LA-UR-08-07337

*Approved for public release;
distribution is unlimited.*

*Title:*

# Petascale Visualization: Approaches and Initial Results

*Author(s):* James Ahrens, Li-Ta Lo, Boonthanome Nouanesengsy, John Patchett, Allen McPherson

*Intended for:* Invited Presentation, SOS13 - Sandia, ORNL and the Swiss (SOS) supercomputing meeting, March 2009

With the advent of the first petascale supercomputer, Los Alamos's Roadrunner, there is a pressing need to address how visualize petascale data. The crux of the petascale visualization performance problem is interactive rendering, since it the most computationally intensive portion of visualization process. At the terascale, commodity clusters with GPUs have been used for interactive rendering. At the petascale, visualization and rendering may be able to run efficiently on the supercomputer platform. In addition to Cell-based supercomputers, such as Roadrunner, we also evaluated rendering performance on multi-core CPU and GPU based processors. To achieve high-performance on multi-core processors, we tested with multi-core optimized ray-tracing engines for rendering. For real-world performance testing and to prepare for petascale visualization tasks we interfaced these rendering engines with vtk and ParaView. Initial results show that rendering software optimized for multi-core CPU and Cell processors provides competitive performance to GPU clusters, for the parallel rendering of massive data. The current architectural multi-core trend suggests multi-core based supercomputers are able to provide interactive visualization and rendering support now and in the future.

# Petascale Visualization: Approaches and Initial Results

James Ahrens
Visualization Team Lead

Ollie Lo, Boonthanome Nouanesengsy,
John Patchett, Allen McPherson
Los Alamos National Laboratory

Dave DeMarle
Kitware, Inc.

## Trends in Petascale Supercomputing

- Lots of compute cycles
  - Multi-core revolution
- Increasing latency from processor to memory, disk and network
  - Many memory-only simulation results
- Very expensive
- Can compute significantly more data than can be saved to disk
  - For example, on RR
    - To disk: 1 Gbyte/sec
    - Compute: 100 Gbytes on a triblade from Cells to Cell memory

## Questions

- 1. What data should be saved from the simulation?
- 2. What are our options for running our visualization software?
- Can we run our visualization software on the supercomputer?

## Can we **efficiently** run our visualization software on the supercomputer?

- The data understanding process is composed of a number of activities:
  - Analysis and statistics
  - Visualization
    - Map simulation data to a visual representation (i.e geometry)
  - Rendering
    - Map geometry to imagery on the screen

- Already runs on the supercomputer
  - Analysis, statistics and visualization

## Can we interactively **render** on the supercomputing platform?

- Fast rendering for interactive exploration
  - 5-10 fps minimum
  - 24-30 fps - HDTV
  - 60 fps - stereo
- Typically provided by commodity graphics in a visualization cluster

## Analysis of tradeoffs

**Rendering on the supercomputer**

Disadvantages

- Cost to port rendering to the supercomputing platform
- Allocate portion of supercomputer to analysis and visualization

Advantages

- Scalable to supercomputer size
- Access to "all" simulation results

**Rendering on visualization cluster**

Disadvantages

- Cost of cluster and infrastructure to connect it
- Less access to data – only data that is written to disk

Advantages

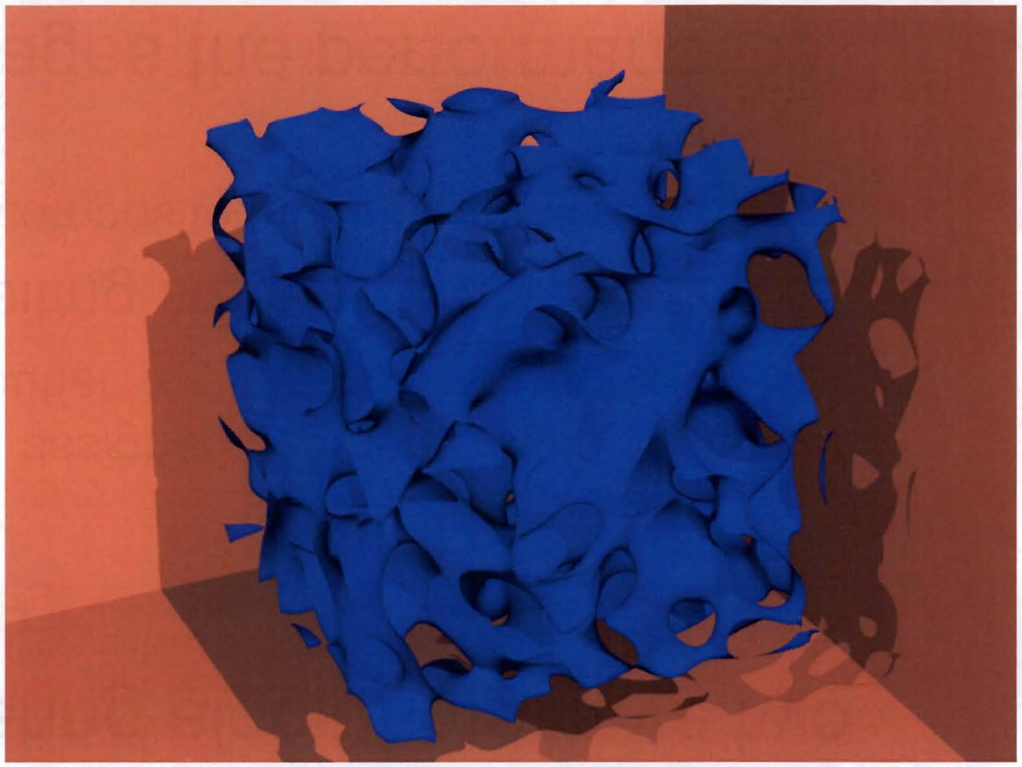
- Independent resource devoted to visualization task
- Very fast especially on smaller datasets

## Sort-last Parallel Rendering of Large Data

- Sort-last parallel rendering algorithms have two stages:
  - 1. Rendering stage
    - The processor renders its assigned geometry into a “distance/ depth” buffer and image buffer
  - 2. Networking / Compositing stage
    - These image buffers are composited together to create a complete result
- Given there are two stages the performance is limited by the slower stage
  - Assuming ***pipelining*** of the stages

## Types of Rendering

- Scan conversion of polygons
  - 1. OpenGL Software
    - Mesa - open-source
  - 2. OpenGL Hardware
    - Graphics cards – Nvidia
- Raytracing
  - Better physics model for the lighting equations
  - Fast multi-core ready implementations
  - 1. Manta Software
    - Multi-core, open-source (Univ. of Utah)
  - 2. iRT Software/Hardware
    - Cell processor



## Results - Incorporate rendering approaches into ParaView

- Paraview (PV) is open-source parallel large-data visualization tool
- 1. Run on two types of supercomputing nodes
  - Multi-core cluster - 1, 2, 4, 8, 16 way
  - Roadrunner - Cell processor
- 2. Run with scan-conversion and ray-tracing
  - PV already uses OpenGL

- Need to incorporate raytracing into PV/vtk
  - Rendering interface
    - Have raytracer implement rendering interface
      - Polygons, texturing, depth buffer
  - Then parallel rendering works as well!

# PV/vtk Rendering Performance

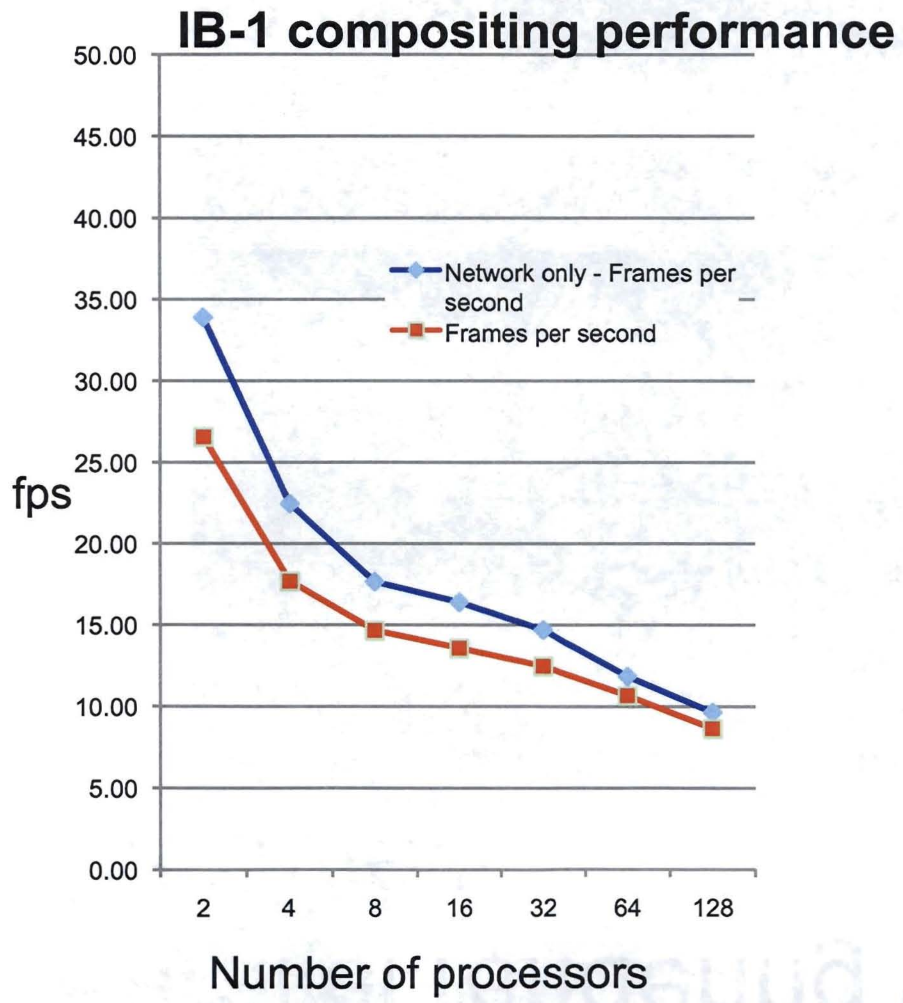
- 1 Million polygons rendering to a 1Kx1K image

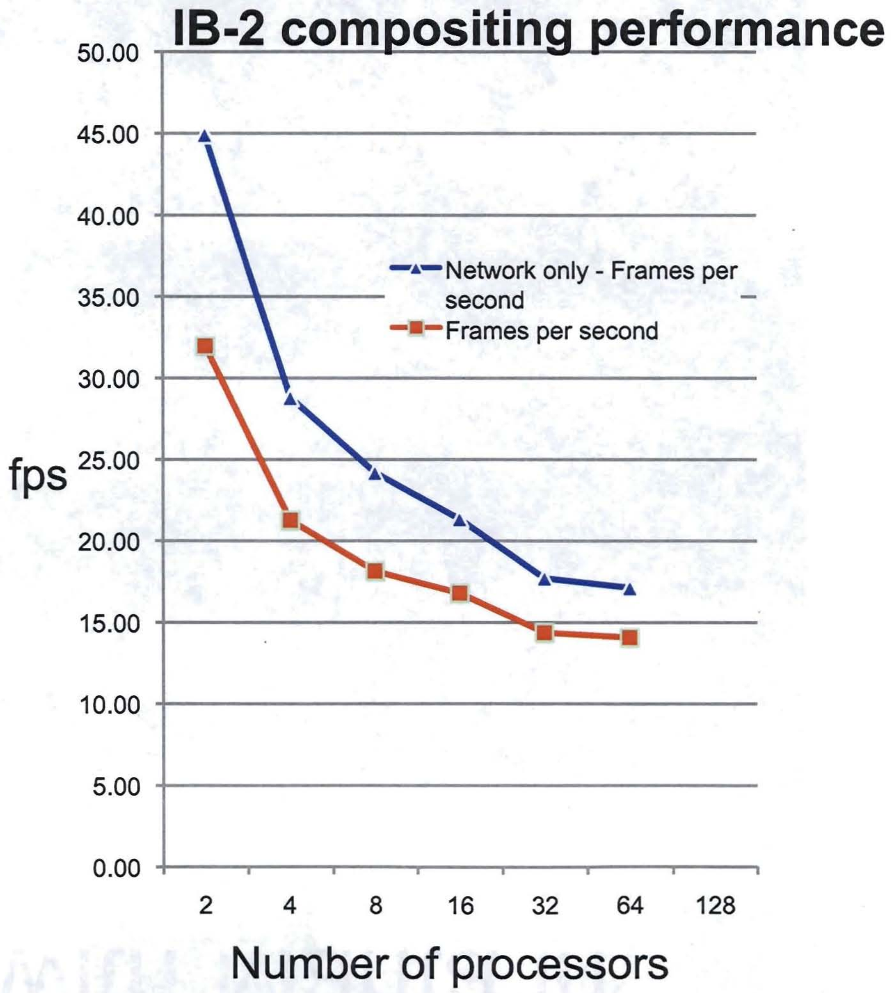
| Rendering Type | Software | Architecture | Frames per second |
|---|---|---|---|
| Scan conversion | OpenGL | Nvidia Quadro FX 5600 | 18.6 |
| Raytracing | iRT | Cell blade (16 SPUs) | 42 |

1. Vtk GPU hardware rendering performance could be improved.
2. iRT is not currently ported to run under PV/vtk.

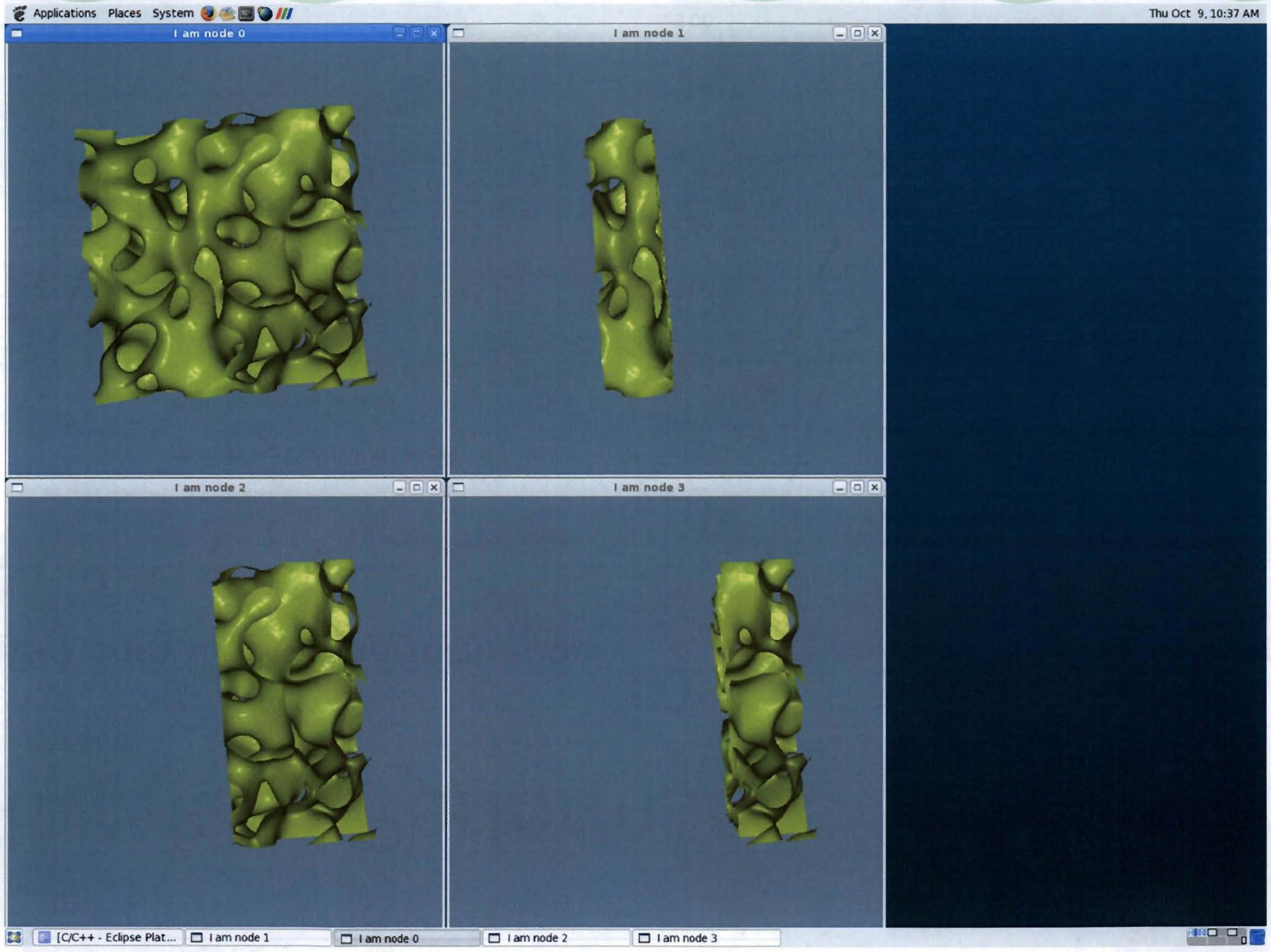
| | | | Frames per second for # of cores | | | | |
|---|---|---|---|---|---|---|---|
| Rendering Type | Software | Architecture | 1 | 2 | 4 | 8 | 16 |
| Scan conversion | Open GL Mesa | Multi-core (4 quad opt.) | 0.7 | 1.2 | 2.0 | 3.2 | 4.6 |
| Raytracing | Manta | Multi-core (4 quad opt.) | 1.6 | 2.8 | 5.6 | 10.9 | 19.4 |

## Networking Performance





## Parallel rendering with Manta in ParaView



## Future Work and Conclusions

- Integration of IBM Cell-based ray-tracer into PV for visualization on RR platform
- Advanced ray-tracing
- This preliminary study suggests that:
  - Multi-core processors are starting to serve some of roles of traditional GPUs such as parallel rendering
  - Using fast software-based rendering methods may offer a path to utilizing our supercomputers for visualization