FTF FREESCALE TECHNOLOGY FORUM 2015

QorIQ Device Virtualization with KVM: Performance vs. Flexibility

FTF-DES-F1254

Mike Caraman, PhD | Software Architect

J U N E . 2 0 1 5

freescale™

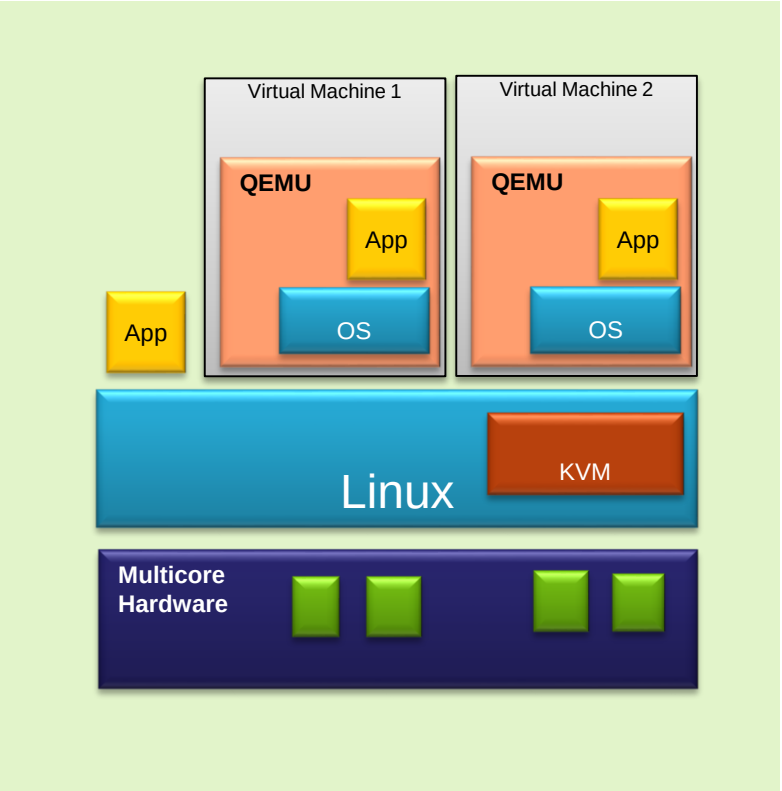
External Use

Agenda

- KVM and Device Virtualization
- Virtio Devices
- Virtio Back-ends
- Virtio Back-end Acceleration
- Device Direct-assignment
- Emulation and Pass-through
- I/O Virtualization Performance
- VM migration
- Conclusions

KVM and Device Virtualization

KVM/QEMU — Overview

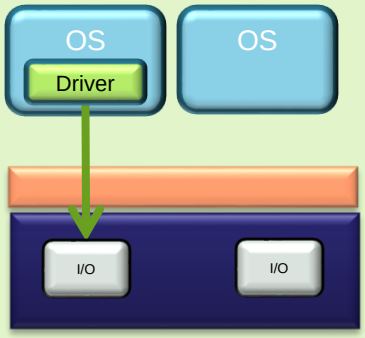


- KVM/QEMU — open source virtualization technology based on the Linux® kernel
- KVM is a Linux kernel module
- QEMU is a user space emulator that uses KVM for acceleration
- Run virtual machines alongside Linux applications
- No or minimal OS changes required
- Virtual I/O capabilities
- Device direct-assignment to VM

Device Usage in Virtual Environments

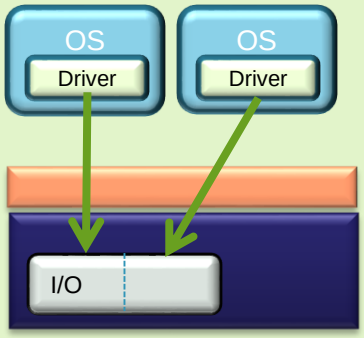
Direct Access

- Fastest native performance
- Direct access to hardware



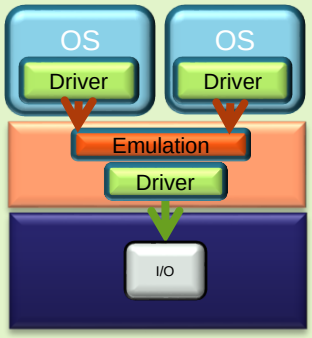
Partitionable HW

- Hardware partitioned
- One hardware block



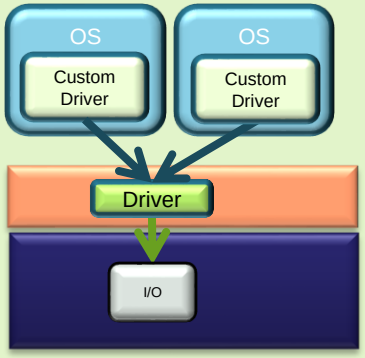
Emulated

- Driver in Hypervisor
- Emulation in Hypervisor
- Unmodified Drivers in Guest



Para-Virtualized

- Driver in Hypervisor
- Modified Drivers in Guest

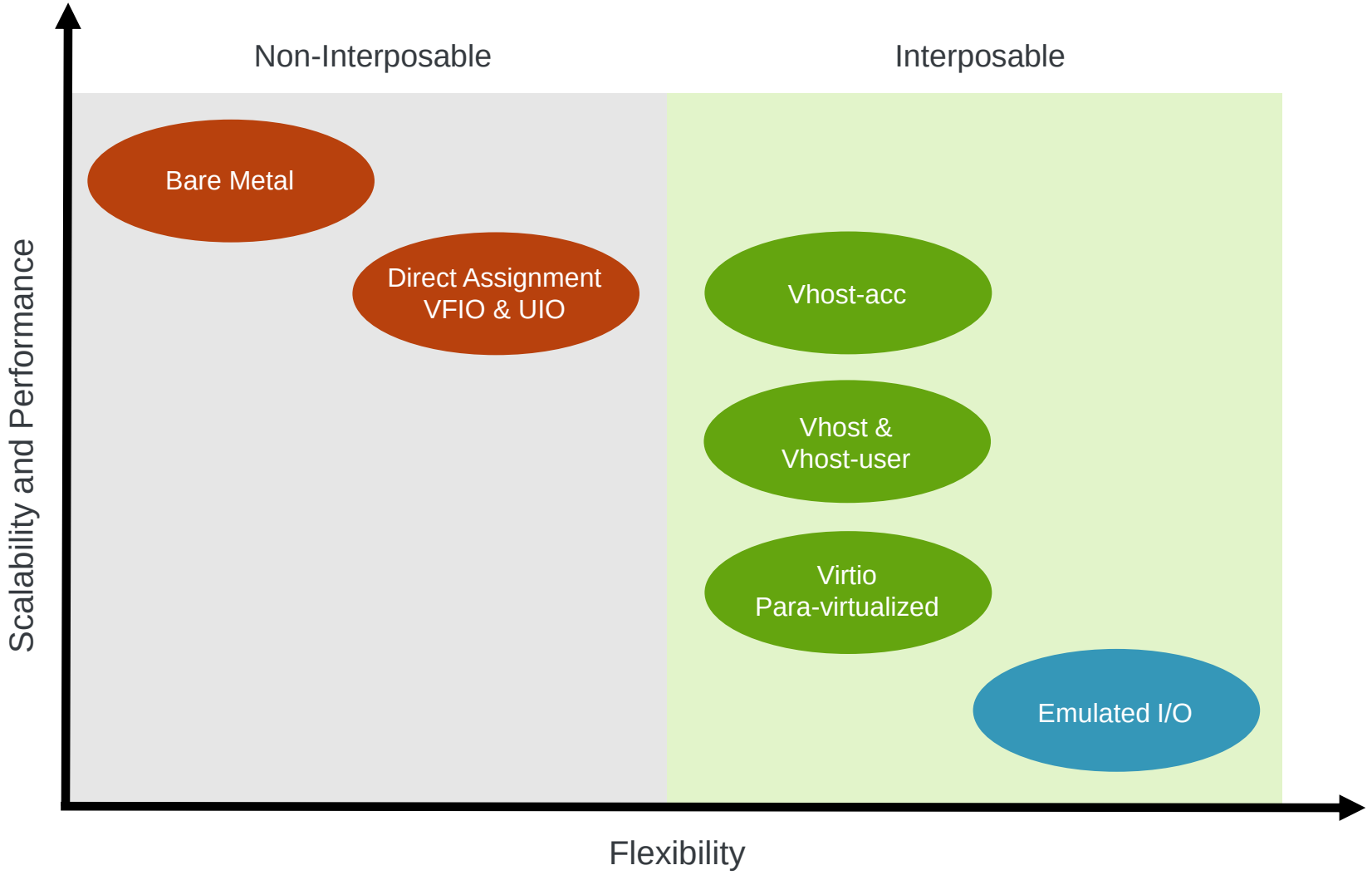


Hardware/software access

Hypercalls

Traps

I/O Virtualization — Performance vs. Flexibility



Virtio Devices

Virtual I/O Device

Virtio family of devices

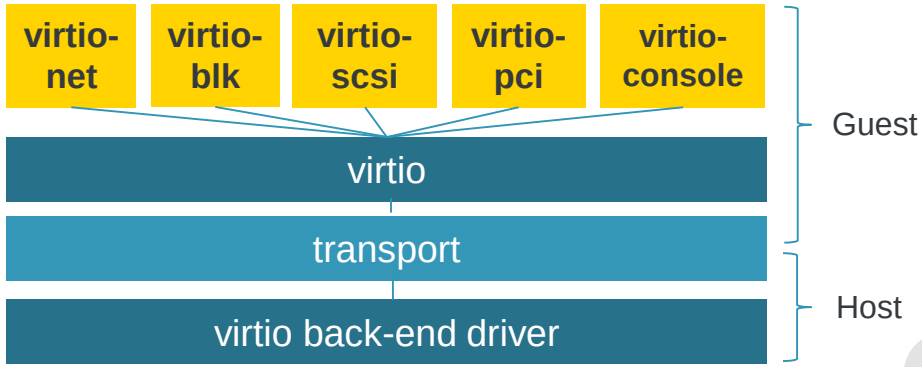
- Found in virtual environments
- By design they look like physical devices
- Use guest standard drivers and discovery mechanisms.
- Specification defined by OASIS technical committee

Virtio specification purpose

- Straightforward - use normal bus mechanisms of interrupts and DMA
- Efficient - rings of descriptors for both input and output, laid out to avoid cache effects
- Standard - makes no assumptions about guest environment beyond supporting MMIO, Channel I/O or PCI bus transports.
- Extensible - devices contain feature bits acknowledged by the guest OS

Virtio device facilities

- Device status field
- Feature bits
- Device Configuration space
- One or more virtqueues

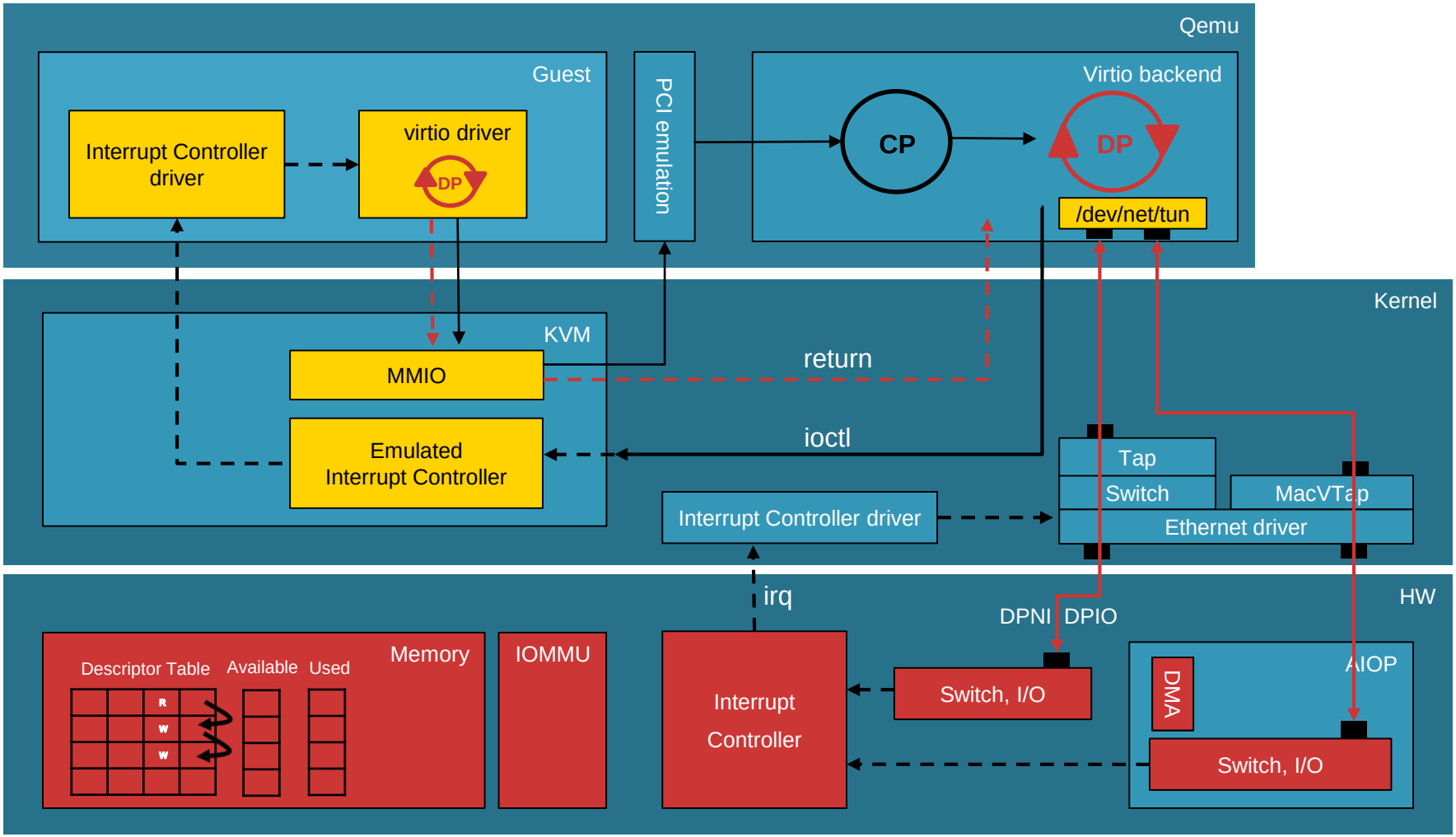


Virtio Back-ends

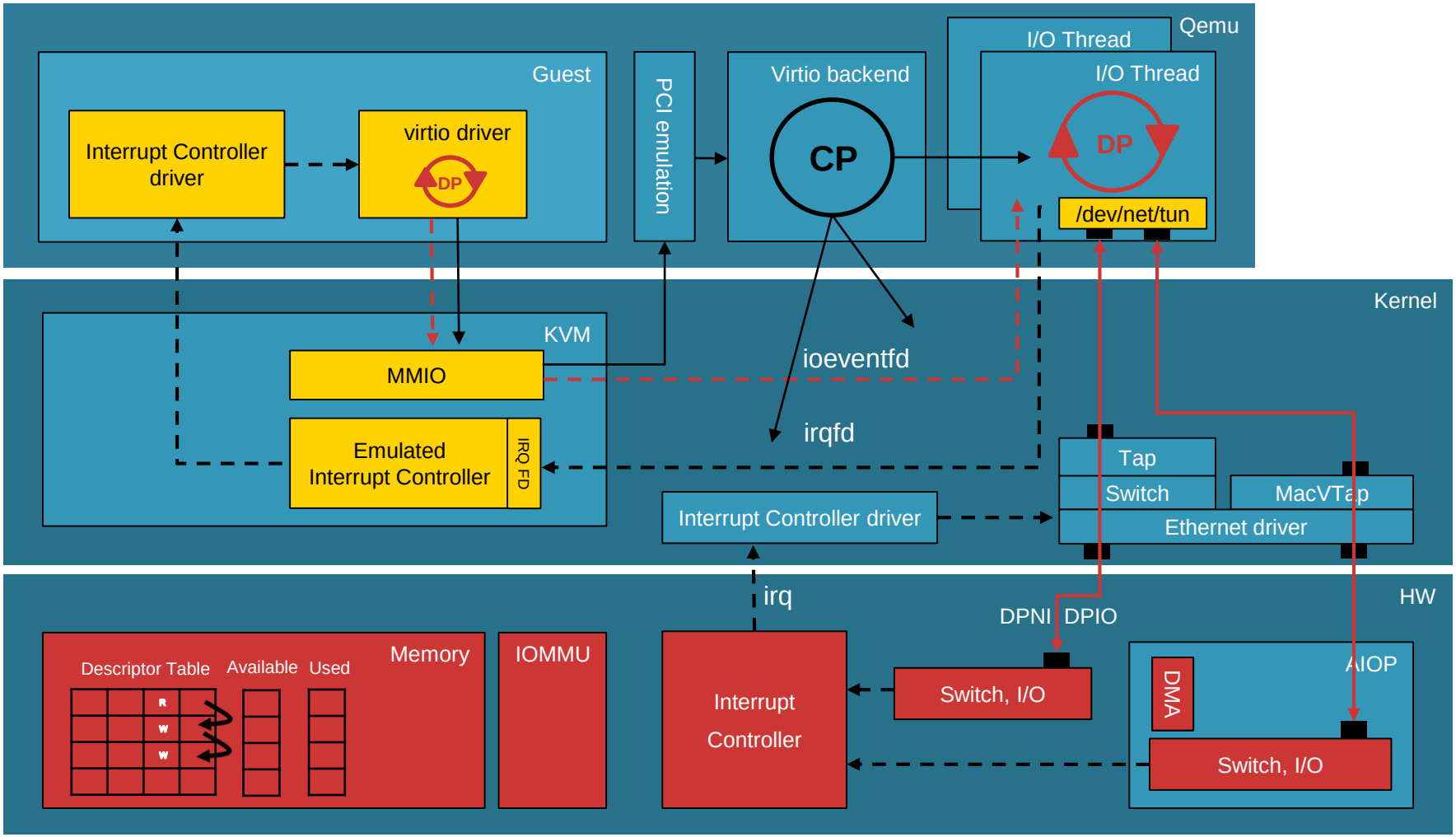
Virtio front-end and back-end drivers

Virtio font-end	Qemu/KVM back-ends	
virtio-net	virtio-net (legacy)	Qemu
	virtio-net (data-plane)	Qemu, I/O thread
	vhost	Kernel
	vhost-user	User space
virtio-blk	virtio-blk	Qemu
	virtio-blk data-plane	Qemu, I/O thread
virtio-scsi	virtio-scsi	Qemu
	vhost-tcm	Kernel

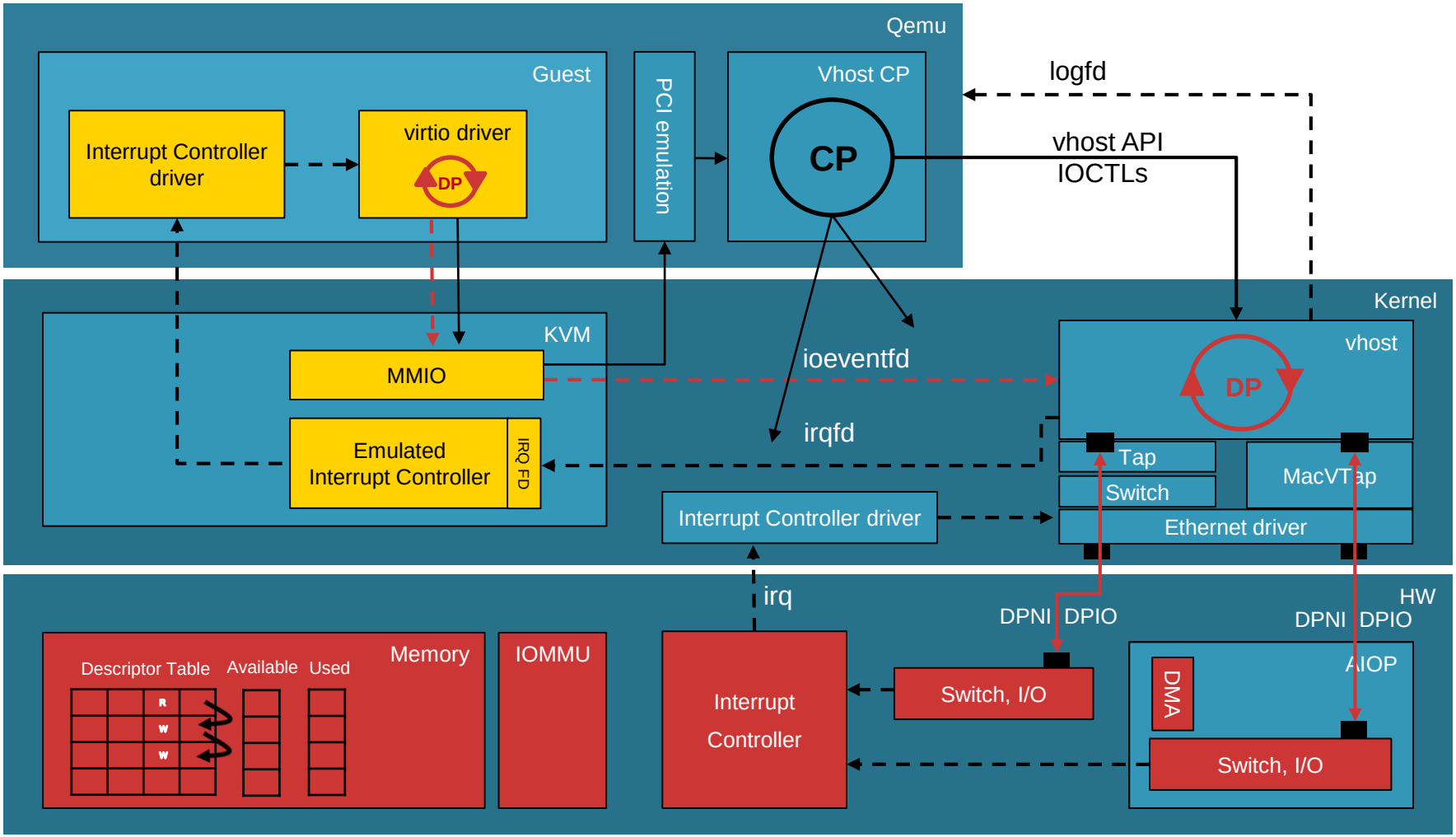
Virtio-net back-ends: virtio-net (legacy)



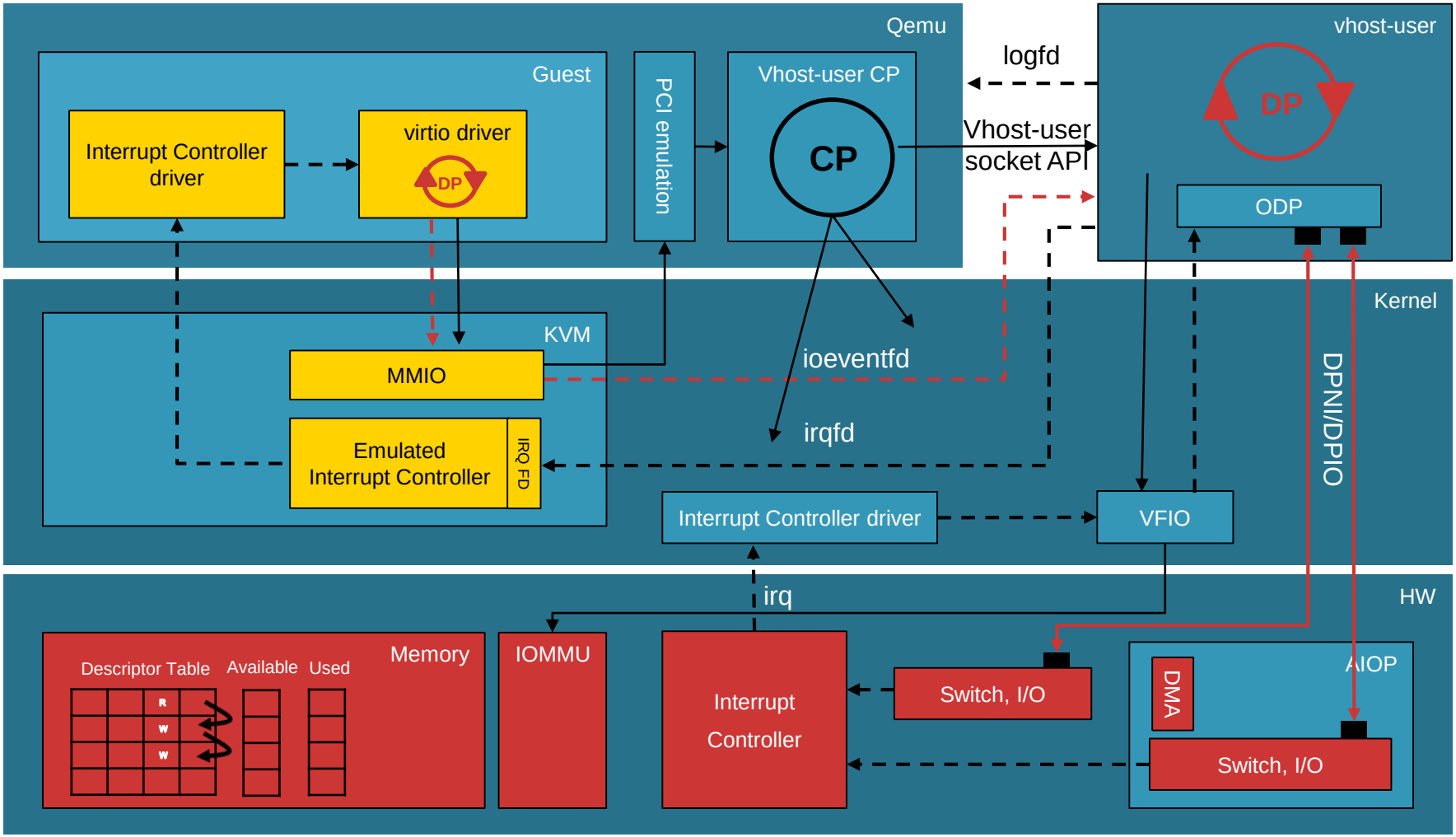
Virtio-net back-ends: virtio-net (data-plane)



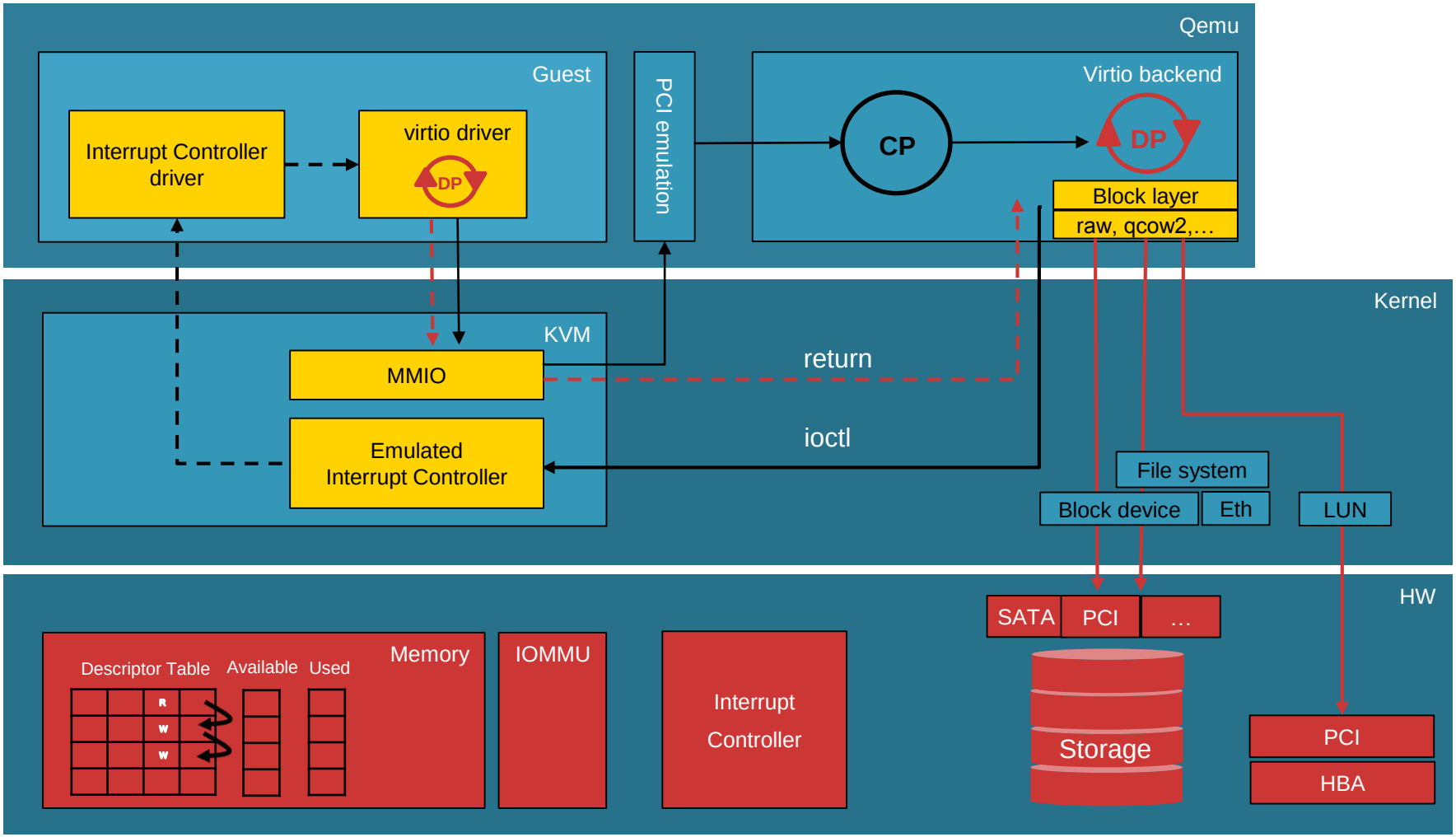
Virtio-net back-ends: vhost



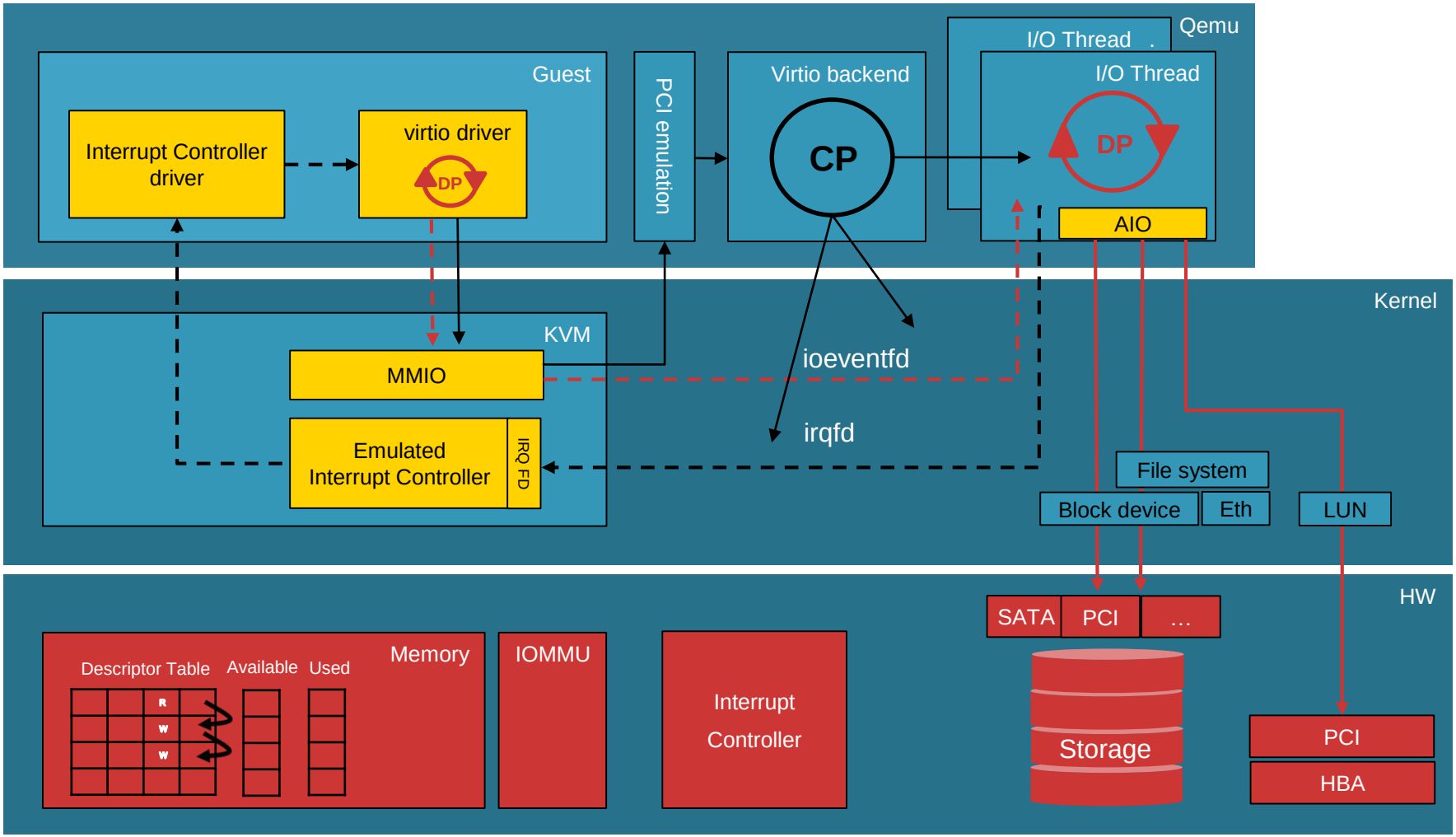
Virtio-net back-ends: vhost-user



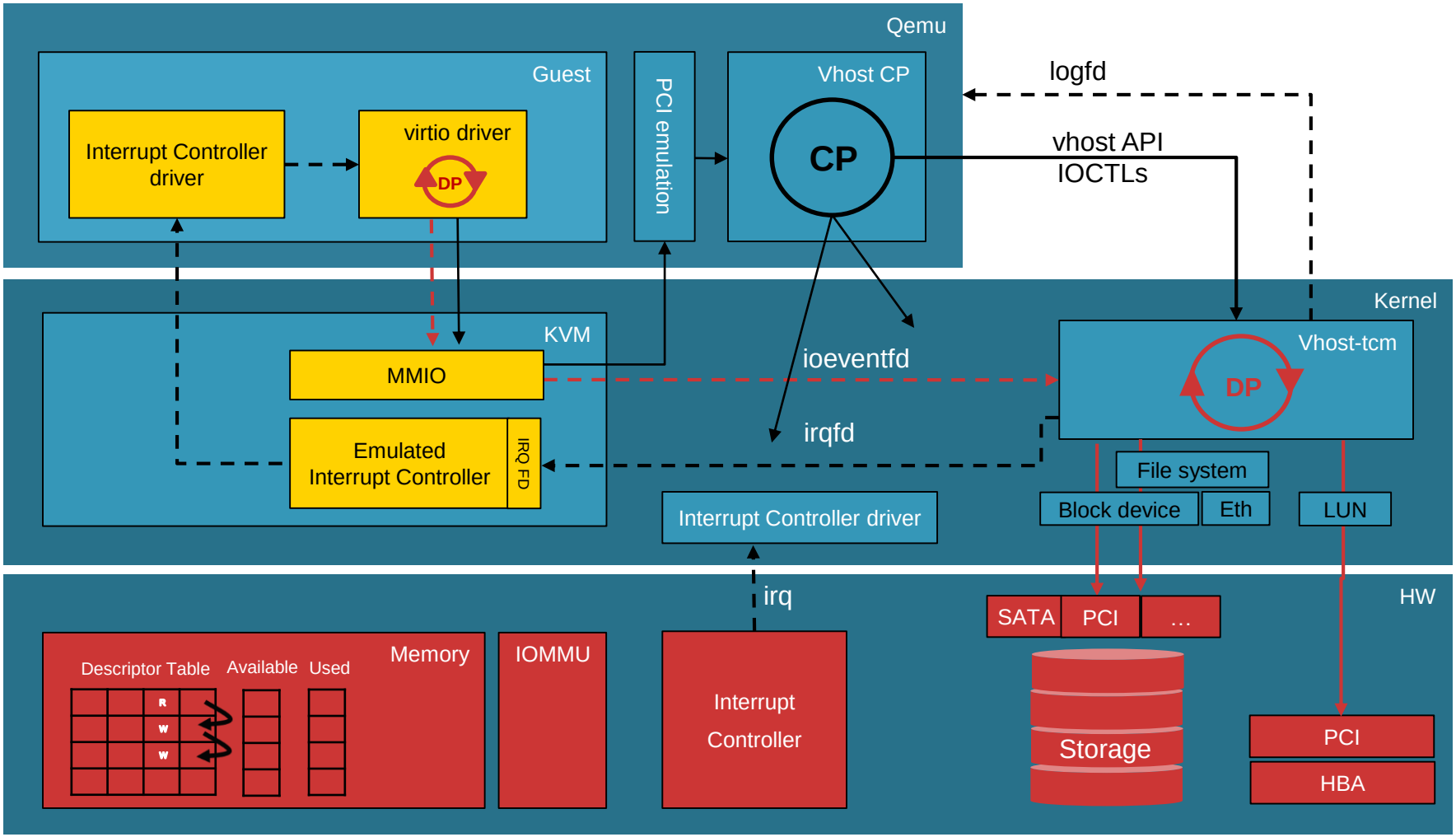
Virtio-blk back-ends: virtio-blk



Virtio-blk back-ends: virtio-blk data-plane

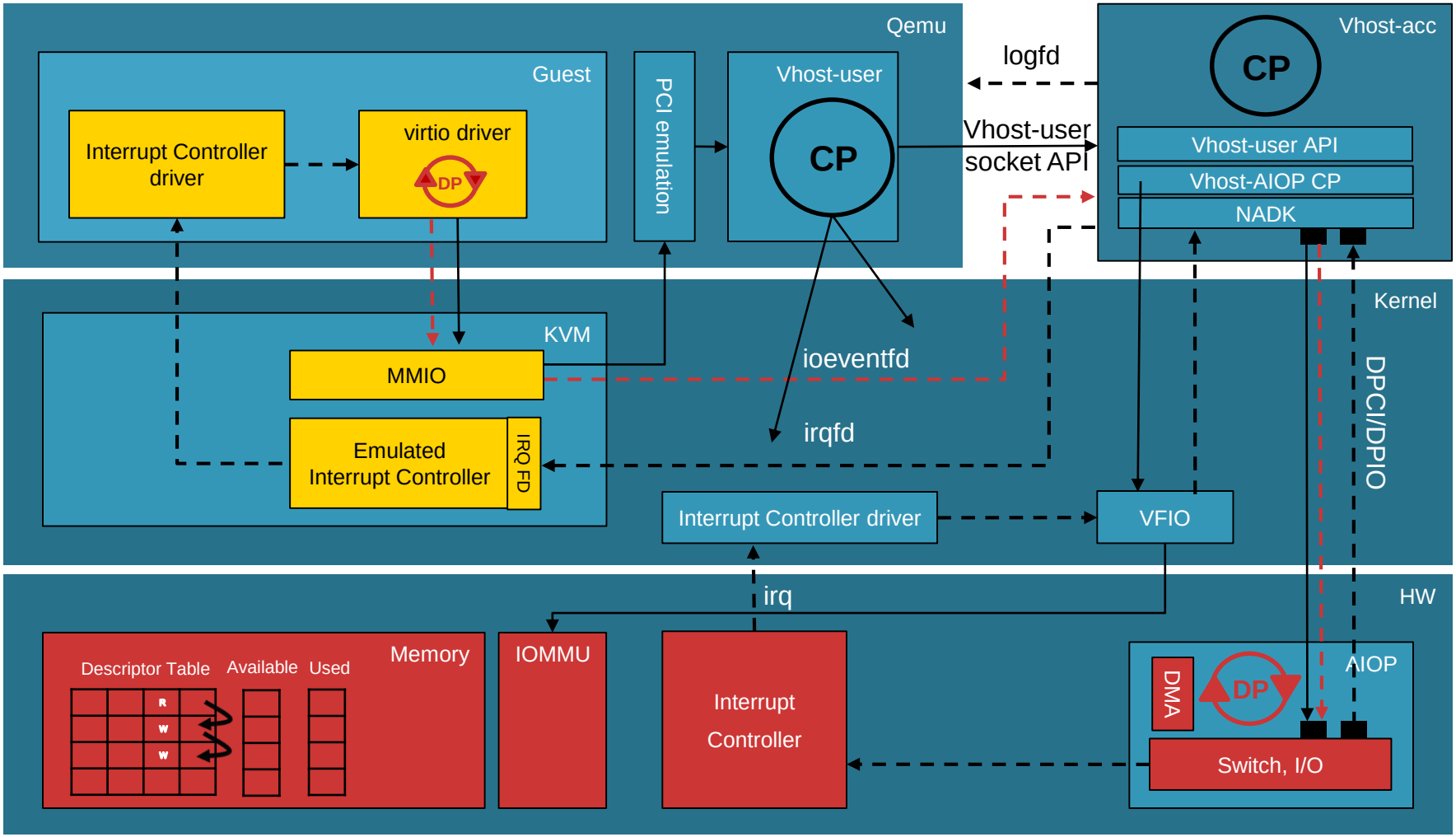


Virtio-scsi back-ends: vhost-tcm



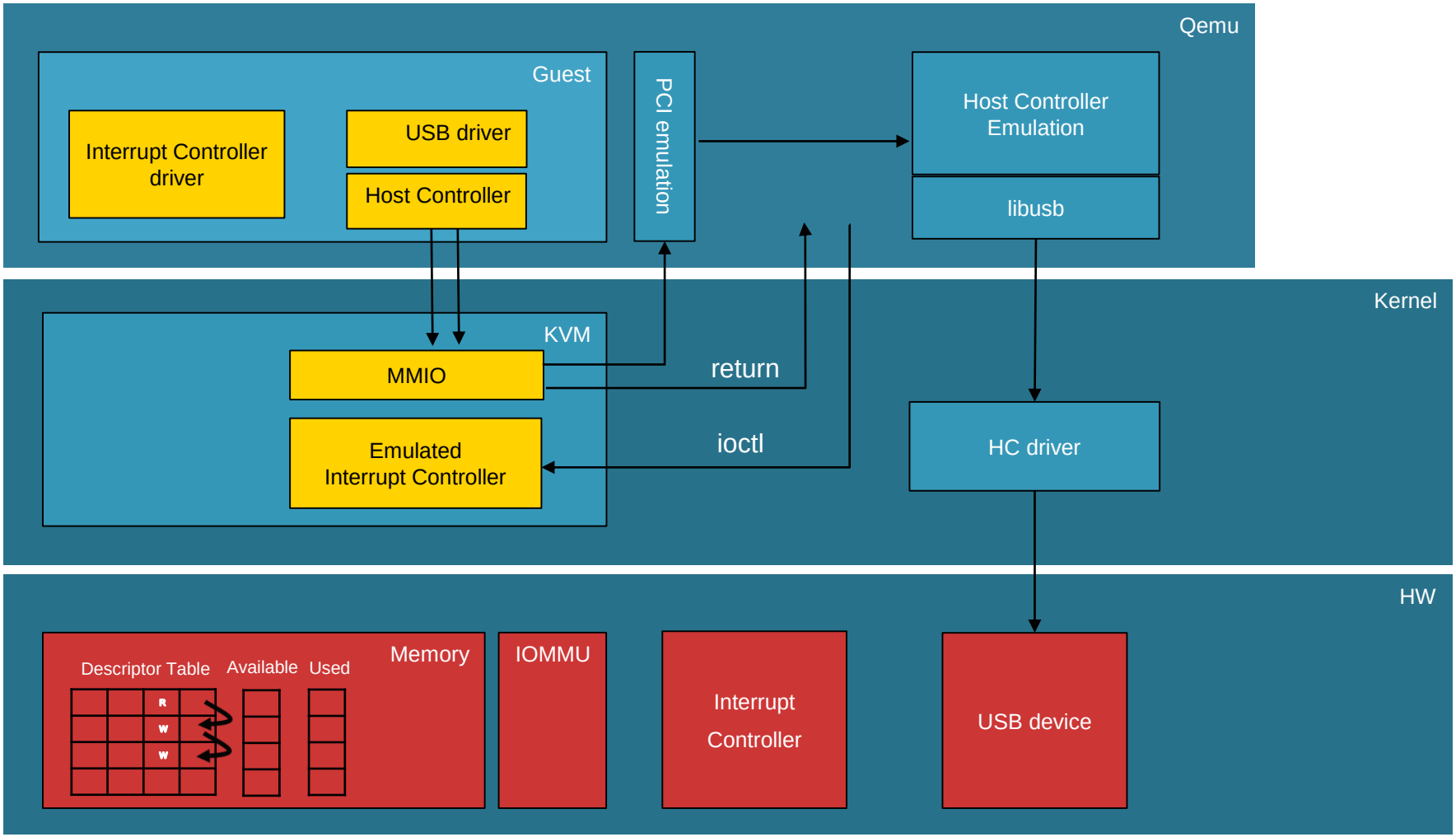
Virtio Back-end Acceleration

Virtio-net back-end: vhost-acc (preliminary design)



Emulation and pass-through

USB pass-through

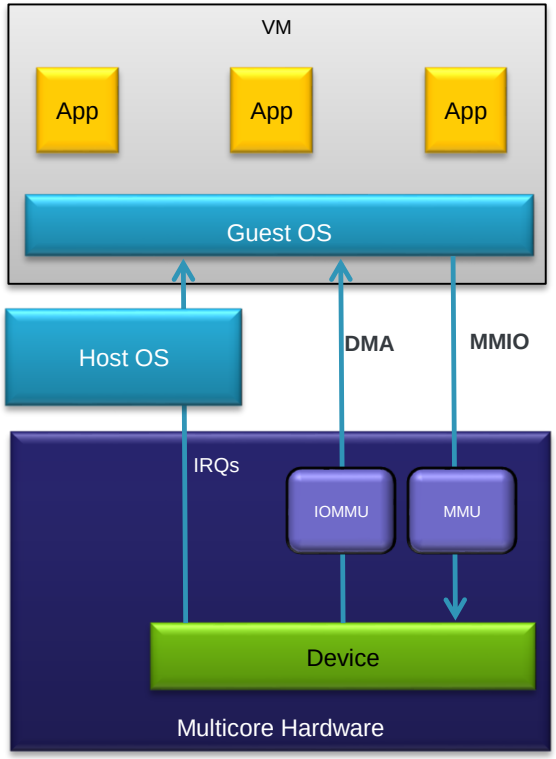


Device Direct-Assignment

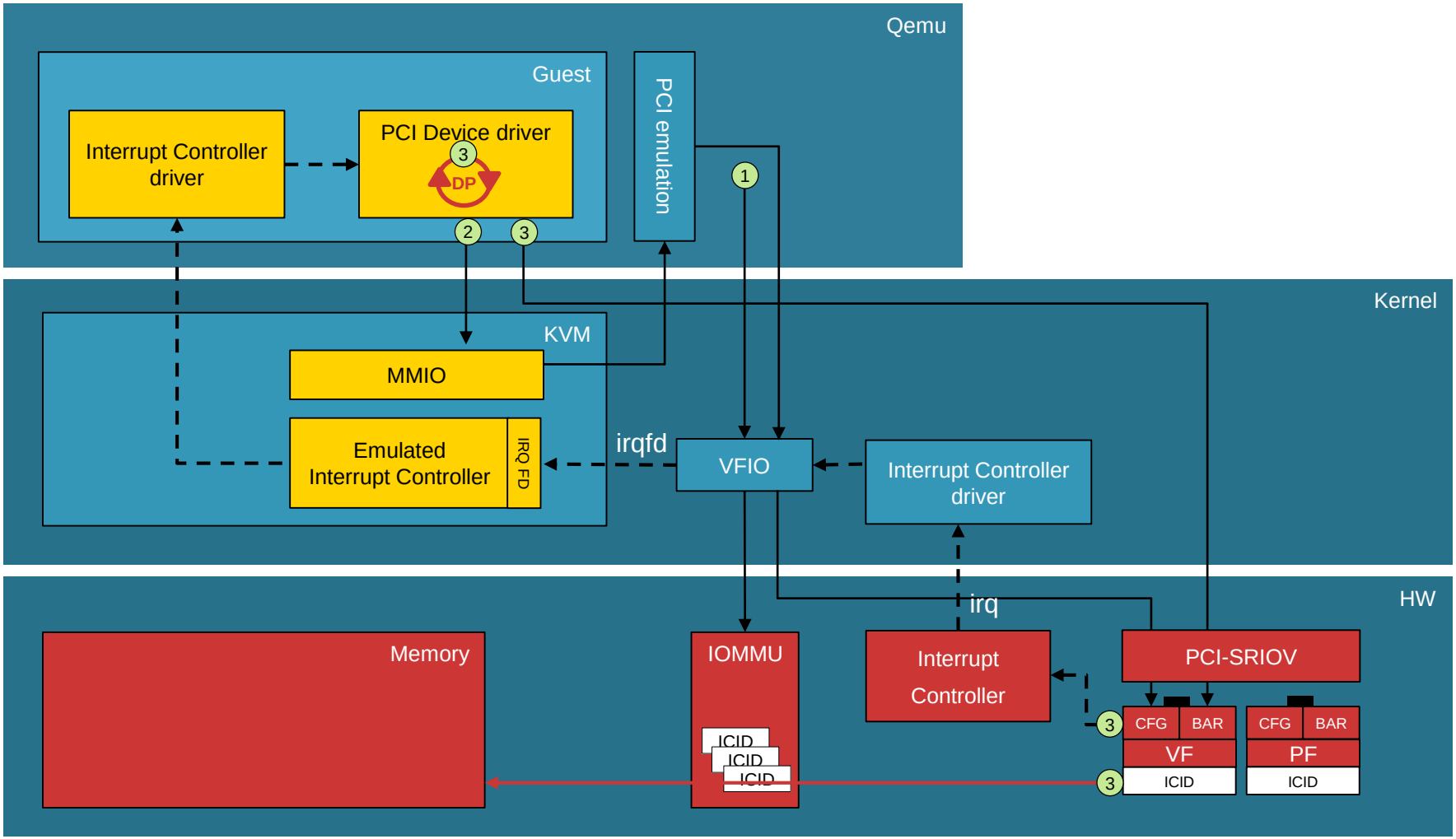
Device Direct-Assignment with VFIO

- VFIO (Virtual Function IO)
  - Linux userspace driver infrastructure
  - Enforces IOMMU protection
- VFIO Provides
  - Device access (mmap device MMIO regions)
  - IOMMU programming interface
  - High performance interrupt support

- VFIO PCI
  - VFIO abstracts devices as Regions and IRQs
  - Regions include
    - PCI configuration space
    - MMIO and I/O port BAR spaces
    - MMIO PCI ROM access
  - IRQs include
    - INTx (legacy interrupts)
    - Message Signaled Interrupts (MSI & MSI-X)

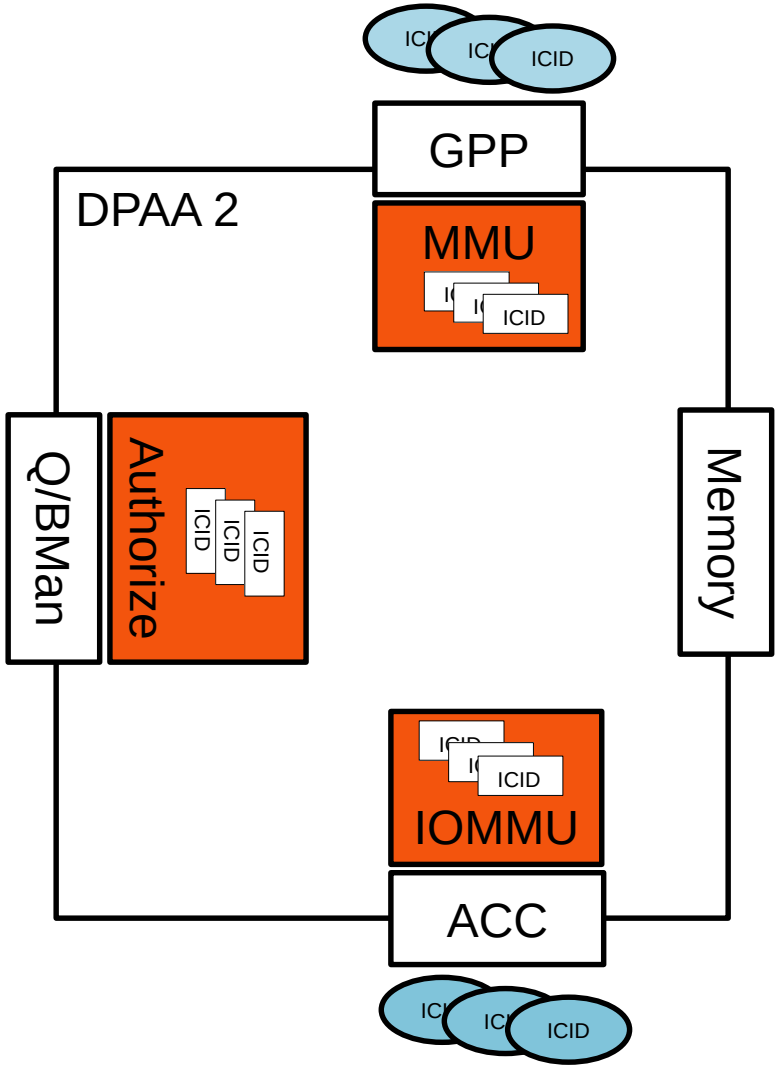


VFIO for PCI Bus

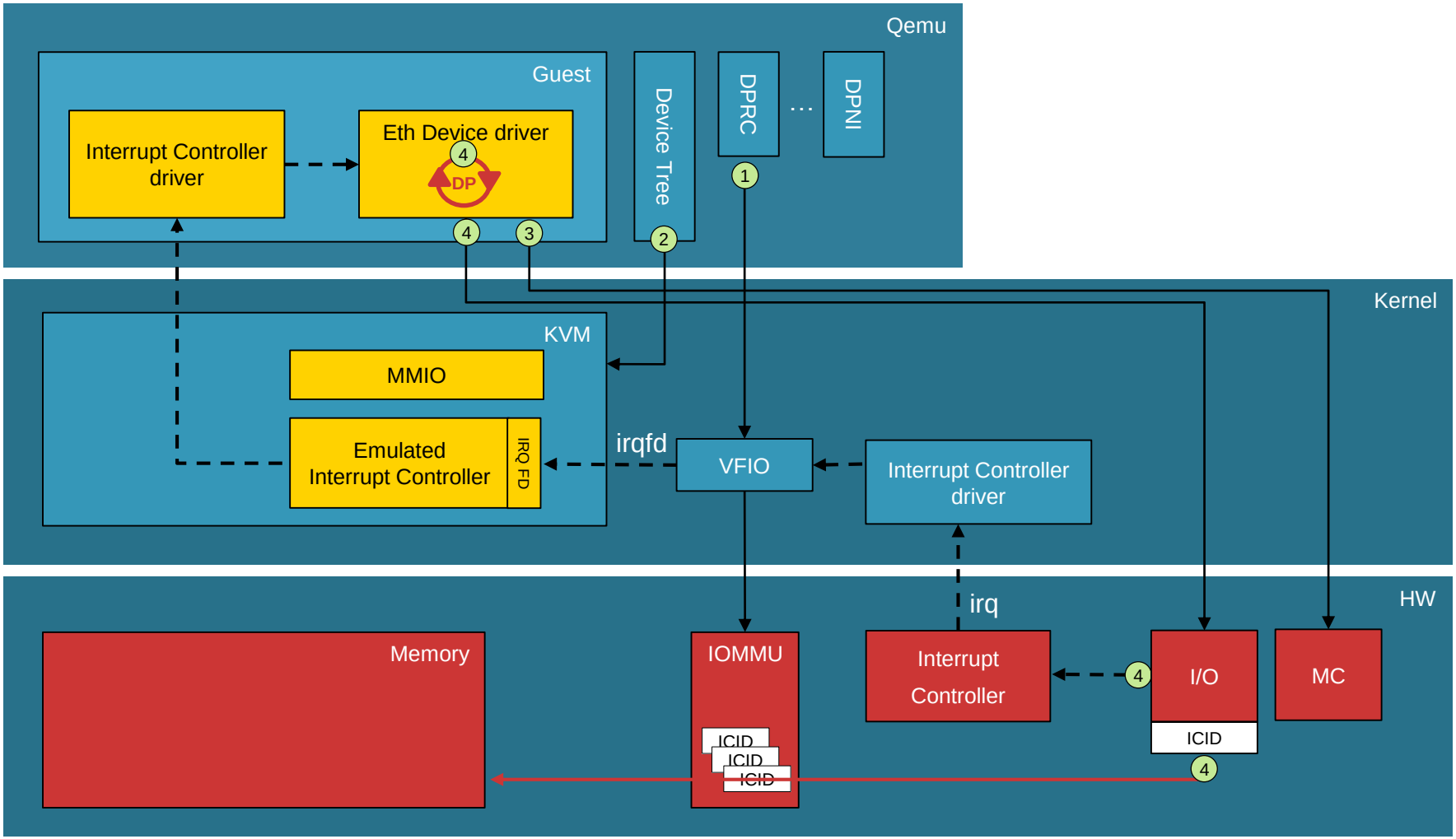


DPAA 2 Secure Direct Assignment

- DPAA 2 architecture
  - Optimized for resource assignment to various software contexts through Management Complex
    - Linux MC bus
    - Resource management tool
  - IOMMU translation and protection for user-space (ODP and QEMU)
    - ICID (StreamID)
    - MC bus integration with VFIO
    - Device reset
  - DPAA secured with Authorization Tables



VFIO for MC Bus



I/O Virtualization Performance

NetPerf UDP Tx and Rx

Environment

- 2 T4240 boards, connected back to back through 10G XAUI — “server” and “client”
- All tests run on client — server only used as the other endpoint, no monitoring
- GCC4.7.3 (-O3 -mcpu=e6500 -m32 -mno-altivec)
- Different Tx / Rx setups
- PCD Rx classification + interrupt steering on host
- Scatter / gather support
- GRO + GSO activated

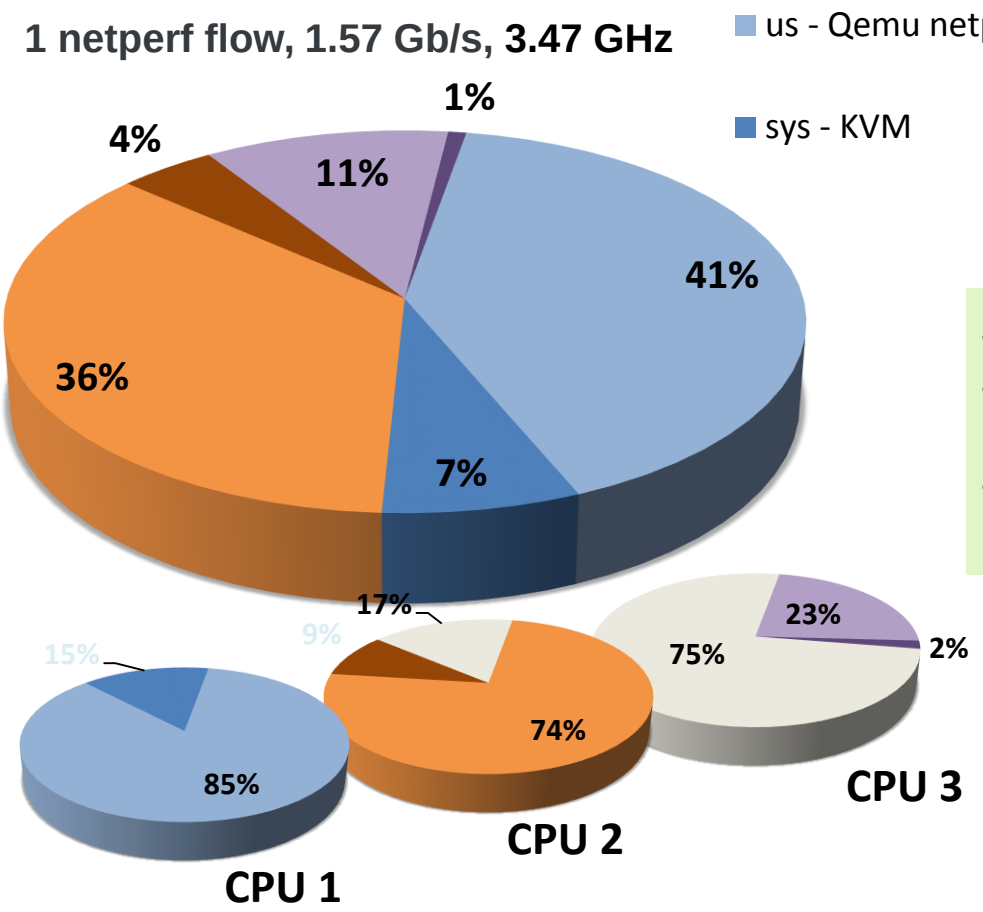
Guest Setup

- 1 GB memory
- Hugepage backed
- **Macvtap + vhostnet + virtio**
- Multiple single-queue interfaces
- VCPU, vhostnet process affinity

Netperf:

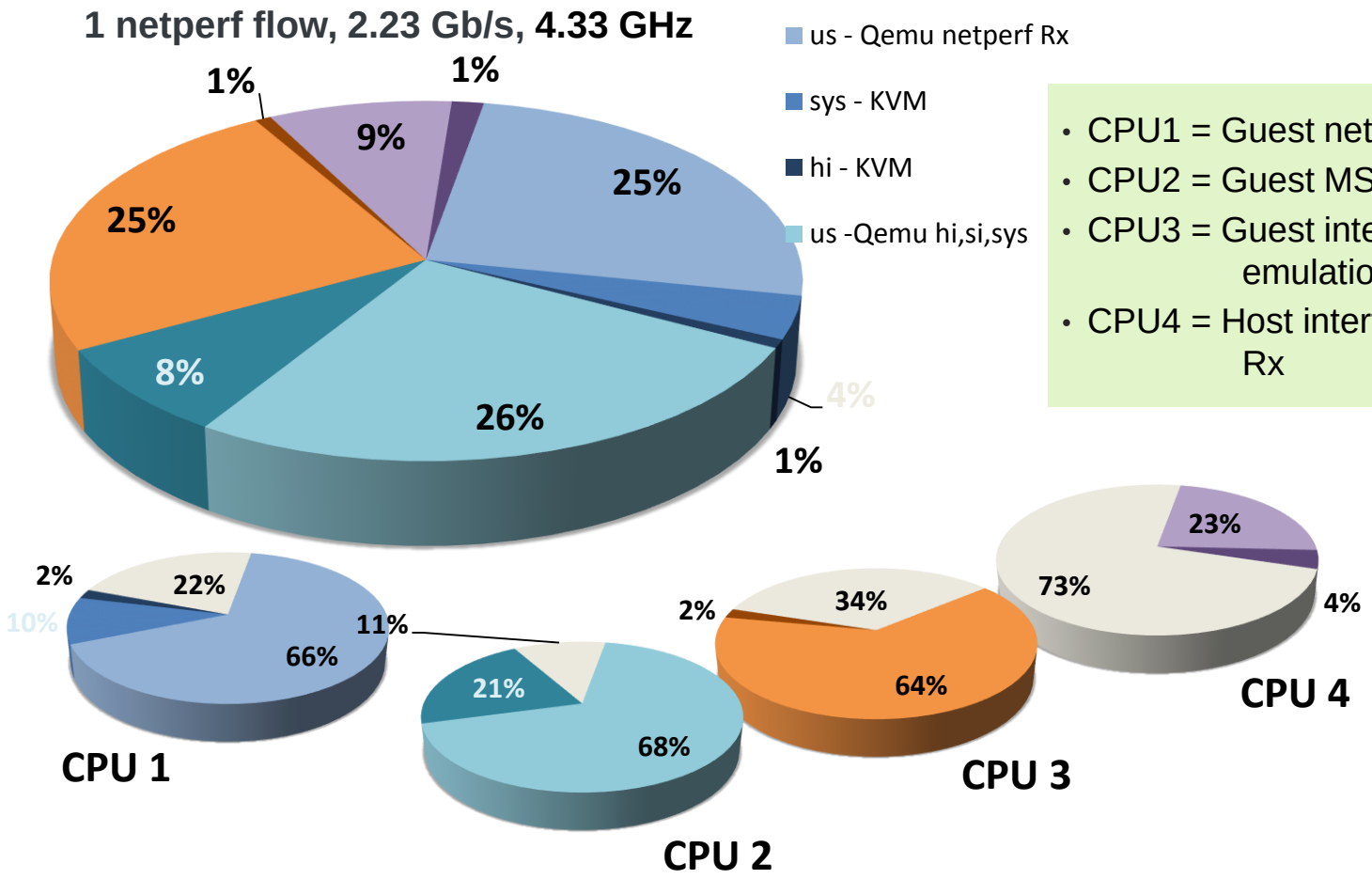
- UDP_STREAM
- Unidirectional
- Message size: 1472
- Core affinity

NetPerf UDP Tx with vhost: CPU Breakdown



- CPU1 = Guest netperf
- CPU2 = Guest interface emulation
- CPU3 = Host interface Tx confirmation

NetPerf UDP Rx with vhost: CPU Breakdown

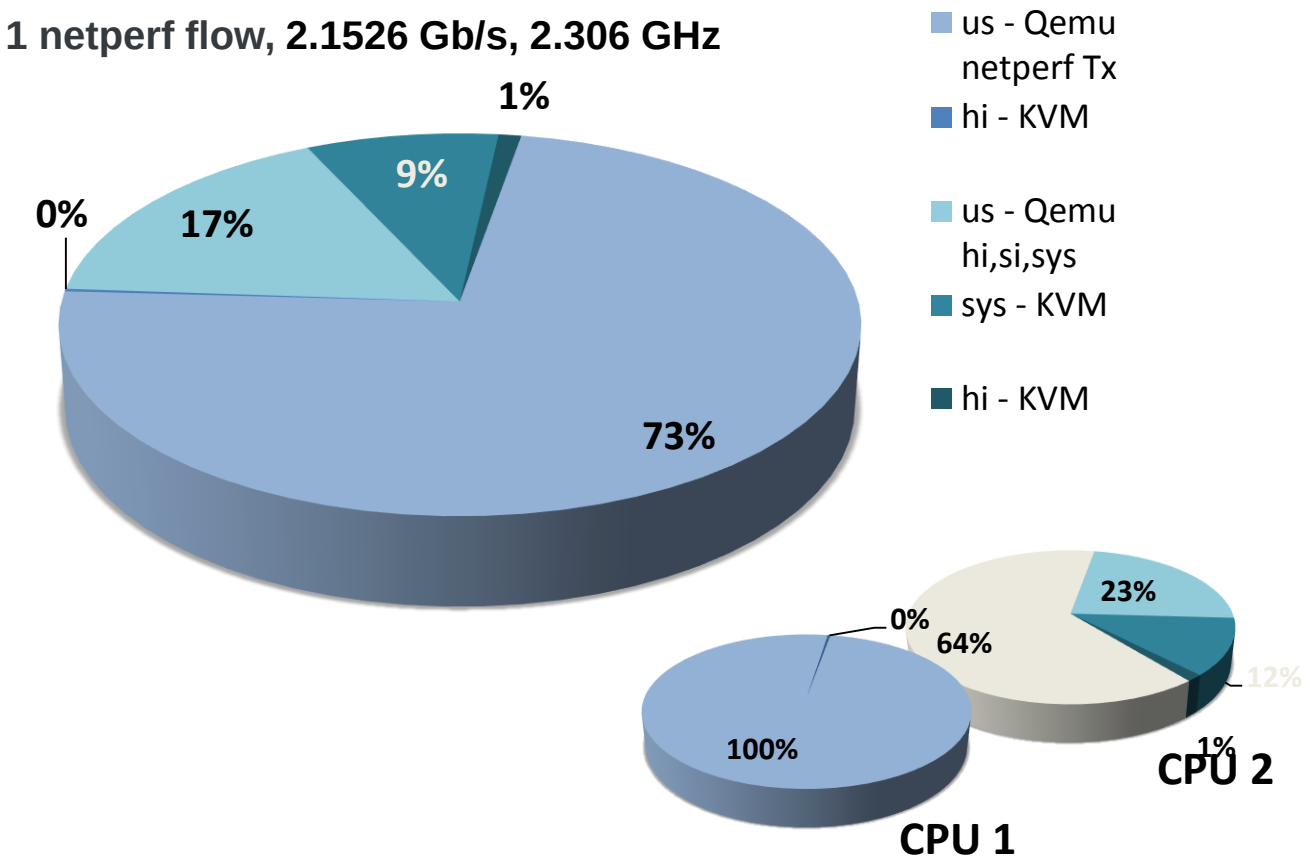


- CPU1 = Guest netperf
- CPU2 = Guest MSI interrupts
- CPU3 = Guest interface emulation
- CPU4 = Host interface Rx

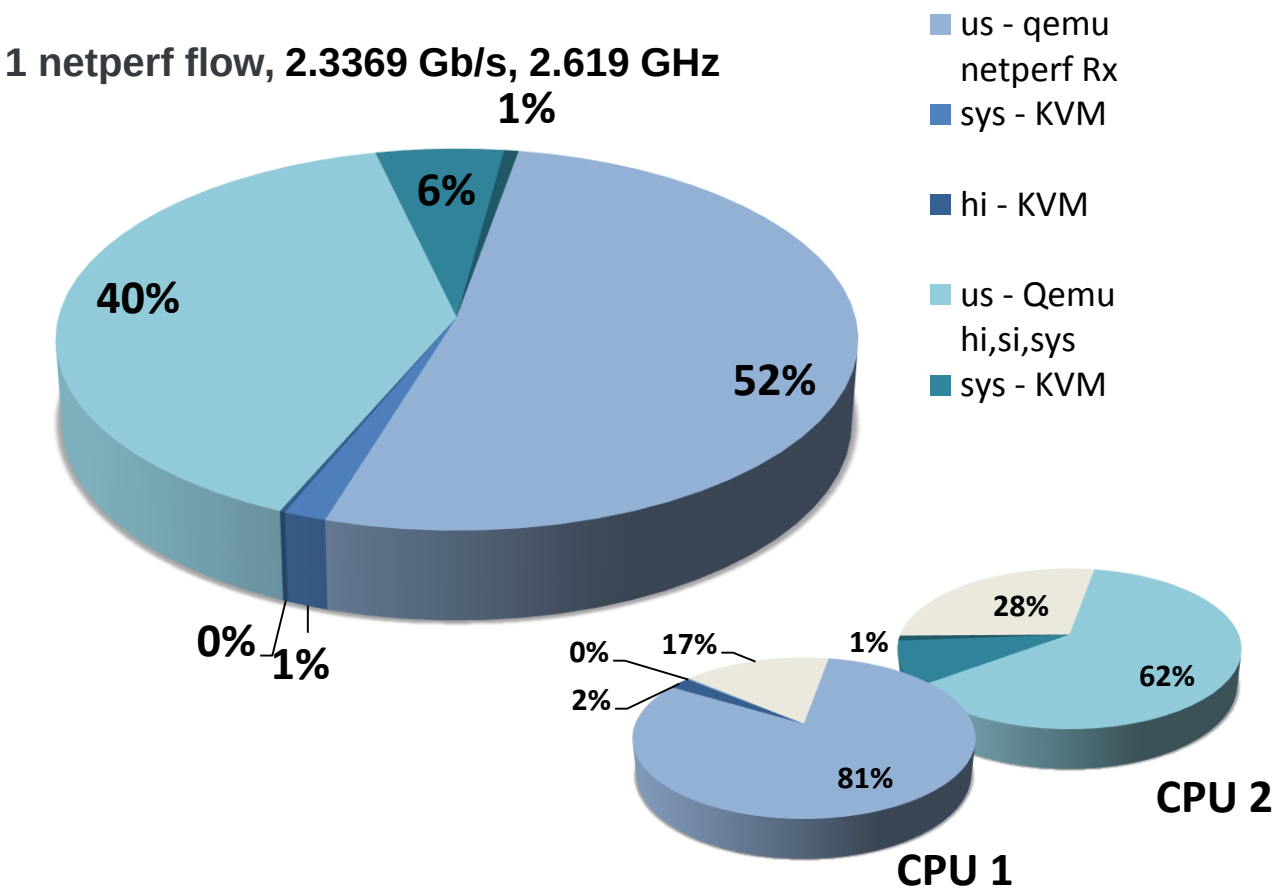
NetPerf UDP Results — Cores to Reach Line Rate (10G)

	Throughput (Mb/s)	Core load (HW Threads)	Degradation (%)
Tx physical interface	9550.4	6.23	-
Tx macvlan	9506.2	8.8	41.79
Tx guest (using macvtap)	9470	10.3	110.27
Rx physical interface	9570.4	4.73	-
Rx macvlan	9583.1	4.79	1.21
Rx guest (using macvtap)	9577.5	10.9	130.21

NetPerf Tx with VFIO PCI - CPU Breakdown



NetPerf Rx with VFIO PCI - CPU Breakdown



NetPerf TCP stream

Environment

- 2 T4240 boards, connected back to back through 10G XAUI — “server” and “client”
- All tests run on client — server only used as the other endpoint, no monitoring
- GCC4.7.3 (-O3 -mcpu=e6500 -m32 -mno-altivec)
- Different Tx / Rx setups
- PCD Rx classification + interrupt steering on host
- Scatter / gather support
- GRO + GSO activated

Guest Setup

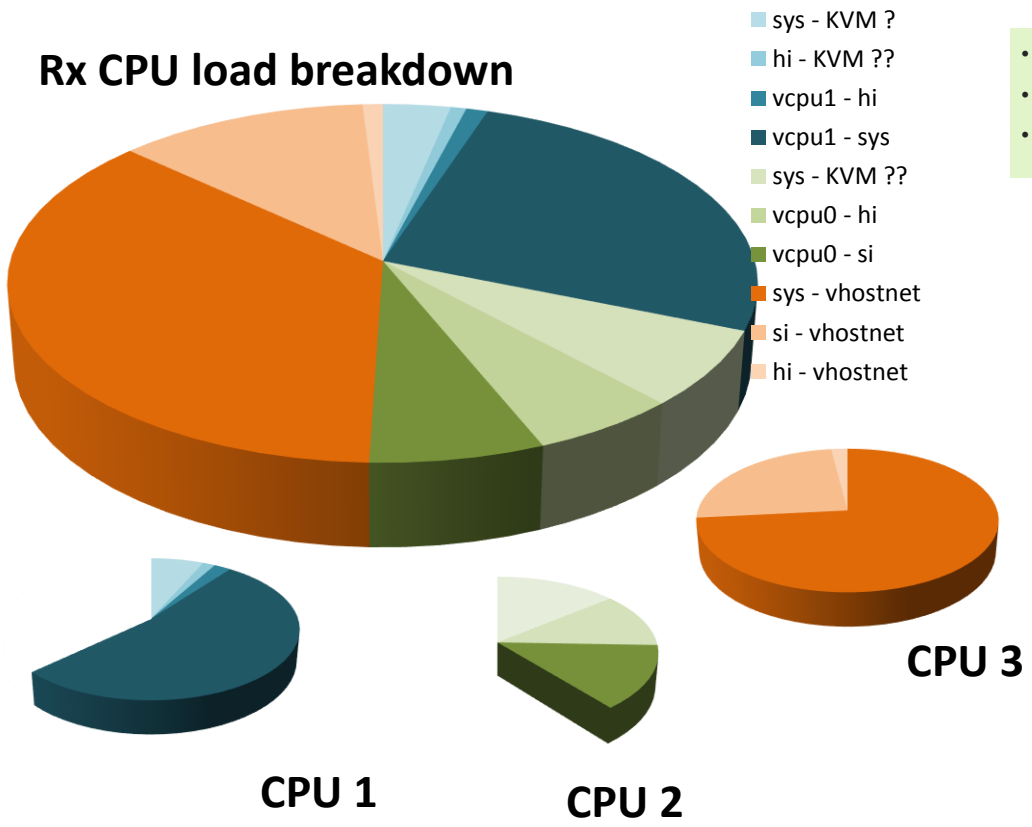
- 1 GB memory
- Hugepage backed
- **Macvtap + vhostnet + virtio**
- Multiple single-queue interfaces
- VCPU, vhostnet process affinity

Netperf:

- TCP_STREAM
- Core affinity

NetPerf TCP stream with vhost (without MSI affinity)

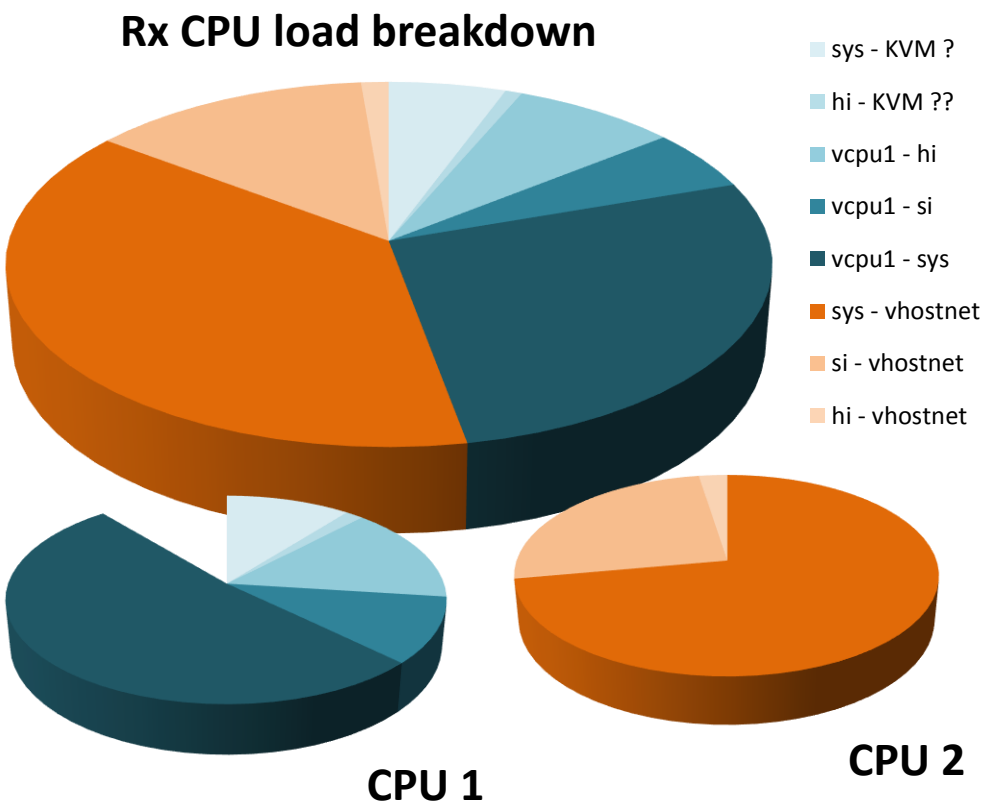
1 TCP_STREAM netperf flow, 2.49 Gb/s, 3.82 GHz



- CPU1 = Guest netperf
- CPU2 = Guest MSI interrupts
- CPU3 = Guest interface emulation

NetPerf TCP stream with vhost (with MSI affinity)

1 TCP_STREAM netperf flow, 2.59 Gb/s, 3.36 GHz

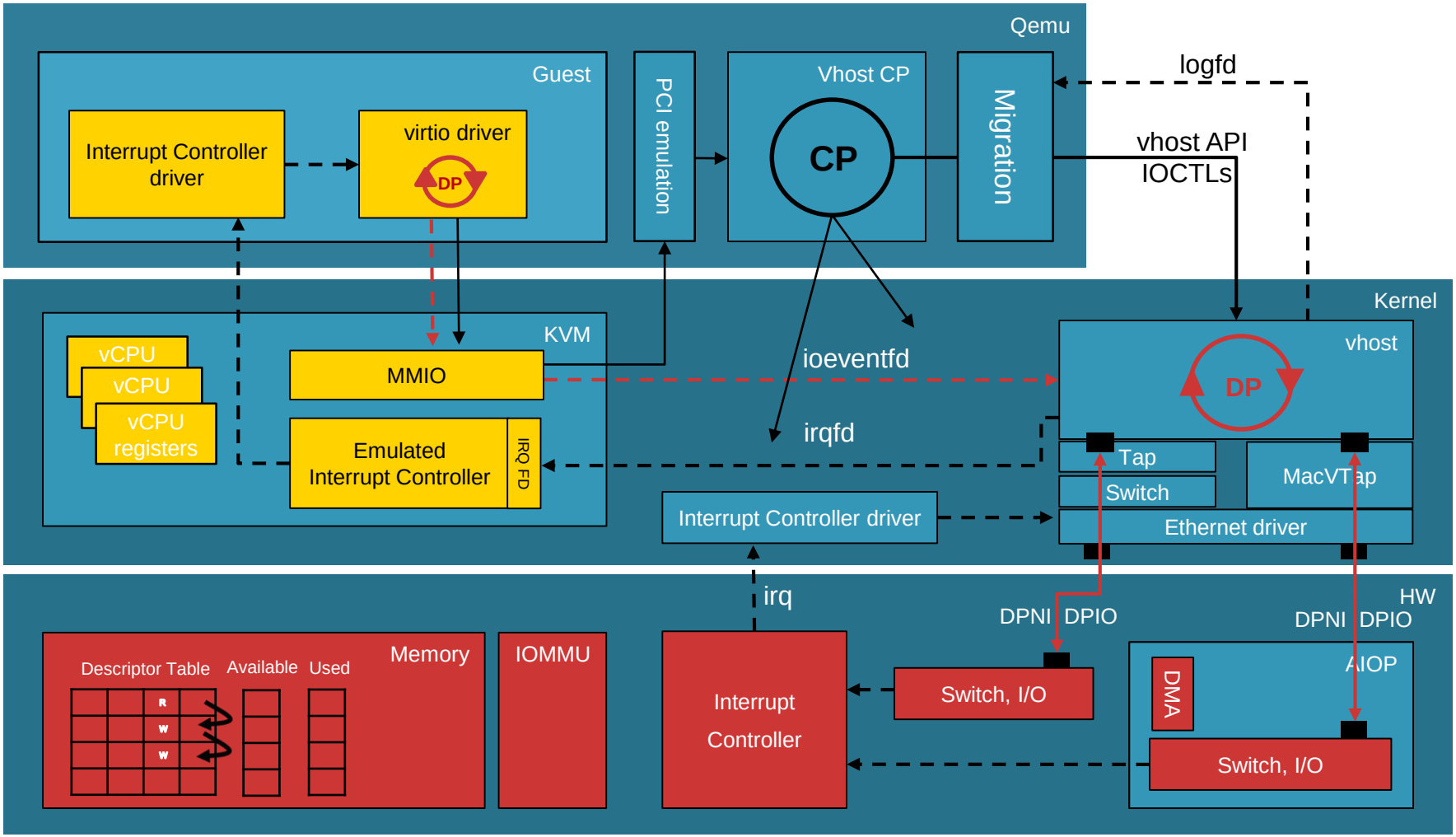


- CPU1 = Guest netperf
- CPU2 = Guest MSI interrupts + interface emulation

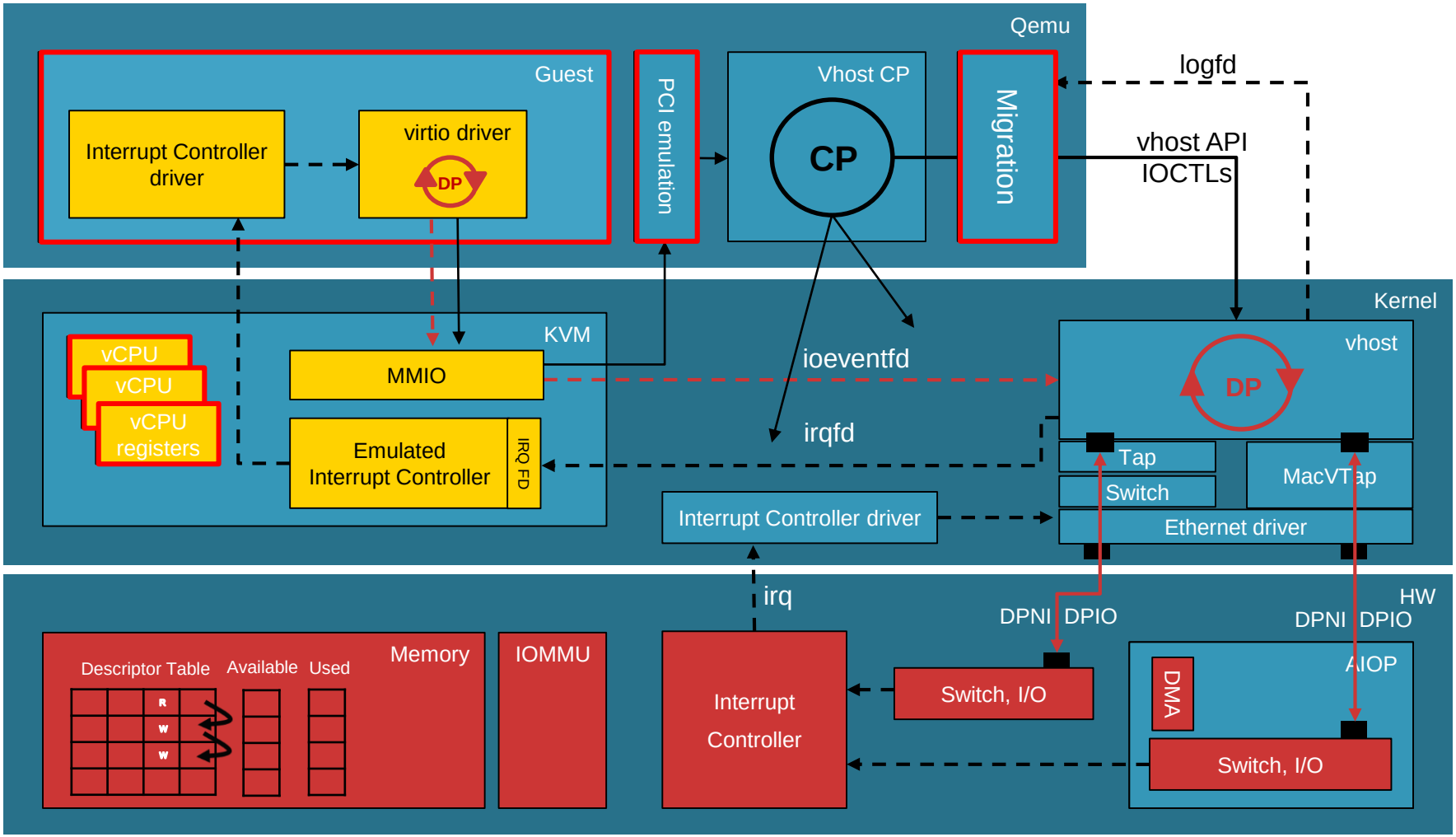
MSI affinity b/Hz Gain: **18%**

VM Live Migration

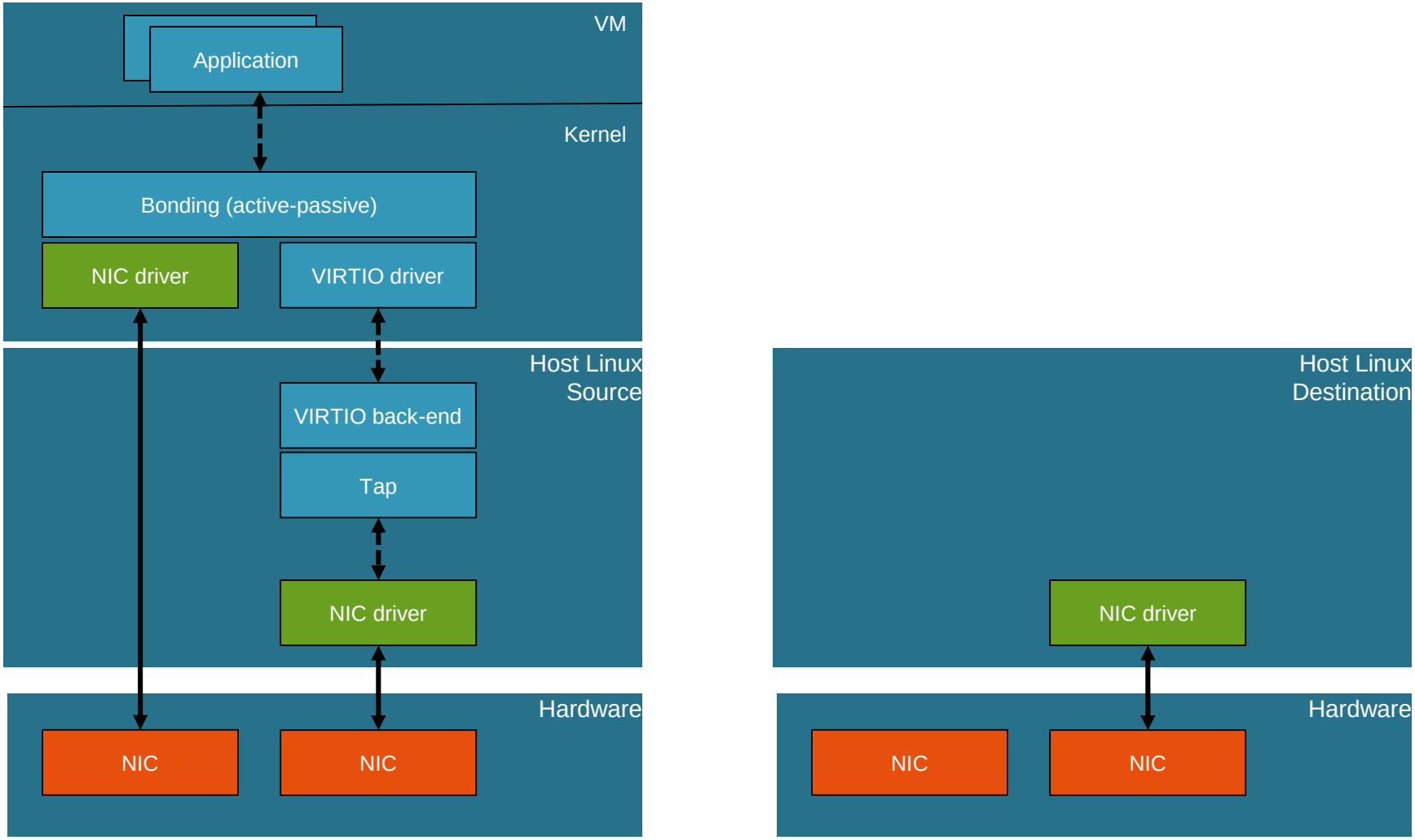
VM Live Migration with virtio devices



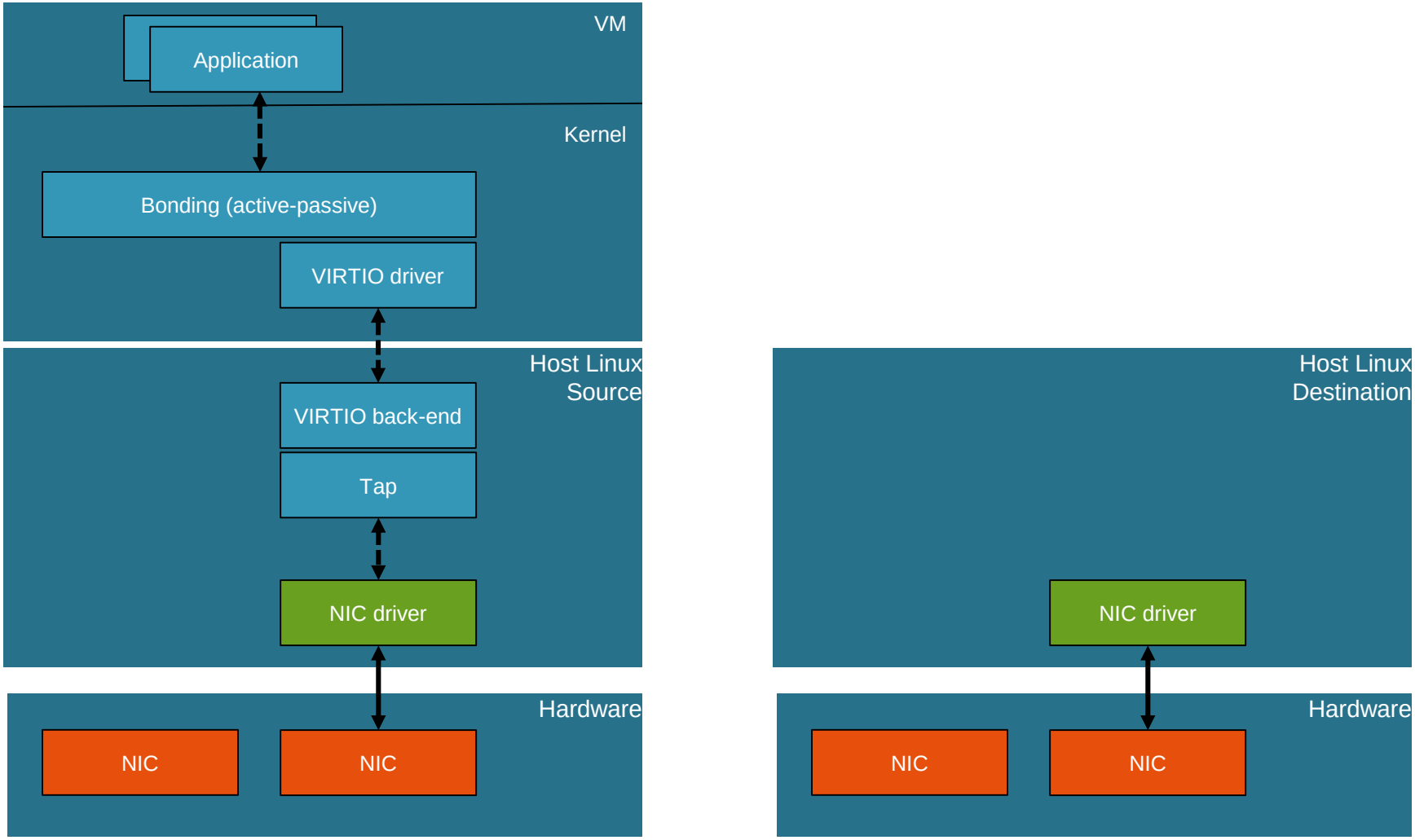
VM Live Migration with virtio devices



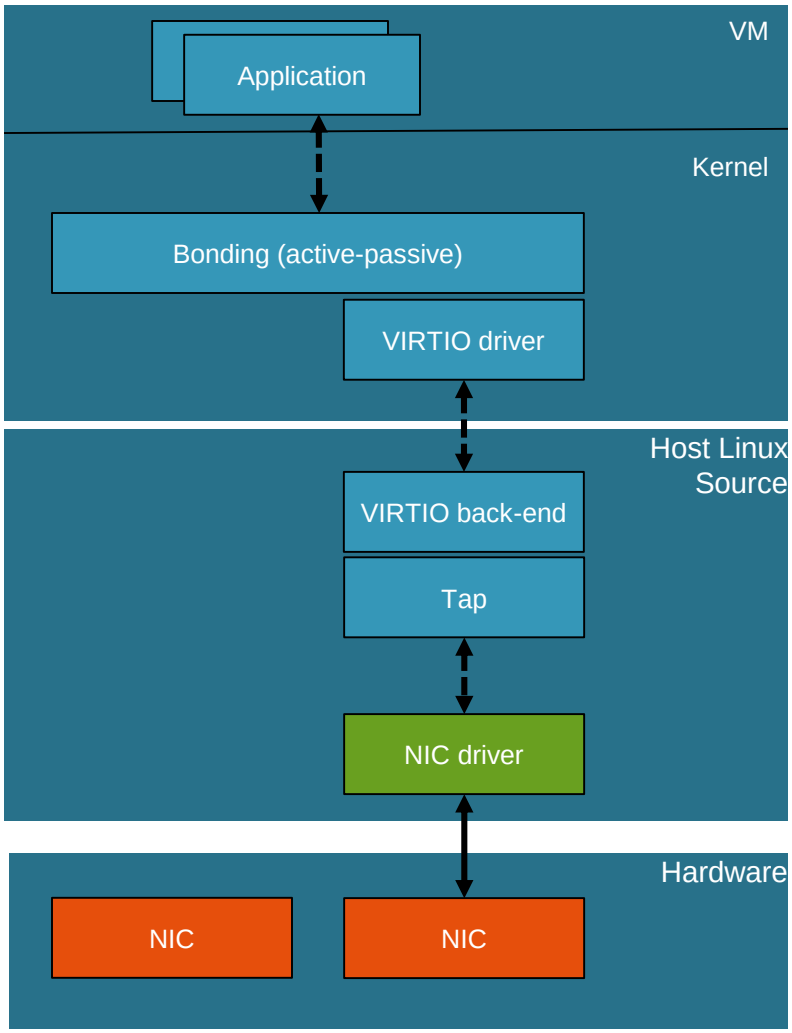
VM Live Migration with Direct Assignment

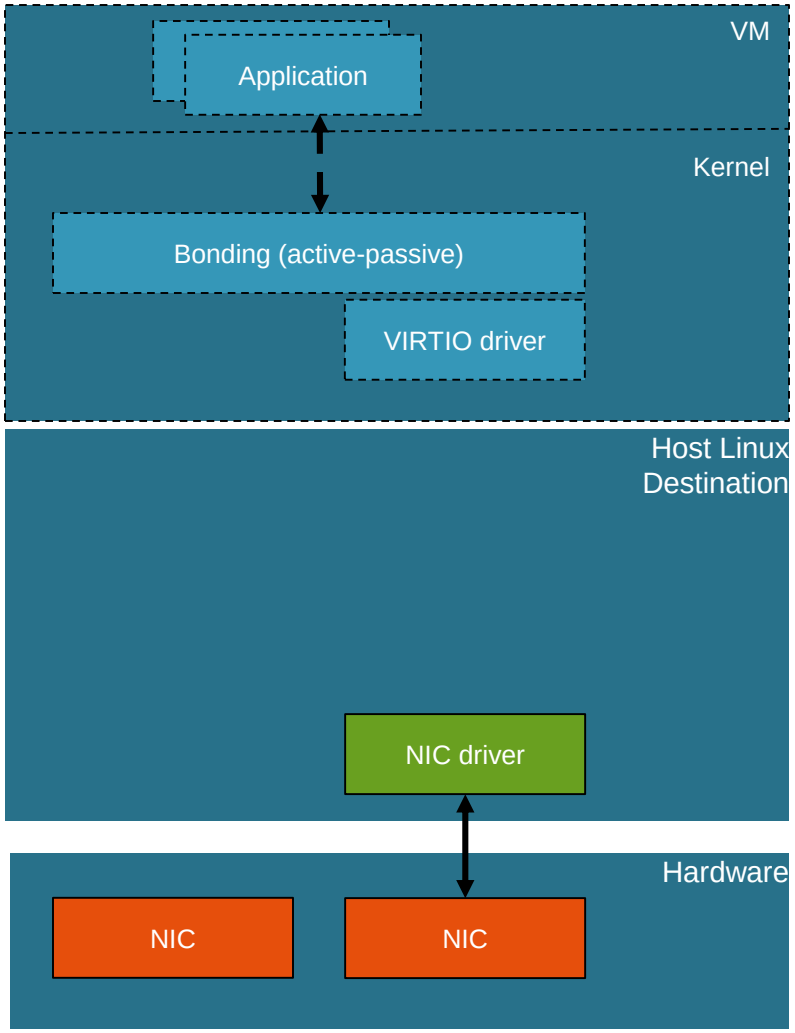


VM Live Migration with Direct Assignment

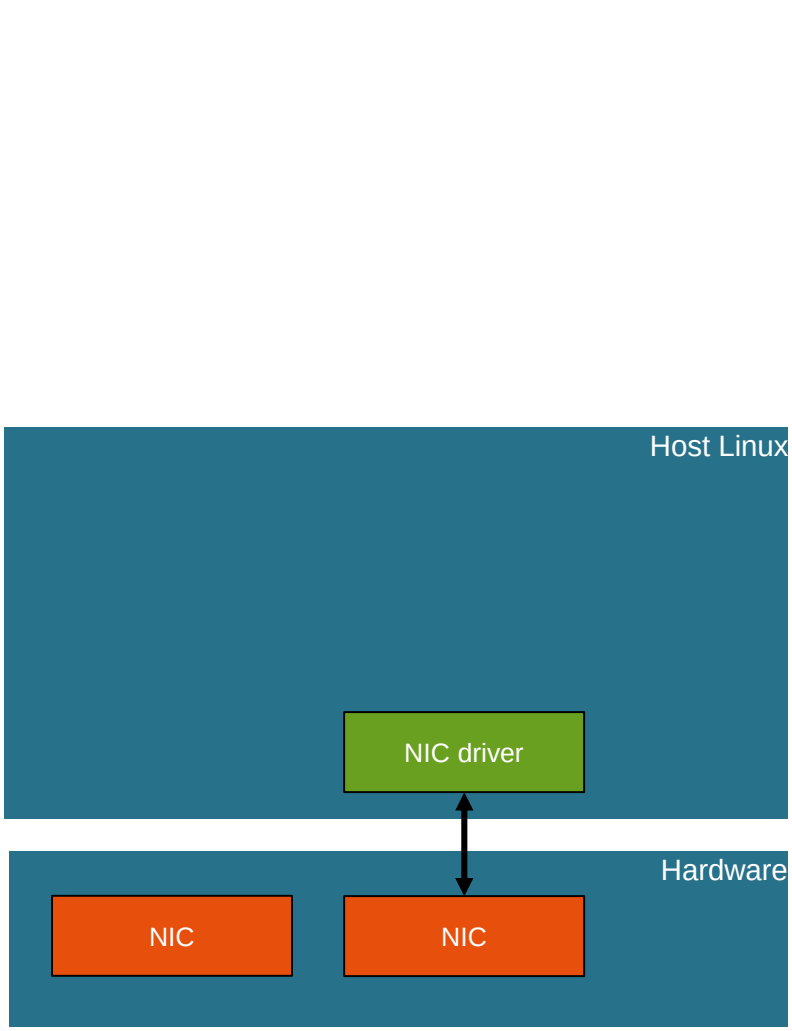


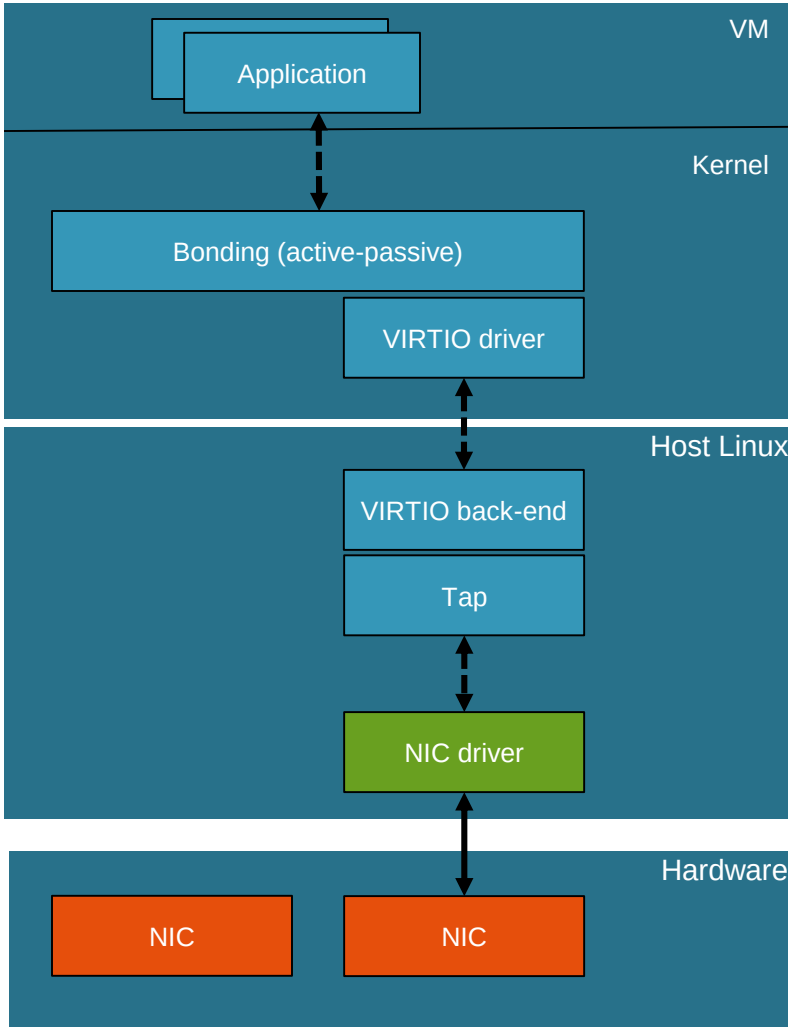
VM Live Migration with Direct Assignment



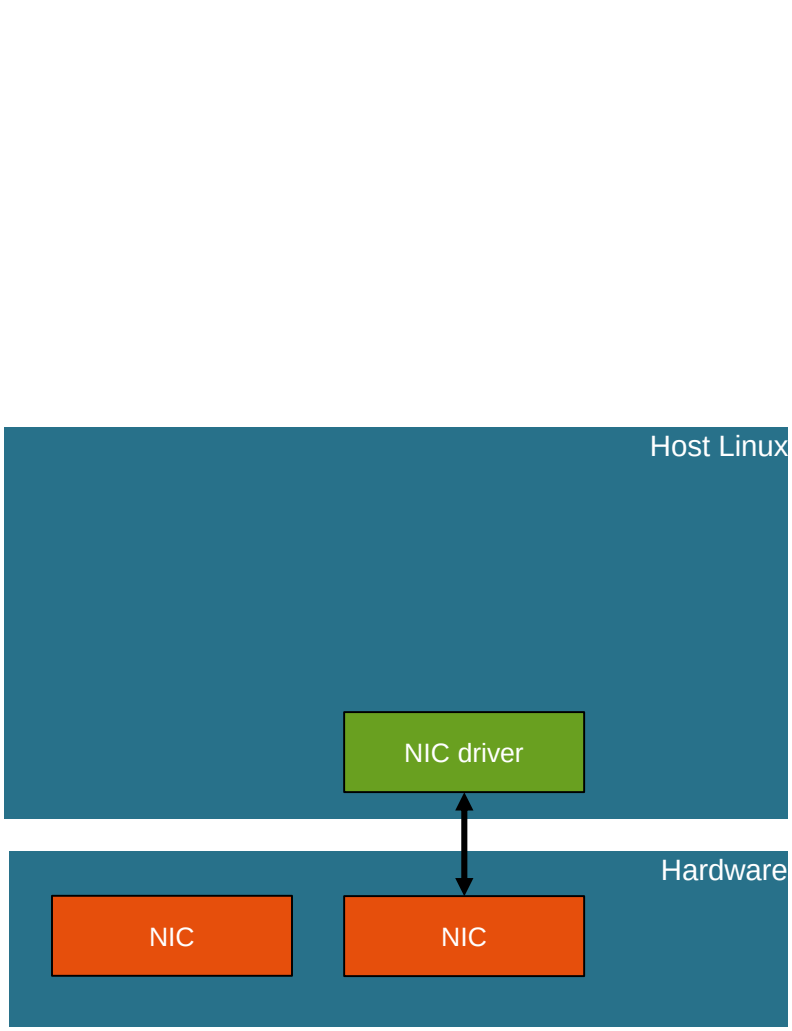


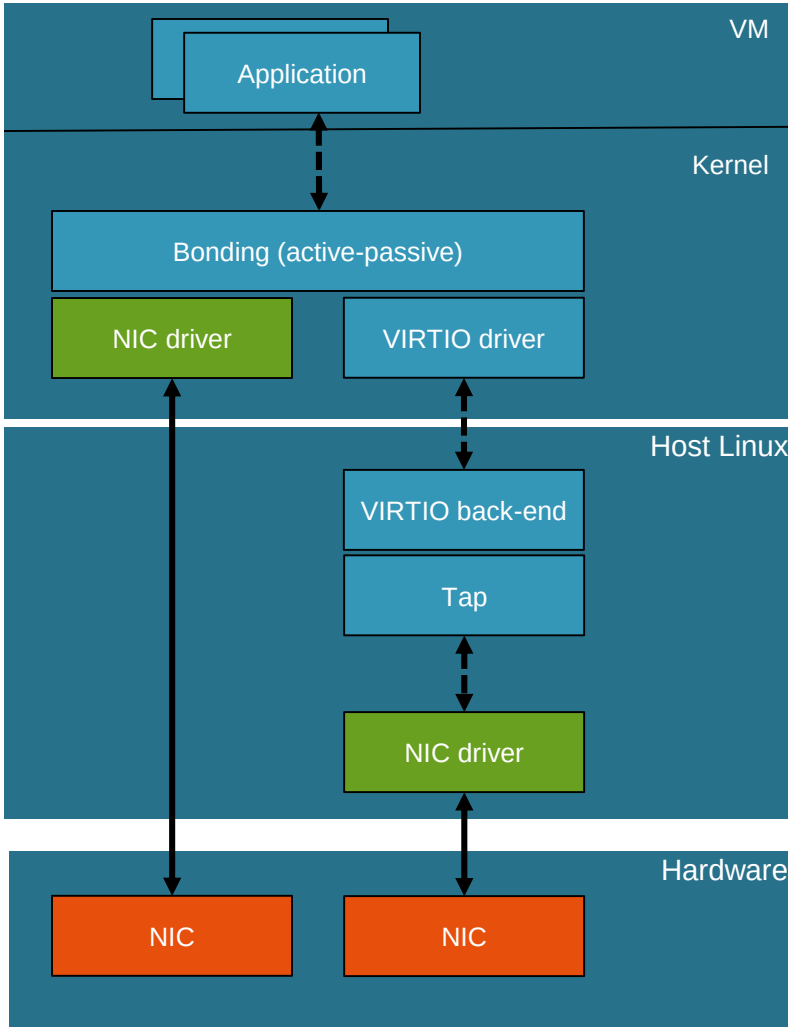
VM Live Migration with Direct Assignment





VM Live Migration with Direct Assignment





Conclusions

Conclusions

- I/O in virtualized environments is driven by new trends NFV, vCPE, vAccess
- Goal is to provide standard I/O support efficient and flexible
- KVM provides virtio, direct-assigned and pass-through devices
- Optimizing virtio, direct-assignment, pass-through is instrumental
- Supporting efficiently the Virtual Machine migration is a differentiator

Q & A

Freescale Has World Class Support….and MORE

Global Technical Information Center
Design & Support Resource

Networking Applications Team
Depth of Expertise & Knowledge

Design With Freescale, Freescale Technology Forum
Training

Networking Software and Services Group

- Commercial Solutions
- Engineering Services
- Guaranteed Performance
- Service Level Agreement Support…and MORE

- Visit Pedestal 415 in the Technology Lab

www.Freescale.com