A Pipeline-based Sort-last Parallel Visualization Architecture

Dongliang Chu and Chase Wu
Department of Computer Science
New Jersey Institute Of Technology
Newark, NJ, 07102, USA

The 3rd International Workshop on High Performance Computing for Big Data
(HPC4BD-2016)
co-located with
The 45th International Conference on Parallel Processing
(ICPP-2016)
August 16, 2016
Agenda

Motivation and Problem Description
Existing Work
Proposed Work
Conclusion and Future Work
Motivation

• Visualization of big data in modern scientific applications
Problem Description

• Sort-last parallel visualization

Sample pictures from tutorial of ParaView.
Existing Work

- Intermediate image rendering on each rendering unit
- Image composition on each compositing unit
- Over-based image composition for same positioned fragments
Proposed Work: V4BD
Very-high-speed Value-added Volume Visualization of Big Data

Real-time over-based image composition at the same fragment position

Intermediate image rendered tile-by-tile on each rendering unit
Tile-oriented composition on each compositing unit
Real-Time *Over* Operator

- Ubiquitous usage of *over* operator in modern graphics world

\[ P_{1,2} = P_1 \overline{\text{over}} P_2 \]
\[ C_{1,2} = C_1 + C_2 g(1 - \alpha_1) \]
Real-Time Over Operator

- Traditional over operator

\[
P'_1 \over P'_2 \over \mathbf{L} \over P'_n
\]

\[
P'_1 \over P'_2 \over \mathbf{L} \over P'_n
\]

- Proposed Over Operator

\[
P'_{RTO} P'_2 RTO P'_3 \over \mathbf{L} RTO P'_n
\]
Real-Time *Over* Operator

- Generalized *over* operator

\[
P_{1,n} = P_1 \text{ over } P_2 \text{ over } L \text{ over } P_n
\]

\[
P_{1,n} = [c_{R_{1,n}}, c_{G_{1,n}}, c_{B_{1,n}}, \alpha_{1,n}]^T
\]

\[
\alpha_{1,n} = \sum_{\text{for any } I \subseteq \{\alpha_1, \alpha_2, \ldots, \alpha_n\}, I \neq \emptyset} (-1)^{|I|+1} g\left(\prod_{\alpha_i \in I} \alpha_i \right)
\]

\[
\alpha_{1,n} = 1 - \prod_{j=1}^{n} (1 - \alpha_j)
\]

\[
c_{1,n} = \sum_{i=1}^{n} \left( \left( \prod_{1 \leq j < i} (1 - \alpha_j) \right) \cdot \alpha_i \cdot C_i \cdot 1 \right)
\]

\[
c_{1,n} = [c_{R_{1,n}}, c_{G_{1,n}}, c_{B_{1,n}}]^T \quad C_i = [c_{R_i}, c_{G_i}, c_{B_i}]^T
\]
Real-Time *Over* Operator

- Fully parallelized *over* operator

\[
\begin{align*}
    c_{1,n} &= \sum_{i=1}^{n} \left( \left( \prod_{1 \leq j < i} (1 - \alpha_j) \right) \cdot \alpha_i \cdot C_i \cdot 1 \right) \\
    \alpha_{1,n} &= 1 - \prod_{j=1}^{n} (1 - \alpha_j)
\end{align*}
\]
Real-Time *Over* Operator

- Experimental results

(a) arriving interval 0s for image size of $2048^2$.

(b) arriving interval 0.1s for image size of $2048^2$.

(c) arriving interval 0s for image size of $3072^2$.

(d) arriving interval 0.1s for image size of $3072^2$. 
Raycasting for Volume Rendering

- Algorithm animations
Parallelized Per-Ray Integration

Sequential Method

\[ S_1 \oplus S_2 \oplus S_3 \oplus L \oplus S_8 \]

Proposed Method

\[ S_1 \oplus S_2 \oplus S_3 \oplus S_4 \oplus S_5 \oplus S_6 \oplus S_7 \oplus S_8 \]

\[ S_{1,2} \oplus S_{3,4} \oplus S_{5,6} \oplus S_{7,8} \]

\[ S_{1,4} \oplus S_{5,8} \]

\[ S_{1,8} \]
Parallelized Per-Ray Integration

• Sampling points determination

\[ S_i = R_{\text{Start}} + igS\text{ampInterval} \]

\[ 0 \leq i \leq n, \quad n = \frac{R_{\text{End}} - R_{\text{Start}}}{\text{SampInterval}} \]
Parallelized Per-Ray Integration

• Sampling points integration

Law of association

\[(S_1 \oplus S_2) \oplus S_3 = S_1 \oplus (S_2 \oplus S_3)\]

\[S_1 \oplus S_2 \oplus \cdots \oplus S_i \oplus \cdots \oplus S_n\]

\[= S_{1,2} \oplus S_{3,4} \oplus \cdots \oplus S_{i,i+1} \oplus \cdots \oplus S_{n-1+n\%2,n}\]

\[= S_{1,4} \oplus \cdots \oplus S_{i,i+3} \oplus \cdots \oplus S_{n-3+(n\%4),n}\]

\[= \cdot \cdot \cdot\]

\[= S_{1,\lfloor \frac{n}{2} \rfloor} \oplus S_{\lfloor \frac{n}{2} \rfloor +1,n}\]
Parallelized Per-Ray Integration

• Experimental results
Pipelined Sort-last parallel visualization

• Related work

Illustration of the existing pipelines, where “RS” represents a rendering step, “TS1” represents a data transfer from a rendering unit to a composition unit, “CS” represents a composition step, “TS2” represents a data transfer from a composition unit to a display unit, and “DS” represents a display step.
Pipelined Sort-last Parallel Visualization

for all $r_i$ ($0 \leq i \leq n-1$) simultaneously do
  for step $k$ ($0 \leq k \leq n-1$)
    render and send tile $t_{(i+k)\%n}$ to $c_{(i+k)\%n}$

for all $c_j$ ($0 \leq j \leq n-1$) simultaneously do
  for step $l$ ($0 \leq l \leq n-1$)
    receive from $r_{(j+n-l)\%n}$ and composite tile $t_j$
Pipelined Sort-last Parallel Visualization

- Comparison with existing pipeline

“RS” represents a rendering step, “TS1” represents a data transfer from a rendering unit to a composition unit, “CS” represents a composition step, “TS2” represents a data transfer from a composition unit to a display unit, and “DS” represents a display step.
V4BD Architecture

- Experimental results of validity test
V4BD Architecture
• Experimental results of performance test

Comparison of the proposed and existing visualization pipeline with 8, 16, and 32 rendering/composition nodes, where “Traditional” represents the traditional sort-last pipeline, “Pipeline+Over” represents the tile-based pipelining structure using only the proposed “over” operator, and “V4BD” represents the tile-based pipelining structure using the proposed rendering and composition methods.
Conclusions

Sort-last parallel visualization architecture – V4BD

- Parallelized per-ray integration method
- Real-time over operator
- Pipelined visualization process
- Correlations between three component techniques
Future Work

• Explore more composition methods
• Optimize memory usage
• Integrate into IceT library to test performance improvement