High Performance Network I/O for Virtual Machines

L. Rizzo, G. Lettieri, V. Maffione

Dipartimento di Ingegneria dell’Informazione
University of Pisa

KVM Forum, 2013
Outline

1. The Problem
2. netmap integration
   - netmap API
   - Integration in QEMU
   - Performance
3. Fast e1000
   - Driver and emulation improvements
   - Paravirtualization
   - Integration with VALE
   - Performance
4. Open problems
The Problem
netmap integration
Fast e1000
Open problems

Problem
- Improving pps throughput between virtual machines and vm/host.
- Desirable for middlebox virtualization

Our solution
Try to use netmap
device-independent ring/buffers in shared memory
kernel/userspace sync only during syscalls (ioctl, poll)
14.88 Mpps on a 10GigE with a single 900 Mhz core
standard in FreeBSD since 9.1-RELEASE
distributed as a separate module for Linux (2.6.32–3.11)
Rizzo, Lettieri, Maffione

High Performance Network I/O for Virtual Machines

```c
fd = open("/dev/netmap", 0);
strcpy(req.nr_name, "eth0");
ioctl(fd, NIOCREGIF, &req);
mem = mmap(NULL, req.nr_memsize, PROT_READ|PROT_WRITE, 0, fd, 0);
nifp = NETMAP_IF(mem, req.nr_offset);
ring = NETMAP_RX_RING(nifp, 0);
for (;;) {
    ...
    poll(/* fd */ ...);
    for ( ; ring->avail > 0; ring->avail-- ) {
        i = ring->cur;
        buf = NETMAP_BUF(ring, i);
        use_data(buf, ring->slot[i].len);
        ring->cur = NETMAP_NEXT(ring, i);
    }
}
```
VALE: Virtual Local Ethernet

An extensible switch using netmap API
Already included in the netmap module
connects:
  virtual ports (netmap API only)
  netmap enabled real NICs
  the host stack
uses batching to improve pps (≈20 Mpps for 64 B pkts between two VPs)

http://info.iet.unipi.it/~luigi/vale/
Use special names starting with “vale”:

```c
fd = open("/dev/netmap");
strcpy(req.nr_name, "valeA:x");
ioctl(fd, NIOCREGIF, &req);
```

Then, same as before.

- virtual ports and switches are created on the fly
- all virtual ports with the same name before the “:” are connected to the same switch
QEMU networking

guest

frontend (e1000, ...)

tap backend

qemu_send_packet

write

tap

read

host kernel
QEMU-VALE integration

guest

frontend (e1000,...)

netmap backend

netmap/VALE

host kernel
QEMU-VALE usage

Attach vm to port x of VALE switch valeA

qemu -device e1000,netdev=n,mac=...  
-netdev netmap,id=n,ifname=valeA:x...

Attach another vm to port y of the same VALE switch.

qemu -device e1000,netdev=n,mac=...  
-netdev netmap,id=n,ifname=valeA:y...
Initial performance (1/2) [1]

KVM guest-guest 64 B UDP throughput

- tap
- VALE
- vhost

Mpps

e1000  virtio  e1000-netmap
Initial performance (2/2) [1]
Initial performance: problems

- terrible base performance of emulated devices
- a bit better for virtio, but not as much as expected
- unstable results
  - slightly slower tx → big improvement w VALE
  - huge packet drops
Fast e1000

- Interrupt moderation (only emulator modified)
- Send combining (only guest driver modified)
- Paravirtualization (both emulator and driver modified)
- Better integration with netmap/VALE
Interrupt moderation and Send combining

Interrupt moderation
- helps amortizing per-packet overheads
- Already merged in QEMU

Send Combining
- Don’t kick the host when a TX interrupt is pending
- Flush pending transmissions when the interrupt comes
- Works well with moderation (bigger batches)
Importing the essence of virtio to e1000

Real hw and emulated TX
TDT register writes used for both:
- updating status (available packets to send)
- notification (status has changed)

Paravirtualized TX emulation
Separate the two functions:
- status only updated in shared memory (Communication Status Block)
- TDT only used for notification
- TX processing in a separate thread
Paravirtualized TX path

- vCPU Thread
- TX ring
- IOThread
- CSB

- Interrupt
- TDT write
- qemu_send_packet

- g_tdt
- g_ntc
- g_wants_kicks
- h_wants_kicks

Rizzo, Lettieri, Maffione

High Performance Network I/O for Virtual Machines
Minimize consumer/producer notifications

**Producer notifications**
- Schedule the IOThread (consumer)
- Notifications disabled while consumer runs

**Consumer notifications**
- Interrupt the guest (producer)
- Notification enabled when TX ring is full (Tx lazy completion)
The Problem
- Netmap integration
- Fast e1000
- Open problems

Driver and emulation improvements
- Paravirtualization
- Performance

Current performance [2]

Graph showing kvm guest-guest 64 B UDP throughput for different devices:
- e1000
- im
- sc+im
- csb
- virtio
- vhost

The graph compares the performance of these devices in terms of Mpps (Millions of Packet Per Second).
Improving VALE batching

Problem

- frontends see batches of packets
- VALE backend may send batches of packets
- but the FE↔BE interface only allows one packet at a time

Implemented solution

- add flags to `qemu_send_packet`
- producer sets a “more packets coming soon” flag
- consumer can take informed batching decisions
Indirect buffers
Indirect buffers + VALE batching

Problem

Frontend does not know when buffers are consumed

Implemented solution

Register a callback from BE to FE
Now common to both e1000-paravirt and virtio

- A fast consumer may cause a high rate of kicks from the producer.
- The producer is slowed down and throughput drops (even by half).
- Unstable and counterintuitive measures.
Open problems (2/2)

**receive livelock**

- when reaching \( \approx 1 \text{ Mpps} \), receiver chokes
- in Linux guests, this invariably happens inside the kernel at the socket queue
- NAPI is still too aggressive w.r.t. the final consumer (i.e., user space)
- FreeBSD polling?

![Graph showing receive livelock response for different drivers](image)
References

L. Rizzo and G. Lettieri.
VALE: a switched ethernet for virtual machines.

L. Rizzo, G. Lettieri, and V. Maffione.
Speeding up packet I/O in virtual machines.