SOLARFLARE Communications

Accelerated Xen networking
Background

- Solarflare does user-level networking
  - TCP/IP stack linked with user app
  - Known as *Etherfabric*

- Smart NIC allows safe access from user-mode
  - Onboard IOMMU for safe DMA
  - Nic’s filter-table demuxes incoming packets to queues
  - Queues get mapped into user-mode processes

- Eliminates interrupts/syscalls/context switches
  - Can also do zero-copy tx from user-mode
Virtual Interfaces and Protection

- NIC has many Virtual Interfaces (VIs)
  - VI = Filter + DMA queue + event queue

- VIs largely about 'protection'
  - Allow untrusted entities to access the NIC without compromising system integrity
  - VIs mapped into untrusted entity’s address space

- Traditionally 'untrusted entity' means user process ... but it might equally mean a guest VM kernel
Xen: I/O via Dom0

- All 'real' drivers live in Dom0
- DomU kernels have pseudo drivers that communicate with Dom0 via the hypervisor
- Necessary because only Dom0 is 'trusted'
But with Etherfabric...

- Accelerated routes set up in Dom0
  - Then DomU can access hardware directly
- DomU can access hardware directly + safely
  - At least most of the time; still slow path via Dom0
Front-end and back-end

- Front-end driver in DomU
  - Where possible, interacts with h/w directly
    - This is known as “fast path”
    - Otherwise packets routed via Dom0: “slow path”
    - Need to be careful about Tx traffic for local machine

- Back-end driver lives in Dom0
  - Sets up VI per guest VM
    - Installs filters “on-demand” as traffic flows in/out
    - Enforces quotas on front-end drivers
Architecture

Back-end driver

Dom0

DomU

Shared pages

Hardware dependent driver

h/w independent

FIFOs

In shared pages:

• VIs
• FIFOs
• Tx/Rx buffs
• Local MAC tbls

RxQ TxQ EvQ

RxQ TxQ EvQ

VI
Front-end driver

- fend communicates with bend over FIFOs
  - Implemented using shared pages

- Front-end driver split into two parts
  - Hardware dependent and independent parts
  - H/w independent part loads h/w dependent part
    - Selects based on message from back-end
    - On Linux, this message is just name of a module
    - Non-fatal error if h/w dependent module not found
On migration, back-end sends ‘h/w unavailable’ message to front-end
  - Everything then goes over slow-path

Possibly followed by ‘new h/w available’ message at the other end

If no accelerated hardware available, or if no driver for that hardware available: slow path!
# Early performance results

<table>
<thead>
<tr>
<th></th>
<th>STREAM tx (Mbps)</th>
<th>STREAM rx (Mbps)</th>
<th>TCP/RR (trans. per sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vanilla Xen</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accelerated</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Native</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Early performance results

STREAM rx (Mbps)
STREAM tx (Mbps)
TCP/RR (trans per 100ms)

Vanilla
Xen
Etherfabric
Xen Dom0
CPU Usage

- STREAM
- rx
- STREAM
- tx
- TCP/RR

Graph showing CPU usage with different graphs labeled:
- Vanilla Xen (Dom0+DomU)
- Etherfabric (Dom0 + DomU)
- Xen Dom0 (Dom0)
What's next?

- Submission of patch to xen-devel
  - Some small changes contributed already
  - Convenient to fold bend and fend into mainline

- Nice features to have in next-gen hardware
  - MAC filtering (on receive and transmit)
  - MSIX (interrupt line per guest)
  - Local dest tx DMA-ed directly to local guest