Multiqueue Networking for KVM

Jason Wang

Senior Software Engineer,

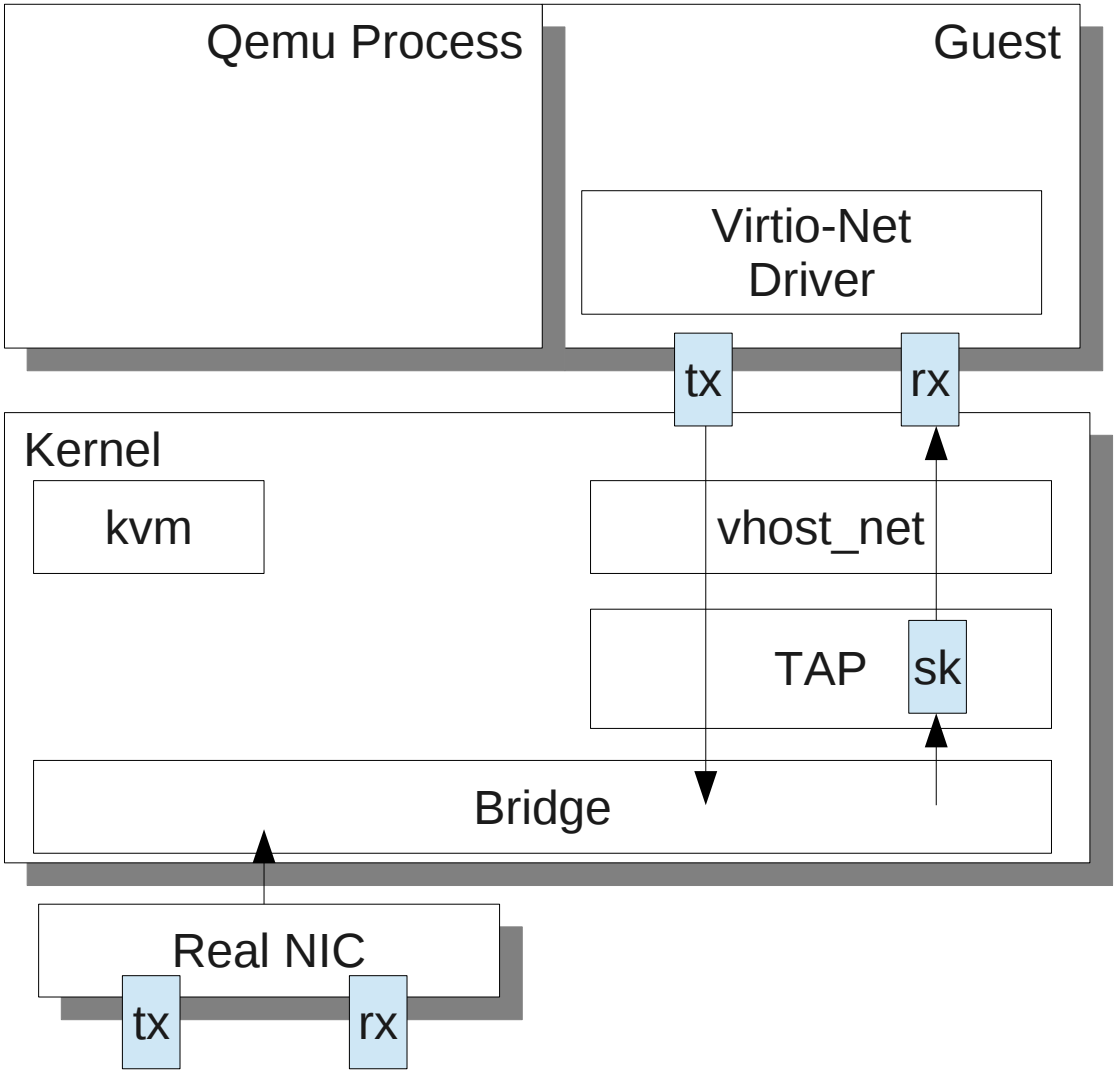
Red Hat

November, 2012

Agenda

- Introduction
- Design & Implementation
- Performance
- TODO
- Q&A
-

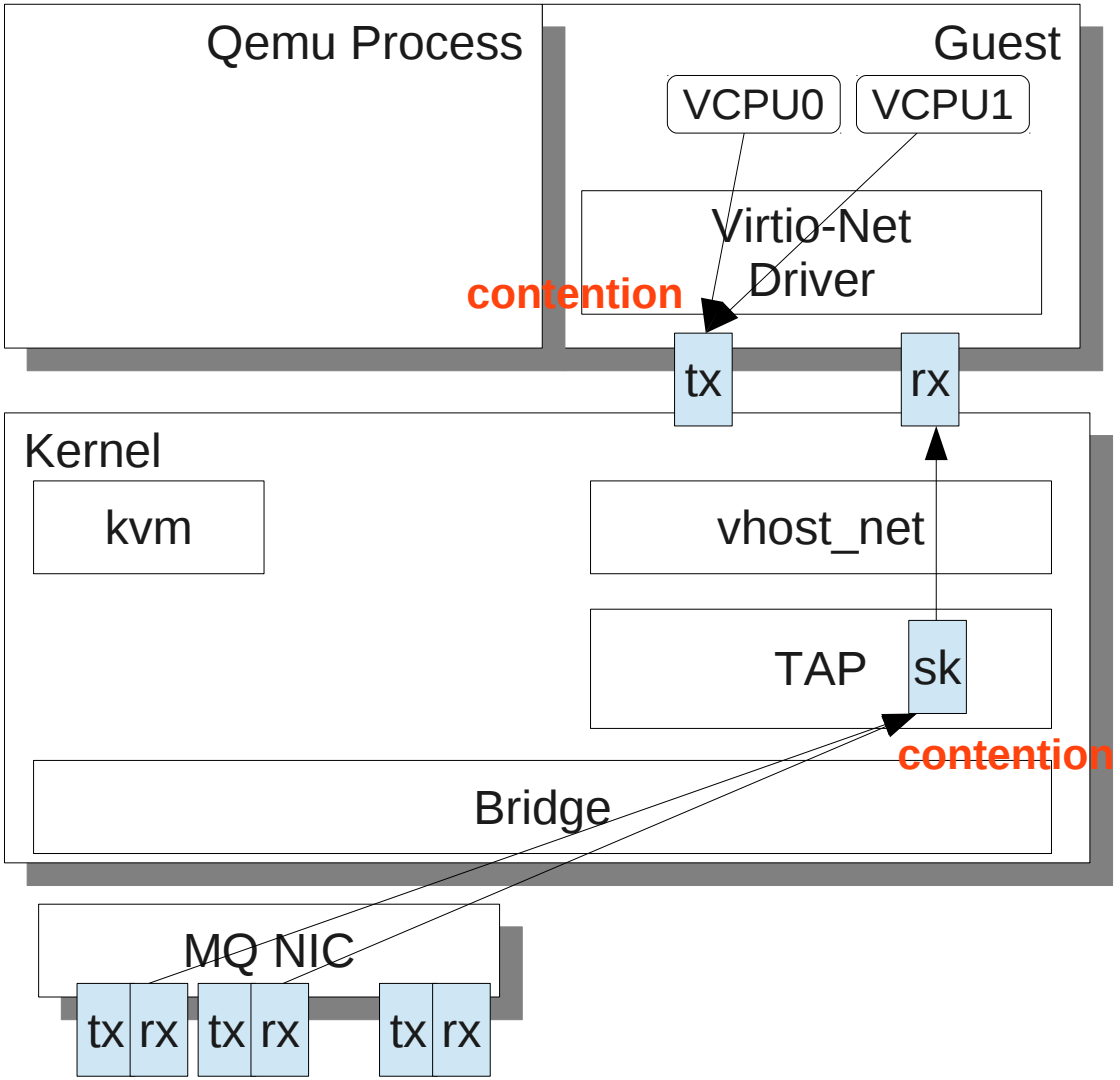
KVM networking with tap/vhost_net



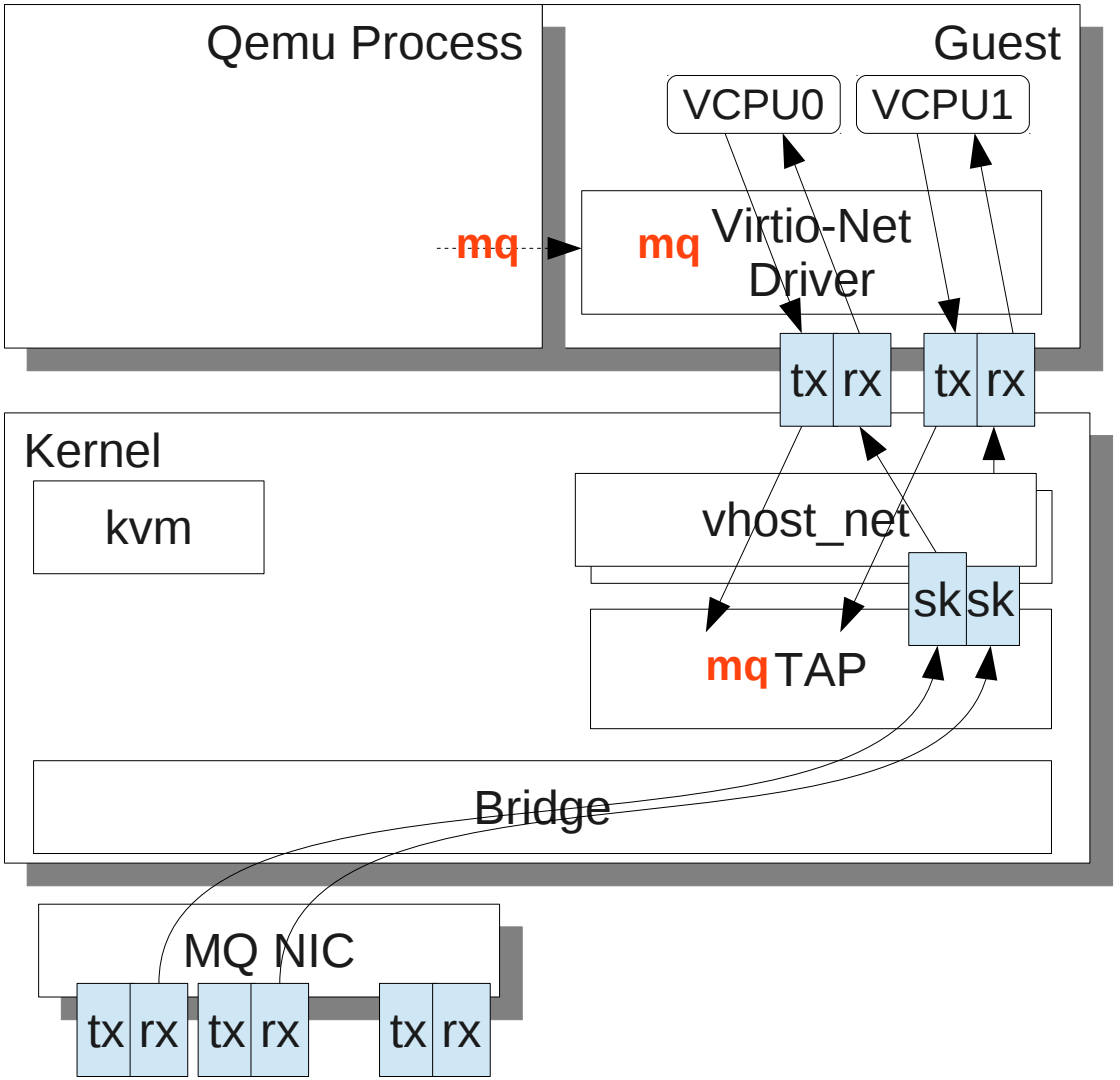
Mq 10gbe with virtio_net

- mq 10gbe become common
  - 40gbe in the future
- 1 txq/rxq
  - only 1vcpu could be used
  - 1q in host could be used (in some card)
  - RFS/RPS
    - IPI would be very expensive in virt
    - recv
- 1 vhost thread
  - overloaded

virtio-net with 10gbe mq card



Multiqueue networking in KVM

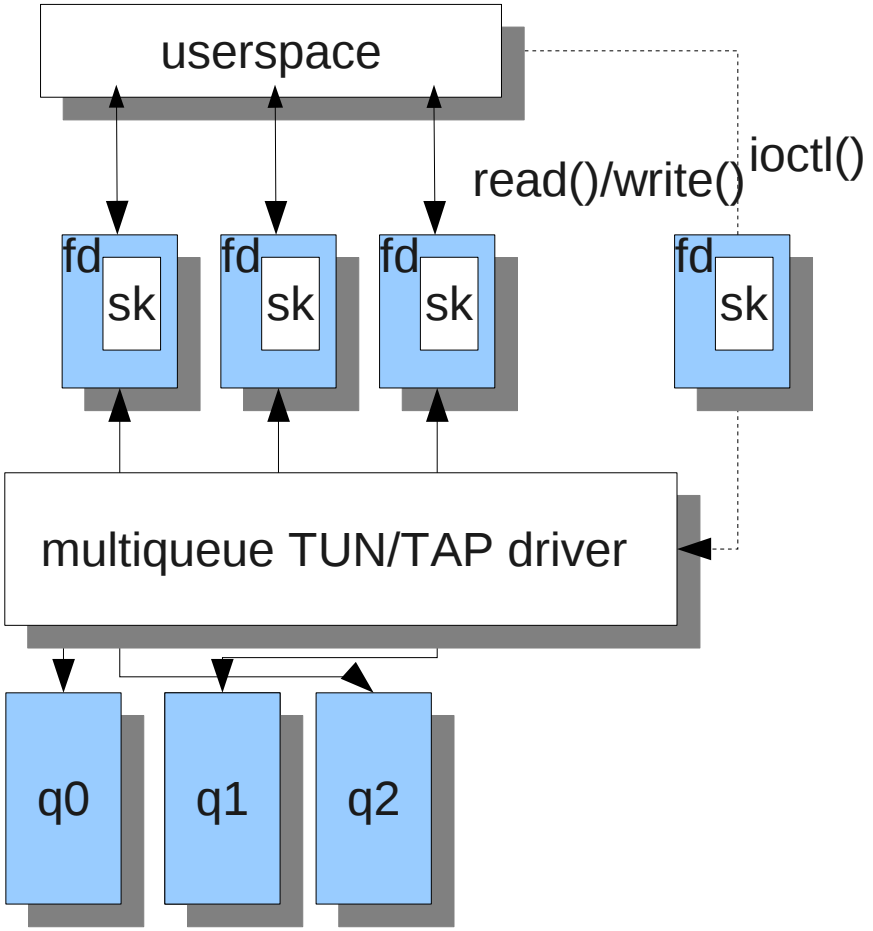


Changes required

Component	mq support?
guest virtio_net driver	convert to mq
qemu	multiple vq paris, config, userspace multiqueue virtio-net, launch multiple vhost threads
vhost_net	no changes?
tun/tap (macvtap?)	convert to multiqueue
host driver	ready

multiqueue tap

- Multiple queue tap
  - move the socket to fd
  - each socket/fd a queue
  - allow fd to be attached
  - expose multiple sk
  - ioctls to attach or detach fds
    - dynamic queue no. configuration
  - API compatible
  - no user visible change
  - useful for non-virtualized user



TUN/TAP MQ API

- TUN_GET_FEATURES
  - IFF_MULTIQUEUE
    - This host can create a multiqueue TUN/TAP
- TUN_SET_IFF
  - TUN_TAP_MQ
    - create a multiqueue netdevice in the host
- TUNSETQUEUE
  - IFF_ATTACH_QUEUE
    - Attach a file to the device (Add a new queue)
  - IFF_DETACH_QUEUE
    - Detach a file from the device (Disable a queue)

Create a TUN/TAP with 2 queue

fd1=open(“/dev/tap”)

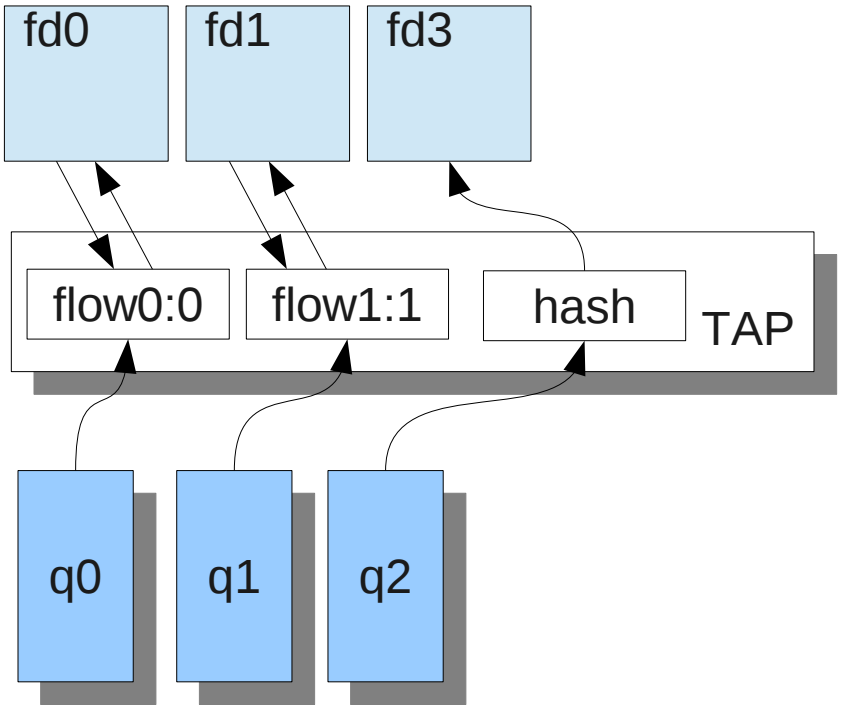
ioctl(fd1, TUNSETIFF, TUN_TAP_MQ)

fd2=open(“/dev/tap”)

ioctl(fd2, TUNSETQUEUE, IFF_ATTACH_QUEUE)

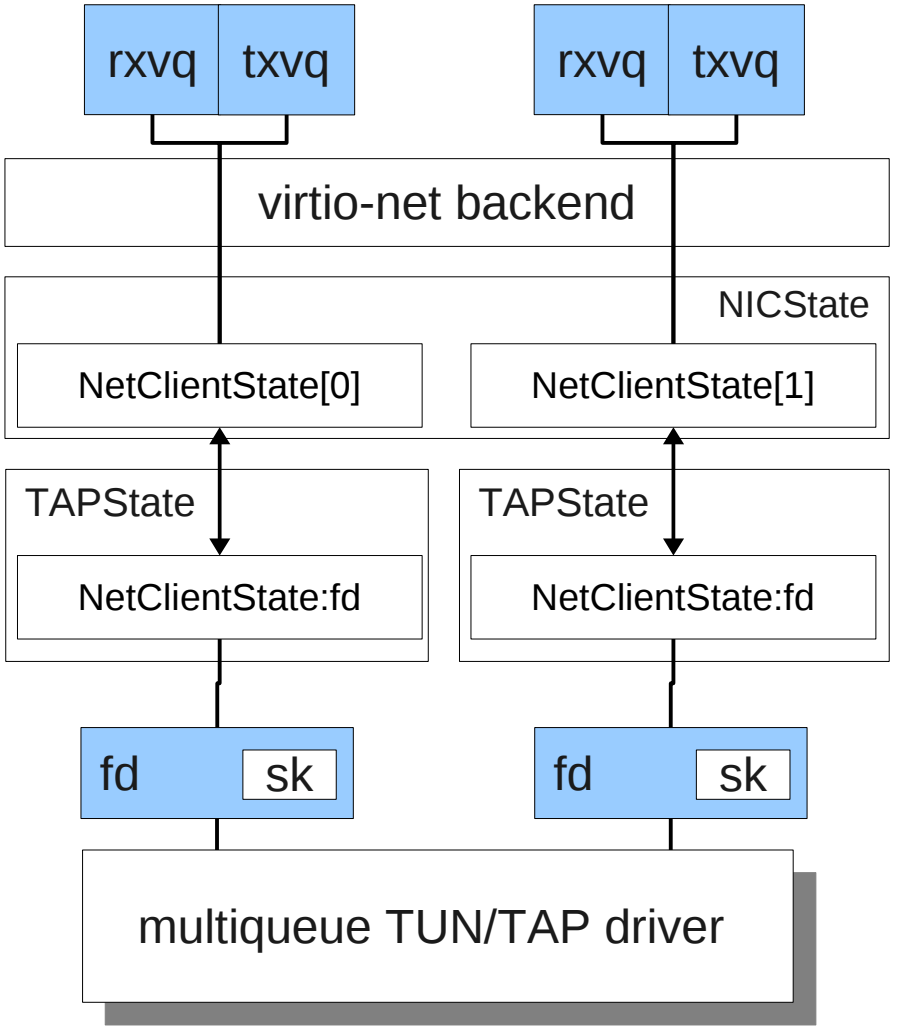
Txq selection in TUN/TAP

- tx follows rx with filter
  - optimize for stream transmission protocol (TCP)
    - A single stream is handled by one queue (userspace thread)
  - flow(hash) to queue table
  - update during receiving
    - or every 100 pkts
    - aging timer to retire old flow
  - query during transmission
  - use pure hash when no mapping



Qemu/virtio-net support

- Qemu mq support
  - pair of NetClientState as backend of txq/rxq
    - one fd in TAPState
  - multiple NetClientState in NICState
  - queues parameter for both netdev and nic
- Userspace multiqueue implementation
  - ?
    - Management, Migration
  - Map the virtqueue to NetClientState
  - Attach/Detach on demand



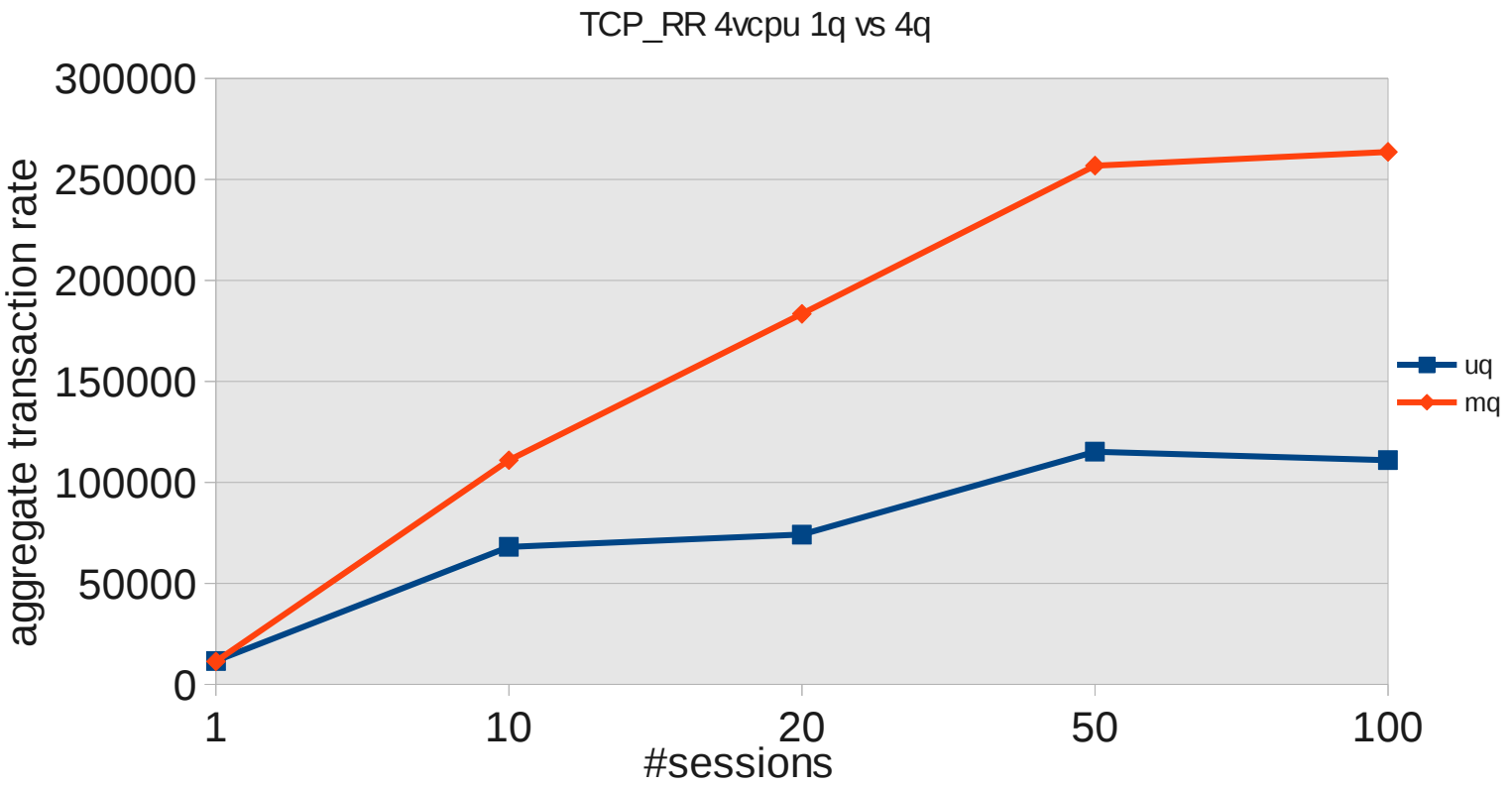
virtio (still in RFC)

- expose the number of queue pairs through config space
- change the number of active queue / steering policy through ctrl vq
  - SINGLE
  - RX_FOLLOW_TX
- use separate virtqueues in the two modes
  - vq 0,1 were reserved for single queue mode
  - eliminate the OOO during mode switching.

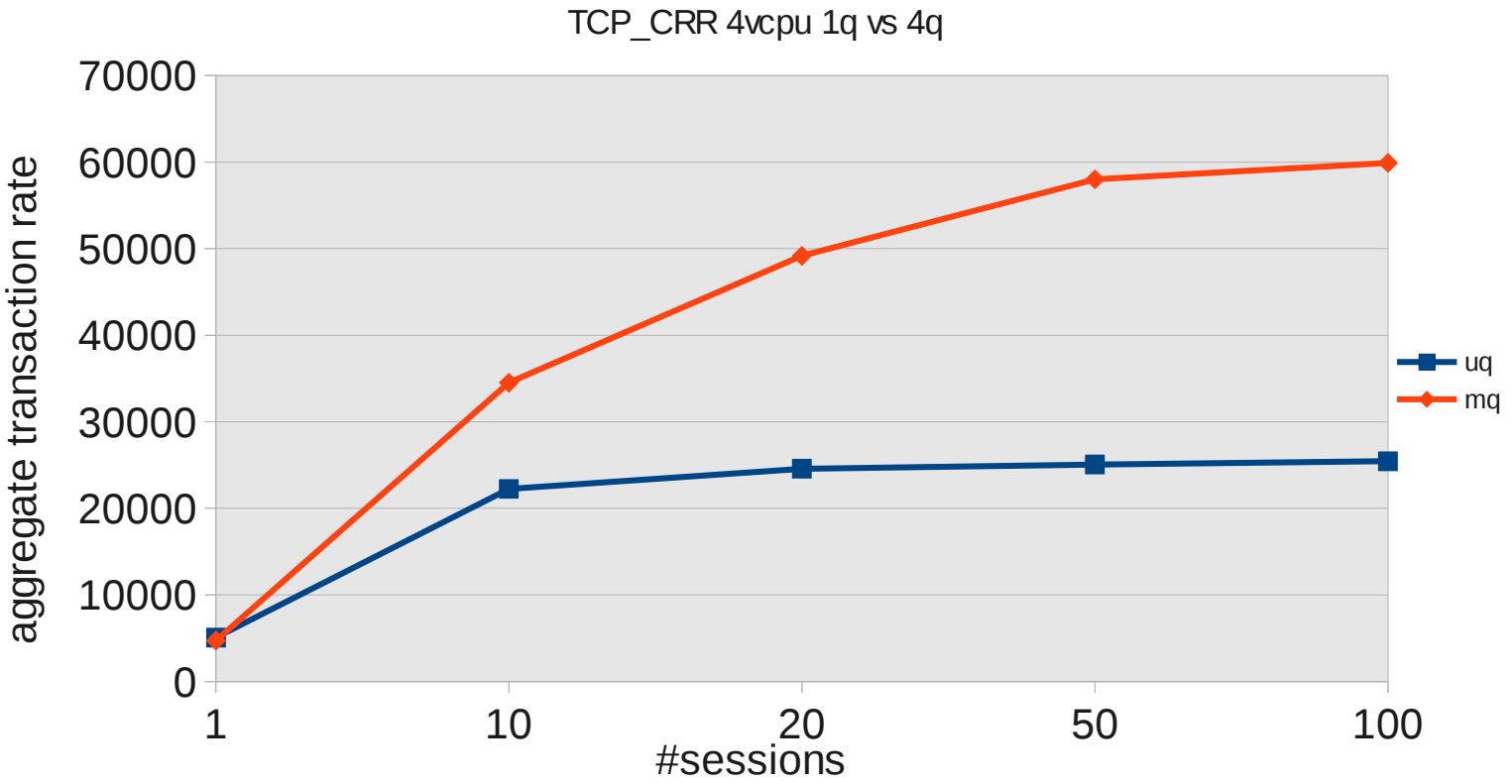
Test & Performance

- Result
  - Aggregated throughput / transaction
- Environment
  - Two E5620 8core 2node
  - Two directed 82599
  - 4vcpu guest
    - vcpu thread were pinned to node 0
    - vhost thread were pinned to node 1
  - Host/Guest kernel: net-next with mq patches
  - netperf (pktgen)?

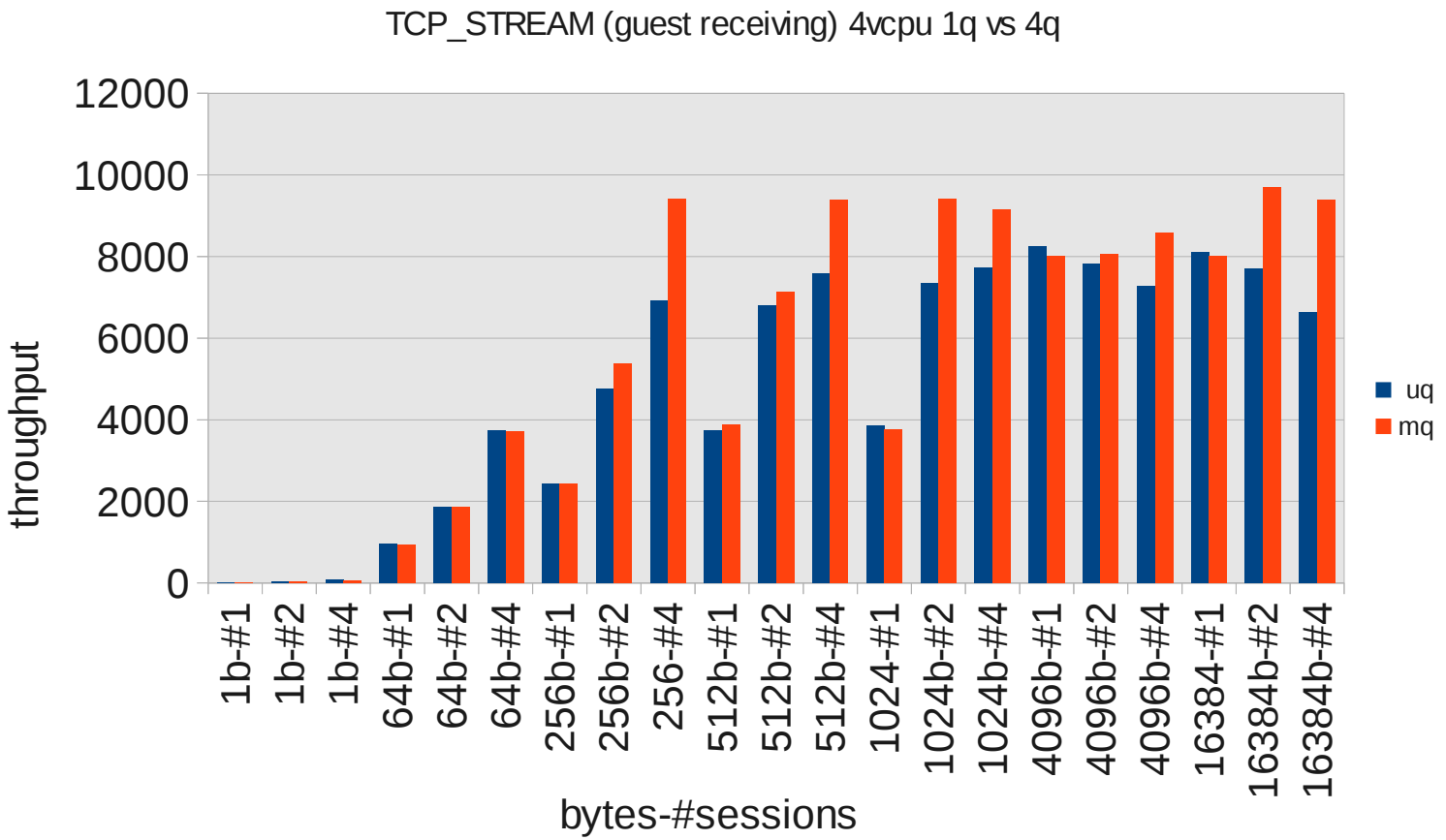
TCP_RR result



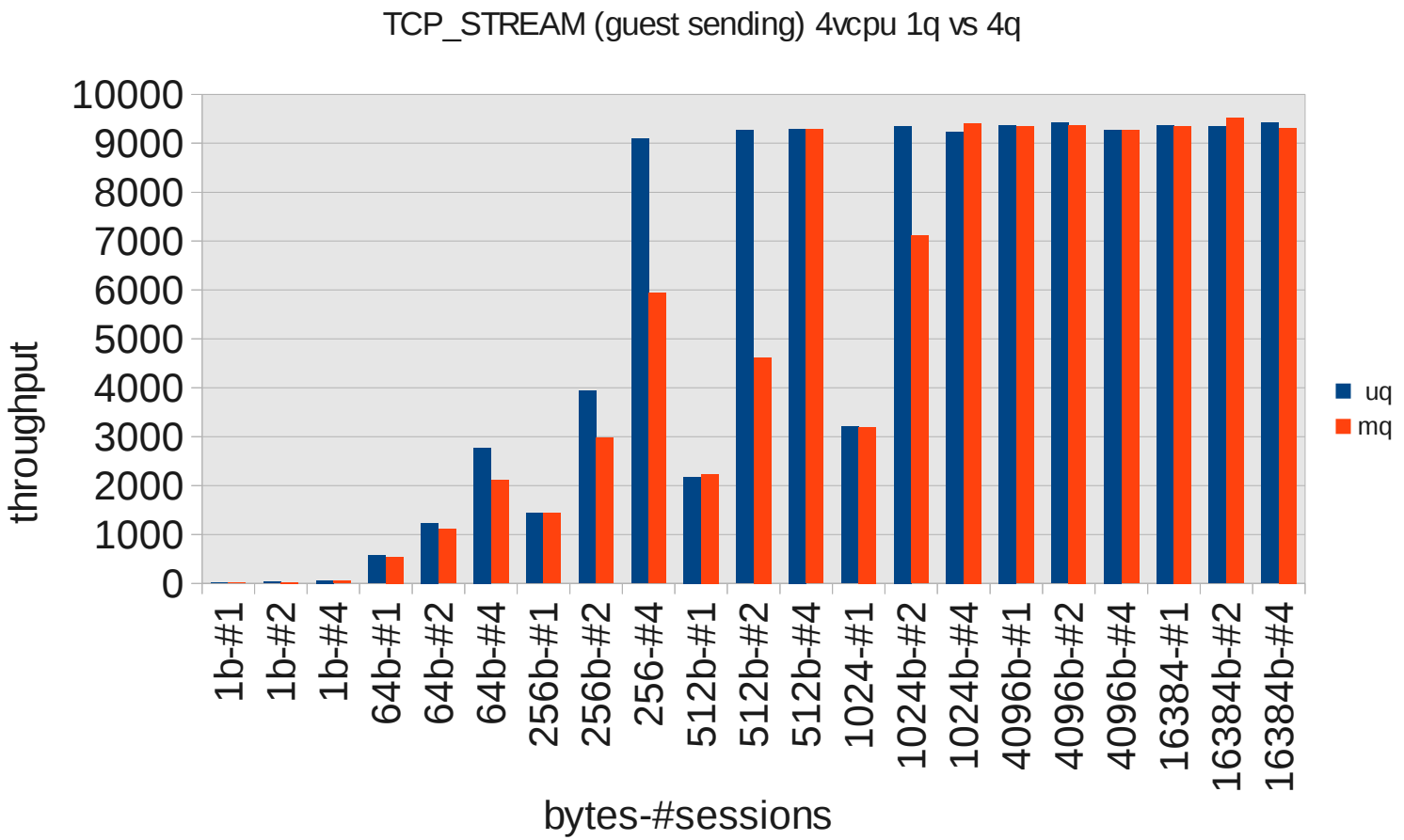
TCP_CRR result



Guest TCP receiving



Guest TCP sending



Performance Discussion

TX	size	sessions	throuput	cpu%	packets_sent	avg_pkt_size
sq	512	2	**9263**	46.84	**2737788**	**5689**
mq	512	2	**4662**	56.11	**3775715**	**1598**
%			**-50%**	+12%	**+38%**	**-72%**

- TCP tends to batch less
  - Latency is improved
  - Optimize in TCP?
  - Automatic mode switch?

TODO

- Performance optimization
  - stream performance
  - more sophisticated flow steering mechanism
  - NUMA consideration since more vhost threads were introduced
  - more test?
    - 40gbe, zero-copy, pv eoi, other card
  - non mq specific optimization

Q&A