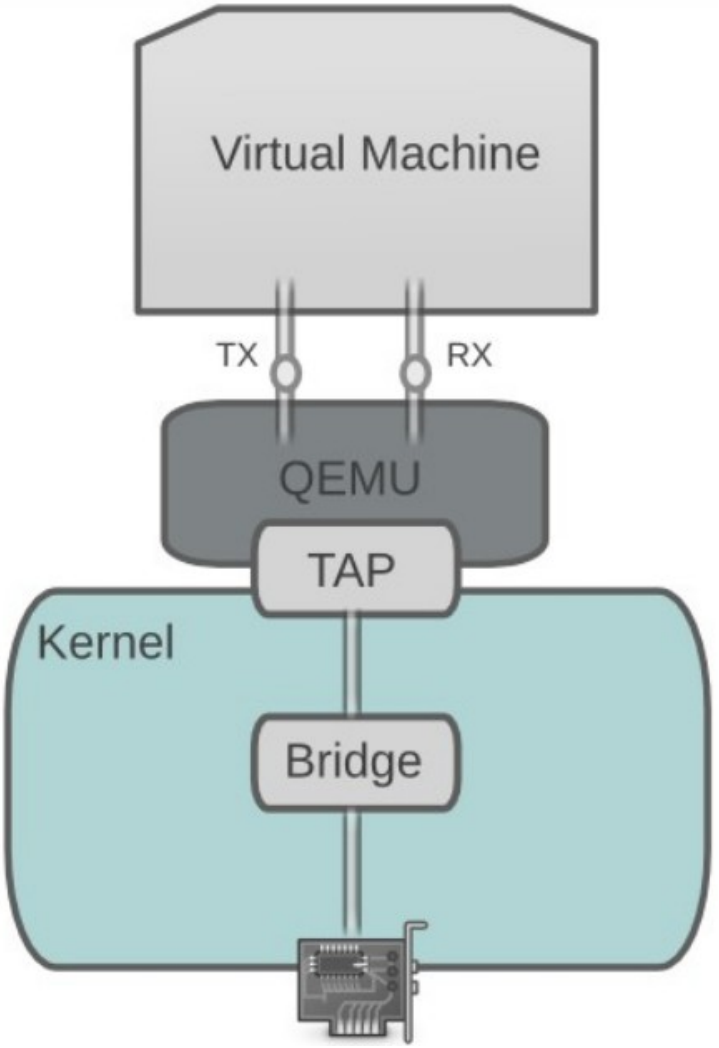
VFIO

PCI device assignment breaks free of KVM

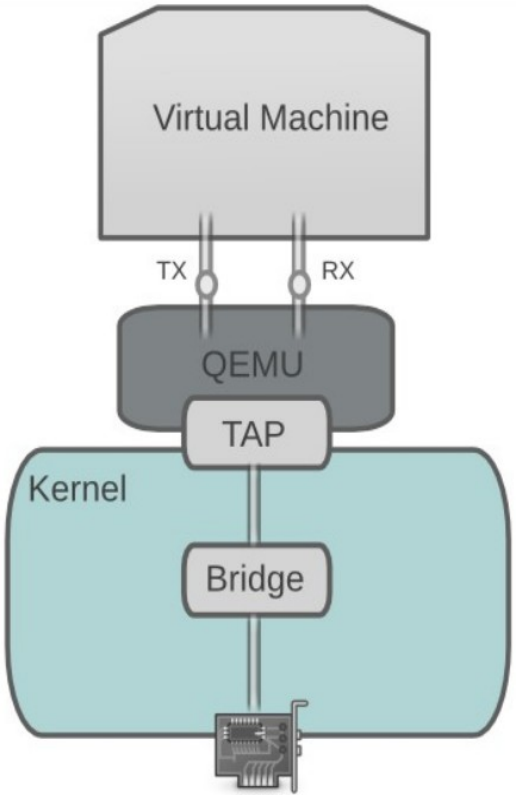
Alex Williamson
<alex.williamson@redhat.com>

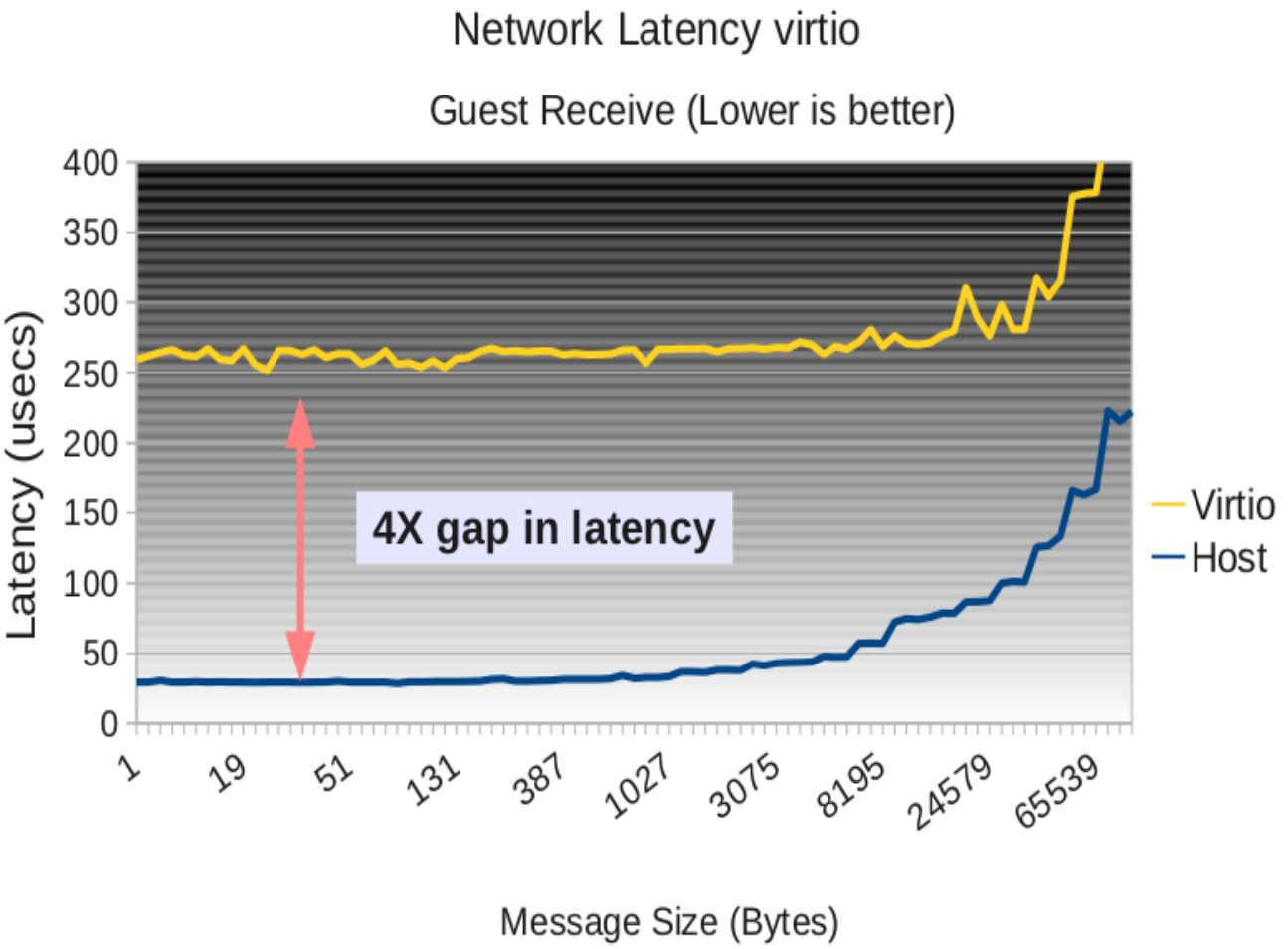
What is device assignment?

A typical NIC

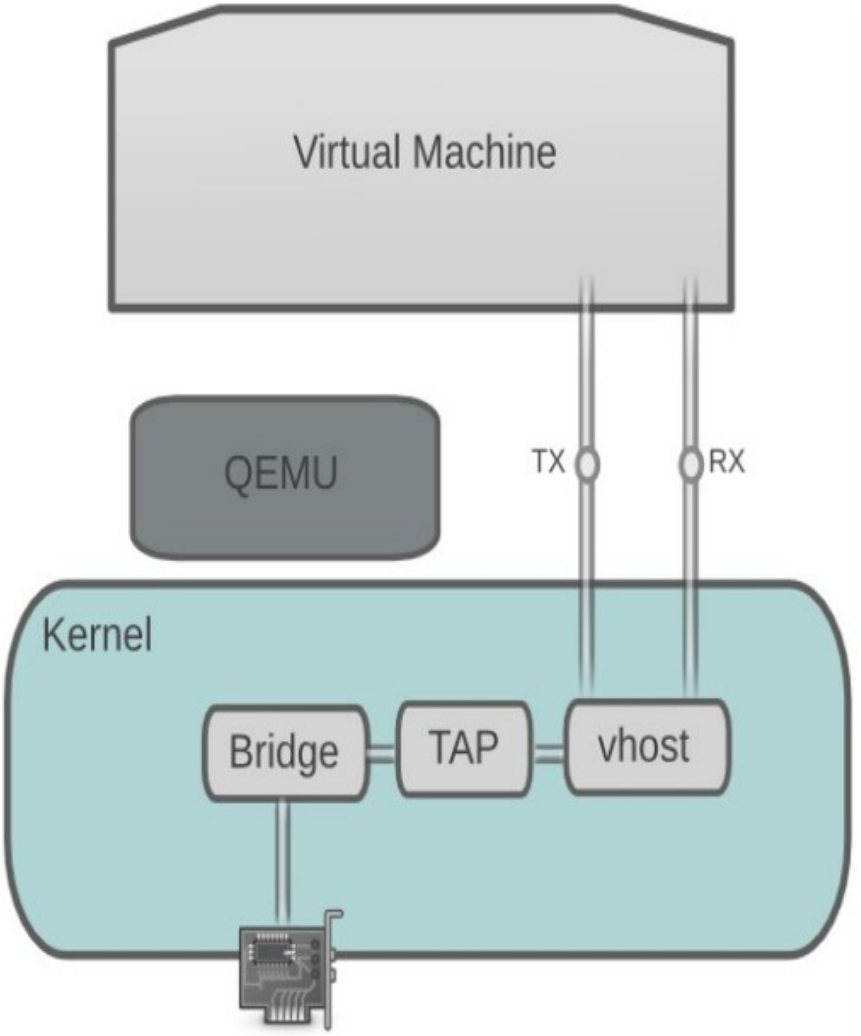


And typical performance...

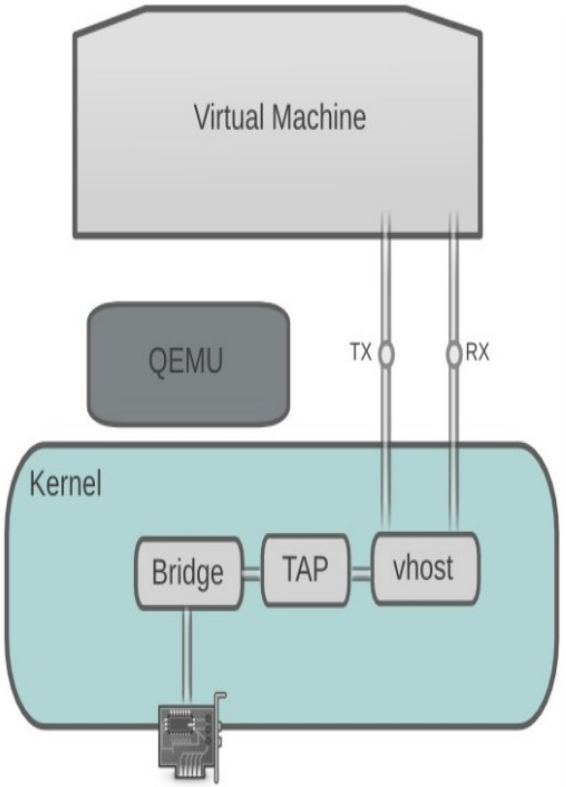


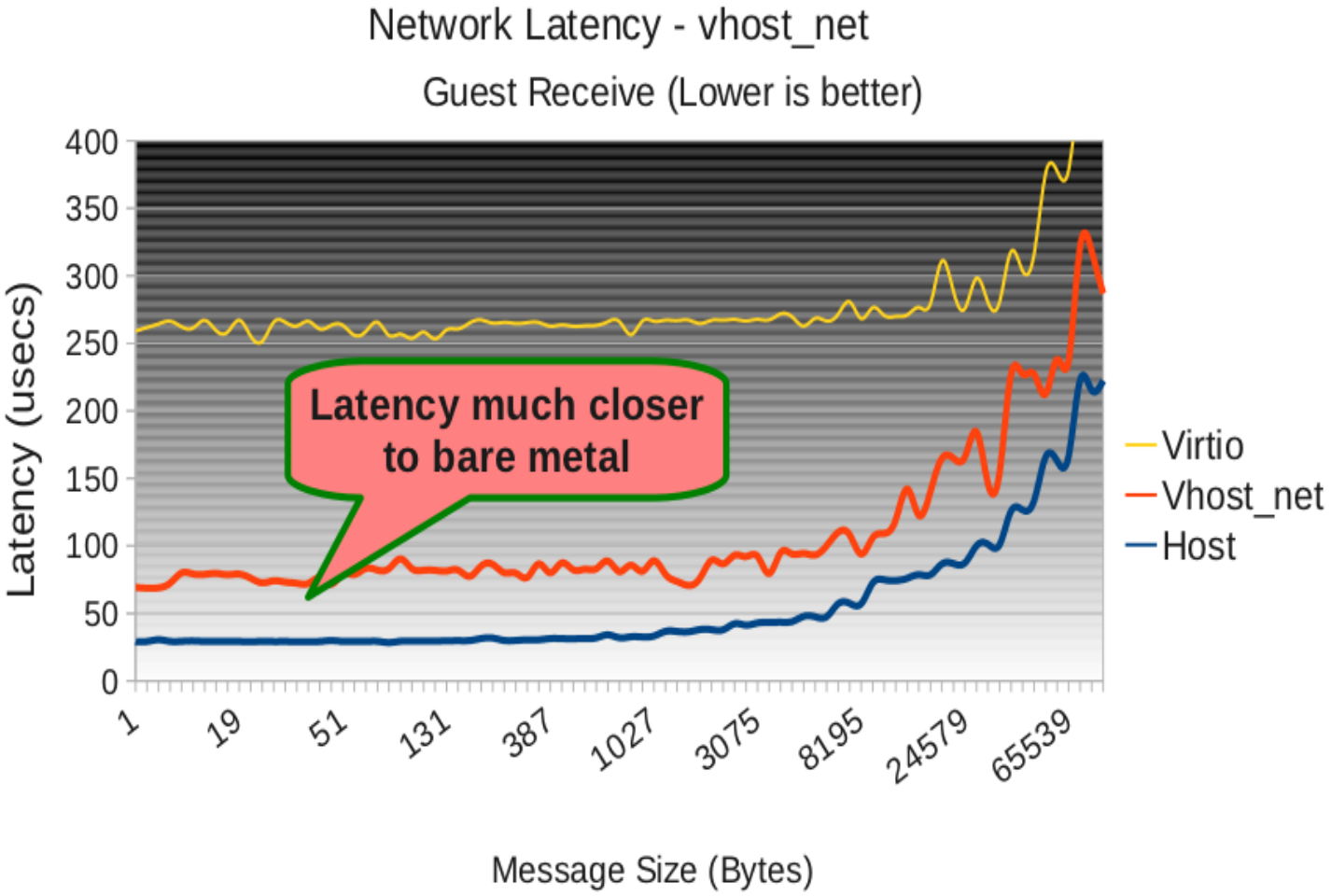


Let's get clever...

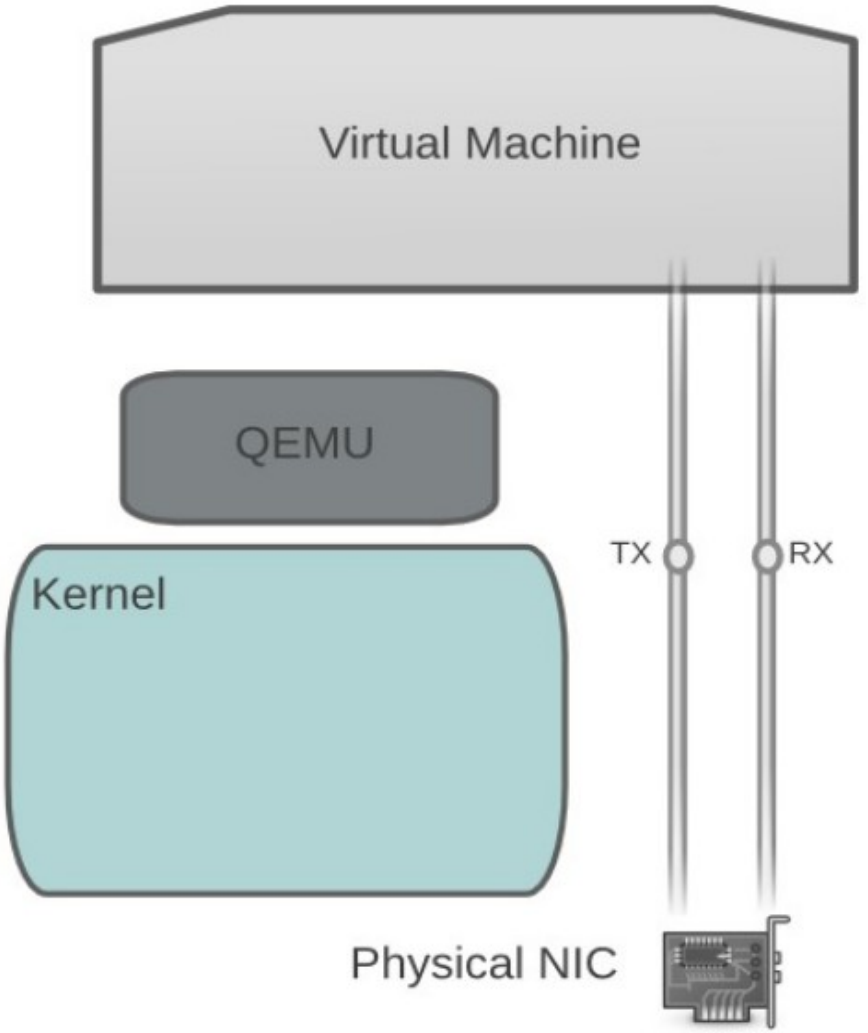


Getting better...

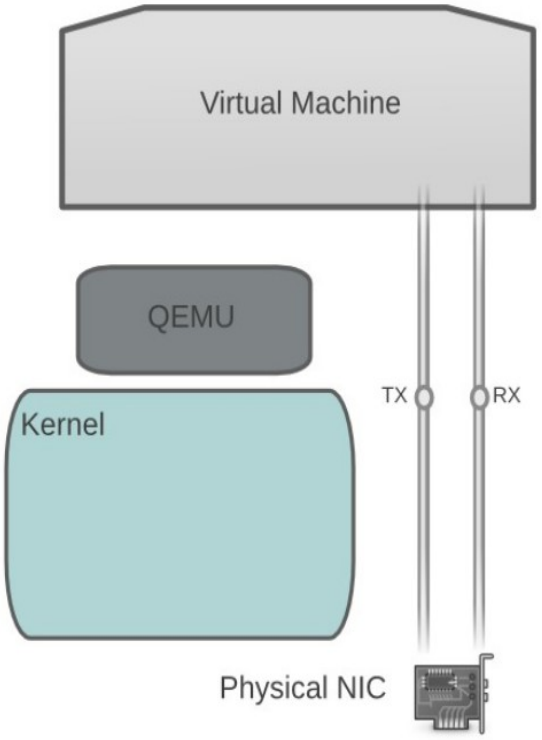


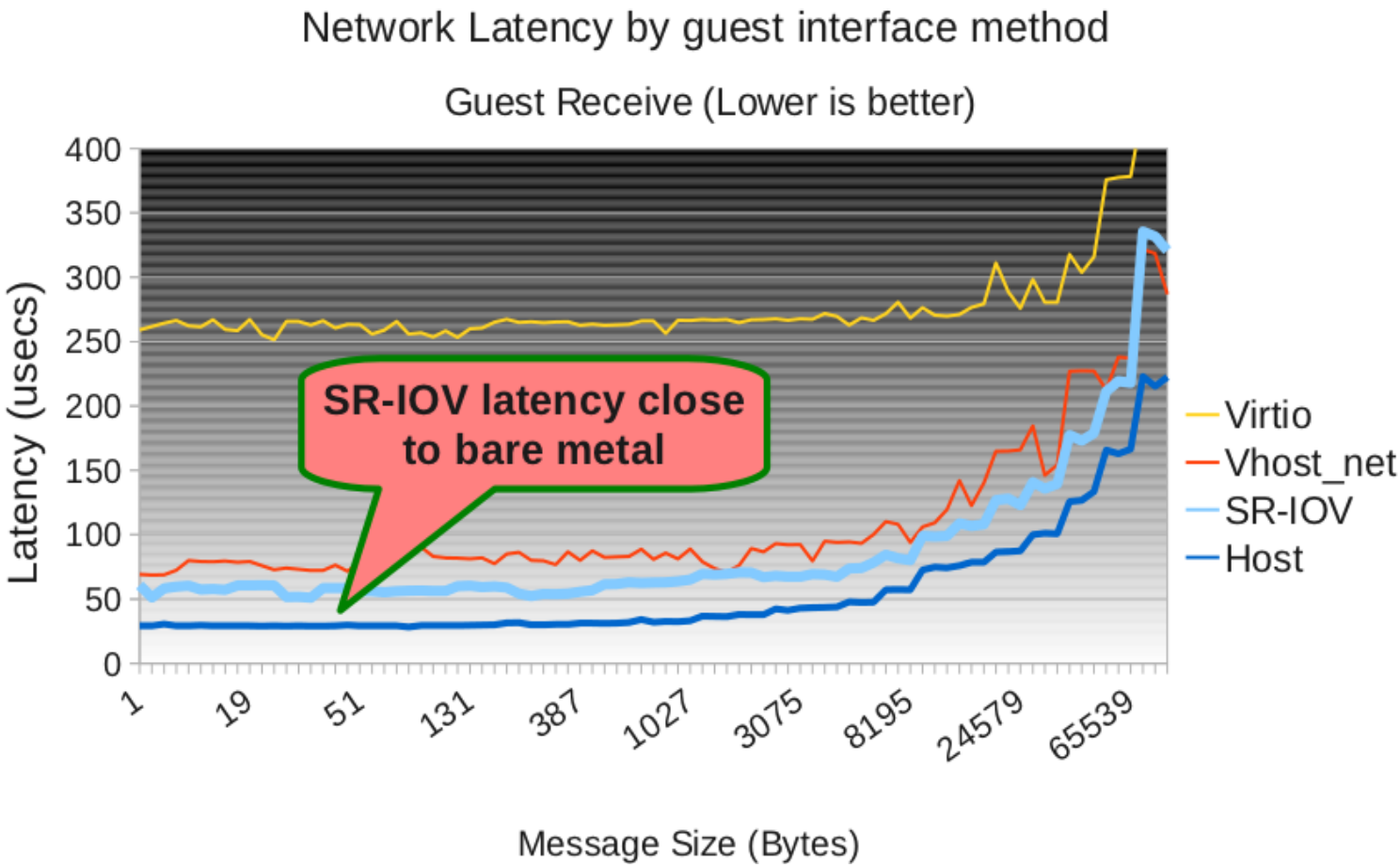


But what if...



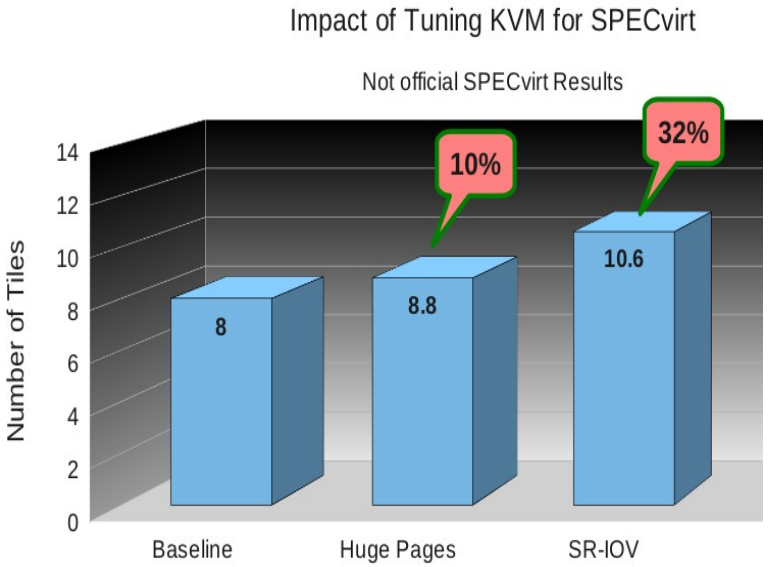
Even better!

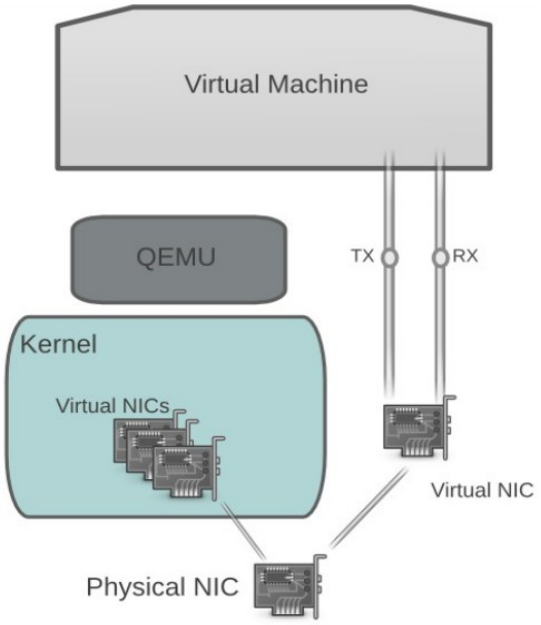

Virtual Machine
QEMU
Kernel
TX
RX
Physical NIC

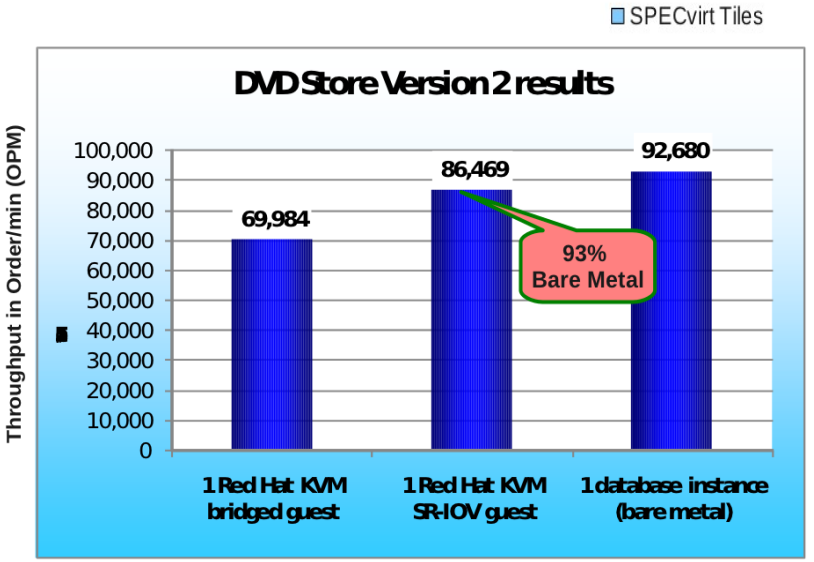

Network Latency by guest interface method
Guest Receive (Lower is better)
Latency (usecs)
400
350
300
250
200
150
100
50
0
SR-IOV latency close to bare metal
1
19
51
131
387
1027
3075
8195
24579
65539
Message Size (Bytes)
Virtio
Vhost_net
SR-IOV
Host

Benefits

- Performance
- Compatibility
- Utilization







At what cost?

- Guest pinned in host memory
  - No page sharing or swapping
- Guest tied to physical host device
  - Migration via bonding/multi-path + hotplug
- Exclusive device usage

PCI device assignment

- PCI Config Space
  - Pass-through/Emulated/Virtualized
- PCI BARs
  - Ideally mmap()'d MMIO
  - Trap & Forward otherwise
- Interrupt
  - Inject as close to guest a possible
  - Likely our biggest latency contributor

Current Implementation

- pci-stub
  - Bind device
- Qemu (qemu-kvm)
  - PCI config virtualization
  - Interrupt management
  - Resource setup
- KVM
  - PCI enablement (request region, enable)
  - IOMMU management, guest mapping
  - Interrupt handling and injection

VFIO

- High performance user-space driver
  - IOMMU based security
  - PCI & Non-PCI support
  - Eventfd based interrupts
- Original PCI implementation: Tom Lyon @Cisco
- A cleaner approach to VM device assignment
  - KVM not required (provides accelerators)
  - Enable non-x86, non-PCI device assignment

VFIO Implementation

- VFIO
  - Bind device
  - PCI enablement & config virtualization
  - IOMMU management
  - Interrupt handling
- Qemu
  - Resource setup
  - Pass-through (config, resources)
  - Interrupt management
- KVM
  - Acceleration only

Platform problems, pt 1

- IOMMU features and granularity
  - Bridges and “partitionable endpoints”
  - IOVA windows vs fully mapped guest
  - ✔ “iommu_group” sysfs attribute
  - ✔ iommu_device_group()
  - ✔ per bus iommu_ops
  - ✗ TBD iommu feature description

Platform problems, pt 2

- Group vs Device vs IOMMU domain management
  - Manage group devices as an atomic set
  - Merging groups to share IOMMU domains
  - Hot-plugs while groups are in use

Platform problems, pt 3

- Device description and access
  - Read() device info
  - Segment file offsets for MMIO/PIO
  - Common and device specific ioctl()s

Stay tuned for VFIO-NG...
Questions/Comments?

Thank you!

Graphics & performance data:
Mark Wagner, Red Hat Summit 2011, "KVM Performance Improvements & Optimizations"
http://www.redhat.com/summit/2011/presentations/summit/decoding_the_code/wednesday/wagner_w_420_kvm_performance_improvements_and_optimizations.pdf