Virtualizing InfiniBand in Xen

Prototype Design, Implementation and Performance

Jiuxing Liu, Bulent Abali (IBM)
Wei Huang, DK Panda (The Ohio State University)

Presented by Jiuxing Liu (jl@us.ibm.com)

Xen Summit 2006
Presentation Outline

- InfiniBand Overview
- Prototype Design
- Performance Results
- Future Plans
InfiniBand

- Interconnect based on industry standard
- High performance
  - Latency as low as several microseconds
  - Bandwidth of 10Gbps (4x) and 30Gbps (12x)
- Intelligent hardware
  - OS-bypass
  - RDMA
InfiniBand Access Models

- **Privileged Access**
  - OS involved
  - Resource management and memory management (opening HCA, creating queue-pairs, registering memory, etc.)

- **Direct Access**
  - Can be done directly in user space (OS-bypass)
  - Queue-pair access (posting send/receive/RDMA descriptors) and CQ polling
QP Access Details

- **Initialization Steps (privileged access)**
  - Map doorbell page (UAR)
  - Allocate and register QP buffers
  - Create QP

- **Communication steps (direct access)**
  - Put descriptors in QP buffer
  - Write to doorbell page
CQ Polling Details

- **Initialization steps (privileged access)**
  - Allocate and register CQ buffer
  - Create CQ

- **Communication steps (direct access)**
  - Poll on CQ buffer for new completion entry
OpenIB Gen2 Driver Stack

- Core module (hardware independent)
- HCA provider module (hardware dependent)
Presentation Outline

- InfiniBand Overview
- Prototype Design
- Performance Results
- Future Plans
IB Virtualization Prototype Overview

- Follows Xen split driver model
- Presents virtual HCAs to guest domains
  - Paravirtualization
- Maintains the same IB VERBS interface
  - Supports existing drivers and applications
- Enables VMM-bypass access
- Reuses existing IB code as much as possible
Implementation Details

- Front-end implemented as a new HCA provider module (reusing core module)
- Backend uses kernel threads to process requests from front-ends (reusing IB drivers in dom0).
- Based on Mellanox MT23108 HCAs (possible to make it hardware independent)
Prototype Structure

- dom0
  - Back-end
  - Core IB Module
  - HCA Provider

- domu
  - User Level IB Provider
  - Core IB Module
  - Virtual HCA Provider
  - VMM-Bypass Direct Access

- HCA Hardware

Privileged Access
Handling Key IB Operations

- **Privileged Access**
  - Most operations
  - Memory registration
  - CQ, QP creation

- **Direct Access (VMM-bypass)**
  - QP access
  - CQ polling

- **Event Handling**
Handling Privileged IB Operations

- Front-end sends request to back-end.
- Back-end does the work, sends back reply.

- Resources are allocated at back-end.
- Front-end uses handles when referring to resources.
Memory Registration

- Front-end does memory pinning and translation.
- Physical (machine) page info is sent to back-end.
- Back-end registers physical pages with HCA.
- Back-end sends local and remote keys to front-end.
CQ, QP Creation

- CQ, QP buffers are allocated at guest domain.
- Front-end registers CQ, QP buffers.
- Front-end sends request to back-end, with CQ, QP buffer keys.
- Back-end create CQ, QP using those keys.
VMM-Bypass Access

- **QP Access**
  - Doorbell page is mapped into address space (needs some support from Xen).
  - Put descriptor into QP buffer (QP buffer is located at front-end.)
  - Ring the doorbell

- **CQ Polling**
  - Can be done directly because CQ buffer is allocated in guest domain
CQ/QP Event Handling

- Uses a dedicated device channel (Xen event channel + shared memory)

- Special event handler registered at back-end for CQs/QPs in guest domains
  - Forwards events to front-end
  - Raise a virtual interrupt

- Guest domain event handler called through interrupt handler
Presentation Outline

- InfiniBand Overview
- Prototype Design
- Performance Results
- Future Plans
Performance Results

- Latency
- Bandwidth
- Latency (blocking)
- Memory registration
- MPI latency and bandwidth
- NAS application
Testbed

- Dual Intel Xeon 3.0 GHz CPUs
- ServerWorks GC LE chipset
- 133 MHz 64 bit PCI-X bus
- Mellanox MT23108 HCAs
- InfiniSwitch
Latency

RDMA Write Latency

Send/Receive Latency
Bandwidth

RDMA Write Bandwidth

Send/Receive Bandwidth
Latency (Blocking)

Blocking Send/Receive Latency

IB-Xen
Native

Bytes
Microseconds

0 5 10 15 20 25 30 35 40
1 2 4 8 16 32 64 128 256 512 1024 2048 4096

IBM Research
© 2006 IBM Corporation
Memory Registration

![Graph showing Memory Registration Cost]

- **Xen Summit 2006**
- **Registration Cost**
- **Number of Pages**
- **Microseconds**

Graph legend:
- **IB-Xen**
- **Native**
MPI Latency and Bandwidth

MPI Latency

Bytes: 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192

Microseconds: 0, 5, 10, 15, 20, 25, 30, 35

IB-Xen
Native

MPI Bandwidth

Bytes: 1, 16, 64, 256, 1024, 4096, 16384, 65536, 262144

MB/s: 0, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000
NAS Parallel Benchmarks

Normalized Execution Time

- **DomU**
- **Native**

<table>
<thead>
<tr>
<th></th>
<th>IS</th>
<th>FT</th>
<th>CG</th>
<th>SP</th>
</tr>
</thead>
<tbody>
<tr>
<td>DomU</td>
<td>0.9</td>
<td>1.2</td>
<td>0.9</td>
<td>1.0</td>
</tr>
<tr>
<td>Native</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>
Future Plans

- Improve code quality
- Xenbus/xenstore integration
- IB management
- Other Upper layer protocols: IPoIB, SDP, etc.
- IB-Xen based HPC
- Checkpointing and migration
Thanks