Scalable Resource Composition in a Flat World

Sudhakar Yalamanchili

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

Sponsors: National Science Foundation, NVIDIA, LogicBlox Inc.

System Diversity

Amazon EC2 GPU Instances
Cray Blue Waters
Phase Change Memory
Photonics

Technology Diversity is mainstream
Composing Memory

While 8 GB DIMMs are most cost-effective, larger DIMM sizes are most power-efficient

Resource Sharing

A Simple Model of Savings

- 1.5 to 16 Watts per node on average
- Savings for a 10,000 core data center would be 3,540 Watts
- Synthetic footprint traces from server applications

**Bandwidth Trends**

DRAM to interconnect bandwidth ratio has been steadily dropping

**Latency Trends**

- MPI latency has steadily approached DRAM read latency
- Hardware switching times in the low hundreds of nanoseconds.
- Note progress in photonics

*Extend the reach of a socket*
The Memory Wall

"Multicore Is Bad News For Supercomputers"
IEEE Spectrum 2008

- Data intensive applications
- Memory bandwidth demand is scaling faster than memory interface capacity

"You can buy bandwidth but you cannot bribe God"
- unknown

Convert Network Bandwidth into Memory Bandwidth

Impact on Clustering

- Combine commodity interconnects and memory systems
- Need flexible hardware level composition of resources
- This is an old idea whose time has come?

Some Examples
- Lim, et al. - Memory Blades for disaggregated memory
- Tolentino, Cameron – Memory Miser OS level support
- Lefurgy, et al. – DRAM server power and DRAM consolidation
- RDMA - Liang ’05 low-level implementation for page swapping
- Memscale – UoH, UPC
- Feng et. Al – Green Supercomputing
Flattening Cluster Hierarchies

**Observation I:**
Everyone is getting closer and we need better sharing but.....

Post Dennard Performance Scaling

\[
\text{Perf} \left(\frac{ops}{s}\right) = \text{Power} (W) \times \text{Efficiency} \left(\frac{ops}{\text{joule}}\right)
\]

Dally, Keynote IITC 2012

Specialization $\rightarrow$ heterogeneity and asymmetry

\[
\text{Operator cost} + \text{Data movement cost}
\]

Three operands x 64 bits/operand

Energy = # bits $\times$ dist $-$ mm $\times$ energy $-$ bit $-$ mm
Hardware Power-Performance Tradeoffs

Customization is key to power performance!

Programmability/Flexibility

GOps/Watt

ASIC

Xilinx Virtex 6

FPGA

NVIDIA Tesla

TMS320671D

GPU

In-Order Processor

DSP (LP)

OOO Processor

Westmere-EP

Programmability/Flexibility

Consolidation on Chip

Programmable Pipeline (GEN6)

Vector Extensions

AES Instructions

Programmable Accelerator

Multiple Models of Computation

Multi-ISA

16, PowerPC cores

Accelerators

• Crypto Engine
• Regular Engine
• XML Engine
• CP-Preserver Engine

Intel Sandy Bridge

Intel Knights Corner
Consolidation in a System

So its not just memory that needs to be shared!

Post Dennard Performance Scaling

\[
\text{Perf} \left( \frac{\text{ops}}{s} \right) = \text{Power}(W) \times \text{Efficiency} \left( \frac{\text{ops}}{\text{joule}} \right)
\]

Dally, Keynote IITC 2012

Specialization \rightarrow \text{heterogeneity and asymmetry}

Operator\_cost + Data\_movement\_cost

Three operands \times 64 \text{ bits/operand}

Energy = \# \text{ bits} \times \text{dist} - \text{mm} \times \text{energy} - \text{bit} - \text{mm}
Scaling: Key Driver is Energy/Power

- Sustain performance scaling through **massive concurrency**
  - New execution models
- **Data movement** becomes more expensive than computation

Cost of Data Movement

Observation II:

You can hide latency, but you cannot hide energy!
A Data Rich World

Mixed Modalities and levels of parallelism

Irregular, Unstructured Computations and Data

System Model

Programming Models
- Domain Specific Languages
- Compiler and Run-Time Support
- Cluster Wide Hardware Consolidation
- Hardware Customization

Data Movement Optimizations

System Abstractions
- e.g. GAS, Virtual DIMMs, etc
Application: Data Warehousing

- On-line and off-line analysis
  - Retail analysis
  - Forecasting
  - Pricing
  - ......
- Combination of data queries and computational kernels
- Current applications process 1 to 50 TBs of data [1]
- Potential to change a company's business model!


Databases: *Not* a Traditional Domain of GPUs

LargeQty(p) <- Qty(q), q > 1000.
......

Relational Computations Over Massive Data Sets
Database Applications on GPUs

- The good
  - Lots of potential data parallelism
  - If data fits in GPU mem, 2x—27x speedup has been shown

- The bad
  - Very large data set (will not even fit in host memory)
  - I/O bound (GPU has no disk)
  - PCI data transfer takes 15–90% of the total time

- The Ugly
  - Irregular/unstructured accesses to data

<table>
<thead>
<tr>
<th>Order</th>
<th>Price</th>
<th>Discount</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10</td>
<td>10%</td>
</tr>
<tr>
<td>1</td>
<td>20</td>
<td>20%</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>15%</td>
</tr>
<tr>
<td>3</td>
<td>51</td>
<td>14%</td>
</tr>
<tr>
<td>4</td>
<td>33</td>
<td>13%</td>
</tr>
<tr>
<td>5</td>
<td>22</td>
<td>10%</td>
</tr>
<tr>
<td>......</td>
<td>......</td>
<td>......</td>
</tr>
</tbody>
</table>


Research Thrusts

- I: Optimized implementations of primitives
  - Relational algebra
  - Data management within the GPU memory hierarchy

- II: In-core processing
  - Cluster wide memory aggregation techniques
  - Change the ratio of host memory size to accelerator memory size

- III: Data movement optimizations
  - Between hosts and (local or remote) accelerators
  - Within an accelerator
I. Relational Algebra Primitives on GPUs

**Raw Performance (C2050)**

*Fastest in GPU*

- Multi-stage algorithm (under review)
- Push to memory-bound
- Simple primitives are close to maximum performance
- Improved primitives under development

---

**TPC-H Query 1-Overall Performance**

*Scale Well*

*Can be optimized 10x by using specialized primitives*

Limit due to GPU memory!
TPC-H Query 1-Breakdown Normalized Performance

- **Solutions (TBD)** use Radix Sort

Sends most time in JOIN and SORT

System Model

- **Programming Models**
  - Domain Specific Languages

- **Data Movement Optimizations**
  - Compiler and Run-Time Support

- **System Abstractions**
  - Cluster Wide Hardware Consolidation
  - e.g. GAS, Virtual DIMMs, etc

- **Hardware Customization**
II. In-Core Processing

- Cluster-based memory aggregation
- Hardware support for global non-coherent, physical address space system
- Change the ratio of host-memory : GPU-memory

Oncilla: Fabrics for Accelerator Clouds

- **Goal**: Efficient memory aggregation for accelerators in data centers
- **Solution**: Use Global Address Spaces (GAS) and commodity fabrics (HT, QPI, PCIe, 10GE, IB)
- Support in-core databases using software from Red Fox project
**Oncilla Infrastructure**

- Low-latency, commodity hardware (Extoll) for efficient memory and GPU aggregation and Red Fox SW layer supports DB queries on remote nodes
- Collaboration with University of Heidelberg (UH), Polytechnic University of Valencia, AIC Inc., LogicBlox Inc.

**Some Candidate Systems Concepts**

1) Young, J., Yalamanchili, S., Dynamic Partitioned Global Address Spaces for Power-Efficient DRAM Virtualization, WIPGC at IGCC, 2010

Estimated two-way latency is on the order of 2.24 µs for 64B cache line read

1) Young, J., Yalamanchili, S., Dynamic Partitioned Global Address Spaces for Power-Efficient DRAM Virtualization, WIPGC at IGCC, 2010

**Remote GPU Access**
Oncilla

- Oncilla GAS runtime simplifies memory management and enables low-latency scheduling across all types of memory.
- Oncilla GAS runtime and API allow for one-sided data movement and optimizations between different memory locations and then export this information to the Red Fox scheduler.
- Combines best features of NVIDIA's UVA, EXTOLL hardware provides a simplified API to scheduler to describe available compute and memory resources.

System Model

- Data Movement Optimizations
- System Abstractions e.g. GAS, Virtual DIMMs, etc
- Programming Models
- Domain Specific Languages
- Compiler and Run-Time Support
- Cluster Wide Hardware Consolidation
- Hardware Customization
III. Data Movement Optimizations

CPU (Multi Core)
2-12 Cores

MAIN MEM
~128GB

GPU
~512 Cores

GPU MEM
~6GB

Intra-node and inter-node transfers

Transfers through the memory hierarchy

Kernel Fusion – Aggregate computation to reuse data

Kernel Fission – Overlap computation with data transfer

Kernel Fusion

A1: 1 2 3 A2: 4 5 6

Kernel A

5 7 9

A3: 2 4 6

Kernel B

Result

A1 A2 A3

Kernel A

Fused Kernel

A1: 1 2 3 A2: 4 5 6 A3: 2 4 6

A1 A2 A3

Fused Kernel A, B

Result
Kernel Fusion Benefits

- Smaller Data Footprint
  - Reduction in Memory Accesses
- Temporal Data Locality
- Reduction in Traffic
- Larger Input Data

- Larger Optimization Scope
  - Common Computation Elimination
  - Improved Compiler Optimization Benefits

Common RA Combinations of TPC-H
Domain Specific Compilation: Red Fox

- Targeting Accelerator Clouds for meeting the demands of data warehousing applications
- In-core databases

Time and Space Gains

\[
\begin{align*}
\text{No PCIe transfers} & & \text{With PCIe Transfer} \\
\text{Avg} &= 2.98X & \text{Avg} &= 1.98X
\end{align*}
\]
Memory and Optimization Scope

Memory Accesses

Impact of Optimization Scope – O3 vs. O0

TPC-H Queries

Query 1

Query 21

Avg = 1.25X (3.18X w/o SORT and PCIe)

Avg = 1.22X
Example of Kernel Fission

Reminiscent of Software Pipelining

1.37x speedup

People

Gregory Diamos
Dynamic optimizations (Harmony, Ocelot, LLVM Bridge)

Andrew Kerr
Program Transformations & Optimizations for Data Parallel Computation (Ocelot, LLVM Bridge, VSIPL)

Haicheng Wu
Dynamic Optimizations (Ocelot)

Jeff Young
Integrated Networks-Memory, Oncilla

Si Li
Correctness & Emulation Tools, GP architectures
Summary

- Refactor cluster architectures for
- Flexible hardware composition of resources and
- Migrate to a communication-centric model of algorithms, systems, and optimizations