Experiments in Pure Parallelism

Dave Pugmire, ORNL
Hank Childs, LBNL/ UC Davis
Brad Whitlock, LLNL
Mark Howison, LBNL
Prabhat, LBNL
Sean Ahern, ORNL
Gunther Weber, LBNL
Wes Bethel LBNL
The story behind this work.....

Deliver production visualization tools to researchers

- Multidiscipline
- Multi-institutional
- Petascale applications
- Running on the largest machines in the world
Motivation and Introduction
- Weak scaling study: The Joule Metric
- Scalability on exascale datasets
- Final thoughts
Motivation

• The landscape is going to dramatically change

• Questions:
  – Are we ready for exascale simulations?
  – How far can we push production visualization tools of today?
  – What bottlenecks will we encounter?
Pure parallelism

• Pure parallelism is data-level parallelism, but…
  – Multi-resolution can be data-level parallelism
  – Out-of-core can be data-level parallelism

• Pure parallelism: “brute force” … processing full resolution data using data-level parallelism

• Pros:
  – Easy to implement

• Cons:
  – Requires large I/O capabilities
  – Requires large amount of primary memory
  – Requires big machines
Pure parallelism
Pure parallelism

Parallel Simulation Code

Data “chunks” on disk
Pure parallelism

Parallel Simulation Code

Parallel visualization program

Process 0
Read → Process → Render

Process 1
Read → Process → Render

Process 2
Read → Process → Render

Data “chunks” on disk

Thursday, July 29, 2010
Pure parallelism

Parallel Simulation Code

Parallel visualization program

Process 0

Read → Process → Render

Process 1

Read → Process → Render

Process 2

Read → Process → Render

Thursday, July 29, 2010
Pure parallelism

Parallel Simulation Code

Parallel visualization program

Process 0
Read \rightarrow Process \rightarrow Render

Process 1
Read \rightarrow Process \rightarrow Render

Process 2
Read \rightarrow Process \rightarrow Render

Thursday, July 29, 2010
Pure parallelism
Outline

✓ Motivation and Introduction

⇒ Weak scaling study: The Joule Metric
  - Scalability on exascale datasets
  - Final thoughts
Scalability of scalar field visualization algorithms

The **Joule** Metric:

DOE Office of Science metric for tracking the efficiency of use of HPC resources

Demonstrate weak scaling by Q4 of a Q2 baseline problem

**2009 Joule codes:**

Thursday, July 29, 2010
Scalability of scalar field visualization algorithms

The **Joule** Metric:

DOE Office of Science metric for tracking the efficiency of use of HPC resources

Demonstrate weak scaling by Q4 of a Q2 baseline problem

**2009 Joule codes:**

- XGC1
- CAM
- Raptor
- VisIt
Joule and VisIt

Isocontouring and volume rendering of a radiation transport code, Denovo
Joule and VisIt

Isocontouring and volume rendering of a radiation transport code, Denovo

**Q2 baseline:**
103M cells on 4K cores
Joule and VisIt

Isocontouring and volume rendering of a radiation transport code, Denovo

**Q2 baseline:**
103M cells on 4K cores

**Q4 benchmark:**
321M cells on 12K cores

Thursday, July 29, 2010
Joule and VisIt

Isocontouring and volume rendering of a radiation transport code, Denovo

**Q2 baseline:**
103M cells on 4K cores

**Q4 benchmark:**
321M cells on 12K cores

As expected, isocontouring exhibited nearly perfect scaling
Scalability of volume rendering

Ray casted volume rendering:
4000 samples per ray, 103M cells

Volume Rendering Time

Sec. per frame

Volume Rendering Time

Number of cores

512 1024 2048 4096

Thursday, July 29, 2010
Barrier to volume rendering scalability
Barrier to volume rendering scalability
Barrier to volume rendering scalability
Barrier to volume rendering scalability

Back to front compositing
Barrier to volume rendering scalability

Back to front compositing
Volume rendering bottleneck

- Compositing puts all samples on a single processor.
- Communication minimized if P2 sends samples to P3
- VisIt determines a communication minimizing assignment
- **However**, this optimization requires all-to-all communication
Volume rendering bottleneck

- Compositing puts all samples on a single processor.
- Communication minimized if $P_2$ sends samples to $P_3$
- VisIt determines a communication minimizing assignment
- **However**, this optimization requires all-to-all communication

All to All communication does not scale
Avoiding **all-to-all** communication dramatically improved scalability

Q2 to Q4 weak scaling of $\sim 5X$ achieved
Outline

✓ Motivation and Introduction
✓ Weak scaling study: The Joule Metric
→ Scalability on exascale datasets
- Final thoughts
Experiments at the exascale

• Is pure parallelism ready for exascale?
• Vary over:
  - Dataset size
  - Data generation
  - Supercomputer architectures
Experiment methodology

1. Dataset generation:
   - Tomorrow’s data not available yet
   - Synthetic data should be reasonable surrogate

2. Read dataset from disk
3. Extract isosurface
4. Render @ 1024x1024

Isosurface of 2 trillion cells, visualized with VisIt on Jaguarpf using 32,000 cores.
Experiment methodology

• Bottlenecks found in volume rendering
• Fixed, but not in time for study
  – Viz runs are opportunistic
  – Hard to get a second chance
  – Render time ~ 5 sec per frame
• Contouring exercises much of the infrastructure

Volume rendering of 2 trillion cells, visualized with VisIt on Jaguarpf using 32,000 cores.
Experiment methodology

• Only used pure parallelism
  – This experiment was about testing the limits of pure parallelism
  – Purposely did not use in situ, multi-resolution, out-of-core, data sub-setting

• Pure parallelism is what production visualization tools use right now
Vary over supercomputer

• Goals:
  – Ensure results aren’t tied to a single machine.
  – Understand results from different architectures.

• Experiment details
  – 1 trillion cells per 16,000 cores
  – 10*Ncores “Brick-of-float” files, gzipped
  – Upsampled data
Vary over supercomputer

• Goals:
  – Ensure results aren’t tied to a single machine.
  – Understand results from different architectures.

• Experiment details
  – 1 trillion cells per 16,000 cores
  – 10*NCores “Brick-of-float” files, gzipped
  – Upsampled data

<table>
<thead>
<tr>
<th>Machine name</th>
<th>Machine type or OS</th>
<th>Total no. of cores</th>
<th>Memory per core (Gbytes)</th>
<th>System type</th>
<th>Clock speed</th>
<th>Peak flops</th>
<th>Top 500 rank (as of Nov. 2009)</th>
</tr>
</thead>
<tbody>
<tr>
<td>JaguarPF</td>
<td>Cray</td>
<td>224,162</td>
<td>2.0</td>
<td>XT5</td>
<td>2.6 GHz</td>
<td>2.33 Pflops</td>
<td>1</td>
</tr>
<tr>
<td>Ranger</td>
<td>Sun Linux</td>
<td>62,976</td>
<td>2.0</td>
<td>Opteron Quad</td>
<td>2.0 GHz</td>
<td>503.8 Tflops</td>
<td>9</td>
</tr>
<tr>
<td>Dawn</td>
<td>Blue Gene/P</td>
<td>147,456</td>
<td>1.0</td>
<td>PowerPC</td>
<td>850.0 MHz</td>
<td>415.7 Tflops</td>
<td>11</td>
</tr>
<tr>
<td>Franklin</td>
<td>Cray</td>
<td>38,128</td>
<td>1.0</td>
<td>XT4</td>
<td>2.6 GHz</td>
<td>352 Tflops</td>
<td>15</td>
</tr>
<tr>
<td>Juno</td>
<td>Commodity (Linux)</td>
<td>18,402</td>
<td>2.0</td>
<td>Opteron Quad</td>
<td>2.2 GHz</td>
<td>131.6 Tflops</td>
<td>27</td>
</tr>
<tr>
<td>Purple</td>
<td>AIX (Advanced Interactive Executive)</td>
<td>12,208</td>
<td>3.5</td>
<td>Power5</td>
<td>1.9 GHz</td>
<td>92.8 Tflops</td>
<td>66</td>
</tr>
</tbody>
</table>
Results

Runtimes for I/O, contouring, and rendering.

Thursday, July 29, 2010
# Results

<table>
<thead>
<tr>
<th>Machine</th>
<th>No. of cores</th>
<th>Data set size (TCells)</th>
<th>Total I/O time (sec.)</th>
<th>Contour time (sec.)</th>
<th>Total pipeline execution time (sec.)†</th>
<th>Rendering time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purple</td>
<td>8,000</td>
<td>0.5</td>
<td>53.4</td>
<td>10.0</td>
<td>63.7</td>
<td>2.9</td>
</tr>
<tr>
<td>Dawn</td>
<td>16,384*</td>
<td>1.0</td>
<td>240.9</td>
<td>32.4</td>
<td>277.6</td>
<td>10.6</td>
</tr>
<tr>
<td>Juno</td>
<td>16,000</td>
<td>1.0</td>
<td>102.9</td>
<td>7.2</td>
<td>110.4</td>
<td>10.4</td>
</tr>
<tr>
<td>Ranger</td>
<td>16,000</td>
<td>1.0</td>
<td>251.2</td>
<td>8.3</td>
<td>259.7</td>
<td>4.4</td>
</tr>
<tr>
<td>Franklin</td>
<td>16,000</td>
<td>1.0</td>
<td>129.3</td>
<td>7.9</td>
<td>137.3</td>
<td>1.6</td>
</tr>
<tr>
<td>JaguarPF</td>
<td>16,000</td>
<td>1.0</td>
<td>236.1</td>
<td>10.4</td>
<td>246.7</td>
<td>1.5</td>
</tr>
<tr>
<td>Franklin</td>
<td>32,000</td>
<td>2.0</td>
<td>292.4</td>
<td>8.0</td>
<td>300.6</td>
<td>9.7</td>
</tr>
<tr>
<td>JaguarPF</td>
<td>32,000</td>
<td>2.0</td>
<td>707.2</td>
<td>7.7</td>
<td>715.2</td>
<td>1.5</td>
</tr>
</tbody>
</table>
### Results

<table>
<thead>
<tr>
<th>Machine</th>
<th>No. of cores</th>
<th>Data set size (TCells)</th>
<th>Total I/O time (sec.)</th>
<th>Contour time (sec.)</th>
<th>Total pipeline execution time (sec.)</th>
<th>Rendering time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purple</td>
<td>8,000</td>
<td>0.5</td>
<td>53.4</td>
<td>10.0</td>
<td>63.7</td>
<td>2.9</td>
</tr>
<tr>
<td>Dawn</td>
<td>16,384*</td>
<td>1.0</td>
<td>240.9</td>
<td>32.4</td>
<td>277.6</td>
<td>10.6</td>
</tr>
<tr>
<td>Juno</td>
<td>16,000</td>
<td>1.0</td>
<td>102.9</td>
<td>7.2</td>
<td>110.4</td>
<td>10.4</td>
</tr>
<tr>
<td>Ranger</td>
<td>16,000</td>
<td>1.0</td>
<td>251.2</td>
<td>8.3</td>
<td>259.7</td>
<td>4.4</td>
</tr>
<tr>
<td>Franklin</td>
<td>16,000</td>
<td>1.0</td>
<td>129.3</td>
<td>7.9</td>
<td>137.3</td>
<td>1.6</td>
</tr>
<tr>
<td>JaguarPF</td>
<td>16,000</td>
<td>1.0</td>
<td>236.1</td>
<td>10.4</td>
<td>246.7</td>
<td>1.5</td>
</tr>
<tr>
<td>Franklin</td>
<td>32,000</td>
<td>2.0</td>
<td>292.4</td>
<td>8.0</td>
<td>300.6</td>
<td>9.7</td>
</tr>
<tr>
<td>JaguarPF</td>
<td>32,000</td>
<td>2.0</td>
<td>707.2</td>
<td>7.7</td>
<td>715.2</td>
<td>1.5</td>
</tr>
</tbody>
</table>

Lustre striping of 2 vs. 4

Thursday, July 29, 2010
## Results

Lustre striping of 2 vs. 4

BG/L has 850 MHz clock

<table>
<thead>
<tr>
<th>Machine</th>
<th>No. of cores</th>
<th>Data set size (TCells)</th>
<th>Total I/O time (sec.)</th>
<th>Contour time (sec.)</th>
<th>Total pipeline execution time (sec.)</th>
<th>Rendering time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purple</td>
<td>8,000</td>
<td>0.5</td>
<td>53.4</td>
<td>10.0</td>
<td>63.7</td>
<td>2.9</td>
</tr>
<tr>
<td>Dawn</td>
<td>16,384*</td>
<td>1.0</td>
<td>240.9</td>
<td>32.4</td>
<td>277.6</td>
<td>10.6</td>
</tr>
<tr>
<td>Juno</td>
<td>16,000</td>
<td>1.0</td>
<td>102.9</td>
<td>7.2</td>
<td>110.4</td>
<td>4.4</td>
</tr>
<tr>
<td>Ranger</td>
<td>16,000</td>
<td>1.0</td>
<td>251.2</td>
<td>8.3</td>
<td>259.7</td>
<td>10.4</td>
</tr>
<tr>
<td>Franklin</td>
<td>16,000</td>
<td>1.0</td>
<td>129.3</td>
<td>7.9</td>
<td>137.2</td>
<td>1.6</td>
</tr>
<tr>
<td>JaguarPF</td>
<td>16,000</td>
<td>1.0</td>
<td>236.1</td>
<td>10.4</td>
<td>246.7</td>
<td>1.5</td>
</tr>
<tr>
<td>Franklin</td>
<td>32,000</td>
<td>2.0</td>
<td>292.4</td>
<td>8.0</td>
<td>300.6</td>
<td>9.7</td>
</tr>
<tr>
<td>JaguarPF</td>
<td>32,000</td>
<td>2.0</td>
<td>707.2</td>
<td>7.7</td>
<td>715.2</td>
<td>1.5</td>
</tr>
</tbody>
</table>
## Results

### Lustre striping of 2 vs. 4

<table>
<thead>
<tr>
<th>Machine</th>
<th>No. of cores</th>
<th>Data set size (TCells)</th>
<th>Total I/O time (sec.)</th>
<th>Contour time (sec.)</th>
<th>Total pipeline execution time (sec.)</th>
<th>Rendering time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purple</td>
<td>8,000</td>
<td>0.5</td>
<td>53.4</td>
<td>10.0</td>
<td>63.7</td>
<td>2.9</td>
</tr>
<tr>
<td>Dawn</td>
<td>16,384*</td>
<td>1.0</td>
<td>240.9</td>
<td>32.4</td>
<td>277.6</td>
<td>10.6</td>
</tr>
<tr>
<td>Juno</td>
<td>16,000</td>
<td>1.0</td>
<td>102.9</td>
<td>7.2</td>
<td>110.4</td>
<td>10.4</td>
</tr>
<tr>
<td>Ranger</td>
<td>16,000</td>
<td>1.0</td>
<td>251.2</td>
<td>8.3</td>
<td>259.7</td>
<td>4.4</td>
</tr>
<tr>
<td>Franklin</td>
<td>16,000</td>
<td>1.0</td>
<td>129.3</td>
<td>7.9</td>
<td>137.3</td>
<td>1.6</td>
</tr>
<tr>
<td>JaguarPF</td>
<td>16,000</td>
<td>1.0</td>
<td>236.1</td>
<td>10.4</td>
<td>246.7</td>
<td>1.5</td>
</tr>
<tr>
<td>Franklin</td>
<td>32,000</td>
<td>2.0</td>
<td>292.4</td>
<td>8.0</td>
<td>300.6</td>
<td>9.7</td>
</tr>
<tr>
<td>JaguarPF</td>
<td>32,000</td>
<td>2.0</td>
<td>707.2</td>
<td>7.7</td>
<td>715.2</td>
<td>4.5</td>
</tr>
</tbody>
</table>

BG/L has 850 MHz clock

7-10 network links failed. Required re-routing
## Results

<table>
<thead>
<tr>
<th>Machine</th>
<th>No. of cores</th>
<th>Data set size (TCells)</th>
<th>Total I/O time (sec.)</th>
<th>Contour time (sec.)</th>
<th>Total pipeline execution time (sec.)</th>
<th>Rendering time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purple</td>
<td>8,000</td>
<td>0.5</td>
<td>53.4</td>
<td>10.0</td>
<td>63.7</td>
<td>2.9</td>
</tr>
<tr>
<td>Dawn</td>
<td>16,384*</td>
<td>1.0</td>
<td>240.9</td>
<td>32.4</td>
<td>277.6</td>
<td>10.6</td>
</tr>
<tr>
<td>Juno</td>
<td>16,000</td>
<td>1.0</td>
<td>102.9</td>
<td>7.2</td>
<td>110.4</td>
<td>10.4</td>
</tr>
<tr>
<td>Ranger</td>
<td>16,000</td>
<td>1.0</td>
<td>251.2</td>
<td>8.3</td>
<td>259.7</td>
<td>4.4</td>
</tr>
<tr>
<td>Franklin</td>
<td>16,000</td>
<td>1.0</td>
<td>129.3</td>
<td>7.9</td>
<td>137.3</td>
<td>1.6</td>
</tr>
<tr>
<td>JaguarPF</td>
<td>16,000</td>
<td>1.0</td>
<td>236.1</td>
<td>10.4</td>
<td>246.7</td>
<td>1.5</td>
</tr>
<tr>
<td>Franklin</td>
<td>32,000</td>
<td>2.0</td>
<td>292.4</td>
<td>8.0</td>
<td>300.6</td>
<td>9.7</td>
</tr>
<tr>
<td>JaguarPF</td>
<td>32,000</td>
<td>2.0</td>
<td>707.2</td>
<td>7.7</td>
<td>715.2</td>
<td>1.3</td>
</tr>
</tbody>
</table>

- Lustre striping of 2 vs. 4
- BG/L has 850 MHz clock
- 7-10 network links failed. Required re-routing

*Note: LG/L has 850 MHz clock*
Varying over data generation

• Concern:
  – Does upsampling produce unrepresentatively smooth surfaces?

• Alternative: replication

Isosurface of 1 trillion cells, visualized with VisIt on Franklin using 16,000 cores.
## Results from data generation test

Test on franklin, using 16,000 cores with unzipped data

<table>
<thead>
<tr>
<th>Data generation</th>
<th>Total I/O time (sec.)</th>
<th>Contour time (sec.)</th>
<th>Total pipeline execution time (sec.)</th>
<th>Rendering time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upsampled</td>
<td>478.3</td>
<td>7.6</td>
<td>486.0</td>
<td>2.8</td>
</tr>
<tr>
<td>Replicated</td>
<td>493.0</td>
<td>7.6</td>
<td>500.7</td>
<td>4.9</td>
</tr>
</tbody>
</table>
Results from data generation test

Test on franklin, using 16,000 cores with unzipped data

<table>
<thead>
<tr>
<th>Data generation</th>
<th>Total I/O time (sec.)</th>
<th>Contour time (sec.)</th>
<th>Total pipeline execution time (sec.)</th>
<th>Rendering time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upsampled</td>
<td>478.3</td>
<td>7.6</td>
<td>486.0</td>
<td>2.8</td>
</tr>
<tr>
<td>Replicated</td>
<td>493.0</td>
<td>7.6</td>
<td>500.7</td>
<td>4.9</td>
</tr>
</tbody>
</table>

More triangles generated, but loading 1 trillion zones dominates
Results from data generation test

Test on franklin, using 16,000 cores with unzipped data

<table>
<thead>
<tr>
<th>Data generation</th>
<th>Total I/O time (sec.)</th>
<th>Contour time (sec.)</th>
<th>Total pipeline execution time (sec.)</th>
<th>Rendering time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upsampled</td>
<td>478.3</td>
<td>7.6</td>
<td>486.0</td>
<td>2.8</td>
</tr>
<tr>
<td>Replicated</td>
<td>493.0</td>
<td>7.6</td>
<td>500.7</td>
<td>4.9</td>
</tr>
</tbody>
</table>

More triangles generated, but loading 1 trillion zones dominates
Replicated data contains more geometry
Pitfalls at scale

• Volume rendering
  – Algorithms scales well up to 2048 processors

• Startup time
  – Loading plugins overwhelmed file system
  – Took ~5 minutes
  – Solution #1:
    • Read plugin information on MPI task 0 and broadcast. (90% speedup)
  – Solution #2:
    • static linking
    • Added in VisIt 2.0
    • Still need to demonstrate at scale
Pitfalls at scale

All to one communication

• Each MPI task needs to report high level information
  – Errors, data, spatial extents, etc

• Previous implementation:
  – Every MPI task sends a direct message to MPI task 0.

• New implementation (Mark Miller, LLNL):
  – Tree communication
# Pitfalls at scale

<table>
<thead>
<tr>
<th>All-to-one?</th>
<th>No. of cores</th>
<th>Data set size (TCells)</th>
<th>Total I/O time (sec.)</th>
<th>Contour time (sec.)</th>
<th>Total pipeline execution time (sec.)</th>
<th>Pipeline minus contour &amp; I/O (sec.)</th>
<th>Date run</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>16,384</td>
<td>1</td>
<td>88.0</td>
<td>32.2</td>
<td>368.7</td>
<td>248.5</td>
<td>June 2009</td>
</tr>
<tr>
<td>Yes</td>
<td>65,536</td>
<td>4</td>
<td>95.3</td>
<td>38.6</td>
<td>425.9</td>
<td>294.0</td>
<td>June 2009</td>
</tr>
<tr>
<td>No</td>
<td>16,384</td>
<td>1</td>
<td>240.9</td>
<td>32.4</td>
<td>277.6</td>
<td>4.3</td>
<td>Aug. 2009</td>
</tr>
</tbody>
</table>
Debugging runs at scale critical to resolving these issues
Continued study

<table>
<thead>
<tr>
<th>Date</th>
<th>Number of zones per timestep</th>
<th>Number of timesteps</th>
<th>Total Size Zones &amp; Bytes</th>
<th>Mesh+Vars</th>
<th>Platform</th>
<th>Visioneers (Visualization-Engineers)</th>
<th>More Info...</th>
</tr>
</thead>
<tbody>
<tr>
<td>October 2009</td>
<td>20,001³ (8 Tz)</td>
<td>1</td>
<td></td>
<td>rect+1</td>
<td>12000 cpus of Graph (Linux)</td>
<td>Cyrus Harrison</td>
<td>More Info...</td>
</tr>
<tr>
<td>June 2009</td>
<td>15,871³ (4 Tz)</td>
<td>1</td>
<td></td>
<td>rect+1</td>
<td>65536 cpus of Dawn (BlueGene/P)</td>
<td>Brad Whitlock</td>
<td>More Info...</td>
</tr>
<tr>
<td>May 2009</td>
<td>12,596³ (2 Tz)</td>
<td>1</td>
<td></td>
<td>rect+1</td>
<td>32000 cpus of Jaguar (Cray)</td>
<td>David Pugmire Sean Ahern</td>
<td>More Info...</td>
</tr>
<tr>
<td>June 2009</td>
<td>12,596³ (2 Tz)</td>
<td>1</td>
<td></td>
<td>rect+1</td>
<td>32000 cpus of Franklin (Cray)</td>
<td>Mark Howison Prabhat Hank Childs</td>
<td>More Info...</td>
</tr>
<tr>
<td>April 2009</td>
<td>10,000³ (1 Tz)</td>
<td>1</td>
<td></td>
<td>rect+1</td>
<td>16000 cpus of Jaguar (Cray)</td>
<td>David Pugmire Sean Ahern</td>
<td>More Info...</td>
</tr>
<tr>
<td>May 2009</td>
<td>10,000³ (1 Tz)</td>
<td>1</td>
<td></td>
<td>rect+1</td>
<td>16000 cpus of Ranger (Sun Linux)</td>
<td>Hank Childs</td>
<td>More Info...</td>
</tr>
</tbody>
</table>
Outline

✓ Motivation and Introduction
✓ Weak scaling study: The Joule Metric
✓ Scalability on exascale datasets
⇒ Final thoughts
Should more tools have been used?

- Could have performed this study with VisIt, ParaView, EnSight, etc.
- Successful test with VisIt validates pure parallelism.
- Of course, I/O is a big problem ... but ParaView, EnSight, etc, are doing the same `fread()`
Trends in I/O

- I/O dominates pure parallelism architectures
  - Usually > 50%
  - Sometimes 98%!
- Amount of data to visualize is typically $O(\text{total mem})$
- Relative I/O is key: $\frac{\text{Mem}}{\text{IO}}$
# Trends in I/O

<table>
<thead>
<tr>
<th>Machine</th>
<th>Year</th>
<th>$\frac{GB/s}{GFLOPS}$</th>
<th>Time to write memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASCI Red</td>
<td>1997</td>
<td>0.30%</td>
<td>300 sec</td>
</tr>
<tr>
<td>ASCI Blue Pacific</td>
<td>1998</td>
<td>0.15%</td>
<td>400 sec</td>
</tr>
<tr>
<td>ASCI White</td>
<td>2001</td>
<td>0.07%</td>
<td>660 sec</td>
</tr>
<tr>
<td>ASCI Red Storm</td>
<td>2004</td>
<td>0.14%</td>
<td>660 sec</td>
</tr>
<tr>
<td>ASCI Purple</td>
<td>2005</td>
<td>0.10%</td>
<td>500 sec</td>
</tr>
<tr>
<td>Jaguar XT4</td>
<td>2007</td>
<td>0.02%</td>
<td>1400 sec</td>
</tr>
<tr>
<td>Roadrunner</td>
<td>2008</td>
<td>0.01%</td>
<td>1600 sec</td>
</tr>
<tr>
<td>Jaguar XT5</td>
<td>2008</td>
<td>0.02%</td>
<td>1250 sec</td>
</tr>
<tr>
<td>Exascale</td>
<td>~2020</td>
<td>????</td>
<td>????</td>
</tr>
</tbody>
</table>
Why is relative I/O getting slower?

• “I/O doesn’t pay the bills”
  – And I/O is becoming a dominant cost in the overall supercomputer procurement.

• Simulation codes are less vulnerable
  – But will be more exposed with proposed future architectures.
Why is relative I/O getting slower?

• “I/O doesn’t pay the bills”
  – And I/O is becoming a dominant cost in the overall supercomputer procurement.

• Simulation codes are less vulnerable
  – But will be more exposed with proposed future architectures.

We MUST de-emphasize I/O in our visualization and analysis tools.
Conclusions

• Pure parallelism works, but is only as good as the underlying I/O infrastructure
  – I/O future looks grim
  – In situ, multi-resolution needed

• Full results available in special issue of Computer Graphics & Applications, special issue on Ultrascale Visualization
Questions ?