Empty Promise:
Zero-copy Receive for vhost

Kalman Meth, Mike Rapoport, Joel Nider
{meth,joeln}@il.ibm.com
rppt@linux.vnet.ibm.com

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreements No 645402 and No 688386.
Motivation

- No copy is better than copy
- Zerocopy TX without RX should feel lonely
- It was 8 years since the last attempt. Can we do better?
More motivation
### Zerocopy: TX vs RX

<table>
<thead>
<tr>
<th>Transmit</th>
<th>Receive</th>
</tr>
</thead>
<tbody>
<tr>
<td>● Downstream routing is easy</td>
<td>● Destination is not yet known</td>
</tr>
<tr>
<td>● Memory is always at hand</td>
<td>● Need memory for DMA</td>
</tr>
<tr>
<td></td>
<td>● Does not exist yet</td>
</tr>
</tbody>
</table>
Assumptions

- **Modern NICs are multiqueue**
  - Dedicate queues to virtual NIC

- **Guest allocates the buffers**
  - Remapping DMA region to guest is more complex

- **Tight coupling between physical and virtual NICs**
  - Restrict zero-copy-RX to macvtap

- **Minimal changes to guest**
Zero-Copy Rx Architecture

- Host
- VM Guest
- VM Guest
- User buffer
- User buffer
- User buffer
- Kernel space
- Guest kernel buffer
- Guest kernel buffer
- Guest kernel buffer
- Socket interface
- KVM Hypervisor
- Ethernet adapter
- DMA
- virtio
- macvtap
- macvlan
- Per-MAC ring buffer
- MAC1
- MAC2
- MAC3
- MAC4
- NIC
- Pass the buffers down through the kernel layers
- network
API changes

netdev

- `->ndo_set_zerocopy_rx(struct net_device *pdev, struct net_device *vdev)`
  - Pass `vdev` down the stack to the ethernet adapter to bind physical and virtual queues.
  - Similar to `->ndo_dfwd_add_station()`

- `->ndo_post_rx_buffer(struct net_device *dev, struct sk_buff *skb)`
  - Passes a single (page aligned) buffer to the ethernet adapter
  - `skb` contains pointer to the upper level device and `ubuf_info`
API changes (cont)

macvtap

- **MSG_ZCOPY_RX_POST**
  - Control message from vhost-net to macvtap to propagate the buffers from guest to the lower levels

- **MSG_ZCOPY_RX**
  - Flag indicating that message contains preallocated buffers that should not be copied to userspace
API changes (cont)

**virtio-net**

- `add_recvbuf_full_page()`
  - Ethernet adapter driver expects page size aligned buffers
  - `Existing add_recvbuf_*()` do not care since the data was always copied
Initialization

- Isolate set of queues in physical NIC
- Create 1:1 correspondence between physical and virtual queues
- Clear RX descriptor ring
- Drop pre-allocated RX buffers in physical NIC driver
Memory allocation

- **virtio-net (guest)**
  - Allocate buffers
    - DMA’able memory (PAGE_SIZE granularity and page aligned)

- **vhost-net**
  - Post buffers to macvtap
    - New control flag `MSG_ZCOPY_RX_POST` for `macvtap_recvmsg()`

- **macvtap**
  - Allocate skb
  - Map iovec to skb (similar to `zerocopy_sg_from_iter`)
  - Pass the buffers to physical NIC
    - New method `ndo_post_rx_buffer()`

- Physical NIC driver adds new buffers to RX descriptor ring
Packet receive

- **Physical NIC driver**
  - DMA directly to the guest buffers
  - Setup skb structure
  - `netif_rx()` and friends

- **macvtap**
  - Queue skb as ready for the userspace
  - Inform vhost-net about the virtio descriptor associated with the skb.
Packet receive (cont)

- **vhost-net**
  - *handle_rx_zero_copy():*
    - Update virtqueue
    - Kick macvtap with `recvmsg(MSG_ZCOPY_RX)`

- **macvtap (again)**
  - *macvtap_do_read_zero_copy()*
    - skb_array_consume
    - Cleanup
Implementation status

- Initial implementation
  - still sub-optimal
- Stable enough to benchmark
- Source:
  - https://github.com/mikelangelo-project/linux-zecorx
  - https://github.com/mikelangelo-project/qemu-zecorx
Test setup

- 2x IBM System x3550 M4 Server
  - Intel® Xeon® Processor E5-2660
    - 8 cores, 16 threads
  - 56G RAM
  - Intel 82599ES 10-Gigabit Network Connection (ixgbe)
- Back to back connection for host NICs
- VM with 4 vCPUs, 2G RAM
- Linux v4.8
- netperf 2.6.0
<table>
<thead>
<tr>
<th>Event 'cycles:pp'</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Baseline</strong></td>
</tr>
<tr>
<td>49.86%</td>
</tr>
<tr>
<td>12.35%</td>
</tr>
<tr>
<td>5.57%</td>
</tr>
<tr>
<td>5.21%</td>
</tr>
<tr>
<td>3.40%</td>
</tr>
<tr>
<td>2.77%</td>
</tr>
<tr>
<td>1.60%</td>
</tr>
<tr>
<td>1.30%</td>
</tr>
<tr>
<td>1.25%</td>
</tr>
<tr>
<td>1.20%</td>
</tr>
<tr>
<td>0.88%</td>
</tr>
<tr>
<td>0.82%</td>
</tr>
<tr>
<td>0.68%</td>
</tr>
<tr>
<td>0.68%</td>
</tr>
<tr>
<td>0.51%</td>
</tr>
<tr>
<td>0.46%</td>
</tr>
</tbody>
</table>
```bash
traffic-gen$ netperf -H guest
```

<table>
<thead>
<tr>
<th></th>
<th>v4.8</th>
<th>v4.8 + ZCRX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput (Mbit/sec)</td>
<td>9255.22</td>
<td>4396.34</td>
</tr>
<tr>
<td>System Utilization (%)</td>
<td>10.9095</td>
<td>12.2941</td>
</tr>
<tr>
<td>CPU usage - vhost (%)</td>
<td>74.676</td>
<td>95.979</td>
</tr>
<tr>
<td>CPU usage - qemu (%)</td>
<td>99.6685</td>
<td>100.83</td>
</tr>
</tbody>
</table>
- RX interrupt, and vhost are on the same CPU
- Retry with forced CPU affinity

```
vm-host$ taskset -cp 4 $(pgrep vhost)
vm-host$ echo 5 > /proc/irq/111/smp_affinity_list
traffic-gen$ netperf -H guest
```

<table>
<thead>
<tr>
<th></th>
<th>v4.8</th>
<th>v4.8 + ZCRX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput (Mbit/sec)</td>
<td>9255.22</td>
<td>8492.17</td>
</tr>
<tr>
<td>System Utilization (%)</td>
<td>10.9095</td>
<td>12.121</td>
</tr>
<tr>
<td>CPU usage - vhost (%)</td>
<td>74.676</td>
<td>86.1722</td>
</tr>
<tr>
<td>CPU usage - qemu (%)</td>
<td>99.6685</td>
<td>100.103</td>
</tr>
</tbody>
</table>
What happened

- New bottleneck is in DMA mapping
- Latency has grown
  - Measured with `rdtsc()` in `vhost-net::handle_rx()`
    - Copy: \( \approx 2 \text{ tscs/byte} \)
    - No copy: \( \approx 3.5 \text{ tscs/byte} \)
- Page recycling in ixgbe replaced with `ndo_post_buffers()`
  - Sequential instead of parallel
  - Move frequent `dma_map*()` / `dma_unmap()`
Can we do better?

- Maybe
- tl;dr
  - Try to re-parallel DMA mapping and RX processing
  - Better batching for memory allocation and dma_map/unmap
  - Major changes to virtio ring
  - AF_XDP based virtio backend in user space
Thank you!