GPU Acceleration of Particle Advection Workloads in a Parallel, Distributed Memory Setting

Presented by David Camp
**What is Advection?**

**ad-vec-tion**  
(ad-vēk'shon)

*n.*

1. The transfer of a property of the atmosphere, such as heat, cold, or humidity, by the horizontal movement of an air mass: *Today’s temperatures were higher due to the advection of warm air into the region.*
2. The rate of change of an atmospheric property caused by the horizontal movement of air.
3. The horizontal movement of water, as in an ocean current.

[Latin advectiō, advectiōn-, act of conveying, from advehēre, past participle of advehere, *to carry to*: ad-, *ad-* + vehere, *to carry*; see wegh- in Indo-European roots.]
Particle Advection is a Foundational Visualization Algorithm
What is a GPU Cluster?

Distributed memory cluster, where tasks communicate over a network. Work can be performed on the CPU or/and on the GPU.
Have We Done Enough GPU Research?

Effective use of GPU clusters

Vis Algorithms on GPU

Mostly GPGPU for simulation.

Vis algorithms on GPU clusters???
Previous Work For Vis Algorithms on GPU Clusters

• Mostly rendering:
  – Surface rendering.
  – Volume rendering.

• Very little published literature outside of rendering.

Notables:
  – Bachtaler, Strengert, Weiskpf, Ertl @ EGPGV06: Texture-based vector field visualization.
  – Martin, Shen, McCormick @ EGPGV10: Load-balanced isosurfacing.
Why Do We Care About Vis Algorithms on GPU Clusters?

• Previous decade:
  – Simulations on supercomputers store snapshots of their state.
  – This data is post-processed by visualization programs.
  – These programs are sometimes run on GPU clusters.

• Upcoming decade:
  – Simulations on supercomputers will often have severe constraints in I/O and power.
  – In these cases, visualization programs must be run in situ with the simulation.
  – Supercomputers are increasingly GPU-accelerated.
  – Visualization algorithms must run on GPU clusters.
What are the Challenges for Designing an Algorithm for a GPU Cluster?

• Complex system: How to map an algorithm to it?
• How to optimize performance?

Tensions between:
• The increased computational power of a GPU versus the latency to access it.
• The few processor cores on a CPU with high clock speeds versus the many processor cores of a GPU with lower clock speeds.
This Study: Particle Advection on GPU Clusters

• **Fundamental research question:**
  If faced with a particle advection problem, should you:
  • Implement a GPU solution?
  • Implement a CPU solution?
  • Does it not matter?
Research Assumptions

• Data is too large to fit on a single node.
  – In situ use case: data already distributed over nodes of a supercomputer.

• We will focus on “parallelize-over-data”.
  – Best fits the in situ case.
“Parallelize-over-Data” Strategy

The data is divided across all tasks.

The particle is advected on Task 1. Then it is transferred to Task 2.

This process is repeated until the particle is finished.
Implementation Details

• GPU version:
  – 1 thread to handle MPI communication.
  – 1 thread to handle GPU advection.

• CPU version:
  – 1 thread to handle MPI communication.
  – 8 threads to handle advection.

• Both used a 4th-order Runge-Kutta technique to advect particles.
Description of Study

• “No two particle advection problems are alike”.

• Differences may come about because of:
  – Number of seed points.
  – Duration of advection.
  – Vector Field Complexity.
  – (and more factors).

• We wanted to be able to draw conclusions about a wider range of particle advection problems than “for this one particular problem…”.
Study Design

• Many configurations:
  – 10 seeding densities.
    • Sparsest corresponded to streamlines, densest to FTLE, places in between corresponded to algorithms like stream surfaces.
  – 5 advection durations (tiny, little, short, med, long).
  – 3 data sets.

• $10 \times 5 \times 3 = 150$ configurations.

• Ran GPU and CPU test for each configuration.
  – 300 tests.
Study Details

- 8 Node cluster Dirac at NERSC, each node had the following:
  - 1 NVidia Tesla C2050 (Fermi), with 448 CUDA cores running at 1.15 GHz.
  - 2 Intel quad-core Nehalem processors, with 8 cores running at 2.4 GHz.

- $500^3$ cell data block per task.
Results

- Vector field has little effect.
- CPU faster with few particles, GPU faster with many particles.
- GPU faster for short durations.
- Long durations produce most extreme results.
Analyzing the 150 Pairs...

This chart is a logarithmic scale, so points below 1 show us tests were the CPU was faster.
Analyzing the 150 Pairs...

Red line spotlights eight tests that all have ~1 million advection steps.
Analyzing the 123 Pairs...

This chart only shows the tests where the GPU was faster.

Best fit line has a slope of 11 and a correlation coefficient of 0.81.
Identifying the Most Dominant Factors for Performance

<table>
<thead>
<tr>
<th>Event</th>
<th>Slope</th>
<th>Correlation Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GPU Advantage</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>High activity</td>
<td>11.6</td>
<td>0.81</td>
</tr>
<tr>
<td>Low activity</td>
<td>-5.3</td>
<td>-0.63</td>
</tr>
<tr>
<td>CPU overhead</td>
<td>-0.05</td>
<td>0.01</td>
</tr>
<tr>
<td>Latency</td>
<td>-0.25</td>
<td>-0.44</td>
</tr>
<tr>
<td>Idle</td>
<td>-6.0</td>
<td>-0.50</td>
</tr>
<tr>
<td><strong>CPU Advantage</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>High activity</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Low activity</td>
<td>3.6</td>
<td>0.5</td>
</tr>
<tr>
<td>CPU overhead</td>
<td>-0.03</td>
<td>-0.333</td>
</tr>
<tr>
<td>Latency</td>
<td>-0.13</td>
<td>-0.69</td>
</tr>
<tr>
<td>Idle</td>
<td>-3.4</td>
<td>-0.50</td>
</tr>
</tbody>
</table>
Summary: Particle Advection on GPU Clusters

• **Fundamental research question:**
  If faced with a particle advection problem, should you:
  
  • Implement a GPU solution?
  • Implement a CPU solution?
  • Does it not matter?
Summary: Particle Advection on GPU Clusters

• Dominant factor is activity:
  – “High activity integration”- dominant: use GPUs.
  – “Low activity integration”- dominant: use CPUs.
  – Lots of idle time: doesn’t matter!

• Surprises (to us):
  – GPU Latency is a virtual non-factor.
  – Long durations lead to extreme cases for either.
Future Work

• Many future directions:
  – Heterogeneous algorithms?
  – Do the results change as a function of concurrency? What about extreme scale? (recall in situ driver)
  – What if we expand the scope of architectures beyond CPU and GPU?
  – Do the results change significantly for unsteady state?
  – Do the results change significantly for different parallelization schemes?
GPU Acceleration of Particle Advection Workloads in a Parallel, Distributed Memory Setting

David Camp, Hari Krishnan, David Pugmire, Christoph Garth, Ian Johnson, E. Wes Bethel, Kenneth I. Joy, and Hank Childs