TOD-Tree: Task-Overlapped Direct Send Tree Image Compositing for Hybrid MPI Parallelism

A.V. Pascal Grosset, Manasa Prasad, Cameron Christensen, Aaron Knoll & Charles Hansen
Distributed Volume Rendering

Volume Rendering on distributed memory systems has 3 stages:
1. Partitioning the data among the nodes & load the data
2. Forming an image from the data
3. Assemble the images (compositing)

Compositing has 2 main operations:
- Send/Receive image among nodes (communication)
- Blend images (computation)
Image Compositing

Why is it important?
- Compositing is the dominant cost when rendering on thousands of nodes
- In-situ visualization needs fast rendering

Has this not been done before?
- Yes, but HPC architectures have evolved:
  - There is a shift from MPI on each core to MPI on each node and threads inside a node
  - Computation is now cheap compared to communication
Aim: Introduce a new compositing algorithm for hybrid MPI Parallelism

Characteristics:
- Focus is on communication avoidance
- Decreased focus on equally distributing the load
- Asynchronous communication to mask communication latency
- Good coding practices
Outline

• Distributed Volume Rendering / Image Compositing
• Introduction
• Outline
• Related Works
• Algorithm
• Setup for Testing
• Profiling
• Results
• Conclusion
• Future Works
Related Works

Classification of Rendering
- Sort-First, Sort-Middle, Sort-Last (Molnar et al. 1994)

Compositing Algorithms
- Direct Send (Hsu 1993)
- Binary Swap (Ma et al. 1993) 2-3 Swap (Yu et al. 2008)
- Radix-k (Peterka et al. 2009)
- IceT (Moreland et al. 2011)

Hybrid Volume Rendering
- Hybrid parallelism for volume rendering on large-, multi-, and many-core systems (Howison et al. 2012)
Algorithm

Task-Overlapped Direct send Tree (TOD-Tree) has three Stages:

i. Direct Send
ii. K-ary Tree compositing
iii. Gather

Two parameters:

i. r (Direct Send)
ii. k (K-ary Tree)

Aim:

- Minimize communication
- Overlap communication with computation to mask communication delays
Algorithm: Direct Send (Stage 1)

Each node:
- Determine the nodes in its locality of size \( r \)
- Creates and advertises receiving buffer
- Do parallel Direct Send
  - Nodes in one locality are split into two sections and in a front to back or back to front order respectively
Algorithm: K-ary Compositing (Stage 2)

Each node:
• Determine if it is sending or receiving

Sending node:
• Sends its image to the receiving node

Receiving node:
• Creates buffer and advertises
• Blend images
• If at the last stage:
  • Blend in an “opaque buffer”
Algorithm: Gather (Stage 3)

Display node:
- Receive from other images

Other nodes:
- Nodes that have images send their data to the display node
Setup for Testing

Datasets:
• Artificial and Combustion
• Images: 2K, 4K & 8K

Algorithms (in IceT):
• Binary-Swap
• Radix-k

Platforms:
• Edison @ NERSC (tested for up to 4096 nodes)
  • Cray X30 supercomputer
  • Intel IvyBridge processor with 24 cores | 460.8 GFLOPS/node

• Stampede @ TACC (tested for up to 1024 nodes)
  • Infiniband FDR network
  • Intel SandyBridge processor with 20 cores | 346 GFLOPS/node
Profiling

Time breakdown:
- Communication (blue and green)
- Computation (red)
- Setting up buffers (yellow)

Profiling for 64 nodes on Edison
Results: Stampede

- TOD-Tree & IceT on Stampede uses -O3 optimization
- TOD-Tree’s performance is quite stable as the number of nodes increase
- Staircase appearance is because same value of r is used for pair of nodes
Results: Edison

Code on Edison uses OpenMP & -O3 optimization

Observations:

- TOD-Tree’s performance is quite stable as the number of nodes increase.
- Sawtooth appearance is because same value of \( r \) is used for pair of nodes (32 & 64 \( r = 16 \), 128 & 256 \( r = 32 \), 512 & 1024 \( r = 64 \), 2048 & 4096 \( r = 128 \)).
- Radix-K is faster for 8K images as we do not use any compression.
  - 2K image: 64 MB
  - 4K image: 256 MB
  - 8K image: 1GB
Results: Stampede vs Edison

Observations:

• Edison is much faster than Stampede at low node counts
  • Fully utilizing OpenMP and SIMD is very important when workload is high

• As the number of nodes increase, their performance are quite similar
  • When the workload per node decreases, communication becomes dominant

• TOD-Tree is faster than the IceT algorithms for the combustion data
  • We suffer less from load imbalance
Conclusion

• We have introduced a new compositing algorithm for Hybrid OpenMP/MPI Parallelism. TOD-Tree:
  a. focuses on communication avoidance instead of balancing the workload
  b. generally performs better than Radix-k on hybrid parallelism environment

• Also, this work shows that some existing algorithms might need to be redesigned to better exploit new hardware
Future Works

• Compression for large image sizes

• Investigate performance on upcoming Intel Knights Landing

• Additional Scaling tests
  • Extend testing to Blue Gene/Q
  • Extend testing to more nodes on Infiniband systems
  • Extend to GPUs
Thank you

Any Questions?