Real-Time Graphics Architecture

Kurt Akeley
Pat Hanrahan

http://www.graphics.stanford.edu/courses/cs448a-01-fall

Parallelism and Communication

with help from
Matthew Eldridge
Topics

1. Why scalability?
2. Types of parallelism
3. Communication patterns and requirements
4. Sorting classification for parallel rendering
5. Case studies
   - Sort-middle interleaved: SGI
   - Sort-middle tiled: Pixel-Planes 5
   - Sort-last image composition: PixelFlow
   - Sort-last fragment: Denali
   - Sort-first: Chromium

Readings

Required
1. S. Molnar, M. Cox, D. Ellsworth, H. Fuchs, A sorting classification of parallel rendering
2. Fuchs et al., A heterogenous multiprocessor graphics system using processor-enhanced memories (PP5).
3. Eyles et al., PixelFlow: The Realization

Recommended
1. F. I. Parke, Simulation and expected performance analysis of multiple processor z-buffer systems
2. H. Fuchs, Distributing a visible surface algorithm over multiple processors
Parallelism and Communication

Parallelism
- Design a single component (either a single stage or a complete graphics pipeline) and replicate it to increase performance

Communication
- Connects components, allowing parallel work to be load balanced

Dependencies and ordering

Scaling Performance

Application
  ▼ Command
  ▼ Geometry
  ▼ Rasterization
  ▼ Texture
  ▼ Fragment
  ▼ Display

Input Rate
  ▼ Triangle Rate
  ▼ Fill Rate
  ▼ Texture Memory
  ▼ Display Resolution
Sources of Parallelism

Task parallelism
- Graphics pipeline

Data parallelism
- Frame-parallel
- Image-parallel
- Object(Geometry)-parallel

Geometry Parallelism (SGI)

![Bar chart showing transform length and width for different models.](chart.png)
Raster/Fragment Processors (SGI)

Communication Requirements

0.880 GB/s
Application

Rough estimate

20 Mvert/s
Command

Vertex 5 Gops

Geometry

Fragment 150 Gops

Rasterization

Texture Memory 4 GB/s

1000 Mpix/s
Texture

Framebuffer 16 GB/s

120 Mpix/s
Display

0.36 GB/s
Ordering

Data dependencies
- Graphics state
-Framebuffer operations
  - Painter’s algorithm
- Write to texture
- Render/Copy/Render
- Readback

```c
glBegin(GL_POLYGON);
glColor(RED);
glVertex3i(0,0,0);
glColor(BLUE);
glVertex3i(1,0,0);
glColor(BLUE);
glVertex3i(0,1,0);
glEnd()
```

Communication Taxonomy

- **Sorting:** Object → Image
  - I. E. Sutherland, R. F. Sproull, R. A. Schumacher, A characterization of ten hidden surface algorithms
  - Classified by order of x, y, z radix sorts
- **Distribution:** Object → Object
- **Routing:** Image → Image
- **Texturing:** Image → Texture
Sorting Taxonomy

Application
  ↓
Command
  ↓
Geometry
  ↓
Rasterization
  ↓
Texture
  ↓
Fragment
  ↓
Display

→ Sort-First

← Sort-Middle

↑ Sort-Last Fragment

← Sort-Last Image Composition

Sort-Middle
**Image-Space Work Distribution**

Parke - Tiled

Fuchs - Interleaved

**Sort-Middle Interleaved**

Geometry work load-balanced, except clipping and tessellation

Broadcast communication does not scale, but supports ordering

Finely interleaved screen tiling insures excellent load balance

*SGI Graphics Workstations: RealityEngine, InfiniteReality*
SGI RealityEngine

Sort-Middle Tiled

Point-to-point communication scales
Coarse tiling incurs load imbalance

LINC PixelPlanes, Stanford Argus
UNC Pixel-Planes4 (1986)

http://www.cs.unc.edu/~pxpl/

Logic-enhanced memory
Tree of 1-bit adders
Compute linear expression
Ax+By+C
Cycle:
Write (A,B,C) to “memory”
Writes 512x512 cells
SIMD compute surface
Efficiency?

Footprint processors

Footprint processors

Area(tri):Area(bbox):
On average ~ 1/6
Average number of pixels inside a tile?
Pixel-Planes 5 Rendering Algorithm

1. Sort primitives into tiles
2. Renderers request tiles from central server
   - Rasterize triangles in tile
   - Send completed tile to FB

Comments
   - Excellent dynamic load balancing
   - Extra pass adds one frame of latency
   - Must demarcate end-of-frame
   - State management difficult in 2-pass algorithms
     e.g. copy to texture in the middle of rendering
   - No longer a stream processors (intermediate memory)
The Overlap Factor

Molnar-Eyles Formula

3 cases

\[ 4 \times \frac{4(w/2)(h/2)}{WH} \]

\[ 2 \times \frac{2(w/2)(H-h) + 2(h/2)(W-w)}{WH} \]

\[ 1 \times \frac{(W-w)(H-h)}{WH} \]

Total \[ O = \left( \frac{H+h}{H} \right) \left( \frac{W+w}{W} \right) \]
**Rasterization Cost**

- **Large tiles**
  - Few tasks, greater variation in work
  - → bad load balance

- **Medium tiles**
  - More tasks, low overlap
  - → good load balance

- **Small tiles**
  - High overlap/more redundancy
  - → best load balance but redundant work

**Sort-Last**
**Z-Composition**

Other combiners possible

**Sort-Last Image Composition**

Exposes rasterization load imbalance to application

Point-to-point ring interconnect scales

Two-stage image composition loses ordering
UNC Pixel Flow

From J. Poulton, J. Eyles, S. Molnar, H. Fuchs,
Pixel Flow: The Realization

CS448 Lecture 9
Kurt Akeley, Pat Hanrahan, Fall 2001
Sort-Last Fragment

- Improved texture locality
- No redundant work in FG
- Exposes rasterization load imbalance to application
- Point-to-point communication scales, but requires more bw
- Finely interleaved screen tiling insures excellent load balance
- Possible, but difficult, to maintain ordering

Kubota Denali, E&S Freedom 3000

Kubota Denali (1993)

Denali Technical Overview 1.0
Kubota Pacific Computer, 1993
Sort-First

Princeton Display Wall, Stanford WireGL

CS448 Lecture 9  Kurt Akeley, Pat Hanrahan, Fall 2001
Sort-First

Ring Parallelism

3DLABs
Sort-Everywhere: Pomegranate

Architecture Comparison

<table>
<thead>
<tr>
<th>Rasterization balanced</th>
<th>Sort-First</th>
<th>Sort-Middle Tiled</th>
<th>Sort-Middle Interleaved</th>
<th>Sort-Last Fragment</th>
<th>Sort-Last Image Comp.</th>
<th>Pomegranate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>❌</td>
<td>❌</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Scalable communication</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>❌</td>
<td>✓</td>
<td>✔</td>
</tr>
<tr>
<td>Temporal load balance</td>
<td>❌</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Ordered</td>
<td>❌</td>
<td>❌</td>
<td>✓</td>
<td>✓</td>
<td>❌</td>
<td>❌</td>
</tr>
</tbody>
</table>
Trends and Predictions

Task- vs. Data-parallelism
  Full cycle? Data parallel better long-term?
  Billion transistor chips

Limits of scalability
  Small to medium scale systems perform well, large?
  Parallel API necessary to get over host bottleneck

Texture “sorting”
  Texture communication is now dominant
  Texture-centric architectures?