Shadowfax: Scaling in Heterogeneous Cluster Systems via GPGPU Assemblies

Alexander Merritt, Vishakha Gupta, Abhishek Verma, Ada Gavrilovska, Karsten Schwan

{merritt.alex,abhishek.verma}@gatech.edu
{vishakha,ada,schwan}@cc.gtaech.edu
Hetereogeneous Computing

• Rapid evolution of hardware used in high-performance computing
  – Many-core, accelerators, complex memory structures

• **GPGPUs predominant**

• Many system and node configurations
  – Cache & memory hierarchies, GPU specs, ...

• Exposure to broader application domains
  – Scientific simulations
  – Data-intensive apps, low-latency codes (finance), high-throughput codes (massive web apps.), ...
Sample Configurations

• Oakridge “Keeneland”
  - Per instance:
    • 33.5 “EC2 Compute Units”
    • 2 GPGPUs (Fermi M2050)
    • 10G Ethernet
    • 22 GiB memory

• Amazon EC2
  - Per instance:
    • 33.5 “EC2 Compute Units”
    • 2 GPGPUS (Fermi M2050)
    • 10G Ethernet
    • 22 GiB memory

• Teragrid
  - “Forge”
    • 64 cores in 8 sockets
    • 8 GPGUs (Fermi M2070)
    • 64 GiB memory
  - Others with older GPGPUs and fewer CPU cores.
Existing Constraints

• Efficiency = fine-tuning of code
  CUDA Best Practices Guide, 76pgs

• Static node configuration
  – Code must match hardware

• Nodes can contain only so much

Georgia Institute of Technology - For VTDC'11
Outline

• Introduction

• GPGPU Assemblies
  – Idea & Design
  – Implementation

• Evaluation

• Conclusion
GPGPU Assembly

- *Virtual platform* of CPUs and GPGPUs
- GPGPUs abstracted as vGPUs to apps

1. Scalability
2. Utilization
3. Portability
Shadowfax: Implementation

CUDA Calls

Guest VM
- CUDA Applications
- CUDA Calls/Responses
  - Interposer Library
- Call Request/Response Packets
  - Frontend Driver
  - Linux

Shared Call Buffer (Ring)

Backend in Dom0
- Per GPU Scheduler
  - Select
  - Per Guest Pollers
    - Request/Response Packets
      - CUDA Call Wrapper
        - CUDA Calls/Responses
          - CUDA Runtime + NVIDIA Driver
            - Linux

Assembly Management

Remote Guests’ Network Handler
- Remote Request/Response Packets
  - Call Packets
    - Marshal/Unmarshal module

Management Domain

Ethernet or Infiniband Network

Management information

Calls request/response

Guest Domain

VCPU VCPU

VCPU VCPU

VCPU VCPU

Xen Hypervisor

Local Node Hardware
- GPGPU
- GPGPU
- CPU
- CPU

Georgia Institute of Technology - For VTDC'11
vGPU

Local vGPU

Call Buffer & Shared Memory
Polling Thread

CUDA Runtime
GPGPU

Remote vGPU

Server Thread
Machine Boundary
CUDA Runtime
GPGPU

Georgia Institute of Technology - For VTDC'11
Assembly: Key Ideas

• Consistent programming model

• Dynamic vGPU ↔ GPU attachment
  (any GPGPU in cluster)

• Match workload to hardware availability
Outline

• Introduction
• GPGPU Assemblies
• Evaluation
  – Experimental System
  – Workloads
  – Results
• Conclusion
Experimental System

Common Hardware:
- 4 CPU cores
- 4 GiB memory

Software Stack:
- Xen 3.2.1
- Linux 2.6.18
- CUDA v1.1
- NV Driver 169.09

Guest Configuration:
- 1 VCPU
- 256 MiB memory
- Linux 2.6.18

GeForce 9800 contains two GPUs internally.

NVIDIA GeForce 8800
NVIDIA GeForce 9800

Application VM
Gigabit Ethernet
Additional Node

Georgia Institute of Technology - For VTDC'11
# Workloads

<table>
<thead>
<tr>
<th>Class</th>
<th>Source</th>
<th>Benchmarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finance</td>
<td>CUDA SDK</td>
<td>Binomial Options, Black-Scholes</td>
</tr>
<tr>
<td>Media</td>
<td>CUDA SDK &amp; Parboil</td>
<td>Matrix Multiply, MRI-FHD</td>
</tr>
<tr>
<td>Science</td>
<td>Parboil</td>
<td>Coulumb Potential</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binomial Options</td>
<td>Few “fat” kernels, some synchronization</td>
</tr>
<tr>
<td>Black-Scholes</td>
<td>Many “small” asynchronous kernels</td>
</tr>
<tr>
<td>Matrix Multiply</td>
<td>Large upfront data movement, 1 kernel</td>
</tr>
<tr>
<td>MRI-FHD &amp; CP</td>
<td>Few kernels, data movement &amp; kernels interleaved</td>
</tr>
</tbody>
</table>
Performance of Single Thread Contending for GPGPU

Binomial Options – Time taken to compute constant workload size.
Performance of Single Thread Contending for GPGPU

Binomial Options – Time taken to compute constant workload size.

Georgia Institute of Technology - For VTDC'11
Increased Aggregate Application Throughput

BinomialOptions (typo in paper says BlackScholes!)
4-vGPU assembly
CPU Contention: Can Limit Performance in Assembly

Multiple VMs, each one instance of Black-Scholes.

Georgia Institute of Technology - For VTDC'11
Sensitivity to Overheads: Different Across Workloads

<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>MRI-FHD</th>
<th>MM</th>
<th>BOp</th>
<th>BS</th>
<th>CP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Time Overhead (VM/Dom0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Local VM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remote VM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Georgia Institute of Technology - For VTDC'11
Insights

• **Scalability** achievable!

• More GPGPUs $\neq$ greater performance  
  – CPUs can become a bottleneck

• Optimizations still needed, but can be pushed to system software layer

• Network overhead remains greatest contributor  
  – Significant for highly synchronous and data-intensive workloads
Outline

• Introduction
• GPGPU Assemblies
• Evaluation

• Conclusion
  – Future & Current Efforts
  – Related Work
Future Work

• Understanding utilization
  – Better characterize workload profiles

• More flexibility + instrumentation
  – Integrate with GPU Ocelot
  – Deeper view into GPGPU utilization
  – Additionally allow CPUs to execute GPGPU code

• Dynamic matching algorithm will need to be intelligent and distributed
  – Incorporate monitoring system at each layer
  – Greater opportunities to share hardware
Related Work

• vCUDA
  – XMLRPC for API marshalling
  – No remote GPGPU access

• GViM – also no remote GPGPU access

• rCUDA
  – Concurrent use of non-local GPGPUs
  – Reduce power consumption
Conclusion

• **Heterogeneous clusters** becoming more diverse
  – GPGPUs dominant accelerator
• **Nodes not reconfigurable**, applications tailored towards specific configuration
• Assemblies provide **flexible framework** for mapping applications to hardware
• Assemblies enable **scalability**, but require optimizations for some workloads
• Provide additional **flexibility** by enabling CUDA to also execute on CPUs.
  – Even greater flexibility for assembly → hardware mapping
Acknowledgements

Student funding provided in part by the National Science Foundation.
Equipment funding provided in part by Intel and NVIDIA.
Thank you to the reviewers for their thorough feedback during submission!
[End of Talk]