Revisiting Large-Scale Parallel Image-Compositing for Sort-Last Rendering

Tom Peterka
tpeterka@mcs.anl.gov
Mathematics and Computer Science Division

MPE profile of binary swap
MPE profile of Radix-k

DOE CGF Research I Session
Definition of Image Compositing

The final stage in sort-last parallel visualization algorithms:
1. Partition data among processes
2. Visualize local data
3. Composite resulting images into one

Composition = communication + computation

The computation is usually an alpha-blend called “Over”

\[ i = (1.0 - \alpha_{old}) \times i_{new} + i_{old} \]

\[ \alpha = (1.0 - \alpha_{old}) \times \alpha_{new} + \alpha_{old} \]

where \( i \) = intensity (R,G,B), \( \alpha \) = opacity

[Porter & Duff, Compositing Digital Images, 1984]
Background: Baseline Performance

Performance of binary swap and MPI collectives for 2 Mpixel image. Binary swap performs 3X faster than reduce-scatter.

No need to gather at one node; output image can be written using collective I/O in parallel.
Direct-send compositing time improved up to 30X. \(1120^3\) data volume, \(1600^2\) image size.

Usually in direct-send, \(n = m\), but setting \(m < n\) can reduce contention when \(n\) is large. On average, \(O(m \times n^{1/3})\) total messages, can get down to \(O(n)\) if \(m = n^{2/3}\).

End-to-End Study of Parallel Volume Rendering on the IBM Blue Gene/P. Peterka et al., ICPP'09
Direct-Send and Binary Swap Operation

Direct-send: maximum parallelism but high number of small messages results in network contention, all messages in one round, non-power-of-two processes ok

Binary swap: fewer messages per round, $\log_2 p$ rounds, $p =$ number of processes, power of 2

[Hsu, Segmented Ray Casting for DataParallel Volume Rendering, 1993]

[Ma et al., Parallel Volume Rendering Using Binary-Swap Compositing, 1994]
Radix-k Operation

Radix-k: More parallel, managed contention, p does not need to be power of 2

A Configurable Algorithm for Parallel Image-Compositing Applications. Peterka et al., SC09
Keys to Success: Increase message concurrency, avoid contention, overlap communication with computation

By being configurable to any architecture

- More participants per group than binary swap ($k > 2$)
- Manage contention by limiting $k$ value ($k < p$)
- Overlap communication with computation (nonblocking communication and careful order of operations)
- Can never do worse than binary swap or direct-send
- No penalty for non-powers-of-two numbers of processes
Radix-k Performance on Blue Gene/P Intrepid

Radix-k improves 40% over binary swap at non-powers-of-two process counts. Left: $p$ varies from 32 to 1024 in steps of 32. Right: $p$ continues from 1024 to 35,000 in steps of 1024.
Optimizations

- Start with efficient implementation of bounding boxes and RLE compression
- Benchmark target k-values for various machines
- Test in a parallel volume renderer

### Intrepid Target k-values

<table>
<thead>
<tr>
<th>P (Mpix)</th>
<th>4 Mpix</th>
<th>8 Mpix</th>
<th>16 Mpix</th>
<th>32 Mpix</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>64</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>128</td>
<td>64</td>
<td>128</td>
<td>128</td>
<td>128</td>
</tr>
<tr>
<td>256</td>
<td>64</td>
<td>128</td>
<td>128</td>
<td>128</td>
</tr>
<tr>
<td>512</td>
<td>64</td>
<td>128</td>
<td>128</td>
<td>128</td>
</tr>
<tr>
<td>1 K</td>
<td>64</td>
<td>32</td>
<td>128</td>
<td>128</td>
</tr>
<tr>
<td>2 K</td>
<td>32</td>
<td>32</td>
<td>128</td>
<td>128</td>
</tr>
<tr>
<td>4 K</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>8 K</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>16 K</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>32 K</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
</tr>
</tbody>
</table>

### Overlap

Overlap = \( \frac{\text{Computation}}{(\text{CommWait} + \text{Computation})} \)

[0.0 (bad) – 1.0 (good)]

**Communication**

<table>
<thead>
<tr>
<th>CommWait time</th>
<th>Computation time</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 Mpix</td>
<td>8 Mpix</td>
</tr>
<tr>
<td>16 Mpix</td>
<td>16 Mpix</td>
</tr>
<tr>
<td>32 Mpix</td>
<td>32 Mpix</td>
</tr>
<tr>
<td>64 Mpix</td>
<td>64 Mpix</td>
</tr>
<tr>
<td>128 Mpix</td>
<td>128 Mpix</td>
</tr>
<tr>
<td>256 Mpix</td>
<td>256 Mpix</td>
</tr>
<tr>
<td>512 Mpix</td>
<td>512 Mpix</td>
</tr>
<tr>
<td>1 K Mpix</td>
<td>1 K Mpix</td>
</tr>
<tr>
<td>2 K Mpix</td>
<td>2 K Mpix</td>
</tr>
<tr>
<td>4 K Mpix</td>
<td>4 K Mpix</td>
</tr>
<tr>
<td>8 K Mpix</td>
<td>8 K Mpix</td>
</tr>
<tr>
<td>16 K Mpix</td>
<td>16 K Mpix</td>
</tr>
<tr>
<td>32 K Mpix</td>
<td>32 K Mpix</td>
</tr>
</tbody>
</table>

**Machine** | **Overlap** | **Range of k-values for 32 Mpix**
--- | --- | ---
Intrepid | 0.58 | 8 to 128
Jaguar | 0.22 | 4 to 64
Lens | 0.29 | 8 to 64
Eureka | 0.10 | 8 to 32

Accelerating and Benchmarking Radix-k Image Compositing at Large-Scale. Kendall et al., EGPGV’10
Large-Scale Results

3X – 6X improvement over optimized binary swap (with bounding boxes and RLE) in many cases. 64Mpix at 32K processes can be composited at .08 s, or 12.5 fps.
Recap

Contributions

- Unifies direct-send, binary swap and points between
- Configurable to architecture
- Benefits from optimizations more than binary swap

Ongoing and future work

- Load balancing
- Polygon compositing
- Implement and benchmark in IceT
Revisiting Large-Scale Parallel Image-Compositing for Sort-Last Rendering

Thank you

Acknowledgments:

Facilities
Argonne Leadership Computing Facility (ALCF)
Oak Ridge National Center for Computational Sciences (NCCS)

Funding
US DOE SciDAC UltraVis Institute

People
Rob Ross, Han-Wei Shen, Jian Huang, Wes Kendall, Dave Goodell, Rajeev Thakur, John Blondin

Tom Peterka
tpeterka@mcs.anl.gov
Mathematics and Computer Science Division