VIRTIO: VHOST DATA PATH ACCELERATION TOWARDS NFV CLOUD

CUNMING LIANG, Intel
Agenda

• Towards NFV Cloud
  – Background & Motivation
• vHost Data Path Acceleration
  – Intro
  – Design
  – Impl
• Summary & Future Work
Towards NFV Cloud

- VIRTIO is a well recognized by Cloud
- DPDK promotes its Perf. into NFV Level
- New accelerators comes, what's the SW impact on I/O virtualization?

Native I/O Perf. by SR-IOV device PT
- Faster simple forwarding by 'cache'
- Remains historical gaps of cloudlization
  - Stock VM and SW vSwitch fallback
  - Cross-platform Live-migration

vDPA: Balanced Perf. and Cloudlization
- Device Pass-thru Like Performance
- Hypervisor native I/O
- Live-migration Friendly
- Stock vSwitch/VMs Support

GOAL

vNF0
vNF1
vNF2

NIC

vNF0’
vNF1
vNF2

NIC w/ Embedded Switch

vNF0’
vNF1
vNF2

NIC w/ Embedded Switch

Cloud vSwitch as NFVi
Accelerated vSwitch as NFVi
Accelerated Cloud vSwitch as NFVi
vDPA Intro
What is vDPA

- As a VMM native device, PV hasn’t shared the I/O VT benefits
  - PV device was born with cloud-lization characters,
  - But it’s lack of performance towards NFV cloud.
- vHost Data Path Acceleration is a framework offering virtualization infrastructure for VRING capable device
  - Decompose DP/CP of VIRTIO device
  - DP pass-thru for VRING capable device
  - CP remains to be emulated, but backed by a DP capable device
  - VRING capable device has ability to ENQ/DEQ VRING and recognize VRING format according to VIRTIO Spec.

<table>
<thead>
<tr>
<th></th>
<th>PV</th>
<th>Dev Pass-thru</th>
</tr>
</thead>
<tbody>
<tr>
<td>VMM</td>
<td>Aware</td>
<td>Unaware</td>
</tr>
<tr>
<td>Performance</td>
<td>~Cloud Qualified</td>
<td>~NFV Qualified</td>
</tr>
<tr>
<td>Direct I/O</td>
<td>N/A(SW Relay)</td>
<td>IOMMU/SMMU</td>
</tr>
<tr>
<td>I/O Bus VT</td>
<td>N/A</td>
<td>SR-IOV, SIOV</td>
</tr>
<tr>
<td>CPU Utilization</td>
<td>Variable</td>
<td>Zero</td>
</tr>
<tr>
<td>SW framework</td>
<td>Emulated device w/ backend Impl.</td>
<td>kvm-pci, vfio-{pci</td>
</tr>
<tr>
<td>Cloud-lization</td>
<td>- LM friendly</td>
<td>- Tricky LM</td>
</tr>
<tr>
<td></td>
<td>- SW fallback</td>
<td>- N/A</td>
</tr>
<tr>
<td></td>
<td>- SW vswitch native</td>
<td>- N/A</td>
</tr>
</tbody>
</table>
Why not device pass-thru for VIRTIO

Statement
- VIRTIO is a SW Spec. continuous evolution
- Unlikely forcing HW to follow ‘uniform’ device definition

Disadvantage
- Inherits all device pass-thru properties
  - “All or nothing” offload, SW fallback in the guest (bonding)
  - Framework limitation to support live-migration in general use
- Becomes VIRTIO Spec. version specific
  - e.g. 0.95 PIO, 1.0 MMIO, etc.
- Lose the benefit of decomposed frontend/backend device framework
  - Diverse backend adaption
vDPA Design
VIRTIO Anatomy

- PCI CSR Trapped
- Device-specific register trapped (PIO/MMIO)
- Emulation backed by backend adapter via VHOST PROTO
- Packet I/O via Shared memory
- Interrupt via IRQFD
- Doorbell via IOEVENTFD
- Diverse VHOST backend adaption
Data Path Pass-thru

- Decomposed VRING Data Path on ACC
  - DMA Enq/Deq VRING via IOMMU
  - Interrupt Notification
    - VFIO INTR eventfd associate with IRQFD
    - IRQFD as token for irq_bypass Prod/Cons
    - Leverage existing posted-interrupt support
  - Doorbell Kick
    - SW Relayed IOEVENTFD to trigger doorbell (PIO)
    - Add guest physical memory slot for doorbell direct mapping (MMIO)
- ACC needs a device framework
  - vhost-net won’t directly associate with driver
Control Path Emulation

- VIRTIO PIO/MMIO trap to QEMU
- Emulation Call → VHOST Req.
- VHOST Req. go thru transport channel to backend
- User space backend
  - Feature message extension
- Kernel space backend
  - Add a new transport channel for vfio mediated device
  - Define transport layout for data path relevant request
Cross Net Client Live-migration

- Live-migration Friendly
- Consistent vhost transport message sequence interact with QEMU live-migration
- Cross net client LM
  - netdev for virtio-net-pci
    - tap w/ vhost=on/off
    - vhost-user
    - vhost-vfio (+)
vDPA Implementation
Construct vDPA via VFIO

#1 QEMU for User Space Driver

vhost-user adapter
- New protocol message extension -- F_VFIO
- SLAVE Request to handover vfio group fd and notify meta data
- vhost-user adapter to map doorbell

Dependence
- Leverage user space device framework (DPDK)

#2 QEMU for Kernel Driver

vhost-vfio adapter
- New netdev client
- Reuse QEMU VFIO interface
- VFIO device as vhost request transport layer
- Leverage vfio/mdev framework

Dependence
- mdev_bus IOMMU support
- Singleton mdev per VF instance in Kernel
QEMU Changes for User Space Driver
-- #1 vhost-user extension

- New Protocol Feature -- VHOST_USER_PROTOCOL_F_VFIO
- Slave Request
  - Meta Data Update: VFIO Group FD, Notify Info
  - Actions: Enable/Disable ACC
- VFIO Group FD
  - Associate VFIO group fd with kvm_device_fd
  - Update GSI routing
- Notify Info
  - Represent for doorbell info (in page boundary)
  - Add guest physical memory slot
QEMU Changes for Kernel Driver

--- #2 vhost-vfio

- New net client for virtio-net-pci
  - `chardev vfio,id=vfio0,sysfsdev=/sys/bus/mdev/devices/$UUID \`
  - `-netdev vhost-vfio,id=net0,chardev=vfio0 -device virtio-net-pci,netdev=net0`

- VFIO device based vhost transport layer
  - vhost request over vfio_device_ops(read, write)
  - data path relevant request: feature, vring, doorbell, log

- Construct context for data path accelerator
  - Leverage QEMU KVM/VFIO interface
  - Memory region mapping for DMA
  - Add guest physical memory slot for doorbell
  - Interrupt/IRQFD via VFIO device ioctl CMD

- Don’t expect other host applications to use the device so far
Relevant Dependence
-- #2 vhost-vfio

• Kernel
  – Leverage VFIO mediated device framework
  – Add IOMMU support for mdev-bus
  – VRING capable device driver to register as mdev
    • Singleton mode only, 1:1 BDF(Bus, Device, Function) with mdev
Summary

- Hypervisor Native I/O
  - virtio-net-pci
- Stock vSwitch/VMs Support
  - Transparent to frontend
- Device Pass-thru Like Performance
  - Data path pass-thru
- Live-migration Friendly
  - Cross net client live-migration
Future Work

- Collect feedback
- Send out RFC patches to Kernel, Qemu and DPDK
- Upstream current Impl. together w/ other relevant patches
- Continue to enable VRING incompatible device
Acknowledgment

- Tiwei Bie
- Jianfeng Tan
- Dan Daly
- Zhihong Wang
- Xiao Wang
- Heqing Zhu
- Kevin Tian
- Rashmin N Patal
- Edwin Verplanke
Thanks!