Unleashing the Performance Potential of GPUs for Atmospheric Dynamic Solvers

Haohuan Fu
haohuan@tsinghua.edu.cn
High Performance Geo-Computing (HPGC) Group
Center for Earth System Science
Tsinghua University

April/5th/2016
Tsinghua HPGC Group


- High performance computational solutions for geoscience applications
  - simulation-oriented research: providing highly efficient and highly scalable simulation applications (exploration geophysics, climate modeling)
  - data-oriented research: data processing, data compression, and data mining

- Combine optimizations from three different perspectives (Application, Algorithm, and Architecture), especially focused on new accelerator architectures
A Design Process That Combines Optimizations from Different Layers

The “Best” Computational Solution
• Exploration Geophysics
  • GPU-based BEAM Migration (sponsored by Statoil)
  • GPU-based ETE Forward Modeling (sponsored by BGP)
  • Parallel Finite Element Electromagnetic Forward Modeling Method (sponsored by NSFC)
• FPGA-based RTM (sponsored by NSFC and IBM)
• Climate Modeling
  • Global-scale atmospheric simulation (800 Tflops Shallow Water Equation Solver on Tianhe-1A, 1.4 Pflops atmospheric simulation 3D Euler Equation Solver on Tianhe-2)
  • FPGA-based atmospheric simulation (selected as one of the 27 Significant papers in the 25 years of the FPL conference)
• Remote Sensing Data Processing
  • Data analysis and visualization (sponsored by Microsoft)
  • Deep learning-based land cover mapping
• Application
  • Parallel Stencil on Different HPC Architectures
  • Parallel Sparse Matrix Solver
  • Parallel Data Compression (PLZMA) (sponsored by ZTE)
  • Hardware-Based Gaussian Mixture Model Clustering Engine: 517x speedup
• Architecture
  • Multi-core/many-core (CPU, GPU, MIC)
  • Reconfigurable hardware (FPGA)
A Highly Scalable Framework for Atmospheric Modeling on Heterogeneous Supercomputers
The Gap between Software and Hardware

China’s models
- pure CPU code
- scaling to hundreds or thousands of cores

China’s supercomputers
- heterogeneous systems with GPUs or MICs
- millions of cores

- millions lines of legacy code
- poor scalability
- written for multi-core, rather than many-core
Our Research Goals

- highly scalable framework that can efficiently utilize many-core accelerators
- automated tools to with the legacy code

China’s models
- pure CPU code
- scaling to hundreds or thousands of cores

China’s supercomputers
- heterogeneous systems with GPUs or MICs
- millions of cores
Our Research Goals

- highly scalable framework that can efficiently utilize many-core accelerators
- automated tools to with the legacy code

100T~1P

China’s models
- pure CPU code
- scaling to hundreds or thousands of cores

China’s supercomputers
- heterogeneous systems with GPUs or MICs
- millions of cores
Example: Highly-Scalable Atmospheric Simulation Framework

- Cube-sphere grid or other grid
- Explicit, implicit, or semi-implicit method
- Cloud resolving
- Algorithm
- Application
- Architecture
- CPU, GPU, MIC, FPGA
- C/C++, Fortran, MPI, CUDA, Java, ...

The "Best" Computational Solution
A Highly Scalable Framework for Atmospheric Modeling on Heterogeneous Supercomputers:

Previous Efforts
Highly-Scalable Framework for Atmospheric Modeling

- 2012: solving 2D SWE using CPU + GPU
  - 800 Tflops on 40,000 CPU cores, and 3750 GPUs

Highly-Scalable Framework for Atmospheric Modeling

- 2012: solving 2D SWE using CPU + GPU
  - 800 Tflops on 40,000 CPU cores, and 3750 GPUs
- 2013: 2D SWE on MIC and FPGA
  - 1.26 Pflops on 207,456 CPU cores, and 25,932 MICs
  - another 10x on FPGA

For more details, please refer to our IPDPS 2014 paper: "Enabling and Scaling a Global Shallow-Water Atmospheric Model on Tianhe-2"; and our FPL 2013 paper: “Accelerating Solvers for Global Atmospheric Equations Through Mixed-Precision Data Flow Engine”.
Highly-Scalable Framework for Atmospheric Modeling

- **2012**: solving 2D SWE using CPU + GPU
  - 800 Tflops on 40,000 CPU cores, and 3750 GPUs

- **2013**: 2D SWE on CPU+MIC and CPU+FPGA
  - 1.26 Pflops on 207,456 CPU cores, and 25,932 MICs
  - another 10x on FPGA

- **2014**: 3D Euler on MIC
  - 1.7 Pflops on 147,456 CPU cores, and 18,432 MICs

A Highly Scalable Framework for Atmospheric Modeling on Heterogeneous Supercomputers:

3D Euler on CPU+GPU
CPU-only Algorithm

- Parallel Version
  - Multi-node & Multi-core
  - MPI Parallelism

25 points stencil

3D channel
CPU-only Algorithm

- Parallel Version
  Multi-node & Multi-core
  MPI Parallelism
- CPU Algorithm
  Workflow

CPU Algorithm per Stencil sweep

For each subdomain
① Update Halo
② Calculate Euler stencil
  a. Compute Local Coordinate
  b. Compute Fluxes
  c. Compute Source Terms

CPU Workflow

Per Stencil Sweep

Halo Updating

Stencil Computation

①

②
Hybrid (CPU+GPU) Algorithm

- Hybrid Partition
  - GPU $\rightarrow$ Inner Stencil Computation
  - CPU $\rightarrow$ Halo Updating & Outer Stencil Computation

- CPU-GPU Hybrid Algorithm

  - **CPU-GPU Hybrid Algorithm Per Stencil Sweep**

    For each subdomain
    - GPU side: **PETSc**
      - Inner-part EulerStencil
    - CPU side:
      1. Update Halo
      2. Outer-part Euler stencil
    - BARRIER 4 layers
    - CPU-GPU Exchange
Hybrid Algorithm Design

Per Stencil Sweep

CPU

Halo Updating → Stencil Computation

①

Per Stencil Sweep

GPU

Inner Stencil Computation

②

CPU

Halo Updating → Outer Stencil Computation

①

Workflow

Barrier

G2C

C2G

②

③
A Highly Scalable Framework for Atmospheric Modeling on Heterogeneous Supercomputers:

GPU-related Optimizations
## Optimizations

<table>
<thead>
<tr>
<th>GPU Opt</th>
<th>Pinned Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SMEM/L1</td>
</tr>
<tr>
<td></td>
<td>AoS -&gt; SoA</td>
</tr>
<tr>
<td></td>
<td>Register Adjustment</td>
</tr>
<tr>
<td></td>
<td>Kernel Splitting</td>
</tr>
<tr>
<td></td>
<td>Other Methods</td>
</tr>
<tr>
<td></td>
<td>Customizable Data Cache</td>
</tr>
<tr>
<td></td>
<td>Inner-thread Rescheduling</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CPU Opt</th>
<th>OpenMP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SIMD Vectorization</td>
</tr>
<tr>
<td></td>
<td>Cache blocking</td>
</tr>
</tbody>
</table>
## Optimizations

<table>
<thead>
<tr>
<th><strong>GPU Opt</strong></th>
<th><strong>Pinned Memory</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>SMEM/L1</td>
<td></td>
</tr>
<tr>
<td>AoS -&gt; SoA</td>
<td></td>
</tr>
<tr>
<td>Register Adjustment</td>
<td></td>
</tr>
<tr>
<td>Kernel Splitting</td>
<td></td>
</tr>
<tr>
<td>Other Methods</td>
<td></td>
</tr>
<tr>
<td>Customizable Data Cache</td>
<td></td>
</tr>
<tr>
<td>Inner-thread Rescheduling</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>CPU Opt</strong></th>
<th><strong>OpenMP</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>SIMD Vectorization</td>
<td></td>
</tr>
<tr>
<td>Cache blocking</td>
<td></td>
</tr>
</tbody>
</table>

---

**Diagram Details**:

- **Device**:
  - **Local**
  - **Global**
    - **Constant**
      - **Texture**
  - **Multiprocessor**
    - **Registers Shared Memory**
  - **Constant and Texture Caches**

- **To Host**

- **Register Configuration**:
  - **AH**
  - **AL**
  - **EAX**
  - **BH**
  - **BL**
  - **EBX**
  - **CH**
  - **CL**
  - **ECX**
  - **DH**
  - **DL**
  - **EDX**
  - **SI**
  - **ESI**
  - **DI**
  - **EDI**
  - **SP**
  - **ESP**
  - **BP**
  - **EBP**

---

21
Optimizations

Theoretic: $T_2 = \frac{1}{3} \times T_1$

Reality: $T_2 < \frac{1}{2} \times T_1$

Virtual Memory $\rightarrow$ Physical Memory $\rightarrow$ GPU

Pinned-memory $\rightarrow$ Physical Memory $\rightarrow$ GPU

Theoretic: $T_2 = \frac{1}{3} \times T_1$
Reality: $T_2 < \frac{1}{2} \times T_1$
## Optimizations

<table>
<thead>
<tr>
<th>GPU Opt</th>
<th>CPU Opt</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pinned Memory</strong></td>
<td><strong>OpenMP</strong></td>
</tr>
<tr>
<td><strong>SMEM/L1</strong></td>
<td><strong>SIMD Vectorization</strong></td>
</tr>
<tr>
<td>AoS -&gt; SoA</td>
<td>Cache blocking</td>
</tr>
<tr>
<td>Register Adjustment</td>
<td></td>
</tr>
<tr>
<td>Kernel Splitting</td>
<td></td>
</tr>
<tr>
<td>Other Methods</td>
<td></td>
</tr>
<tr>
<td>Customizable Data Cache</td>
<td></td>
</tr>
<tr>
<td>Inner-thread Rescheduling</td>
<td></td>
</tr>
</tbody>
</table>

Compiler option
-Xptxas dlc= ca
### Optimizations

<table>
<thead>
<tr>
<th>GPU Opt</th>
<th>CPU Opt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pinned Memory</td>
<td>OpenMP</td>
</tr>
<tr>
<td>SMEM/L1</td>
<td>SIMD Vectorization</td>
</tr>
<tr>
<td><strong>AoS -&gt; SoA</strong></td>
<td>Cache blocking</td>
</tr>
<tr>
<td>Register Adjustment</td>
<td></td>
</tr>
<tr>
<td>Kernel Splitting</td>
<td></td>
</tr>
<tr>
<td>Other Methods</td>
<td></td>
</tr>
<tr>
<td>Customizable Data Cache</td>
<td></td>
</tr>
<tr>
<td>Inner-thread Rescheduling</td>
<td></td>
</tr>
</tbody>
</table>

![Diagram showing Pinned Memory and other optimization techniques](image)
## Optimizations

### GPU Opt

<table>
<thead>
<tr>
<th>Pinned Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMEM/L1</td>
</tr>
<tr>
<td>AoS -&gt; SoA</td>
</tr>
<tr>
<td><strong>Register Adjustment</strong></td>
</tr>
<tr>
<td>Kernel Splitting</td>
</tr>
<tr>
<td>Other Methods</td>
</tr>
<tr>
<td>Customizable Data Cache</td>
</tr>
<tr>
<td>Inner-thread Rescheduling</td>
</tr>
</tbody>
</table>

### CPU Opt

<table>
<thead>
<tr>
<th>OpenMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIMD Vectorization</td>
</tr>
<tr>
<td>Cache blocking</td>
</tr>
</tbody>
</table>

---

### Streaming Multi-Processor

- **64K Register**
- **2048 threads**

Rt: Register per thread

**Occupancy** = \( \frac{(64 \times 1024)}{(2048 \times \text{Rt})} \)

---

256 registers per threads

Rt = 256

1 Block per SM

**Occupancy** = \( \frac{(64 \times 1024)}{(2048 \times \text{Rt})} = 12.5\% \)
## Optimizations

<table>
<thead>
<tr>
<th>GPU Opt</th>
<th>Pinned Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SMEM/L1</td>
</tr>
<tr>
<td></td>
<td>AoS -&gt; SoA</td>
</tr>
<tr>
<td><strong>Register Adjustment</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Kernel Splitting</td>
</tr>
<tr>
<td></td>
<td>Other Methods</td>
</tr>
<tr>
<td></td>
<td>Customizable Data Cache</td>
</tr>
<tr>
<td></td>
<td>Inner-thread Rescheduling</td>
</tr>
</tbody>
</table>

### CPU Opt

<table>
<thead>
<tr>
<th></th>
<th>OpenMP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SIMD Vectorization</td>
</tr>
<tr>
<td></td>
<td>Cache blocking</td>
</tr>
</tbody>
</table>

### Streaming Multi-Processor

- **64K Register**
- **2048 threads**

### Formula

\[
\text{Occupancy} = \frac{64 \times 1024}{2048 \times \text{Rt}}
\]

### GPU Opt

- **Register per thread**

### CPU Opt

- **64 registers per thread**
- **Rt = 64**
- **4 Block per SM**

### Compiler option

- `-maxrregcount = 64`
## Optimizations

<table>
<thead>
<tr>
<th>GPU Opt</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pinned Memory</td>
<td>SMEM/L1</td>
<td>AoS -&gt; SoA</td>
<td>Register Adjustment</td>
<td>Kernel Splitting</td>
<td>Other Methods</td>
<td>Other Methods</td>
</tr>
<tr>
<td></td>
<td>Customizable Data Cache</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Inner-thread Rescheduling</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| CPU Opt | | | | | | | |
|---------|--------------|--------------|--------------|--------------|--------------|--------------|
|         | OpenMP | SIMD Vectorization | Cache blocking | | | | |

![Diagram showing optimization techniques]
Optimizations

Algorithm 8 Demonstration of Euler State Reconstruction Step

X-axis original code:  \((qe[0], qe[1], qw[0] \text{ and } qw[1])\) are intermediate variables in the step of State Reconstruction

1:  \(qw[0] = (24*x[k,j,i] + 8*x[k,j,i-1] - 4*x[k,j,i+1] - (x[k,j+1,i] + x[k,j-1,i]
+ x[k+1,j,i] + x[k-1,j,i])) / 24;\)

2:  \(qe[0] = (24*x[k,j,i] + 8*x[k,j,i+1] - 4*x[k,j,i-1] - (x[k,j+1,i] + x[k,j-1,i]
+ x[k+1,j,i] + x[k-1,j,i])) / 24;\)

3:  \(qe[1] = (24*x[k,j,i+1] + 8*x[k,j,i] - 4*x[k,j,i+2] - (x[k,j+1,i+1] + x[k,j-1,i+1]
+ x[k+1,j,i+1] + x[k-1,j,i+1])) / 24;\)

4:  \(qw[1] = (24*x[k,j,i-1] + 8*x[k,j,i] - 4*x[k,j,i-2] - (x[k,j+1,i-1] + x[k,j-1,i-1]
+ x[k+1,j,i-1] + x[k-1,j,i-1])) / 24;\)
Algorithm 8 Demonstration of Euler State Reconstruction
Step

X-axis original code: \((qe[0], qe[1], qw[0] \text{ and } qw[1])\text{ are intermediate variables in the step of State Reconstruction)\)

1. \(qw[0] = \frac{(24*x[k,j,i] + 8*x[k,j,i-1] - 4*x[k,j,i+1] - (x[k,j+1,i] + x[k,j-1,i] + x[k+1,j,i] + x[k-1,j,i]))}{24};\)
2. \(qe[0] = \frac{(24*x[k,j,i] + 8*x[k,j,i+1] - 4*x[k,j,i-1] - (x[k,j+1,i] + x[k,j-1,i] + x[k+1,j,i] + x[k-1,j,i]))}{24};\)
3. \(qe[1] = \frac{(24*x[k,j,i+1] + 8*x[k,j,i] - 4*x[k,j,i+2] - (x[k,j+1,i+1] + x[k,j-1,i+1] + x[k+1,j,i+1] + x[k-1,j,i+1]))}{24};\)
4. \(qw[1] = \frac{(24*x[k,j,i-1] + 8*x[k,j,i] - 4*x[k,j,i-2] - (x[k,j+1,i-1] + x[k,j-1,i-1] + x[k+1,j,i-1] + x[k-1,j,i-1]))}{24};\)
Optimizations
A Highly Scalable Framework for Atmospheric Modeling on Heterogeneous Supercomputers:

Results
# Experimental Result

<table>
<thead>
<tr>
<th>CPU Opt</th>
<th>CPU Opt</th>
<th>OpenMP</th>
<th>SIMD Vectorization</th>
<th>Cache blocking</th>
<th>Pinned Memory</th>
<th>SMEM/L1</th>
<th>AoS -&gt; SoA</th>
<th>Kernel Splitting</th>
<th>Register Adjustment</th>
<th>Other Methods</th>
<th>Customized Data Cache</th>
<th>Inner-thread Rescheduling</th>
</tr>
</thead>
<tbody>
<tr>
<td>19.7s</td>
<td>5.91s</td>
<td>1.80s</td>
<td>0.92s</td>
<td>70%</td>
<td>69%</td>
<td>49%</td>
<td>31.64x</td>
<td>speedup over 12-core CPU (E5-2697 v2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Experimental Result
First-Round Optimizations
- Five general-used optimizations
- A 80GFlops performance result is achieved on a single Tesla K40

Customized Optimizations
- A customized cache mechanism & Inter-thread Rescheduling
- A 146GFlops performance result is achieved on a single Tesla K40

Experimental Results
- 451GFLOPs on a single Tesla K80, which is 31.64x speedup over a 12-core CPU (E5-2697 v2)
- 16.87% of peak based on Tesla K80

Weak Scaling Result
- 98.7% among 32 Node
Acknowledgement

National Natural Science Foundation of China

Tsinghua University

Statoil

IBM

inspur 浪潮

Schlumberger

MAXELLER Technologies

NVIDIA

BGP INC., CHINA NATIONAL PETROLEUM CORPORATION

Microsoft
Thank You!

haohuan@tsinghua.edu.cn