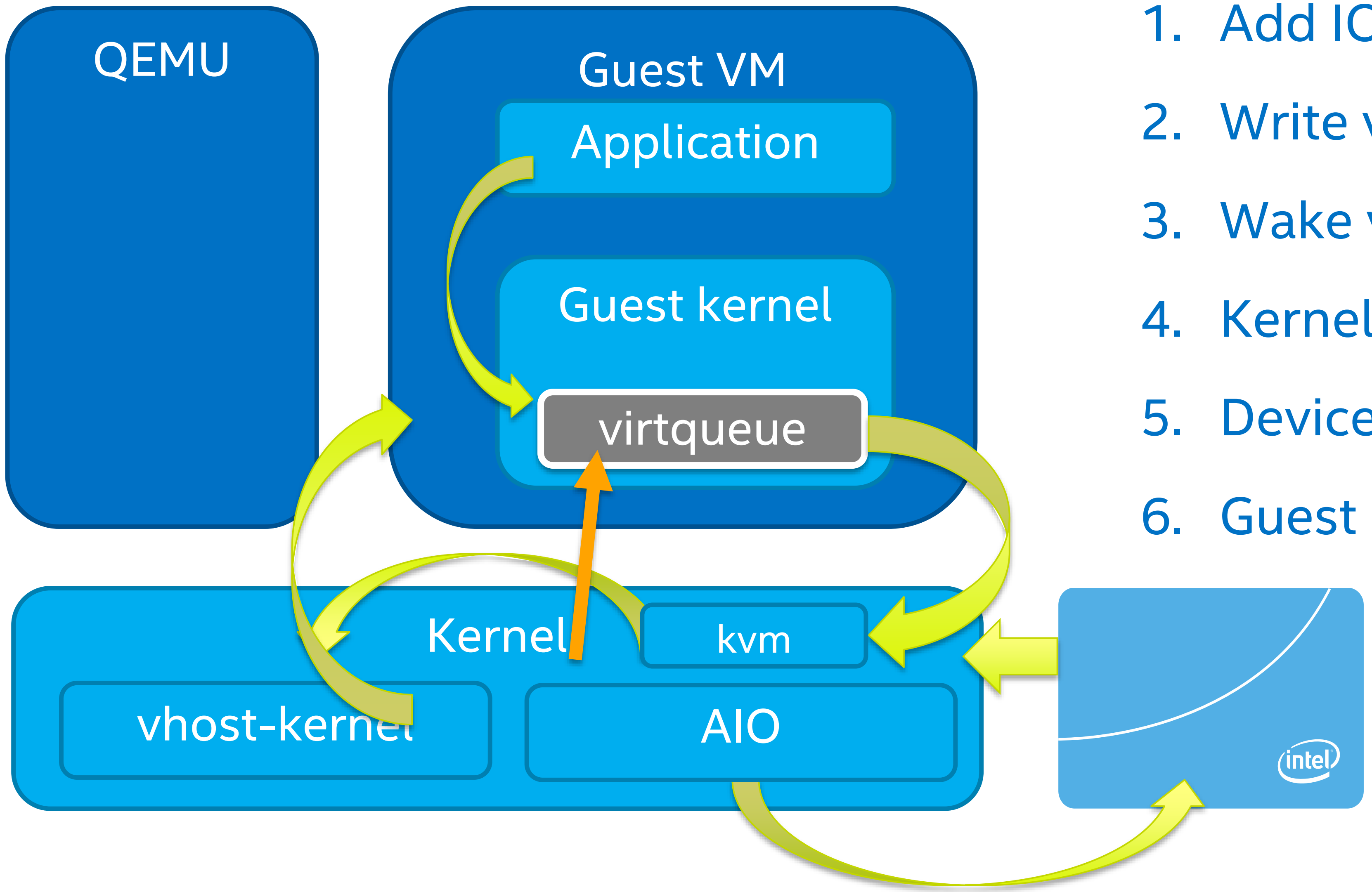
VHOST (KERNEL)

VHOST


Guest VM
(Linux*, Windows*, FreeBSD*, etc.)
virtio front-end drivers
virtqueue
virtio back-end drivers
device emulation
vhost
vhost
Hypervisor (i.e. QEMU/KVM)
vhost target (kernel or userspace)

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

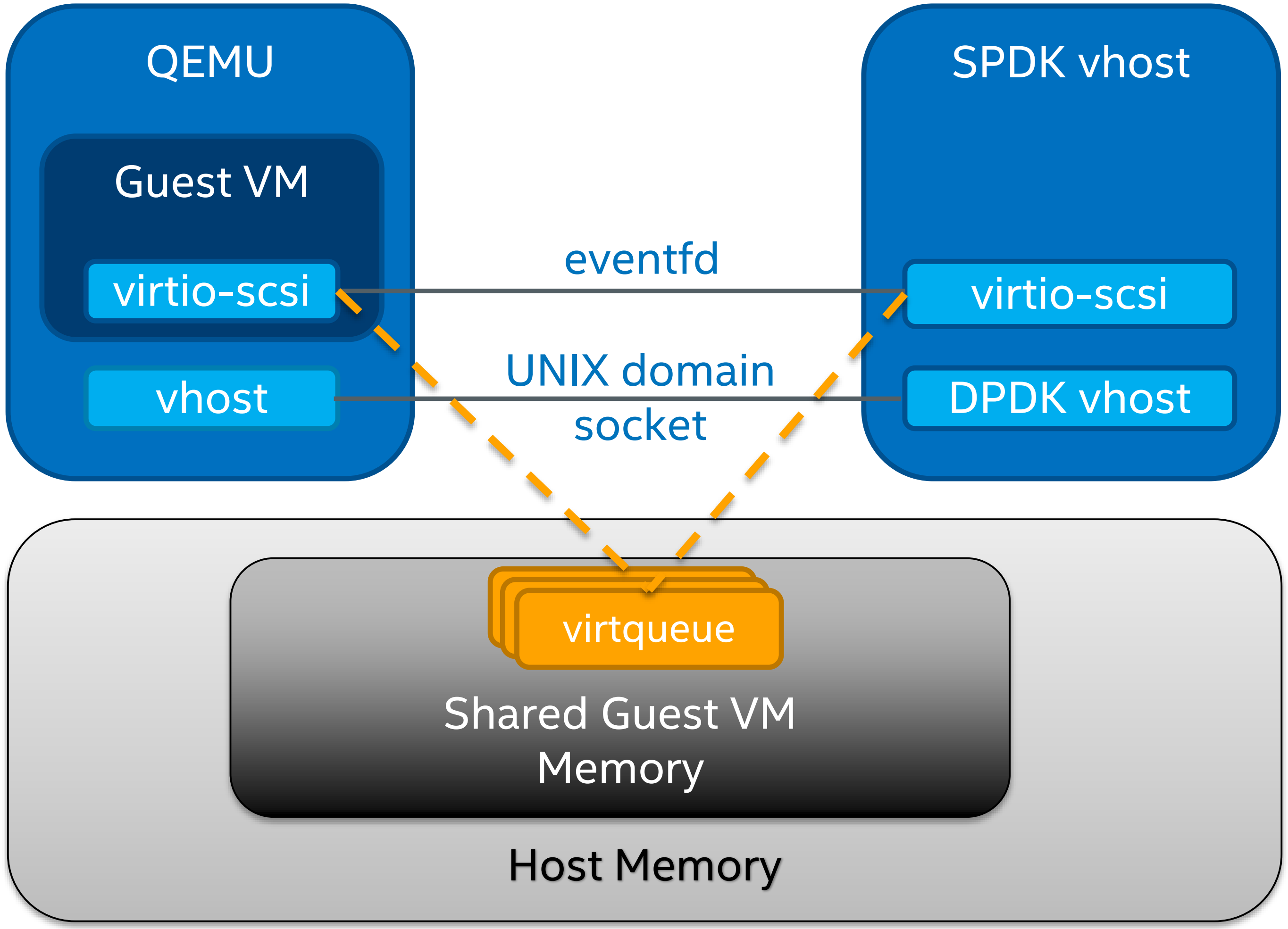
KERNEL VHOST



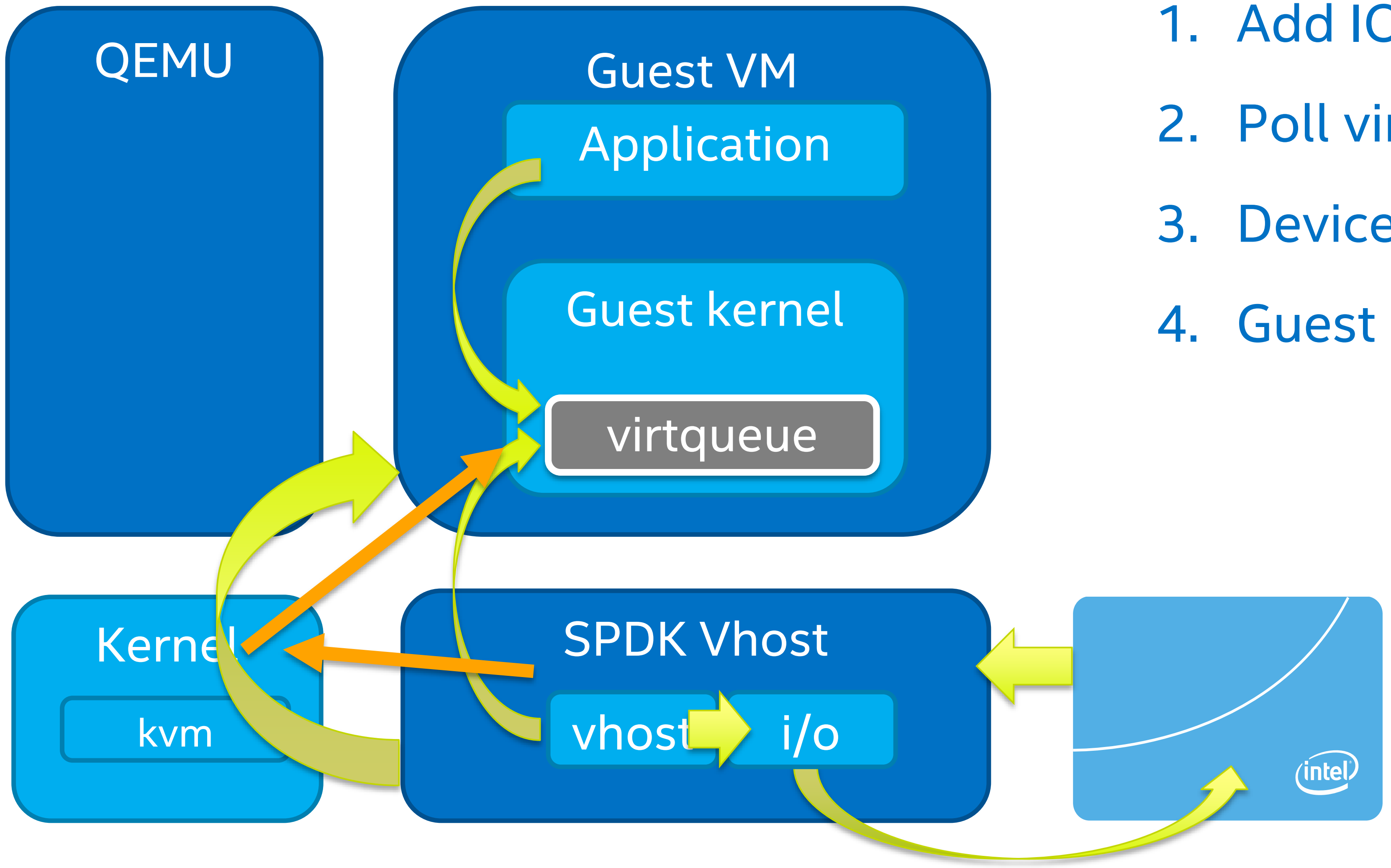
1. Add IO to virtqueue
2. Write virtio doorbell
3. Wake vhost kernel
4. Kernel pins memory
5. Device executes IO
6. Guest completion interrupt

$$\begin{tikzcd}
\text{QEMU VIRTIO} \arrow[d] \\
\text{VHOST (KERNEL)} \arrow[d] \\
\text{VHOST (USERSPACE)}
\end{tikzcd}$$

SPDK VHOST ARCHITECTURE

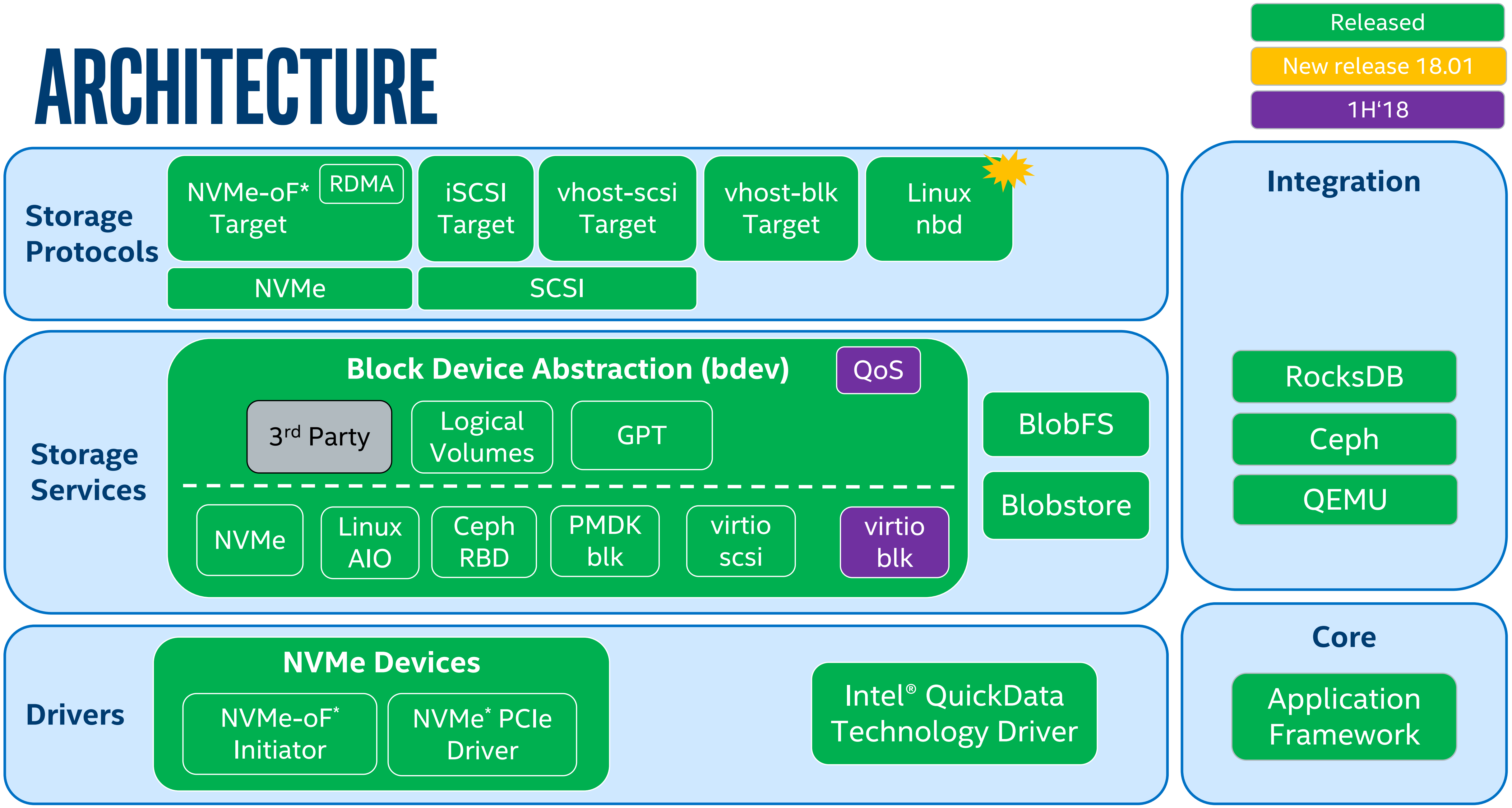


SPDK VHOST



1. Add IO to virtqueue
2. Poll virtqueue
3. Device executes IO
4. Guest completion interrupt

ARCHITECTURE



Sharing SSDs in userspace

Typically not 1:1 VM to local attached NVMe SSD

- otherwise just use PCI direct assignment

What about SR-IOV?

- SR-IOV SSDs not prevalent yet
- precludes features such as snapshots

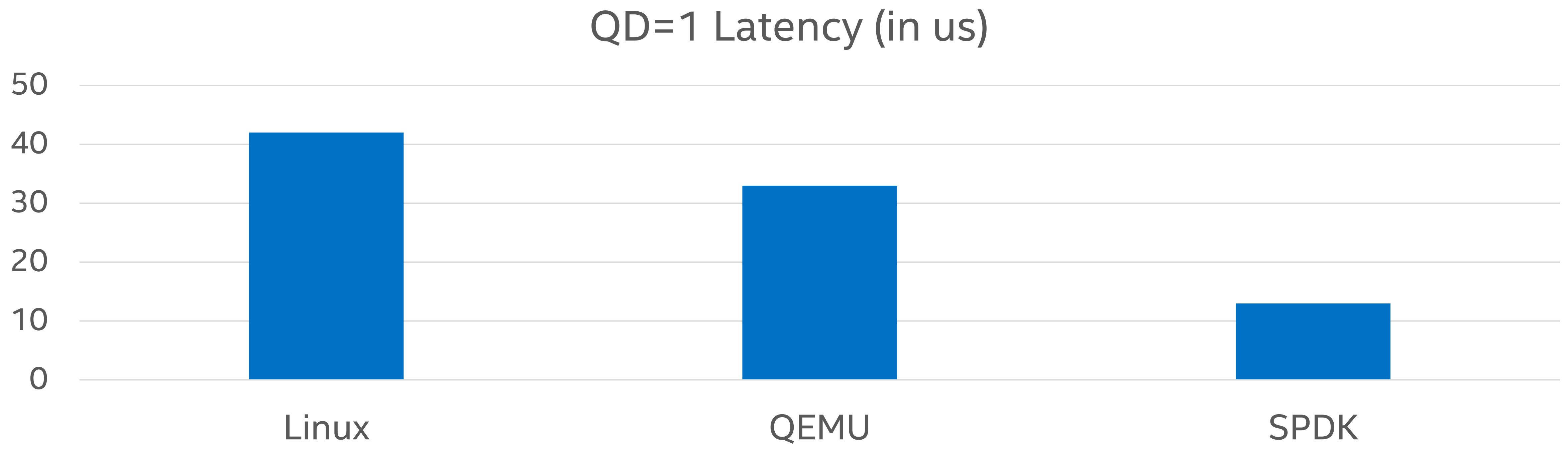
What about LVM?

- LVM depends on Linux kernel block layer and storage drivers (i.e. nvme)
- SPDK wants to use userspace polled mode drivers

SPDK Blobstore and Logical Volumes!

KVM/QEMU VIRTUALIZATION PERFORMANCE & EFFICIENCY

SPDK vhost Performance



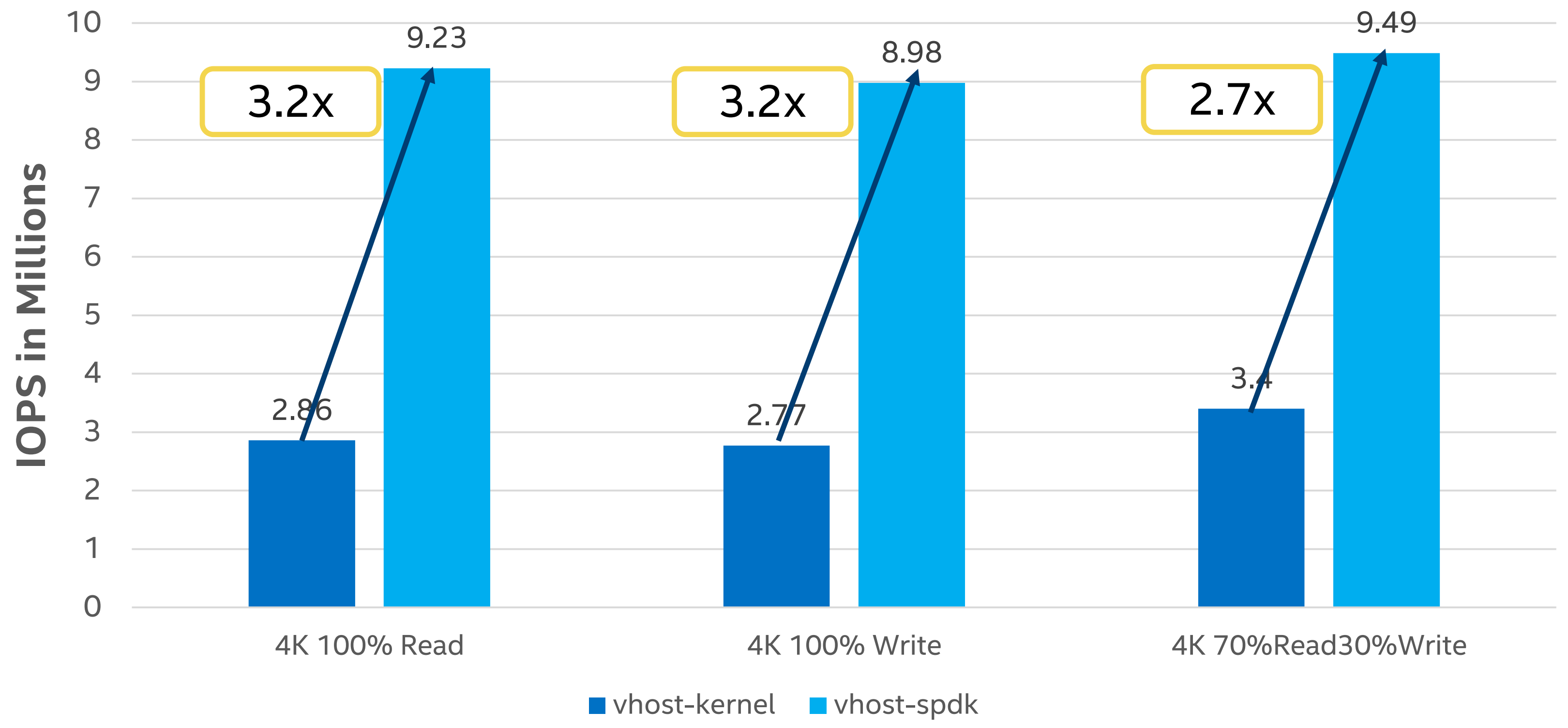
SPDK up to 3x better efficiency and latency

System Configuration: 2S Intel® Xeon® Platinum 8180: 28C, E5-2699v3: 18C, 2.5GHz (HT off), Intel® Turbo Boost Technology enabled, 12x16GB DDR4 2133 MT/s, 1 DIMM per channel, Ubuntu* Server 16.04.2 LTS, 4.11 kernel, 23x Intel® P4800x Optane SSD – 375GB, 1 SPDK lvolstore or LVM lvgroup per SSD, SPDK commit ID c5d8b108f22ab, 46 VMs (CentOS 3.10, 1vCPU, 2GB DRAM, 100GB logical volume), vhost dedicated to 10 cores
As measured by: fio 2.10.1 – Direct=Yes, 4KB random read I/O, Ramp Time=30s, Run Time=180s, Norandommap=1, I/O Engine = libaio, Numjobs=1
Legend: Linux: Kernel vhost-scsi QEMU: virtio-blk dataplane SPDK: Userspace vhost-scsi

48 VMs: vhost-scsi performance (SPDK vs. Kernel)

Intel Xeon Platinum 8180 Processor, 24x Intel P4800x 375GB

2 partitions per VM, 10 vhost I/O processing cores

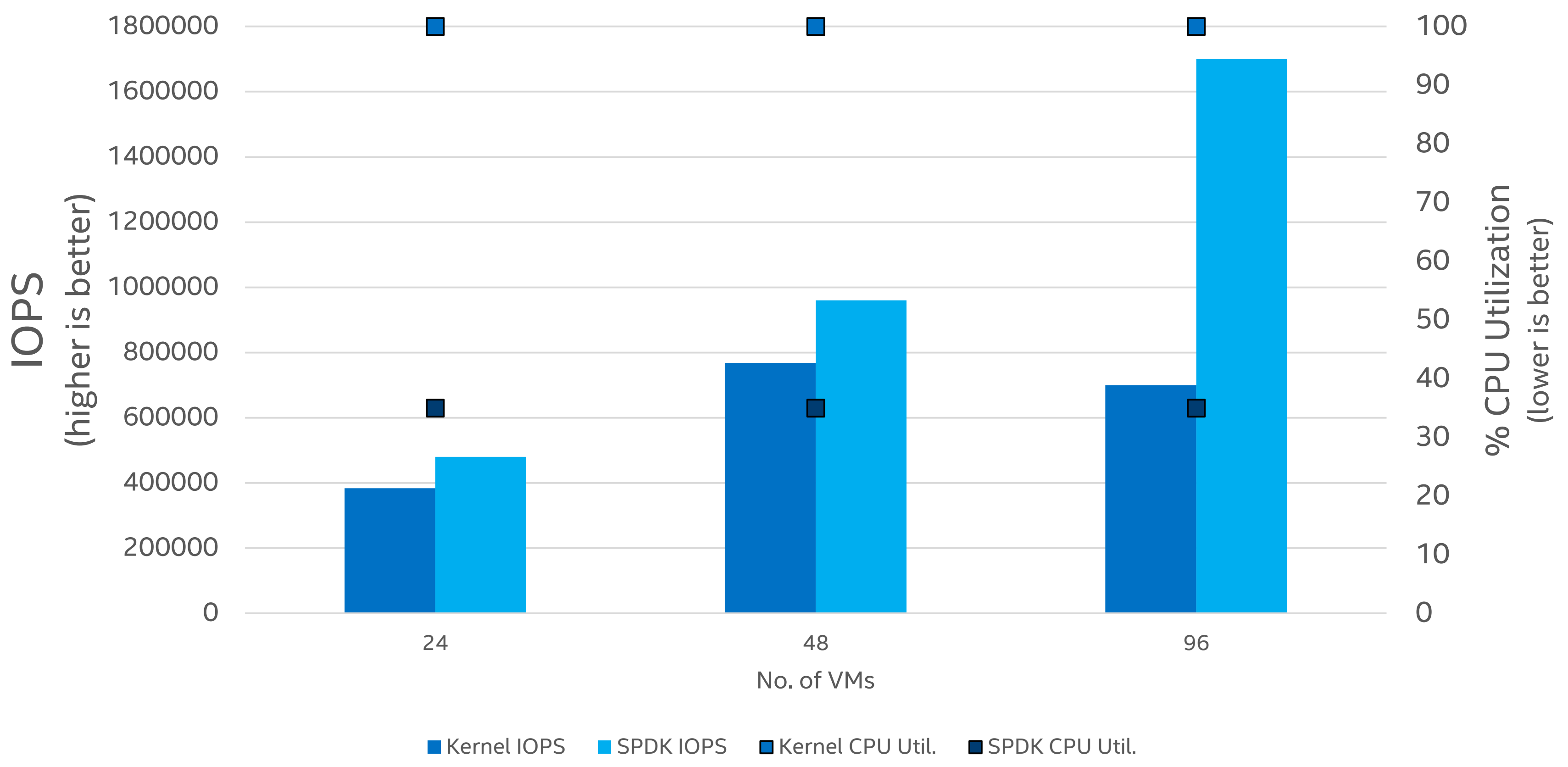


- Aggregate IOPS across all 48x VMs reported. All VMs on separate cores than vhost-scsi cores.
- 10 vhost-scsi cores for I/O processing
- SPDK vhost-scsi up to 3.2x better with 4K 100% Random read I/Os
- Used cgroups to restrict kernel vhost-scsi processes to 10 cores

System Configuration:Intel Xeon Platinum 8180 @ 2.5GHz. 56 physical cores 6x 16GB, 2667 DDR4, 6 memory Channels, SSD: Intel P4800x 375GB x24 drives, Bios: HT disabled, p-states enabled, turbo enabled, Ubuntu 16.04.1 LTS, 4.11.0 x86_64 kernel, 48 VMs, number of partition: 2, VM config : 1core 1GB memory, VM OS: fedora 25, blk-mq enabled, Software packages: Qemu-2.9, libvirt-3.0.0, spdk (3bfecec994), IO distribution: 10 vhost-cores for SPDK / Kernel. Rest 46 cores for QEMU using cgroups, FIO-2.1.10 with SPDK plugin, io depth=1, 8, 32 numjobs=1, direct=1, block size 4k

VM Density: Rate Limiting 20K IOPS per VM

Intel Xeon Platinum 8180 Processor, 24x Intel P4800x 375GB 10 vhost-scsi cores

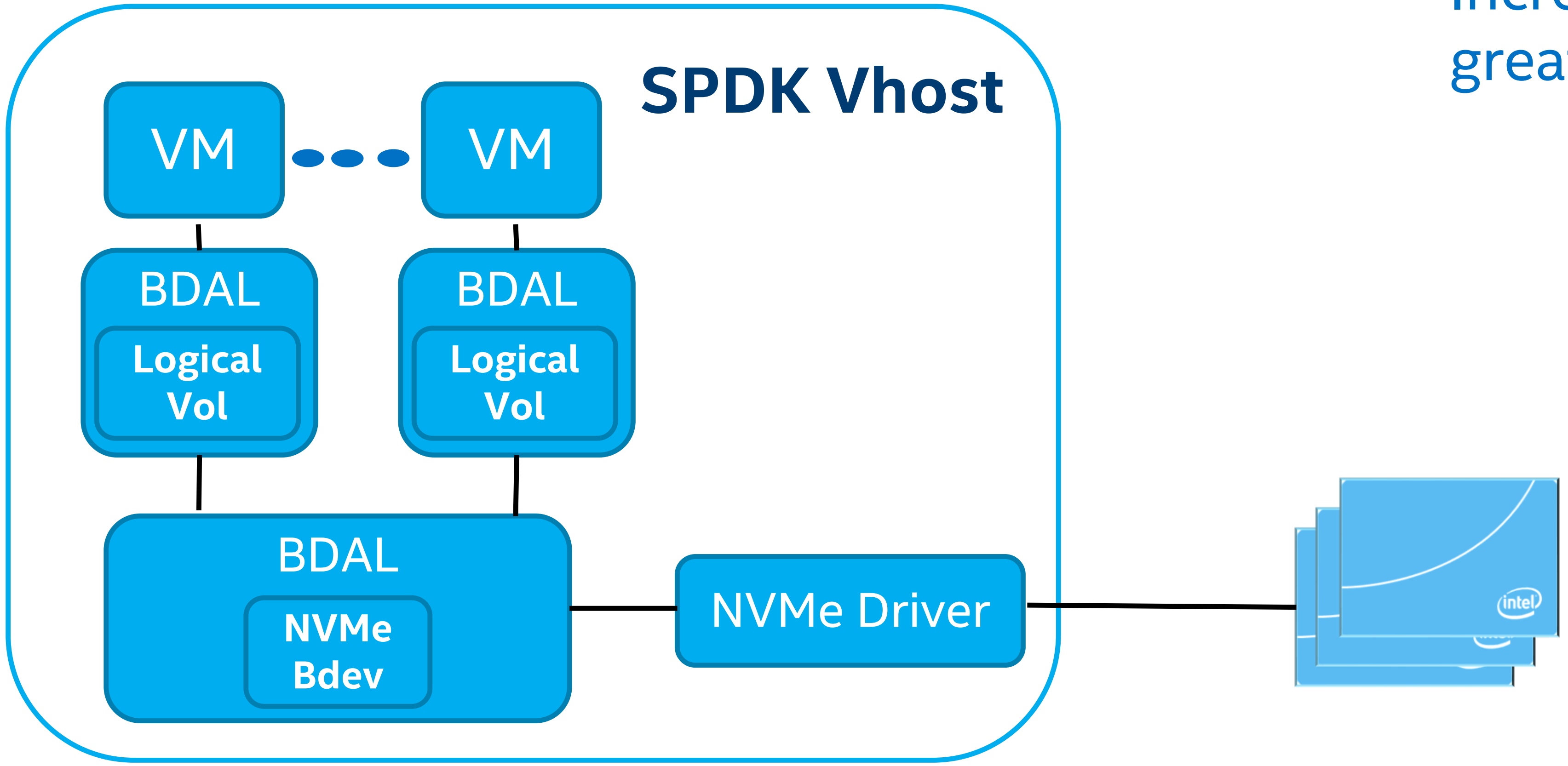


- % CPU utilized shown from VM side
- Each VM was running queue depth=1, 4KB random read workload
- Hyper threading enabled to allow 112 cores.
- Each VM rate limited to 20K IOPS using cgroups
- SPDK able to scale to 96 VMs, supporting 20K per VM. Kernel scale till 48 VMs. Beyond 48 VMs, 10 vhost-cores seem bottleneck

System Configuration:Intel Xeon Platinum 8180 @ 2.5GHz. 56 physical cores 6x 16GB, 2667 DDR4, 6 memory Channels, SSD: Intel P4800x 375GB x24 drives, Bios: HT disabled, p-states enabled, turbo enabled, Ubuntu 16.04.1 LTS, 4.11.0 x86_64 kernel, 48 VMs, number of partition: 2, VM config : 1core 1GB memory, VM OS: fedora 25, blk-mq enabled, Software packages: Qemu-2.9, libvirt-3.0.0, spdk (3bfecec994), IO distribution: 10 vhost-cores for SPDK / Kernel. Rest 46 cores for QEMU using cgroups, FIO-2.1.10 with SPDK plugin, io depth=1, 8, 32 numjobs=1, direct=1, block size 4k

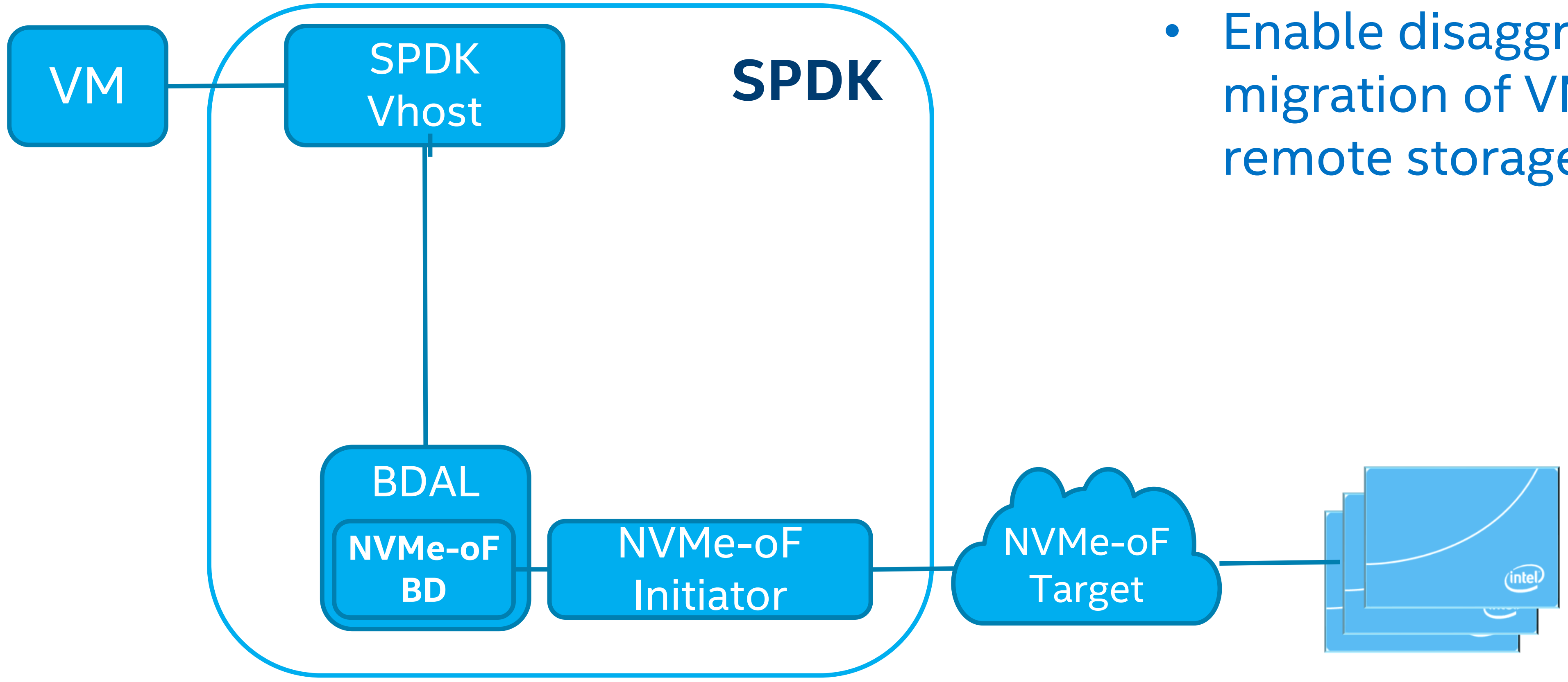
USE CASES FOR VHOST

VM EPHEMERAL STORAGE



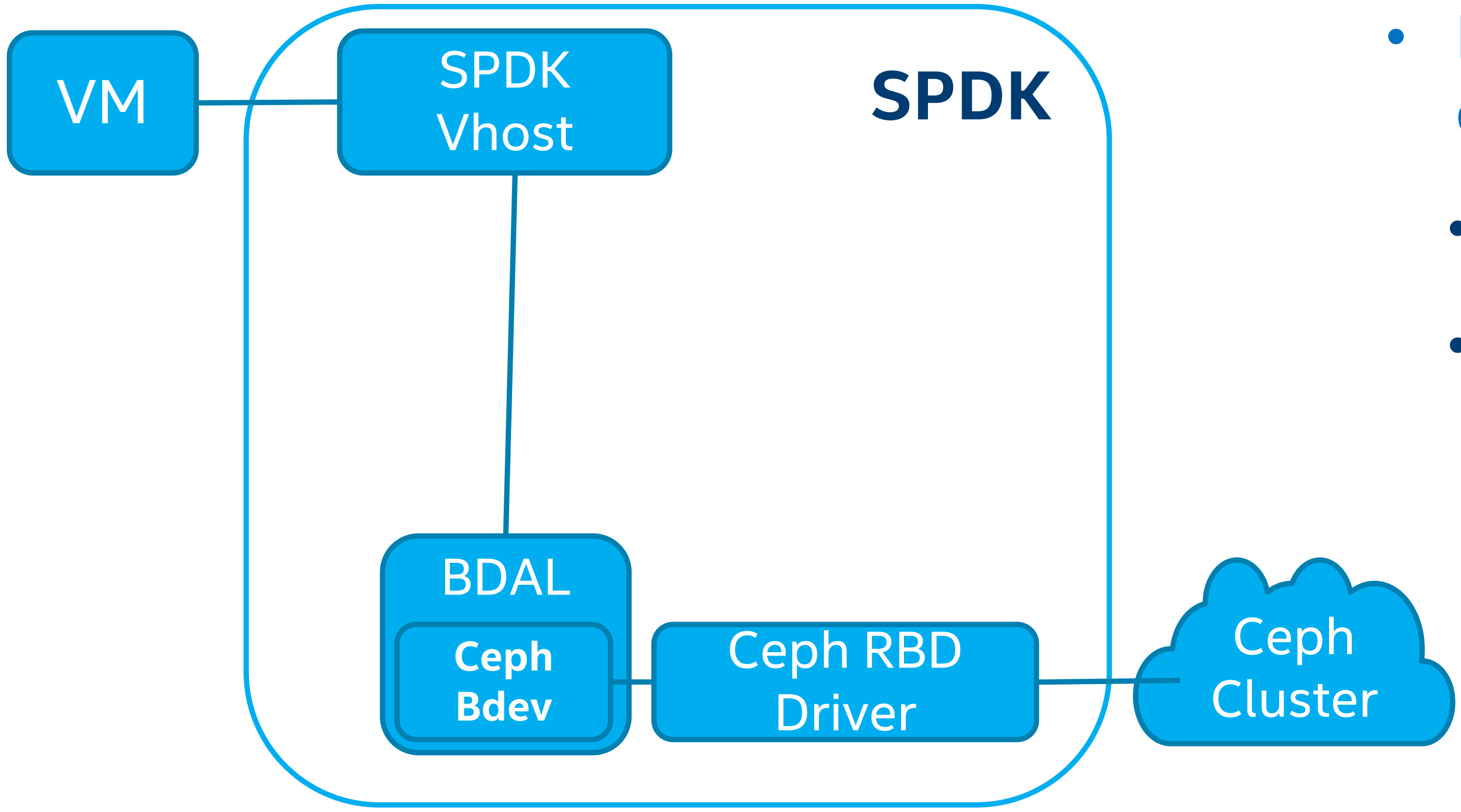
- Increased efficiency yields greater VM density

VM REMOTE STORAGE



- Enable disaggregation and migration of VMs using remote storage

VM CEPH STORAGE



- Potential for innovation in data services
  - Cache
  - Deduplication

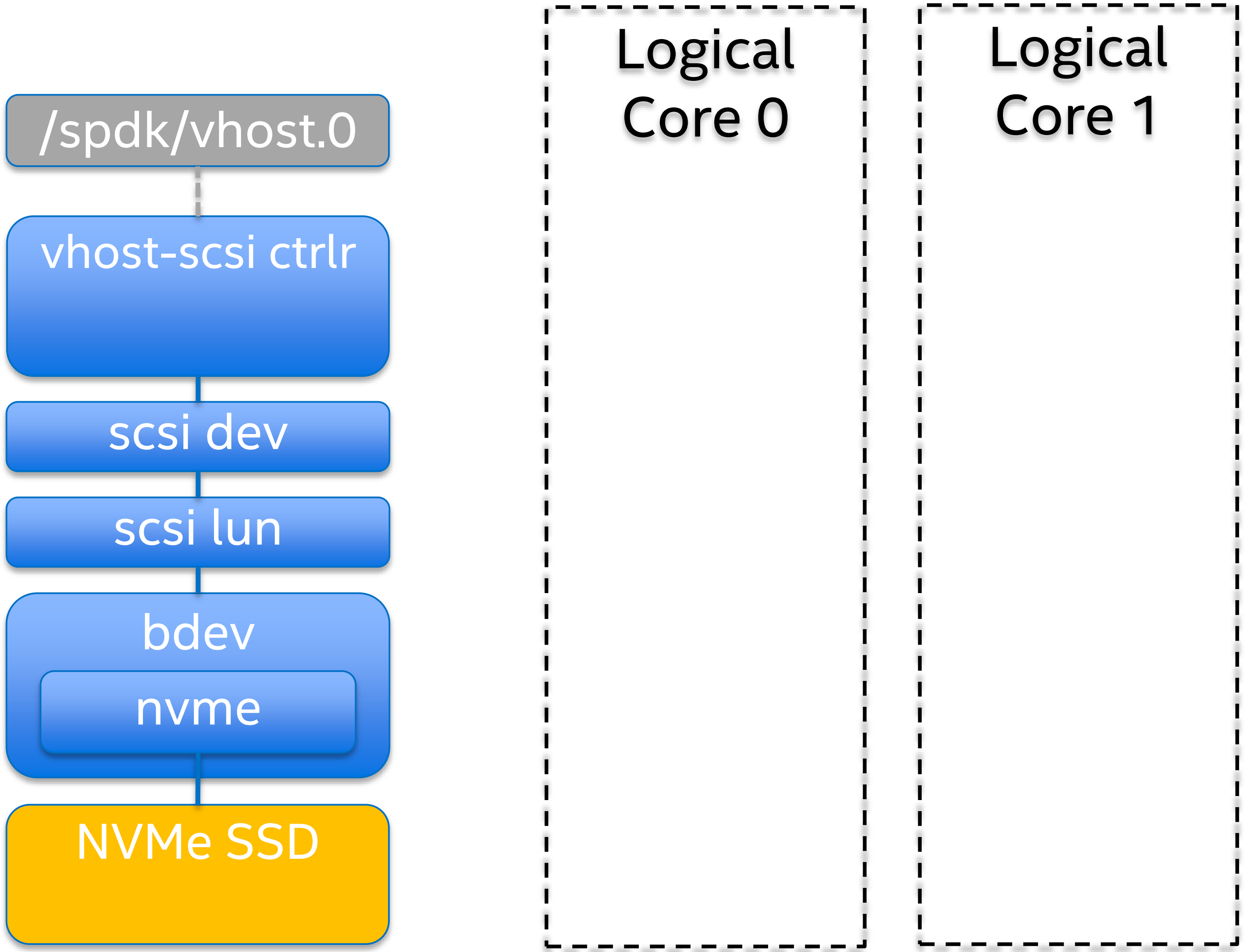
For More information on SPDK

- Visit ***SPDK.io*** for tutorials and links to github, maillist, IRC channel and other resources
- Follow ***@SPDKProject on twitter*** for latest events, blogs and other SPDK community information and activities

Basic Architecture

Configure vhost-scsi controller

- JSON RPC
- creates SPDK constructs for vhost device and backing storage
- creates controller-specific vhost domain socket



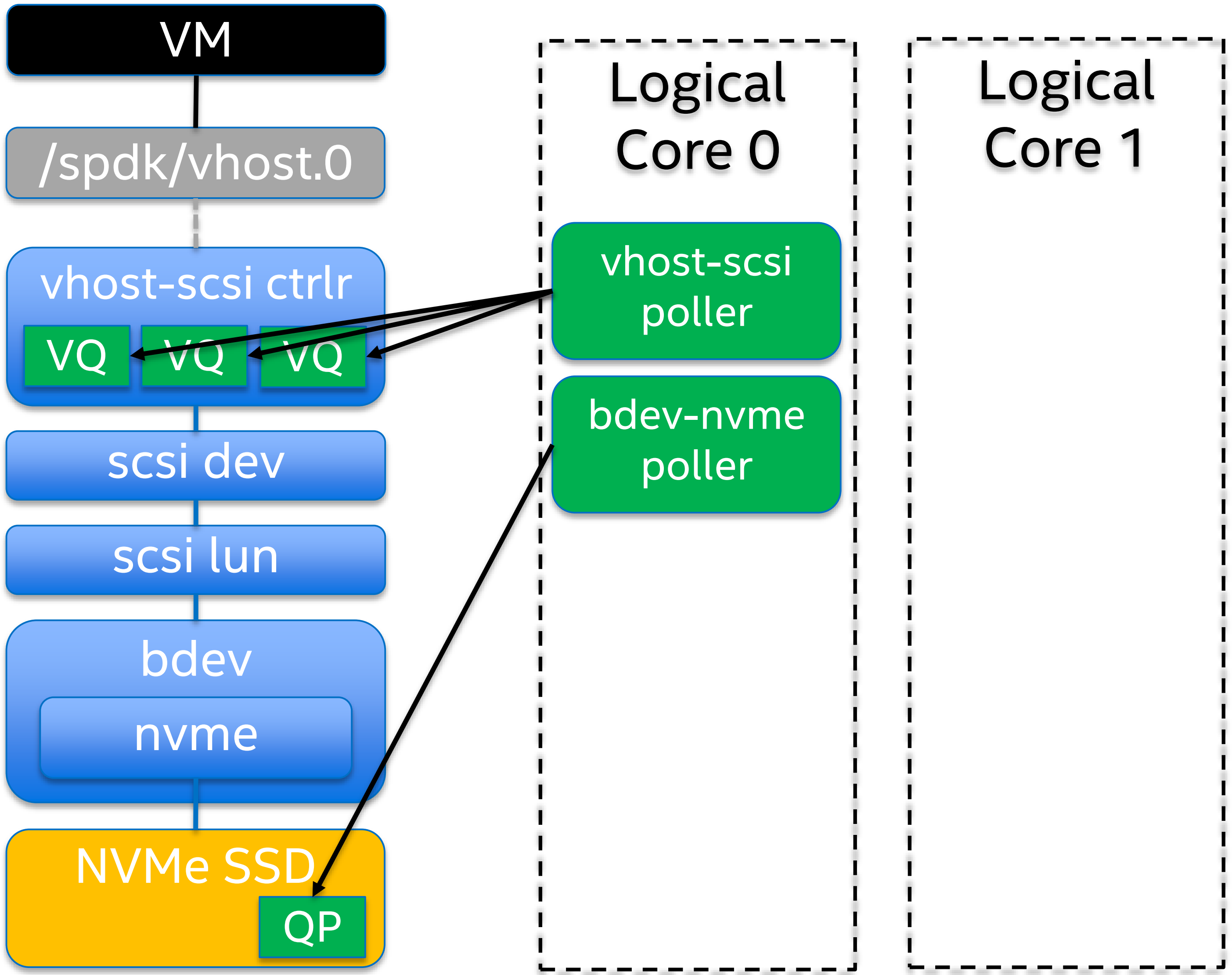
Basic Architecture

Launch VM

- QEMU connects to domain socket

SPDK

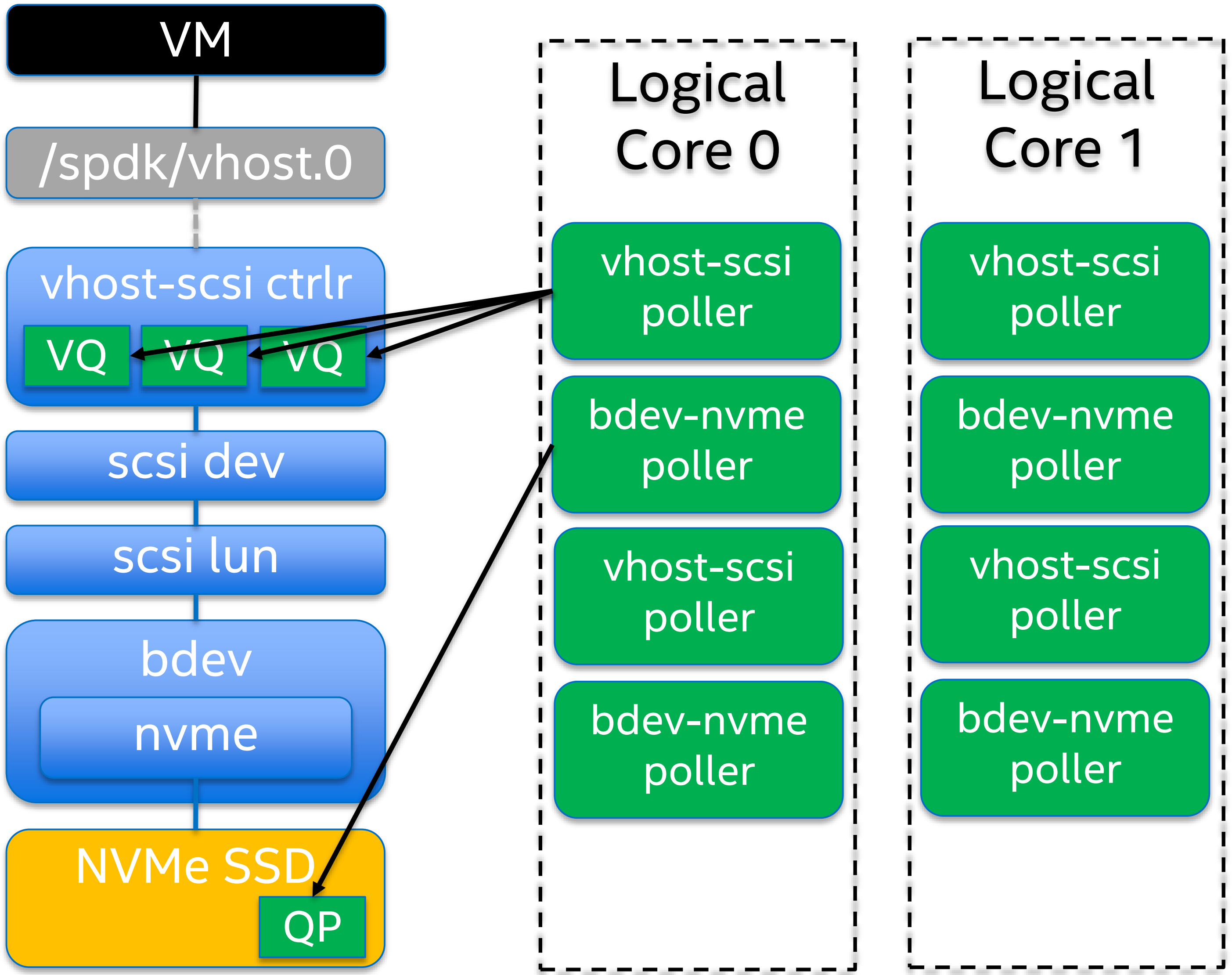
- Assigns logical core
- Starts vhost dev poller
- Allocates NVMe queue pair
- Starts NVMe poller



Basic Architecture

Repeat for additional VMs

- pollers spread across available cores



LOGICAL VOLUMES AND BLOBSTORE

Blobstore Design – Design Goals

- Minimalistic for targeted storage use cases like Logical Volumes and RocksDB
- Deliver only the basics to enable another class of application
- Design for fast storage media

Blobstore Design – High Level

Application interacts with chunks of data called blobs

- Mutable array of pages of data, accessible via ID

Asynchronous

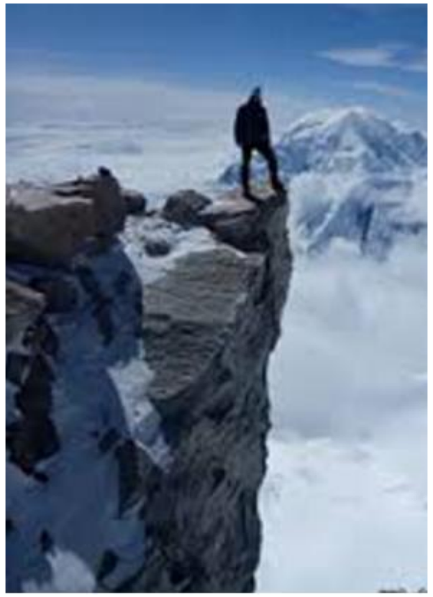
- No blocking, queuing or waiting

Fully parallel

- No locks in IO path

Atomic metadata operations

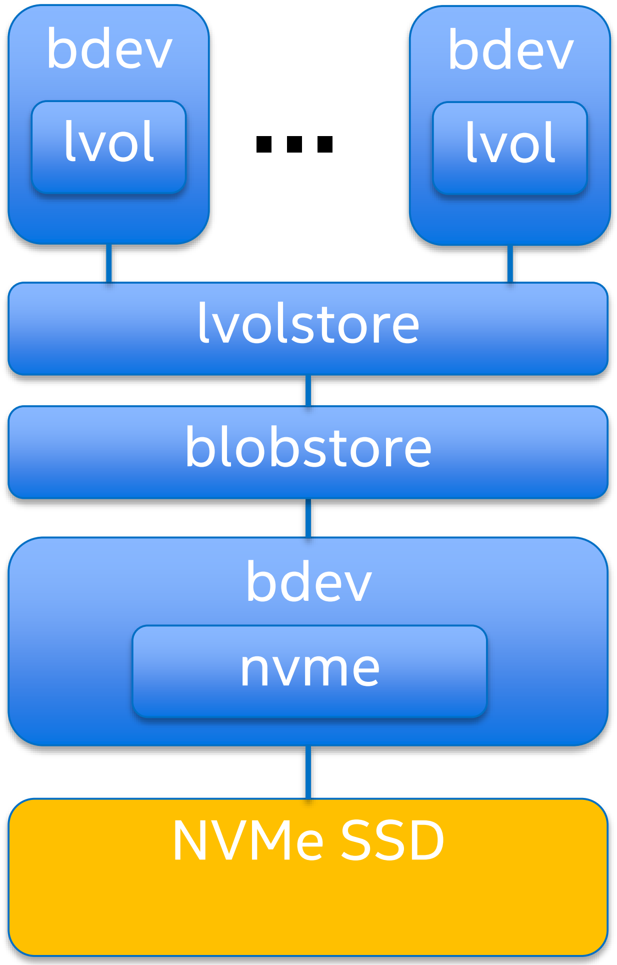
- Depends on SSD atomicity (i.e. NVMe)
- 1+ 4KB metadata pages per blob



Logical Volumes

Blobstore plus:

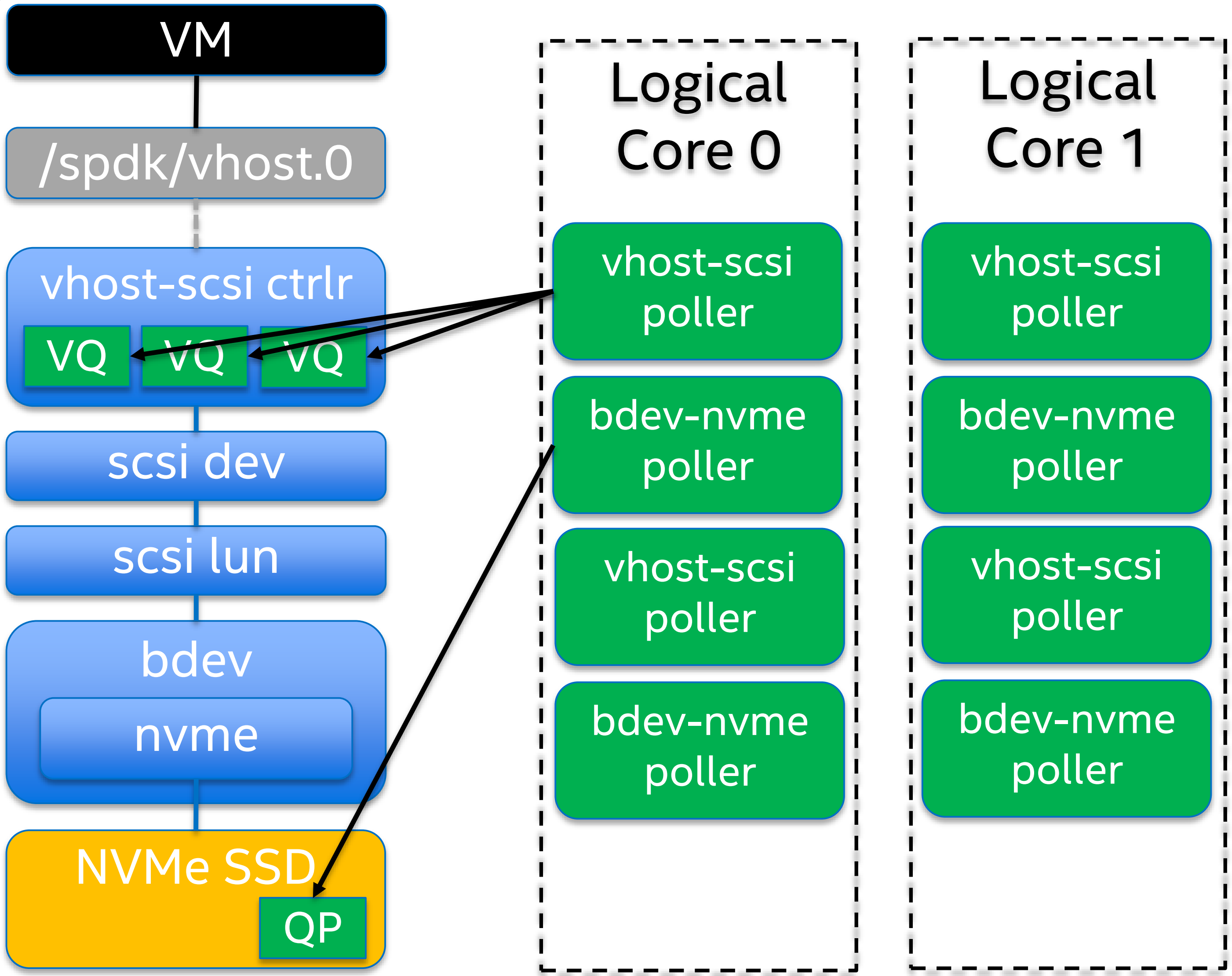
- UUID xattr for lvolstore, lvols
- Friendly names
  - lvol name unique within lvolstore
  - lvolstore name unique within application
- Future
  - snapshots (requires blobstore support)



Asynchronous Polling

Poller execution

- Reactor on each core
- Iterates through pollers round-robin
- vhost-scsi poller
  - poll for new I/O requests
  - submit to NVMe SSD
- bdev-nvme poller
  - poll for I/O completions
  - complete to guest VM


VM
/spdk/vhost.0
vhost-scsi ctrlr
VQ
VQ
VQ
scsi dev
scsi lun
bdev
nvme
NVMe SSD
QP
Logical Core 0
vhost-scsi poller
bdev-nvme poller
vhost-scsi poller
bdev-nvme poller
Logical Core 1
vhost-scsi poller
bdev-nvme poller
vhost-scsi poller
bdev-nvme poller