M02: High Performance Computing with CUDA

Overview
Speakers

Ian Buck
NVIDIA

Massimiliano Fatica
NVIDIA

Patrick Legresley
NVIDIA

Paulius Micikevicius
NVIDIA

Scott Morton
Hess Corporation

Jim Phillips
University of Illinois Urbana-Champaign

John Stone
University of Illinois Urbana-Champaign
Schedule

8:30  Introduction
  *Motivation, GPU computing & CUDA, tutorial overview*
  
9:00  Programming CUDA
  *Execution & memory model, C extensions, examples*
  
9:45  CUDA Toolkit
  *Compiler, Libraries, calling CUDA from other languages*
  
10:30 Break

11:00 Optimizing CUDA
  *Optimization strategies, algorithmic strategies*
  
12:00 Lunch
<table>
<thead>
<tr>
<th>Time</th>
<th>Case Studies</th>
<th>Speaker</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:30</td>
<td>Seismic imaging</td>
<td>Morton</td>
</tr>
<tr>
<td></td>
<td><em>Algorithms and Porting &amp; Production experiences</em></td>
<td></td>
</tr>
<tr>
<td>2:15</td>
<td>Molecular visualization &amp; analysis</td>
<td>Stone</td>
</tr>
<tr>
<td></td>
<td><em>Direct Coulomb summation, integrating CUDA &amp; VMD</em></td>
<td></td>
</tr>
<tr>
<td>3:00</td>
<td>Break</td>
<td>Phillips</td>
</tr>
<tr>
<td>3:30</td>
<td>Molecular dynamics</td>
<td>Phillips</td>
</tr>
<tr>
<td></td>
<td><em>NAMD, optimization walkthroughs, performance</em></td>
<td></td>
</tr>
<tr>
<td>4:15</td>
<td>Computational Fluid Dynamics</td>
<td>LeGresley</td>
</tr>
<tr>
<td></td>
<td><em>Mapping PDE algorithms to GPU, handling boundary conditions</em></td>
<td></td>
</tr>
<tr>
<td>5:00</td>
<td>Wrap!</td>
<td></td>
</tr>
</tbody>
</table>

M02: High Performance Computing with CUDA
M02: High Performance Computing with CUDA

Introduction
GPU Performance History

- GPUs are massively multithreaded many-core chips
  - Hundreds of cores, thousands of concurrent threads
  - Huge economies of scale
  - Still on aggressive performance growth
Parallelism is Scaling Rapidly

• CPUs and GPUs are parallel processors
  - CPUs now have 2, 4, 6, ... processors
  - GPUs now have 32, 64, 128, 240, ... processors

• Parallelism is increasing rapidly with Moore’s Law
  - Processor count is doubling every 18 - 24 months
  - Individual processor cores no longer getting faster

• Challenge: Develop parallel application software
  - Scale software parallelism to use more and more processors in a transparent manner
  - Same source for parallel GPUs and CPUs
CUDA is C for Parallel Processors

• CUDA is industry-standard C
  - Write a program for one thread
  - Instantiate it on many parallel threads
  - Familiar programming model and language

• CUDA is a scalable parallel programming model
  - Program runs on any number of processors without recompiling

• CUDA parallelism applies to both CPUs and GPUs
  - Compile the same program source to run on different platforms with widely different parallelism
  - Map CUDA threads to GPU threads or to CPU vectors
CUDA Uses Extensive Multithreading

- **CUDA threads** express fine-grained data parallelism
  - Map threads to GPU threads or CPU vector elements
  - Virtualize the processors
  - You must rethink your algorithms to be aggressively parallel

- **CUDA thread blocks** express coarse-grained parallelism
  - Map blocks to GPU thread arrays or CPU threads
  - Scale transparently to any number of processors

- **GPUs execute thousands of lightweight threads**
  - One DX10 graphics thread computes one pixel fragment
  - One CUDA thread computes one result (or several results)
  - Provide hardware multithreading & zero-overhead scheduling
GPU Sizes Require CUDA Scalability

- **32 SP Cores**
  - SM
  - I-Cache
  - MT Issue
  - C-Cache
  - SP
  - SP
  - SP
  - SP
  - SFU
  - SFU
  - Shared Memory

- **128 SP Cores**
  - SM
  - I-Cache
  - MT Issue
  - C-Cache
  - SP
  - SP
  - SP
  - SP
  - SP
  - SP
  - SFU
  - SFU
  - Shared Memory

- **240 SP Cores**
  - SM
  - I-Cache
  - MT Issue
  - C-Cache
  - SP
  - SP
  - SP
  - SP
  - SP
  - SFU
  - SFU
  - Shared Memory

M02: High Performance Computing with CUDA
CUDA runs on NVIDIA GPUs...
Over 86 Million CUDA GPUs Deployed

GeForce®
Entertainment

Quadro®
Design & Creation

Tesla™
High-Performance Computing

and now on multi-core CPUs !!!!!!
Pervasive CUDA Parallel Computing

- CUDA brings data-parallel computing to the masses
  - Over 86 M CUDA-capable GPUs deployed since Nov 2006

- Wide developer acceptance
  - Download CUDA from www.nvidia.com/cuda
  - Over 50K CUDA developer downloads
  - A GPU “developer kit” costs ~$100 for several hundreds GFLOPS

- Data-parallel supercomputers are everywhere!
  - CUDA makes this power readily accessible
  - Enables rapid innovations in data-parallel computing

- Parallel computing rides the commodity technology wave
University’s Teaching Parallel Programming With CUDA

- Duke
- Erlangen
- ETH Zurich
- Georgia Tech
- Grove City College
- Harvard
- IIT
- IIIT
- Illinois Urbana-Champaign
- INRIA
- Iowa
- ITESM
- Johns Hopkins
- Kent State
- Kyoto
- Lund
- Maryland
- McGill
- MIT
- North Carolina - Chapel Hill
- North Carolina State
- Northeastern
- Oregon State
- Pennsylvania
- Polimi
- Purdue
- Santa Clara
- Stanford
- Stuttgart
- Suny
- Tokyo
- TU-Vienna
- USC
- Utah
- Virginia
- Washington
- Waterloo
- Western Australia
- Williams College
- Wisconsin
CUDA Computing Sweet Spots

• Parallel Applications:
  - High arithmetic intensity:
    Dense linear algebra, PDEs, \( n \)-body, finite difference, ...
  - High bandwidth:
    Sequencing (virus scanning, genomics), sorting, database, ...
  - Visual computing:
    Graphics, image processing, tomography, machine vision, ...
  - Computational modeling, science, engineering, finance, ...
CUDA Zone: www.nvidia.com/cuda

- Resources, examples, and pointers for CUDA developers
Introducing Tesla T10P Processor

1.4 billion transistors
1 Teraflop of processing power
240 SP processing cores
30 DP processing cores with IEEE-754 double precision

…NVIDIA’s 2nd Generation CUDA Processor
CUDA Computing with Tesla T10

- 240 SP processors at 1.45 GHz: 1 TFLOPS peak
- 30 DP processors at 1.45Ghz: 86 GFLOPS peak
- 128 threads per processor: 30,720 threads total
# Double Precision Floating Point

<table>
<thead>
<tr>
<th></th>
<th>NVIDIA GPU</th>
<th>SSE2</th>
<th>Cell SPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>IEEE 754</td>
<td>IEEE 754</td>
<td>IEEE 754</td>
</tr>
<tr>
<td>Rounding modes for FADD and FMUL</td>
<td>All 4 IEEE, round to nearest, zero, inf, -inf</td>
<td>All 4 IEEE, round to nearest, zero, inf, -inf</td>
<td>Round to zero/truncate only</td>
</tr>
<tr>
<td>Denormal handling</td>
<td>Full speed</td>
<td>Supported, costs 1000’s of cycles</td>
<td>Flush to zero</td>
</tr>
<tr>
<td>NaN support</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Overflow and Infinity support</td>
<td>Yes</td>
<td>Yes</td>
<td>No infinity, clamps to max norm</td>
</tr>
<tr>
<td>Flags</td>
<td>No</td>
<td>Yes</td>
<td>Some</td>
</tr>
<tr>
<td>FMA</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Square root</td>
<td>Software with low-latency FMA-based convergence</td>
<td>Hardware</td>
<td>Software only</td>
</tr>
<tr>
<td>Division</td>
<td>Software with low-latency FMA-based convergence</td>
<td>Hardware</td>
<td>Software only</td>
</tr>
<tr>
<td>Reciprocal estimate accuracy</td>
<td>24 bit</td>
<td>12 bit</td>
<td>12 bit</td>
</tr>
<tr>
<td>Reciprocal sqrt estimate accuracy</td>
<td>23 bit</td>
<td>12 bit</td>
<td>12 bit</td>
</tr>
<tr>
<td>log₂(x) and 2^x estimates accuracy</td>
<td>23 bit</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
### Tesla C1060 Computing Processor

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Processor</strong></td>
<td>1x Tesla T10P</td>
</tr>
<tr>
<td><strong>Core GHz</strong></td>
<td>1.33 GHz</td>
</tr>
<tr>
<td><strong>Form factor</strong></td>
<td>Full ATX: 4.736” (H) x 10.5” (L) Dual slot wide</td>
</tr>
<tr>
<td><strong>On-board memory</strong></td>
<td>4 GB</td>
</tr>
<tr>
<td><strong>System I/O</strong></td>
<td>PCIe x16 gen2</td>
</tr>
<tr>
<td><strong>Memory I/O</strong></td>
<td>512-bit, 800MHz DDR 102 GB/s peak bandwidth</td>
</tr>
<tr>
<td><strong>Display outputs</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Typical power</strong></td>
<td>160 W</td>
</tr>
</tbody>
</table>
# Tesla S1070 1U System

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Processors</strong></td>
<td>4 x Tesla T10P</td>
</tr>
<tr>
<td><strong>Core GHz</strong></td>
<td>1.5 GHz</td>
</tr>
<tr>
<td><strong>Form factor</strong></td>
<td>1U for an EIA 19” 4-post rack</td>
</tr>
<tr>
<td><strong>Total 1U system memory</strong></td>
<td>16 GB (4.0GB per GPU)</td>
</tr>
<tr>
<td><strong>System I/O</strong></td>
<td>2 PCIe x16</td>
</tr>
<tr>
<td><strong>Memory I/O per processor</strong></td>
<td>512-bit, 800MHz GDDR 102 GB/s peak bandwidth</td>
</tr>
<tr>
<td><strong>Display outputs</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Typical power</strong></td>
<td>700 W</td>
</tr>
<tr>
<td><strong>Chassis dimensions</strong></td>
<td>1.73” H × 17.5” W × 28.5” D</td>
</tr>
</tbody>
</table>
M02: High Performance Computing with CUDA

Applications
Folding@home Performance Comparison

<table>
<thead>
<tr>
<th>OS Type</th>
<th>Current TFLOPS*</th>
<th>Active CPUs</th>
<th>Total CPUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>208</td>
<td>218169</td>
<td>2172051</td>
</tr>
<tr>
<td>Mac OS X/PowerPC</td>
<td>7</td>
<td>8563</td>
<td>120066</td>
</tr>
<tr>
<td>Mac OS X/Intel</td>
<td>23</td>
<td>7332</td>
<td>62880</td>
</tr>
<tr>
<td>Linux</td>
<td>60</td>
<td>35036</td>
<td>330960</td>
</tr>
<tr>
<td>ATI GPU</td>
<td>447</td>
<td>4065</td>
<td>9396</td>
</tr>
<tr>
<td>NVIDIA GPU</td>
<td><strong>1450</strong></td>
<td>13185</td>
<td>27428</td>
</tr>
<tr>
<td>PLAYSTATION®3</td>
<td>1078</td>
<td>38229</td>
<td>599812</td>
</tr>
<tr>
<td>Total</td>
<td>3273</td>
<td>324579</td>
<td>3322593</td>
</tr>
</tbody>
</table>

nano seconds of simulation per day

F@H kernel based on GROMACS code
Lattice Boltzmann

1000 iterations on a 256x128x128 domain

Cluster with 8 GPUs: 7.5 sec
Blue Gene/L 512 nodes: 21 sec

10000 iterations on irregular 1057x692x1446 domain with 4M fluid nodes

<table>
<thead>
<tr>
<th></th>
<th>Speed</th>
<th>MLUPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 C870</td>
<td>760 s</td>
<td>53 MLUPS</td>
</tr>
<tr>
<td>2 C1060</td>
<td>159 s</td>
<td>252 MLUPS</td>
</tr>
<tr>
<td>8 C1060</td>
<td>42 s</td>
<td>955 MLUPS</td>
</tr>
</tbody>
</table>

Blood flow pattern in a human coronary artery, Bernaschi et al.
Desktop GPU Supercomputer Beats Cluster

CalcUA

256 Nodes (512 cores)

8 GPUs in a Desktop

- 256 Sun Fire V20z rekennodes (dual AMD Opteron 250, 2.4 GHz):
  - 192 rekennodes beschikken over 4 GB intern geheugen,
  - 64 rekennodes beschikken over 8 GB intern geheugen en zijn onderling verbonden via Myrinet
- 2 Sun Fire V440 servers
- 2 Sun StorEdge 3510 FC Array 6 TB centrale schijfcapaciteit

Life Sciences: Autodock for Cancer Research

National Cancer Institute reports 12x speedup

“Wait for results” reduced from 2 hours to 10 minutes
Weather Research and Forecast (WRF) model

4000+ registered users worldwide

20% speedup with 1% of WRF on CUDA

Saves 1 week analysis time
3D Ultrasound CT

• Safer, more comfortable procedure
• Exam and results in single office visit
• 4 GPU system fits in the office
• 20x faster than CPUs
Accelerating MATLAB®

Pseudo-spectral simulation of 2D Isotropic turbulence

Use MEX files to call CUDA from MATLAB, 17x speed-up

1024x1024 mesh, 400 RK4 steps, Windows XP, Core2 Duo 2.4Ghz vs GeForce 8800GTX

Applications in several fields

146X Interactive visualization of volumetric white matter connectivity
36X Ionic placement for molecular dynamics simulation on GPU
19X Transcoding HD video stream to H.264
17X Simulation in Matlab using .mex file CUDA function
100X Astrophysics N-body simulation

149X Financial simulation of LIBOR model with swaptions
47X GLAME@lab: An M-script API for linear Algebra operations on GPU
20X Ultrasound medical imaging for cancer diagnostics
24X Highly optimized object oriented molecular dynamics
30X Cmatch exact string matching to find similar proteins and gene sequences