Parallel Volume Rendering on the IBM Blue Gene/P

Tom Peterka
tpeterka@mcs.anl.gov

Radix Laboratory for Scalable Parallel System Software
Mathematics and Computer Science Division
Argonne National Laboratory
Intro: Wanted - € 50 Reward

Rob Ross, ANL

Kwan-Liu Ma, UCSD

Hongfeng Yu, UCSD

Tom Peterka, ANL
Intro: Supercomputer software rendering
- Déjà vu?

Bigger graphics clusters?
- Cost of scalability
- Power, cooling, space
- Not a rendering bottleneck

Availability
- Short runs available ad hoc
- Longer runs scheduled
- Dedicated resources

Cost effectiveness
- Short runs easy to schedule
- Backfill available cycles
- Policy decisions
Background: Big iron

Argonne National Labs
500TF BG/P (intrepid)

BUT
NO GRAPHICS!
Background: Data

- Supernovae shock wave
- Structured grid
- $864^3$
- 5 variables in netCDF
- Preprocess to extract single variable
- Time-varying, 200 time steps
- Each time step 2.5 GB
- Courtesy John Blondin, NC State, and Tony Mezzacappa, ORNL
Background: Parallel volume rendering

- MPI programming model
- Distributed memory

\[ t_{\text{frame}} = t_{\text{io}} + t_{\text{render}} + t_{\text{composite}} \]
Implementation: Data distribution & I/O

Process view: processes
- MPI_File_open(…)
- MPI_Type_create_subarray(…)
- MPI_Type_commit(…)
- MPI_File_set_view(…)
- MPI_File_read_all(…)

Dataset view: logical blocks

File view: bytes

P0

P1

Pn
Implementation: Ray casting

- Trilinear interpolation
- Front-to-back color, opacity accumulation
- Early ray termination
- Static data partition
- Each process completes subimage of its subvolume data

$$i = (1.0 - \alpha_{old}) \cdot i_{new} + i_{old}$$

$$\alpha = (1.0 - \alpha_{old}) \cdot \alpha_{new} + \alpha_{old}$$

where $i = \text{intensity}$, $\alpha = \text{opacity}$
Implementation: Direct-send compositing

- Load balanced
- Non-scheduled
- $O\left(n^{4/3} + n\right)$ where $n = \text{number of cores}$
- $O(m)$ where $m = \text{number of pixels}$
Results: Strong scaling

• End-to-end results, including file I/O
• Vis-only time .8 s
Results: Absolute time distribution
Results: Relative time distribution

864^3 data, 1600^2 image

% of frame time vs. Number of processors

- i/o
- render
- composite
Results: Efficiency

864^3 data, 1600^2 image

io eff.  render eff.  total eff.
Results: Improved efficiency

render efficiency $864^3$ data, $1600^2$ image

- default
- round robin
Displaying the image

- **compute node**
- **io node**
- **tree network**
- **TCP socket**
- **daemon**
- **login node**
- **64 nodes**
- **10 Gbps TCP**
Conclusions

Successes
- Visualization on leadership machines at ultrascale
- End-to-end performance is a delicate balance
- I/O matters (E2E time, not just rendering time)
- Combination of systems and visualization solutions

Current work
- Load balance
- Hiding I/O cost
- Improved image quality
- Larger data

Future work
- Less structured data
- Interaction
- Novel display environments
- Exploit multi-cores
- Other architectures
Questions and challenges

Technical
• Compositing
• Interactivity
• In situ visualization: implementation

Nontechnical
• Machine availability
• Machine utilization
• In situ visualization: collaboration
Parallel Volume Rendering on the IBM Blue Gene/P

Tom Peterka
tpeterka@mcs.anl.gov

Radix Laboratory for Scalable Parallel System Software
Mathematics and Computer Science Division
Argonne National Laboratory