Virtio/vhost status update

Yuanhan Liu
<yuanhan.liu@linux.intel.com>

Aug 2016

outline

- **Performance**
  - Multiple Queue
  - Vhost TSO
- **Functionality/Stability**
  - Live migration
  - Reconnect
  - Vhost PMD
- **Todo**
  - Vhost-pci
  - Vhost Tx zero copy

Multiple Queue – Why

- **virtio/vhost is a CPU intensive workload, meaning CPU is the bottleneck**
- **Use multiple queue with multiple CPU could result to linear performance boost**

Multiple Queue – Howto

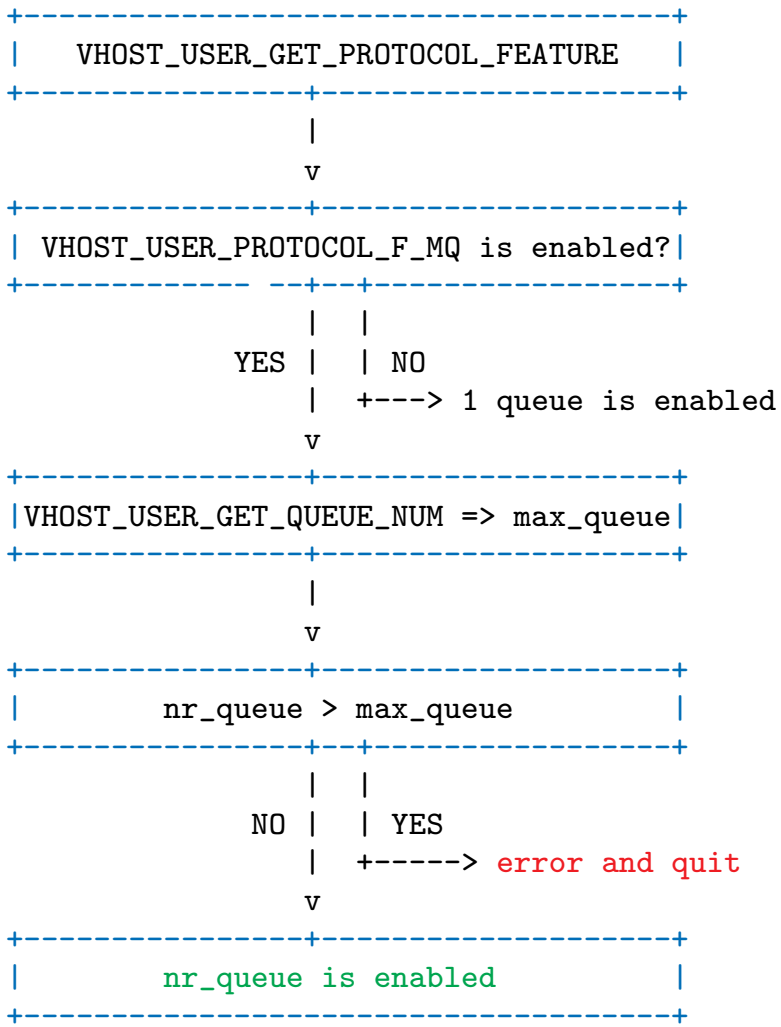
QEMU

```
-enable-kvm -m 4G -smp 4 -cpu host \
-object memory-backend-file,id=mem,size=4G,mem-path=/dev/hugepages,share=on \
-numa node,memdev=mem -mem-prealloc \
 \
-chardev socket,id=char0,path=$socket_path \
-netdev type=vhost-user,id=net0,chardev=char0,queues=2 \
-device virtio-net-pci,netdev=net0,mac=$mac,mq=on,vectors=6
```

Inside guest

- **Linux kernel virtio-net driver (one queue is enabled by default)**
  $ ethtool -L combined 2 eth0
- **DPDK Virtio PMD**
  $ testpmd -c 0x7 -- -i --rxq=2 --txq=2 --nb-cores=2 ...

Multiple Queue – Under the hood



Vhost TSO – Why

- **VM2VM performance sucks**
- **It could be improved dramatically with TSO enabled**

Vhost TSO – Howto

TSO is enabled by default, nothing special needed.

Vhost TSO – Under the hood

- **VM2VM on the same host**
  - no actual segment: virtio net supports transforming/receiving big packets
  - no checksum generating/calculating: packets are transfered on the same host; we can assume the data is valid.
- **VM2NIC**
  - set correct mbuf flags
  - NIC will do the TSO and CSUM offload

NOTE: OVS hasn't enabled NIC TSO yet. Thus the VM2NIC case won't work. Patch has been submitted though:
`http://openvswitch.org/pipermail/dev/2016-June/072871.html`

Live migration – Why

A very important feature for production usage, such as host maintainance

Live migration – Howto

src.sh

```
-chardev socket,id=char0,path=$socket_path      \
-netdev type=vhost-user,id=net0,chardev=char0 \
-device virtio-net-pci,netdev=net0,mac=$mac   \
-monitor telnet::3333,server,nowait \
```

dst.sh

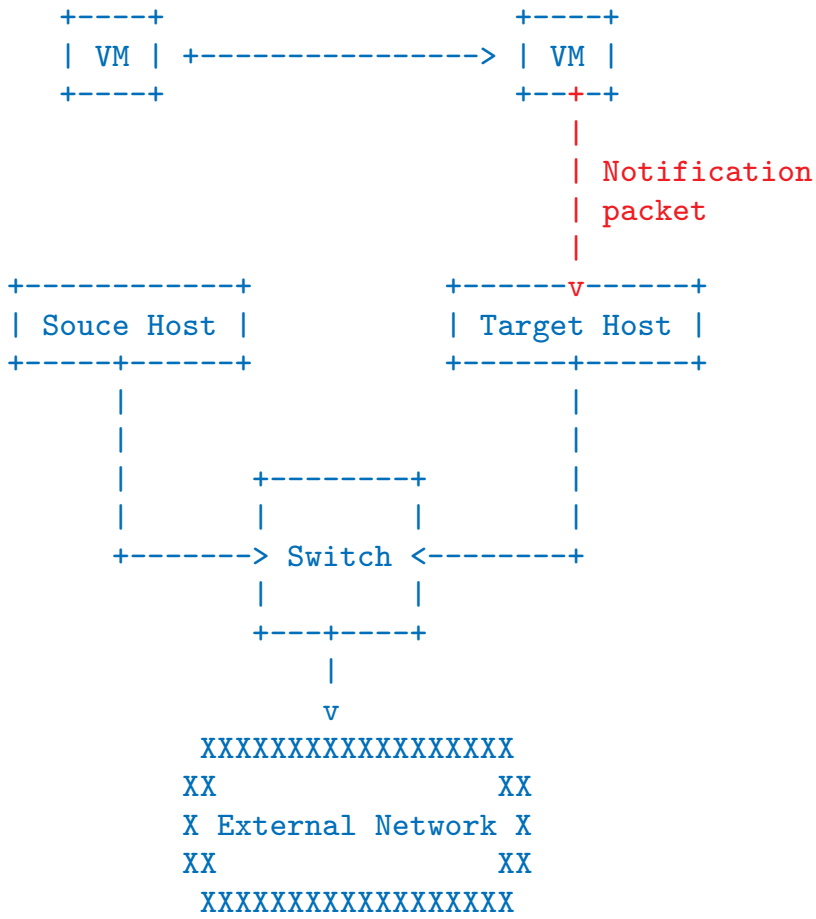
```
-chardev socket,id=char0,path=$socket_path      \
-netdev type=vhost-user,id=net0,chardev=char0 \
-device virtio-net-pci,netdev=net0,mac=$mac   \
-incoming tcp:0:4444
```

trigger live migration: run on src host

```
$ telnet 0 3333
> migrate -d tcp:$dst_host:4444
```

Live migration – Under the hood

- **log vring memory changes**
  - used vring update
  - desc buffer
- **notification the location change**
  - A **GARP** packet will be sent by the driver if it supports GUEST_ANNOUNCE feature (kernel v3.5 or above).
  - Otherwise, vhost constructs and sends an **RARP** packet



Reconnect – The issue

A crash of DPDK app (such as OVS) need restart all VMs connected to it, which is

- **A disaster for production usage**
- **Also not handy for development, that building and restarting DPDK happens often**

Reconnect – Howto

```
-chardev socket,id=char0,path=$socket_path,server \
-netdev type=vhost-user,id=net0,chardev=char0 \
-device virtio-net-pci,netdev=net0,mac=$mac   \
```

NOTE: DPDK v16.07 (or above) and QEMU v2.7 (or above) are needed!

Reconnect – Under the hood

- **When DPDK starts (or restarts), it invokes "connect()" it may**
  - succeed, when the server (QEMU) is up.
  - fail, when the server is not up yet.
    DPDK vhost lib then creates a reconnect thread to keep calling "connect()" until the server is up.

- **When QEMU restarts, DPDK detect a connection broken, which in turn puts it to the reconnect thread.**

Vhost PMD – What

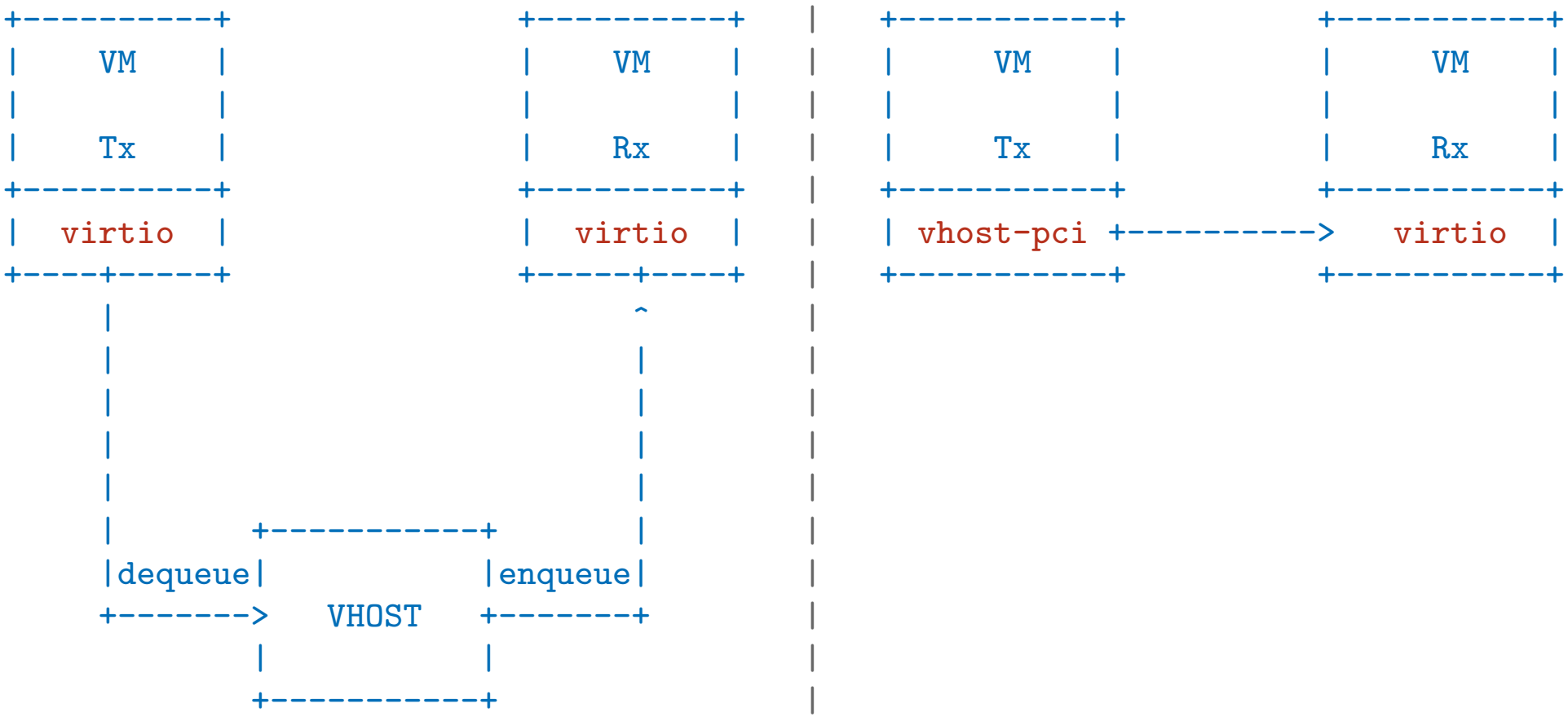
- **A virtual PMD wraps vhost APIs**
- **Common eth APIs apply**
  - rte_eth_dev_configure
  - rte_eth_rx_queue_setup
  - rte_eth_tx_queue_setup
  - rte_eth_dev_start
  - rte_eth_rx_burst
  - rte_eth_tx_burst
  - ...

Vhost PMD – Howto

```
$ testpmd  -c 0x1f -n 4 \
        --socket-mem 4096,0                          \
        --vdev "eth_vhost0,iface=$socket_path1,queues=2" \
        --vdev "eth_vhost1,iface=$socket_path2,queues=2" \
        ...
```

Vhost-pci – What

A method to Tx packets from one VM to another VM **directly**



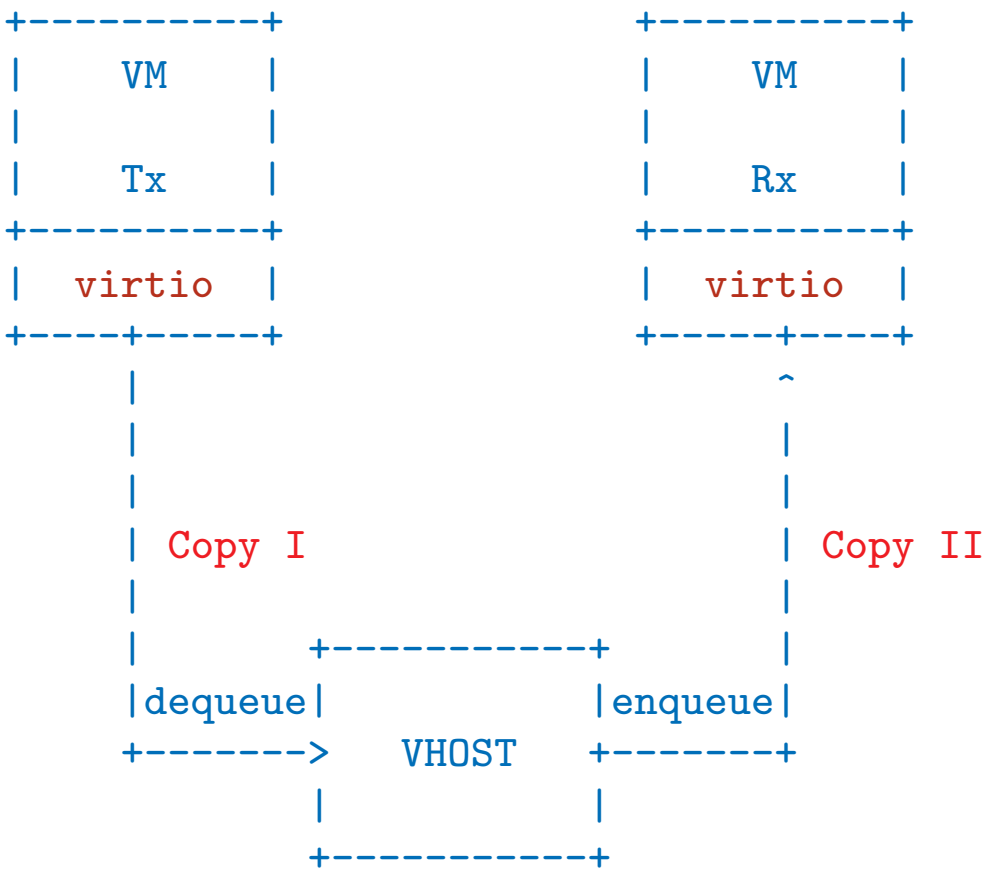
Generic vhost datapath

Vhost-pci datapath

Vhost-pci – Under the hood

- A PCI device
- 2nd VM's memory is mapped and stored in 1st VM'c PCI BAR 2
- The 2nd VM will send some vhost-pci messages to tell necessary info, such as vring addr
- 1st VM is then able to directly Tx packets to the 2nd VM's Rx vring.

Vhost zero copy



Generic vhost datapath

Instead of doing "Copy I", let mbuf reference the virtio desc buffer directly.

Therefore, one copy is saved.

Thanks

Thanks!