S4599: An Adventure in Porting: Adding GPU-Acceleration to Open-Source 3D Elastic Wave Modeling

Robin M. Weiss and Jeffery Shragge

March 26, 2014
GPU Technology Conference
Seismic Data Modeling

• Elastic modeling
  – Anisotropic and heterogeneous materials
• Seismic imaging with elastic models
  – Increasing interest in this area
  – Solved as an inverse problem
  – Requires accurate, fast forward solvers
• Computational issues
  – Large model sizes
  – Generally requires a ton of compute power
Open-Source 3D Elastic Modeling

• Open-source computational geophysics package
  – Madagascar – www.reproducibility.org (RSFSRC/user/rweiss)
  – Targets reproducibility, transparency, and clarity
• Madagascar’s CPU-based (w/ OpenMP) EWE solver implementation:
  – Paul Sava, Colorado School of Mines
  – Anisotropic elastic wave equation
  – Stress-Stiffness formulation
  – 3D Finite-difference solver, centered derivatives, regular grid
    • 8th order spatial, 2nd order time

Weiss and Shragge, 2013, Solving 3D anisotropic elastic wave equations on parallel GPU devices, Geophysics
Finite-Difference and GPUs

- Represents the bulk of the compute work
- Compact FD stencils map nicely to GPU
  - Regular memory access pattern
  - Some data reuse (use shared memory)
  - But still memory bound
    - GTC 2013 - S3176

Micikevicius, 2009, *3D Finite Difference Computation on GPUs using CUDA, NVIDIA*
Optimize: The usual suspects...

- Stash data in shared memory and/or registers
- Find good block dimensions
- Repeated expression replacement
- Kernel fusing
  - Keep an eye on register usage (`-xptxas -v`)
- CUDA Visual Profiler is your friend
GPU-CPU Displacement Comparison

depth-component of displacement field overlaid on velocity model
Off to the races…

• All experiments run on N³ data sets for 2000 timesteps
• CPU results computed with Intel Sandy Bridge E5-2670 2.60GHz
  – 2x 8-core, IBM iDataPlex dx360 M4
  – University of Chicago’s Midway Compute Cluster
• GPU results computed with Nvidia K40
  – 4x GPUs per node
  – Nvidia PSG cluster – thanks, Nvidia!
Off to the races...

<table>
<thead>
<tr>
<th>Cube Size</th>
<th>GPU Time (seconds)</th>
<th>16-CPU Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>1</td>
<td>1.5</td>
</tr>
<tr>
<td>128</td>
<td>2</td>
<td>3.5</td>
</tr>
<tr>
<td>256</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>512</td>
<td>8</td>
<td>15</td>
</tr>
</tbody>
</table>
Off to the races...

Throughput (Mpoints/sec)

<table>
<thead>
<tr>
<th>Cube Size</th>
<th>Disp. To Strain</th>
<th>Strain to Stress</th>
<th>Stress to Accel.</th>
<th>Advance Disp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1405</td>
<td>1657</td>
<td>1064</td>
<td>2976</td>
</tr>
<tr>
<td>148</td>
<td>1333</td>
<td>1702</td>
<td>1053</td>
<td>3077</td>
</tr>
<tr>
<td>196</td>
<td>1467</td>
<td>1714</td>
<td>1164</td>
<td>3051</td>
</tr>
<tr>
<td>244</td>
<td>1449</td>
<td>1662</td>
<td>1054</td>
<td>3051</td>
</tr>
<tr>
<td>292</td>
<td>1404</td>
<td>1703</td>
<td>993</td>
<td>2950</td>
</tr>
<tr>
<td>340</td>
<td>1282</td>
<td>1721</td>
<td>1021</td>
<td>3170</td>
</tr>
</tbody>
</table>
Overall Throughput

![Graph showing overall throughput vs cube size.](#)
Speedup

GPU vs. 16-CPU

Speedup Factor

Cube Size

100 148 196 244 292 340 388
Speedup

Cube Size

GPU vs. 16-CPU
GPU vs. 1-CPU
While this is all well and good...
…CPU implementation has us beat
GPU Memory Wall

• Limited GPU memory becomes an issue
• In our 3D implementation, each grid point requires:
  – 1 density value
  – 9 stiffness coefficients
  – 3 displacement components (x3 time steps)
  – 6 strain/stress field components
  – 3 acceleration components
• GlobalMemory ≥ 28 * gridPoints * sizeof(float)
Multiple Devices

- Increase GPU memory (in aggregate) by distributing grid across multiple devices
- Not quite embarrassingly parallel
  - FD stencil needs halo cells from neighbor device(s)
  - Halo dance happens each iteration
  - Communication can become costly
- Slice on the slowest varying axis
  - Easy to compute stride
Multiple Devices

GPU 0

GPU 1

GPU 2

GPU 3
Multiple Devices: Take 1

- One host-process per device
- Communicate through MPI
Multiple Devices: Take 1

- Generalizes nicely, scales to multiple nodes
- Handle arbitrary* number of devices & nodes
Multiple Devices: Take 1

- But: the data dance is awkward (read: slow)
Multiple Devices: Take 1

**Cube Size**

- **MPI-1**
- **MPI-2**
- **MPI-4**
- **MPI-8**

**Time (seconds)**

- 16
- 32
- 64
- 128
- 256
- 512
- 1024
- 2048
Multiple Devices: Take 1

• Can we fix this?
  – CUDA-aware MPI...?
  – GPU Direct...?
  – RDMA...?
  – Black magic...?

• Open-source research code considerations
  – Keep it simple!
Multiple Devices: Take 2

- One host-process *per node*
- Communicate through peer-to-peer *memcpy*
Multiple Devices: Take 2

- P2P-1
- P2P-2
- P2P-4

Time (seconds) vs Cube Size

- 16
- 32
- 64
- 128
- 256
- 512
- 1024
Overall Throughput

![Overall Throughput Graph](image)
Multiple Devices: Take 2

• But we’re still limited to one node...
Multiple Devices: Take 2.1

- Scale to multiple nodes with MPI
- Handle arbitrary* number of devices & nodes

\* arbitrary number

![Diagram showing multiple GPUs and nodes connected with MPI and P2P communications.](image-url)
Multiple Devices: Take 2.1

Cube Size

- P2P-1
- P2P-2
- P2P-4
- P2P-4(2)
Multiple Devices: Take 2.1

<table>
<thead>
<tr>
<th>Cube Size</th>
<th>Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>16</td>
</tr>
<tr>
<td>128</td>
<td>32</td>
</tr>
<tr>
<td>256</td>
<td>64</td>
</tr>
<tr>
<td>512</td>
<td>128</td>
</tr>
<tr>
<td>1024</td>
<td>256</td>
</tr>
</tbody>
</table>

Graph comparing P2P-4(2) and MPI-8 performance.
Wrapping up

• Elastic wave modeling represents a complex computation problem
  – But, GPU is a promising route for acceleration
  – Flexible multi-GPU implementation offers considerable speedup on large grids
• Represents a seed for further research, modifications, and extensions

• Check out our code: [www.reproducibility.org](http://www.reproducibility.org)
  – Exec: sfewefd2d_gpu, sfewefd3d_gpu_p2p, sfewefd3d_multiNode
  – Source: user/rweiss
  – Improvements and extensions welcomed...
    ...bug reports welcome as well
S4599: An Adventure in Porting: Adding GPU-Acceleration to Open-Source 3D Elastic Wave Modeling

Robin M. Weiss and Jeffery Shragge

March 26, 2014
GPU Technology Conference