Scalable Parallel Out-of-core Terrain Rendering

Prashant Goswami     Maxim Makhinya     Jonas Boesch
Renato Pajarola
Organization of Talk

- Introduction
- Related Work
- Preview to Terrain Renderer
- Parallelizing Terrain Renderer
- Results
- Conclusion
• Introduction
• Related Work
• Preview to Terrain Renderer
• Parallelizing Terrain Renderer
• Results
• Conclusion
Introduction

• Interactive visualization of huge terrain data
• Advancement of hardware (CPU/GPU)
• Increasing precision of data acquisition
• Level-of-detail (LOD) based solutions
  - GPU-oriented
• Parallel rendering solutions
• Parallel + LOD rendering?
LOD - Single Machine
Data Size : 32 k X 32 k
Pixel Error : 6
LOD - Single Machine

Pixel Error : 4
LOD - Single Machine

Pixel Error : 2
LOD - Single Machine

Pixel Error : 1
Introduction

- Parallelizing LOD based algorithms
  - Division strategy
  - Choice of algorithm
  - Performance comparison
• Introduction
• Related Work
• Preview to Terrain Renderer
• Parallelizing Terrain Renderer
• Results
• Conclusion
Related Work

- LOD based terrain rendering
  - Pajarola[PG07]

- Realistic terrain images rendering in parallel
  - Vezina[VR91], Agranov[AG95]
  - Not interactive, cant handle large datasets

- Rendering on PC cluster
  - Yin[YJSZ06]
Related Work

- Shared resources from community
  - Johnson [JLMVK06]
- Remote visualization parallel streaming
  - Hu [HTMS07]
- What we address?
  - Parallel task decomposition strategies
  - Comparative analysis of performance
• Introduction
• Related Work
• Preview to Terrain Renderer
• Parallelizing Terrain Renderer
• Results
• Conclusion
RASTeR : Preview

- RASTeR uses two units [BGP09]:
  - K-Patches : Triangulation unit
  - M-Blocks : Data unit
RASTeR: Preview

- K-Patch
  - One of 8 isosceles right triangles
  - K vertices along each edge
  - Triangles within K-Patch arranged as a triangle strip
  - Macro triangles of bintree

RASTeR : Preview

- M-Block
  - Square block of terrain height data
  - All M-Blocks have same size $M \times M$
  - $M = 2^m + 1$
  - Forms quadtree hierarchy
RASTeR : Preview

K-Patch and M-Block relation
RASTeR: Preview

- View dependent saturated error metric
- Error per K-Patch
- Textures for M-Blocks
- Asynchronous fetching for M-Blocks and their textures
Why RASTeR?

• GPU-oriented efficient rendering
• Asynchronous fetching
• Complete disentanglement of
  - Height data
  - Triangulation data
• Easy to parallelize
Parallel Terrain Rendering

- Parallelized using **Equalizer**[EMP09]
  - Framework for parallel rendering
  - Driven by Server-Client approach
RASTeR on Equalizer

- Each machine runs an independent application
- RASTeR modified to obtain from Equalizer
  - Task division parameters
    - Frustum
    - Database range
- Mouse, keyboard controls, pixel error
  - Same across all nodes
RASTeR on Equalizer

- RASTeR is multithreaded
  - OpenGL context handling via Equalizer
- Task division managed by Equalizer server
Task Division Modes

- Sort-Last / Database Decomposition
- Sort-First / Screen Decomposition
Task Division Modes

- **sort-last**
  - Graphics
  - Database
  - Geometry processing
  - Fragment processing
  - Sort fragments (depth visibility)
  - Display

- **sort-middle**
  - Graphics
  - Database
  - Sort screen-space primitives
  - Display

- **sort-first**
  - Graphics
  - Database
  - Bucketization (sort)
  - Display

Diagram shows the flow of tasks for different division modes, including sorting, geometry processing, and fragment processing.
Task Division Modes

• Sort-Last / Database Decomposition
• Sort-First / Screen Decomposition
Optimal Parallelization

- Requires
  - Task is almost equally divided among rendering machines
  - Per-frame inter communication between machines is kept minimal
Sort-Last Decomposition

• Given
  - N machines
  - A range between $R = [0, 1]$ to each of them
    ($R_i = [i/N, (i+1)/N]$)

• Divide the visible rendering data as equally as possible in database domain

• All machines obtain same frustum from Equalizer
Linear Enumeration

- Each machine in parallel:
  - Gets range $R_i = [l, r]$
  - Traverses bintree
  - Selects $K$-Patches with $M$-Blocks having origin $O_M$

$$l^*X_{\max} \leq O_M(x) \leq r^*X_{\max}$$

($X_{\max} = \text{Max X coordinate}$)
Linear Enumeration
Linear Enumeration

- **Drawbacks**
  - Traversal coherence lost
    - Division follows M-Blocks, not K-Patches
    - Susceptible to changes on translation and rotation
  - Data distribution among machines unequal
Quadtree Enumeration

- Enumerate nodes of quadtree
- Assign intervals \([L,R]\) to each node in bottom-up fashion
  - Range of a node covers that of all its descendants
- Given range \(R_i = [l, r]\), select M-Blocks
  \(l \times n_{\text{max}} \leq [L, R] \leq r \times n_{\text{max}}\)
  
  \(n_{\text{max}}\): maximum number of leaf nodes
Quadtree Enumeration
Quadtree Enumeration

• Evaluation:
  - More coherence in tree traversal
  - Less susceptible to changes upon rotation, translation
  - More uniformly distributed data
  - Can’t ensure that each machine gets similar amount of rendering data
Optimal Task Distribution

- In both Linear Enumeration and Quadtree Enumeration
  - Machine with no data visible will be idle
Active K-Patch Enumeration

- This is where RASTeR helps!
  - Each K-Patch has the same number of triangles
- K-Patch list for a given frame similar on all machines
- Post K-Patch list selection, divide them equally among all machines
- Range $R_i$ used for mapping
  - No inter-machine communication needed
Active K-Patch Enumeration

K-Patch list

<table>
<thead>
<tr>
<th>l=0.0</th>
<th>l=0.25</th>
<th>l=0.5</th>
<th>l=0.75</th>
</tr>
</thead>
<tbody>
<tr>
<td>r=0.25</td>
<td>r=0.5</td>
<td>r=0.75</td>
<td>r=1.0</td>
</tr>
</tbody>
</table>

Meta bintree

a b c d e f g h
Active K-Patch Enumeration
Active K-Patch Enumeration

• Evaluation:
  - Coherence in tree traversal
  - Less susceptible to changes upon rotation, translation
  - Uniformly distributed data
  - Ensures that each machine gets similar amount of rendering data
  ‣ Automatic load balancing
Task Division Modes

- Sort-Last / Database Decomposition
- Sort-First / Screen Decomposition
Sort-First Decomposition

• Task division in screen space
• Each machine updates its frustum to the one it obtains from Equalizer server
• Different machines render mutually exclusive part of terrain
  - Final image assembly does not require z-depth or alpha-compositing
• Load balancing through Equalizer server
Sort-First Decomposition
• Introduction
• Related Work
• Preview to Terrain Renderer
• Parallelizing Terrain Renderer
• Results
• Conclusion
Results

- Equalizer and RASTeR : C++ / GLSL
- 10 Linux Machines in Cluster:
  - 2 Gbit/s Myrinet for Image Compositing
  - 1 Gbit/s network for data-retrieval
  - Dual 2.2 GHz AMD Opteron CPU
  - 4 GB RAM
  - GeForce 9800 GX2 graphics
Results - DB Decomposition

- Linear Block and Quadtree Enumeration
  - Need load balancing from Equalizer server
  - Do not provide scalable sort-last rendering
- Active K-Patch enumeration
  - Provides automatic load-balancing
  - Performance scales with # of machines
Results - 2D Decomposition

- 2 kinds of 2D decomposition
  - Vertical Tiling
    ▶ Uneven distribution of data per tile
  - Horizontal Tiling
    ▶ More even distribution of data per tile
Puget Sound: 16k X 16k

**Turn; 16k; 1280x1024; 2Gbit/s**
- 2D - Rendering
- DB - Rendering
- DB
- Linear

**Zoom; 16k; 1280x1024; 2Gbit/s**
- 2D - Rendering
- DB - Rendering
- DB
- Linear
SRTM : 32 k X 32 k
Performance Analysis

- Pure rendering scales at least linearly
- Pure rendering scales better in Sort-Last
Performance Analysis

• Overall rendering performance also depends on compositing
  - Reading partial images
  - Transmission over network
  - Assembling at destination machine
• Sort-last transmits twice the amount of data per frame than sort-first
Results - 2D Decomposition

Display Wall Configuration
Conclusion

• We have presented:
  - Parallel solution for real-time multi-resolution out-of-core terrain visualization
  - Efficient LOD based adaptive solution for automatic load balancing

• We have addressed:
  - Challenges in distributed environment
Thank You!