Virtio Code Walkthrough

Mark McLoughlin
2008-11-05

virtio

- “A maze of twisty abstractions”
- Buffer queue
- Interrupts
- Device enumeration
- Device configuration

Buffer Queue Abstraction

- virtqueue_ops
- err = ops->add_buf(vq, sg, out, in, data);
- data = ops->get_buf(vq, &len);
- ops->kick(vq);
- ops->disable_cb();
- more_used = ops->enable_cb();

virtio_net TX

- Entry is net_device->hard_start_xmit()
- Add skb buffers to the queue
- Notify the guest
- On completion, detach and kfree_skb()
- skbs can have linear and paged data
- Header to pass GSO/checksum metadata

virtio_net RX

- Allocate skbs, add them to the queue
- On interrupt, vq_ops->get_buf()
- Translate GSO/checksum metadata
- Pass up the stack with netif_receive_skb()

virtio_net TX – Host Side

- Guest notifies of available packet buffers
- At some point we flush the queue
- pop() from queue and send it along
- push() back onto the queue once
- Notify the host that we're done

virtio_ring – What Lies Behind

- Ring == circular list of buffer descriptors
- Lockless add/remove
- virtio queue implemented with two rings
- Producer adds to “avail” ring
- Consumer adds to “used” ring
- Allows out-of-order consumption

virtio_ring

- Buffer desc table – addr, len, flags, next
- avail ring – desc idx, current position
- used ring – same, but also buffer size
- Both rings have flags e.g. NO_NOTIFY
- 128 entries == 8k
- 256 entries == 12k
- 1k entries == 32k

Segmentation Offload

- Larger packets == less overhead
- Partial checksums
- Scatter-gather I/O
- Generic segmentation

struct virtio_net_hdr

- flags – checksum not completed?
- gso_type – TSO vs. UFO
- hdr_len – headers/payload boundary
- gso_size – payload size per segment
- csum_start – where to start summing
- csum_offset – where to place the result

New tun/tap Features

- TUNSETIFF w/ IFF_VNET_HDR
- TUNGETIFF
- TUNGETFEATURES
- TUNSETOFFLOAD