System Impact of Integrated Interconnects

Sudhakar Yalamanchili

Computer Architecture and Systems Laboratory
Center for Experimental Research in Computer Systems
School of Electrical and Computer Engineering
Georgia Institute of Technology
Acknowledgements

- This talk covers work @GT contributed by students and colleagues

- Graduate Students
  - Kendall Chuang, Greg Diamos, Vishakha Gupta, Jeffery Young

- Faculty
  - Ada Gavrilovska, Jose Duato, Karsten Schwan, Fede Silla
The Traditional Cost of Communication

- Local physical address space
- Remote physical address space
- Cluster Network
- Message
- Sender
  - Sending overhead
  - Transmission time (bytes/bandwidth)
- Receiver
  - Time of flight
  - Transmission time (bytes/bandwidth)
  - Receiving overhead
  - Interaction with the operating system
- Overhead
- Total latency
On-Chip Integrated Interconnects

Low latency, high bandwidth interconnect

HW distance from the wire to memory controllers has dropped dramatically

What about synergistic trends?

From http://elnexus.com/mail/07-09-barcelona.jpg
Network Technology Trends

- Link and switch cut-through latencies have been dropping faster than DRAM latencies.
- We already see commodity board-level, multi-socket coherent, NUMA machines.
- Lesson: Memory reach can be extended beyond board with small performance penalties.
Memory Cost and Power Trends

- Memory cost is a non-linear function of size and chip density
- Power expended in the memory system has become significant
- Lesson: Need effective sharing of memory
Memory Pin Bandwidth

**Trends**
- Performance via core scaling
- Relatively slower growth in pin bandwidth
- Consolidation via virtualization → increased memory pin bandwidth demand

**Memory bandwidth/core decreasing → increased memory pressure**

**Lessons:**
- Use network pins to increase (remote) memory BW
- Need more pins → 3D stacks, network-on-package, packetized memory controllers

---

3) International Technology Roadmap for Semiconductors, Executive Summary 2007
Modern Data Center & Cloud Computing

- Modern data centers are presenting challenges in terms of cost and power efficiency
  - Statistics suggest that 1.5% of all U.S. energy costs go to datacenters and this cost could double by 2011.
  - Focus has been on CPU power and overall cooling of datacenter.
  - Memory cost and power has an increasing impact.

Rethink architectural concepts in how we integrate and communicate with components.

---

1) EPA Report to Congress on Server and Data Center Efficiency, 2007

Photo from http://eetd.lbl.gov
Consequences

- On-Demand Sharing
  - Extend memory reach
  - Extend reach of virtualization

- Redefine traditional boundaries
  - Integrated on-chip memory controller/network interface

- Evolution to new system abstractions
1. On Demand Memory Sharing

- Today: provision blades for worst case memory footprint

- Add capacity via additional memory on a blade or additional blades

- System wide memory fragmentation
  - Application footprints are time-varying
  - Provisioning for simultaneous peak demand is inefficient
Virtualizing Memory “Location”

- Provision memory/blade for average demand not peak demand
- Dynamically size NUMA machines
- Islands of coherence
  - Variable amount of physical memory per coherence domain
The Software View

- Abstractions for “borrowing”
  - Lease model \(\rightarrow\) time dependent
  - Spill model \(\rightarrow\) a few fat nodes

- Memory configuration optimization for the data center
System View

Dynamic Partitioned Global Address Space (DPGAS)

- Extend the preceding model to a system wide global address space
  - Flexible assignment of physical address space of individual processes to memory controllers
- Protection issues handled by virtual memory system
  - Bridge mapping handled and coordinated by OS
  - System-wide RDMA
Impact

Young, Yalamanchili, Duato, Silla (2008, 2009)

- Simulated allocation of benchmark memory footprints
- Using 64GB(56GB)/node
- Potential cost savings - 15% to 26% ($30,000 and $200,000)
Impact on Data Centers

- System wide optimization
  - Fat blades that serve memory on demand
  - Neighbors get you most of the way
  - Placement of fat blades

- Local interconnects
  - Native nearest-neighbor topologies based on packaging
  - Move the network into the memory system

- System Abstractions
  - Dynamic partitioned global address spaces (DPGAS)
  - Platform models
2. Integrating Memory and Communication

- The hardware distance from the wire to the memory controller is getting smaller

- Can we move the NI functionality to DRAM?

- Consequences
Integrated Scheduling of Memory Traffic

From Core

Qos Scheduler

Memory Scheduler

TX FIFO

RX FIFO


Application #1

Pipelined scheduling logic

Application #n

Scheduler

- VPID 1
- VPID 2
- VPID 3

Time

0 1/5 2/5 3/5 4/5 1
Consequences

- Memory controller demand bandwidth increases

- Distinct behaviors at the memory controller
  - Different locality behaviors → scheduling disciplines
  - Direct feedback from multiple cores

- Quality of service scheduling disciplines at the memory controller
  - High speed architectures for Dynamic Window Constrained Scheduling (West & Schwan)
Consequences (cont.)

- Integrated NICs for a fast messaging path
  - Reduce system complexity and consequently power and cost
  - VELO (UoH) and EV7 (DEC)

- Dedicated core to deal with communication data structures
  - Helper threads (Schlansker, et.al. SC07)
  - Asymmetric multicore $\rightarrow$ simpler helper cores
3. Using Accelerators

**Coupled Accelerators**

Using Accelerators

- Multicore Host
- NVIDIA 8800 GTX

**Lightweight multiprocessor**

PCIe 8 GB/s

DDR Memory

- 20GB/s PCIe 8 GB/s
- DDR Memory

School of Electrical and Computer Engineering | Georgia Institute of Technology

CASL 20
A Risk Assessment Application

- Inner loop replaced by GPU equivalent

```java
Model
readInputs();
computeInvariants();
for all chunks
{
    simulateChunk();
}
generateResults();
```

Diamos, Yalamanchili (2008)

Data parallel implementation

```java
for all chunks
{
    simulateChunk();
}
```

Generated offline

C Implementation

CUDA Implementation

Link to one implementation
Productivity Aspects

- Consequences of tight coupling
  - Fine grained sharing
  - Predictable performance gains
  - Operate as multicore and not multinode

- GPU Software development requires:
  - Programming and execution models
  - Data copy operations to and from GPU memory
Problem Scaling - Measured

With latest CPUs (2x faster) and GPUs (4x faster), GPU advantage should grow by 2x
Problem Scaling - Projected

For 1M points/min
For one million data points / min

<table>
<thead>
<tr>
<th>Language</th>
<th>#Processors</th>
<th>Cost (processors)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>356 CPUs</td>
<td>$30616</td>
</tr>
<tr>
<td>C</td>
<td>83 CPUs</td>
<td>$7138</td>
</tr>
<tr>
<td>C/CUDA</td>
<td>20 GPUs</td>
<td>$2925</td>
</tr>
</tbody>
</table>

Slope depends on PCIe latency

Using per processor costs for web vendors
Scalable Portable Execution – Harmony Runtime

Diamos, Yalamanchili (2008)

Cap Model 3
readInputs();
computeInvariants();
for all chunks
{
    simulateChunk();
}
generateResults();

Harmony Run-time

- **Transparent** scheduling, execution management of chunks
- **Binary** compatibility across system sizes

Network (e.g., Hypertransport)

V2 of Harmony runtime complete
- Single, node multiple GPUs
- “Distance matters”
Example: MultiCell Accelerator

Gupta, Yalamanchili, Duato, 2007

- Implementation for a MultiCell architecture
- Using a parallel, multicell simulator based on MAMBO
- Version of a Platform API for application development
Heterogeneous Virtualized Manycores

- Virtualized execution for sharing multiple GPUs
- Sensitivity to GPU-Core latency
  - Scheduler must account for latency cost
- Feasibility of fine grained virtualization over HT (University of Heidelberg)
Developer sees a pool of accelerators, homogeneous multicore processors, and a flexible memory hierarchy

- Configure a virtual machine to meet the needs of the application
- Accelerators are slaves to homogeneous cores

Flexible, low latency interconnect to couple and isolate platforms
Future Thrusts – Some Thoughts

Partitioning/Virtualization
- Direct vs. indirect
- Regular vs. irregular
- High radix vs. low radix
- Universal (emulation) vs. custom

Reliability
- Oblivious vs. adaptive
- Source routed vs. switch

Thermal/Power
- Wormhole, packet, Circuit, VCT
- Hybrid switching
- Virtual channels
- Flow control & clocking
- Buffering & arbitration
- Noise margins

Design Space
Thank You!
v1 Implementation Detail

[Diagram of v1 Implementation Detail]

1. Interposer Library detours calls to the duplicate library
2. Duplicate CUDA shared library
3. CUDA bridge device
4. Each block contains CUDA Call Packets
5. Kernel device handles the write call issued by the duplicate CUDA library
6. User
7. Backend is just another application for CUDA library
8. Actual CUDA shared library
9. Shared Memory Ring
10. Event Channel
Eliminating Copies

Map Page Directory across dom0 and domU

- Each entry page with frame numbers
- Lets us deal with larger parameter sizes

Let us deal with contiguous memory wrt to all domains (dom0 and domU and physical) no shared Mm

CUDA application uses Mmap operation for allocating kernel memory to user space pointer

Pre-shared memory from DomU to Dom0 using page directory

Allocated pages for the current application from shared memory

Backend Application reads mfn's of the shared memory using page directory and maps its pointer

no shared Mm

with shared Mm (no copy)