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

Kernel
- kvm

Qemu Process
- ...mq...

Guest
- VCPU0
- VCPU1
- Virtio-Net Driver
- tx rx
- tx rx

Bridge
- MQ NIC
- tx rx
- tx rx
- tx rx

vhost_net
- mq
- mq TAP
- sk sk
## Changes required

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

- Multiple queue tap
  - move the socket to fd
  - each socket/fd a queue
  - allow fd to be attached
  - expose multiple sk
  - ioctl's 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_SETIFF**
  - **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

```c
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_RR 4vcpu 1q vs 4q

aggregate transaction rate

#sessions

TCP_RR 4vcpu 1q vs 4q

- blue line: uq
- red line: mq
TCP_CRR result

TCP_CRR 4vcpu 1q vs 4q

aggregate transaction rate

#sessions

1 10 20 50 100

TCP_CRR 4vcpu 1q vs 4q

0 10000 20000 30000 40000 50000 60000 70000

uq

mq
Guest TCP receiving

TCP_STREAM (guest receiving) 4vcpu 1q vs 4q

throughput

bytes-#sessions

uq
mq
Guest TCP sending

TCP_STREAM (guest sending) 4vcpu 1q vs 4q

throughput

bytes-#sessions

10000
9000
8000
7000
6000
5000
4000
3000
2000
1000
0

1b-#1
1b-#2
1b-#4
64b-#1
64b-#2
64b-#4
256b-#1
256b-#2
256b-#4
512b-#1
512b-#2
512b-#4
1024-#1
1024-#2
1024-#4
4096-#1
4096-#2
4096-#4
16384-#1
16384-#2
16384-#4

uq
mq
Performance Discussion

<table>
<thead>
<tr>
<th>TX</th>
<th>size</th>
<th>sessions</th>
<th>throughput</th>
<th>cpu%</th>
<th>packets_sent</th>
<th>avg_pkt_size</th>
</tr>
</thead>
<tbody>
<tr>
<td>sq</td>
<td>512</td>
<td>2</td>
<td><strong>9263</strong></td>
<td>46.84</td>
<td><strong>2737788</strong></td>
<td><strong>5689</strong></td>
</tr>
<tr>
<td>mq</td>
<td>512</td>
<td>2</td>
<td><strong>4662</strong></td>
<td>56.11</td>
<td><strong>3775715</strong></td>
<td><strong>1598</strong></td>
</tr>
</tbody>
</table>

| %  | -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