Using GPUs to Run Weather Prediction Models

Mark Govett
Tom Henderson, Jacques Middlecoff,
Paul Madden, Jim Rosinski

November 2010
Organizational Structure

- NOAA
  - National Weather Service
  - National Centers for Environmental Prediction (NCEP)
  - Oceanic & Atmospheric Research
    - Earth System Research Laboratory
      - Global Systems Division
        » Advanced Computing Section
Model Development Activities

• Regional, Local Models (1-5 KM)
  – NOAA HRRR, WRF-ARW, WRF-NMM, etc
    • Hurricanes, Aviation, Fires, Chemistry / Ash
  – Ensembles (15-30KM)

• Global Models (10-30 KM)
  – NOAA FIM model
  – Improved hurricane forecasts

Computing Requirements

– 3000 cores:
  • 15KM Global FIM
– 126,000 cores
  • 21 member 30 KM ensemble
Cloud Resolving Models

– Benefits
  • Clouds have a major influence on weather and climate
  • Improvements in 5-20 day forecasts, climate
  • Improved Hurricane track and intensity

– Active developments within the research community
  • NICAM: University of Tokyo
  • GCRM: Colorado State University
  • NIM: NOAA Earth System Research Laboratory

– Non-hydrostatic Icosahedral Model (NIM)
  • Cloud Resolving Scale (target 2KM resolution)
  • Uniform, hexagonal-based, icosahedral grid
  • Novel indirect addressing scheme used that permits concise, efficient code
    – Used in the hydrostatic FIM model (Operational at NCEP in 2011)
CPU Computing Requirements

• Research and Development
  – CSU’s 4KM GCRM was run on 80,000 cores of DOE Jaguar
  – Simulations ran at ~50 percent of real-time

• Operations
  – Models must run at 1-2 percent of real-time
  – NIM performance & scaling study indicates about 200,000 cores would be needed to get to ~3 % real-time

  • System reliability, power requirements, uncertainties in model scaling are big concerns
GPU / Multi-core Technology

- **NVIDIA**: Fermi chip first to support HPC
  - Formed partnerships with Cray, IBM on HPC systems
  - #1, #3 systems on TOP500 (Fermi, China)
- **AMD/ATI**: Primarily graphics currently
  - #7 system on TOP500 (AMD-Radeon, China)
  - Fusion chip in 2011 (5 TeraFlops)
- **Intel**: Knights Ferry (2011), 32-64 cores

NVIDIA: Fermi (2010)

- 1.2 TeraFlops
- 8x increase in double precision
- 2x increase in memory bandwidth
- Error correcting memory

NVIDIA: Tesla (2008)

- CPU: 2008
  - 933 Gflops
- CPU: 2008
  - ~45 Gflops

November 2010
CPU – GPU Comparison

• CPUs focus on per-core performance
  – Chip real estate devoted to cache, speculative logic
  – Westmere: 6 cores, 140 Gflops, 130 Watts (~1 GFlop /Watt)

• GPUs focus on parallel execution
  – Fermi: 512 cores, 1100 Gflops, 220 Watts (~5 Gflops / Watt)
Next Generation Weather Models

- Models being designed for global cloud resolving scales (3-4km)
- Requires PetaFlop Computers

**DOE Jaguar System**
- 2.3 PetaFlops
- 250,000 CPUs
- 284 cabinets
- 7-10 MW power
- ~ $100 million
- Reliability in hours

**Equivalent GPU System**
- 2.3 PetaFlop
- 2000 Fermi GPUs
- 20 cabinets
- 1.0 MW power
- ~ $10 million
- Reliability in weeks

- Large CPU systems (>100 thousand cores) are unrealistic for operational weather forecasting
  - Power, cooling, reliability, cost
  - Application scaling

Valmont Power Plant
~200 MegaWatts
Boulder, CO
Application Performance

- 20-50x is possible on highly scalable codes
- Efficient use of memory is critical to good performance
  - 1-2 cycles to access shared memory
  - Hundreds of cycles to access global memory

<table>
<thead>
<tr>
<th>Memory</th>
<th>Tesla</th>
<th>Fermi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared</td>
<td>16K</td>
<td>64K</td>
</tr>
<tr>
<td>Constant</td>
<td>16K</td>
<td>64K</td>
</tr>
<tr>
<td>Global</td>
<td>1-2GB</td>
<td>4-6GB</td>
</tr>
</tbody>
</table>
Execution Flow-control  
(Accelerator Approach)

– Copy between CPU and GPU is non-trivial
  • Performance benefits can be overshadowed by the copy
  • WRF demonstrated ~6x for one subroutine including data transfers (Michalakes, 2009)
    – ~10x without data transfers
Execution Flow-control
(run everything on GPUs)

- Eliminates copy every model time step
- CPU-GPU copies only needed for input /output, inter-process communications
- JMA: ASUCA model, reported a 80x performance improvement
  - Rewrote the code in CUDA
  - SC2010 Paper: Tuesday 2:30 – 3:00 PM

November 2010
Code Parallelization (2009)

- Developed the Fortran-to-CUDA compiler (F2C-ACC)
  - Commercial compilers were not available in 2008
  - Converts Fortran 90 into C or CUDA-C
  - Some hand tuning was necessary
- Parallelized NIM model dynamics
  - Result for a single GPU
  - Communications only needed for I/O

<table>
<thead>
<tr>
<th>Resolution</th>
<th>HorizPts</th>
<th>Harpertown</th>
<th>Tesla</th>
<th>Nehalem</th>
<th>Fermi</th>
</tr>
</thead>
<tbody>
<tr>
<td>G4-480km</td>
<td>2562</td>
<td>2.13</td>
<td>0.079 (26.9)</td>
<td>1.45</td>
<td>0.054 (26.7)</td>
</tr>
<tr>
<td>G5-240km</td>
<td>10242</td>
<td>8.81</td>
<td>0.262 (33.5)</td>
<td>5.38</td>
<td>0.205 (26.2)</td>
</tr>
</tbody>
</table>
Model Parallelization (2010)

• Updated NIM Model Parallelization
  – Active model development
  – Code optimizations on-going
• Evaluate Fortran GPU compilers
  – Use F2C results as benchmark
• Evaluate Fermi
• Run on Multiple GPUs
  – Modified F2C-ACC GPU compiler
  – Uses MPI-based Scalable Modeling System (SMS)
  – Testing on 10 Tesla & 10 Fermi GPUs
Fortran GPU Compilers

• General Features
  – Do not support all Fortran language constructs
  – Converts Fortran into CUDA for further compilation

• CAPS – HMPP
  – Extensive set of parallelization directives to guide compiler analysis and optimization
  – Optionally generates OpenCL

• PGI
  – **ACCELERATOR** – directive-based accelerator
  – **CUDA Fortran** – Fortran + language extensions to support Kernel calls, GPU memory, etc

• F2C-ACC
  – Developed at NOAA for our models
  – Requires hand tuning for optimal performance
# Run Times for Single GPUs vs. Single Nehalem CPU Core

2652 points / GPU (NIM – G4)

<table>
<thead>
<tr>
<th></th>
<th>Harpertown CPU Time</th>
<th>F2C-ACC CUDA-C Tesla GPU Time</th>
<th>HMPP Tesla GPU Time</th>
<th>PGI Tesla GPU Time</th>
<th>F2C-ACC CUDA-C Fermi GPU Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>vdmints</td>
<td>88.86</td>
<td>2.05</td>
<td>2.35</td>
<td>4.78</td>
<td>1.92</td>
</tr>
<tr>
<td>vdmintv</td>
<td>37.73</td>
<td>0.94</td>
<td>0.98</td>
<td>0.97</td>
<td>0.75</td>
</tr>
<tr>
<td>flux</td>
<td>17.97</td>
<td>0.55</td>
<td>1.05</td>
<td>2.51</td>
<td>0.30</td>
</tr>
<tr>
<td>vdn</td>
<td>12.77</td>
<td>0.56</td>
<td>0.73</td>
<td>--</td>
<td>0.53</td>
</tr>
<tr>
<td>diag</td>
<td>5.13</td>
<td>0.086</td>
<td>0.085</td>
<td>0.077</td>
<td>0.09</td>
</tr>
<tr>
<td>force</td>
<td>5.34</td>
<td>0.11</td>
<td>0.19</td>
<td></td>
<td>0.08</td>
</tr>
<tr>
<td>trisol</td>
<td>8.41</td>
<td>1.38</td>
<td>1.38</td>
<td>--</td>
<td>1.14</td>
</tr>
<tr>
<td>Total</td>
<td>190.26</td>
<td>6.54 (29.0)</td>
<td>8.12 (23.4)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Run Times for Single GPUs vs. Single Nehalem
10242 points / GPU (NIM-G5)

<table>
<thead>
<tr>
<th></th>
<th>Nehalem CPU Time 10424 pts</th>
<th>F2C-ACC Fermi 10242 pts</th>
</tr>
</thead>
<tbody>
<tr>
<td>vdmints</td>
<td>221.37</td>
<td>7.73 (28.6)</td>
</tr>
<tr>
<td>vdmintv</td>
<td>102.58</td>
<td>2.86 (35.9)</td>
</tr>
<tr>
<td>flux</td>
<td>56.84</td>
<td>1.17 (48.5)</td>
</tr>
<tr>
<td>vdn</td>
<td>17.67</td>
<td>2.02 (  8.8)</td>
</tr>
<tr>
<td>diag</td>
<td>18.02</td>
<td>0.36 (50.2)</td>
</tr>
<tr>
<td>force</td>
<td>15.00</td>
<td>0.37 (40.0)</td>
</tr>
<tr>
<td>trisol</td>
<td>9.25</td>
<td>5.4 (  1.7)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>467.38</strong></td>
<td><strong>21.5 (21.7)</strong></td>
</tr>
</tbody>
</table>
Parallel Performance Projections

• A doubling of model resolution implies:
  – A 4x increase in horizontal points
  – 2x increase in model time step
  – 4x increase in memory

• GPU global memory limits scaling

<table>
<thead>
<tr>
<th></th>
<th>G4</th>
<th>G5</th>
<th>G6</th>
<th>G7</th>
<th>G8</th>
<th>G9</th>
<th>G10</th>
<th>G11</th>
</tr>
</thead>
<tbody>
<tr>
<td>resolution</td>
<td>480KM</td>
<td>240KM</td>
<td>120KM</td>
<td>60KM</td>
<td>30 KM</td>
<td>15 KM</td>
<td>7 KM</td>
<td>3.5 KM</td>
</tr>
<tr>
<td>horizontal points</td>
<td>2.5K</td>
<td>10K</td>
<td>40K</td>
<td>160K</td>
<td>640K</td>
<td>2560K</td>
<td>10,000K</td>
<td>40,000K</td>
</tr>
<tr>
<td>memory</td>
<td>.25GB</td>
<td>1GB</td>
<td>4GB</td>
<td>16GB</td>
<td>64GB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tesla</td>
<td>26x</td>
<td>33x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fermi</td>
<td>26x</td>
<td>26x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># GPUs</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>16</td>
<td>64</td>
<td>256</td>
<td>1024</td>
</tr>
</tbody>
</table>
Parallel Performance Considerations

- Application scaling will be limited by the fraction of time spent doing inter-process communications
- Using GPUs, if we get a $\sim20x$ speedup in computation time, communications now becomes 50 percent of the runtime.

<table>
<thead>
<tr>
<th></th>
<th>Input / output time</th>
<th>GPU time</th>
<th>Inter-process communications</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Time</td>
<td>100 sec 5 100 sec 5 100 sec 5 100 sec 5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GPU Time</td>
<td>5 5 5 5 5 5 5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Minimize data transfers and frequency
  - Trade communications for extra computations
    - GPU computes are “cheap”
- Overlap communications with computations
  - CPU is idle and available
  - Move inter-process communications from just before data is needed to just after the data is updated.
GPUs and the Challenges Ahead

– Performance and Portability
  • Models are becoming increasingly complex
  • Challenge to maintain a single source
    • Operations, research, collaboration
    – Especially for models under active development

– New codes are easier to parallelize
  • Models can be designed to run on GPUs, Multi-core
  • Collaboration between model developers, computer scientists

– Legacy codes will be harder to convert
  • Similar to transition from Vector to MPPs
GPU Performance Portability

- Reliance on NVIDIA, AMD compilers
  - Register allocation inefficient
  - Loop fusion, in-lining, data re-use optimizations are rudimentary
  - Commercial compilers overcome some performance issues
- Requires code changes to achieve good results
  - 2-3x performance benefit was observed (10x becomes 20x in NIM)
  - Different optimizations necessary on CPU than GPU
    - Cache, memory hierarchy
  - GPU architectures will affect performance