Experience Applying Fortran GPU Compilers to Numerical Weather Prediction

Tom Henderson  
NOAA Global Systems Division  
Thomas.B.Henderson@noaa.gov

Mark Govett, Jacques Middlecoff  
Paul Madden, James Rosinski, Craig Tierney
Hurricane Irene 7-Day Forecast

- Forecast valid @ 8/21/2011 12Z UTC
- Black line is observed track
  - White diamonds are storm location at 00z UTC each day
- Green & cyan lines are forecast tracks from NWP models
Hurricane Irene
5-Day Forecast

Forecast valid @ 8/23/2011 12Z UTC
Hurricane Irene 3-Day Forecast

- Forecast valid at 8/25/2011 12Z UTC
- Forecast uncertainty (estimated from spread of forecast tracks) already smaller than the hurricane
Hurricane Irene
1-Day Forecast

Forecast valid@ 8/27/2011 12Z UTC
Correlation of Forecast Skill and Compute Power

NCEP Operational Forecast Skill
36 and 72 Hour Forecasts @ 500 MB over North America
[100 ° (1-S/70) Method]

- 36 Hour Forecast
- 72 Hour Forecast

MPP (1992-)
GPU (2008-)

1st Operational MPP (2000)
GPU Fine-Grained Parallelism

- “Blocks” of “threads”
  - Many threads per core (1000s)
  - Needs code that vectorizes well
- Large on-chip ("global") memory
  - ~3x higher bandwidth than CPUs
  - High latency (100s of cycles)
    - Need lots of threads to hide memory latency
- Limited per-thread fast ("shared") memory & registers
  - User-managed cache
  - Limits number of threads
- **Slow** data transfers between CPU & GPU
For NWP Use CPU as a “Communication Co-Processor”

- Invert traditional “GPU-as-accelerator” model
  - Model state lives on GPU
  - Initial data read by the CPU and passed to the GPU
  - Data passed back to the CPU only for output & message-passing
  - GPU performs all computations
    - Fine-grained parallelism
  - CPU controls high level program flow
    - Coarse-grained parallelism
- Minimizes overhead of data movement between CPU & GPU
- “Reverse offload” in MIC terms
NIM NWP Dynamical Core

- NIM = “Non-Hydrostatic Icosahedral Model”
  - Research NWP dynamical core
  - Target: global “cloud-permitting” resolutions ~3km (42 million columns), 2% of real-time
  - Rapidly evolving code base

- “GPU-friendly” (also good for CPU)
  - Single-precision floating-point computations
  - Computations structured as simple vector ops with indirect addressing and inner vertical loop

- Coarse-grained parallelism via Scalable Modeling System (SMS)
  - Directive-based approach to distributed-memory parallelism (implemented with MPI)
Icosahedral (Geodesic) Grid

- Near constant resolution over the globe
- Always 12 pentagons

(slide courtesy Dr. Jin Lee)
NIM Source Code Requirements

- Must maintain single Fortran source code for all desired execution modes
  - Single and multiple CPU
  - Single and multiple GPU
  - Prefer a directive-based Fortran approach for GPU that interoperates with SMS directives
GPU Fortran Compilers

- Commercial directive-based compilers
  - CAPS HMPP 3.0.5
  - Portland Group PGI Accelerator 11.10
  - Cray (beta)
- Open-source directive-based compiler developed at GSD
  - F2C-ACC (Govett, 2008)
    - “Application-specific” Fortran->CUDA-C compiler for performance evaluation
    - Provide feedback to commercial compiler developers
      - Bug identification
      - Performance improvement
Initial Performance Results

- "G5-L96" test case
  - 10242 columns, 96 levels, 1000 time steps
  - Expect similar number of columns on each GPU at ~3km target resolution

- CPU = Intel Westmere (2.66GHz)
- GPU = NVIDIA C2050 “Fermi”
- Optimize for both CPU and GPU
  - Some code divergence
  - Always use fastest code
Good Performance on CPU

- Used PAPI to count flops (Intel compiler)
  - Requires –O1 (no vectorization) to accurately count FLOPs
  - 2nd run with –O3 (vectorization) to get wallclock

\[
\frac{2.8 \times 10^{12}}{940} = 2.98 \text{Gflops/sec}
\]

~27% of peak on Westmere 2.8 GHz
Fermi GPU vs. Single/Multiple Westmere CPU cores, “G5-L96”

<table>
<thead>
<tr>
<th>NIM routine</th>
<th>CPU 1-core Time (sec)</th>
<th>CPU 6-core Time (sec)</th>
<th>F2C-ACC GPU Time (sec)</th>
<th>HMPP GPU Time (sec)</th>
<th>PGI GPU Time (sec)</th>
<th>F2C-ACC Speedup vs. 6-core CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total*</td>
<td>8654</td>
<td>2068</td>
<td>449</td>
<td>--**</td>
<td>--</td>
<td>4.6</td>
</tr>
<tr>
<td>vdmints</td>
<td>4559</td>
<td>1062</td>
<td>196</td>
<td>192</td>
<td>197</td>
<td>5.4</td>
</tr>
<tr>
<td>vdmintv</td>
<td>2119</td>
<td>446</td>
<td>91</td>
<td>101</td>
<td>88</td>
<td>4.9</td>
</tr>
<tr>
<td>flux</td>
<td>964</td>
<td>175</td>
<td>26</td>
<td>24</td>
<td>26</td>
<td>6.7</td>
</tr>
<tr>
<td>vdn</td>
<td>131</td>
<td>86</td>
<td>18</td>
<td>17</td>
<td>18</td>
<td>4.8</td>
</tr>
<tr>
<td>diag</td>
<td>389</td>
<td>74</td>
<td>42</td>
<td>33</td>
<td>--</td>
<td>1.8</td>
</tr>
<tr>
<td>force</td>
<td>80</td>
<td>33</td>
<td>7</td>
<td>11</td>
<td>13</td>
<td>4.7</td>
</tr>
</tbody>
</table>

* Total time includes I/O, PCIe, etc.
** Recent result: HMPP now complete and faster

2/22/12
Early Work With Multi-GPU Runs

- F2C-ACC + SMS directives
  - Identical results using different numbers of GPUs
  - Reduced scaling because compute has sped up but communication has not
  - Working on communication optimizations

- Demonstrates that single source code can be used for single/multiple CPU/GPU runs
Conclusions

- Some grounds for optimism
  - Fermi is ~4-5x faster than 6-core Westmere
  - Once compilers mature, expect level of effort similar to OpenMP for “GPU-friendly” codes like NIM
- Debugging and validation are more difficult on GPUs
- For NWP we need:
  - More memory bandwidth
  - More per-thread resources
  - Eliminate PCIe overhead
  - More mature compilers
Work In-Progress

- “FIM” (more mature NWP code scheduled for operational implementation)
- Intel MIC
- Cray GPU compiler
- Address multi-GPU scaling issues
- Continue to improve GPU performance
  - Tuning options via F2C-ACC and commercial compilers
  - Convince compiler vendors to support key optimizations
Thanks to…

- Guillaume Poirier, Yann Mevel, and others at CAPS for assistance with HMPP
- Dave Norton and others at PGI for assistance with PGI Accelerator
- Pete Johnsen at Cray for assistance with beta Cray GPU compiler
- We want to see multiple successful commercial directive-based Fortran compilers for GPU/MIC
Thank You