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

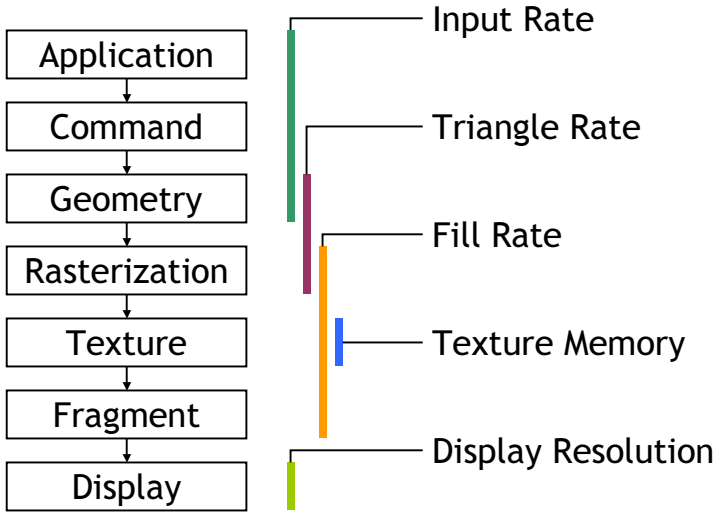
- Design a single component (either a single stage or a complete graphics pipeline) and replicate it to increase performance

Communication

- Connects components, allowing parallel work to be load balanced

Dependencies and ordering

Scaling Performance



Sources of Parallelism

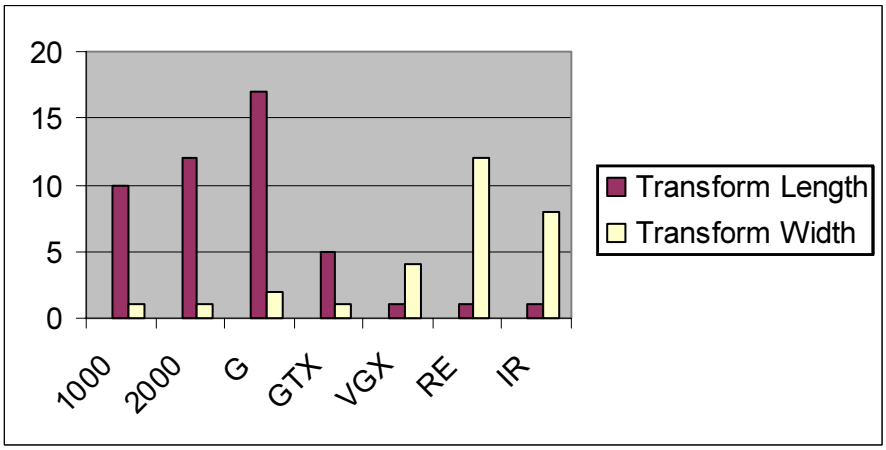
Task parallelism

- Graphics pipeline

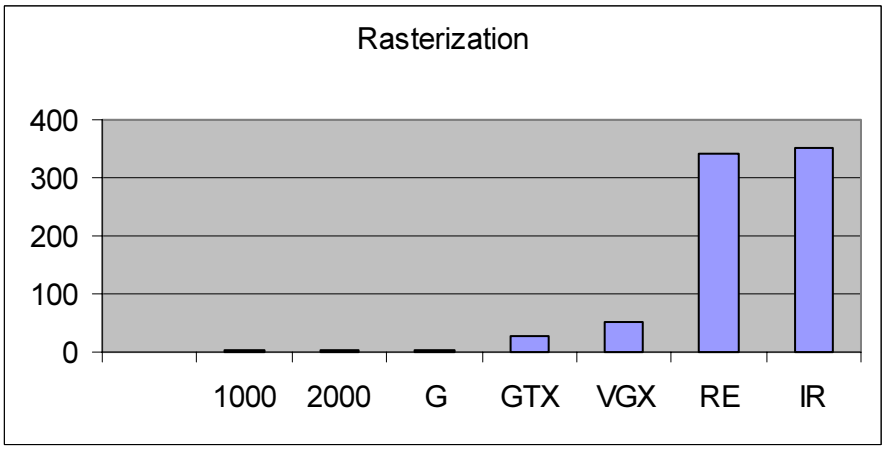
Data parallelism

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

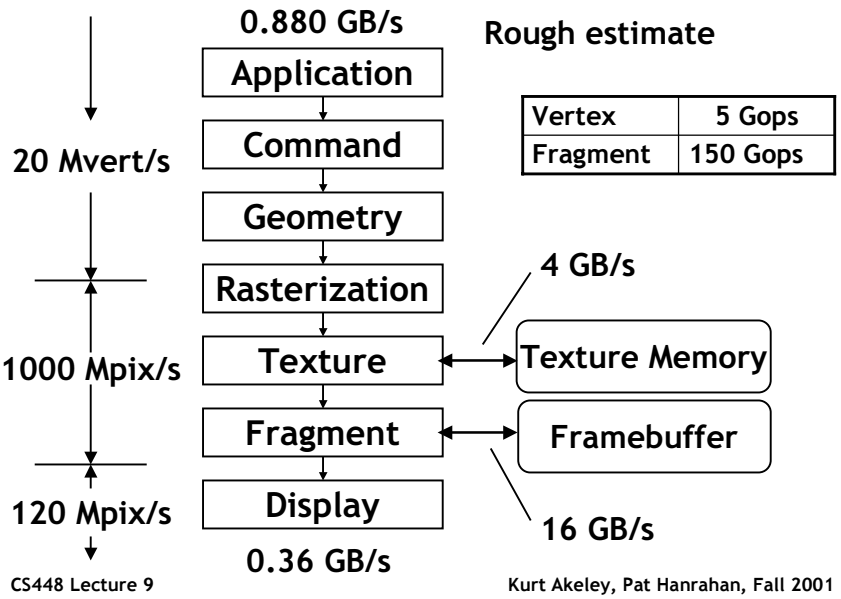
Geometry Parallelism (SGI)



Raster/Fragment Processors (SGI)



Communication Requirements



Ordering

Data dependencies

- Graphics state
- Framebuffer operations
  - Painter's algorithm
- Write to texture
  - Render/Copy/Render
- Readback

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

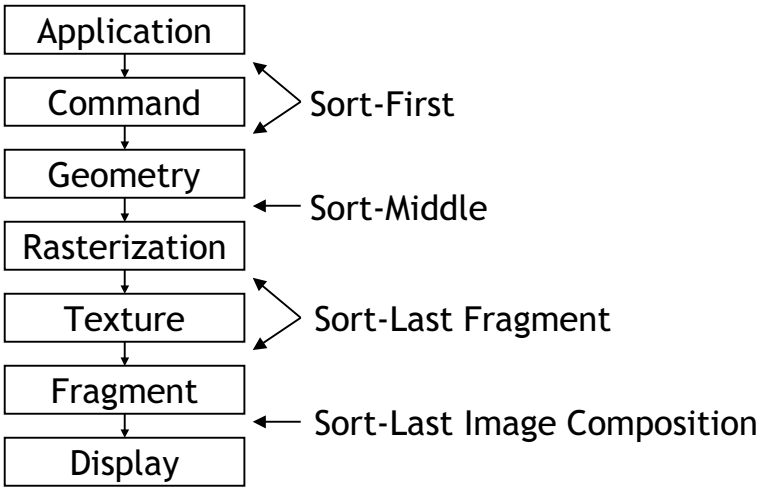

Communication Taxonomy

- 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



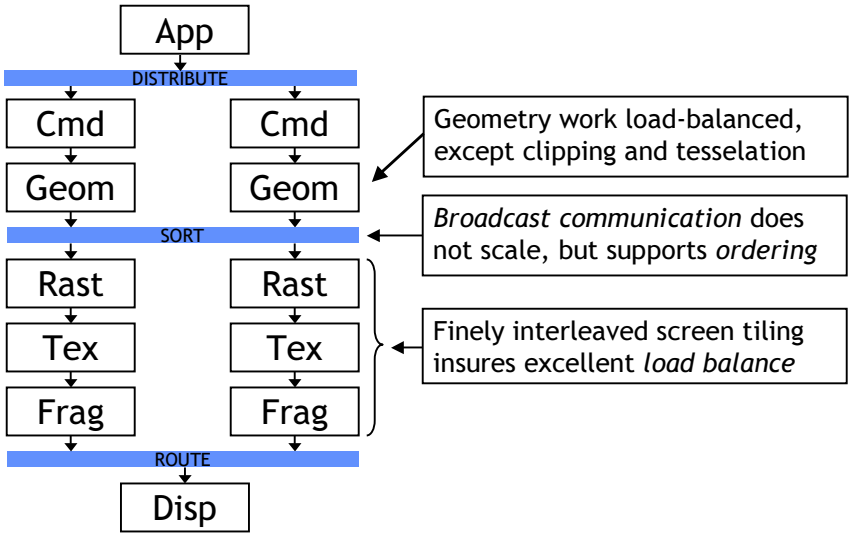
Sort-Middle

Image-Space Work Distribution

Parke - Tiled

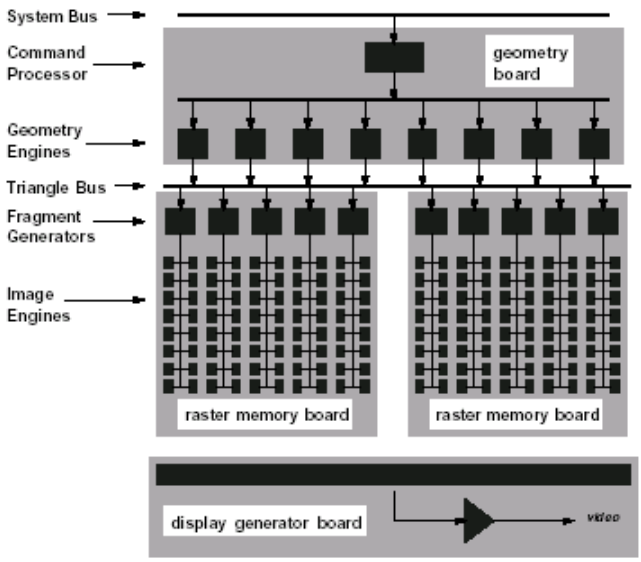
Fuchs - Interleaved

Sort-Middle Interleaved

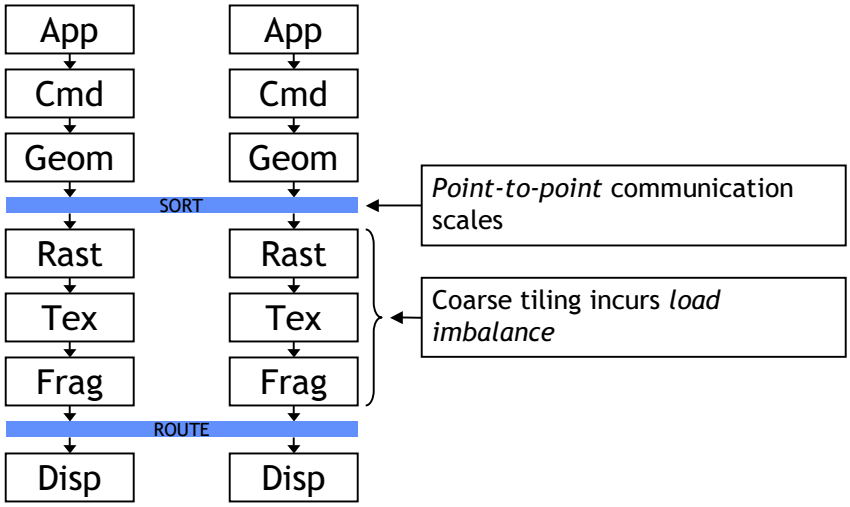


SGI Graphics Workstations: RealityEngine, InfiniteReality

SGI RealityEngine



Sort-Middle Tiled



UNC PixelPlanes, Stanford Argus

UNC Pixel-Planes4 (1986)

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

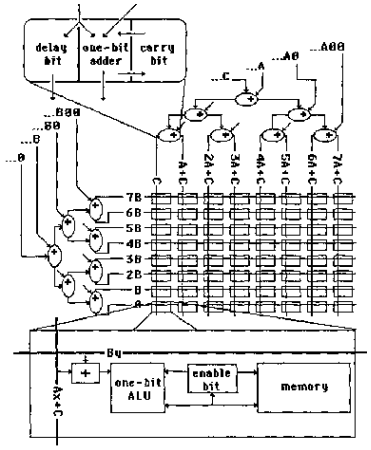


Fig. 1: Conceptual design of an 8×8 Pixel-Planes chip.

Logic-enhanced memory

Tree of 1-bit adders

Compute linear expression

Ax+By+C

Cycle:

Write (A,B,C) to "memory"

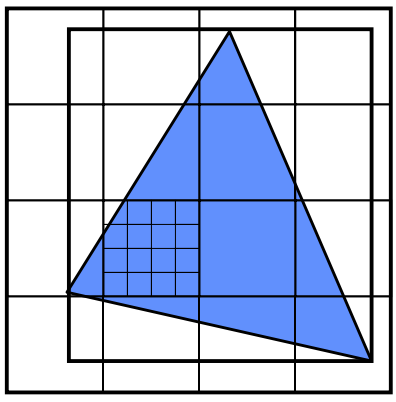
Writes 512x512 cells

SIMD compute surface

Efficiency?

Footprint processors

Footprint processors

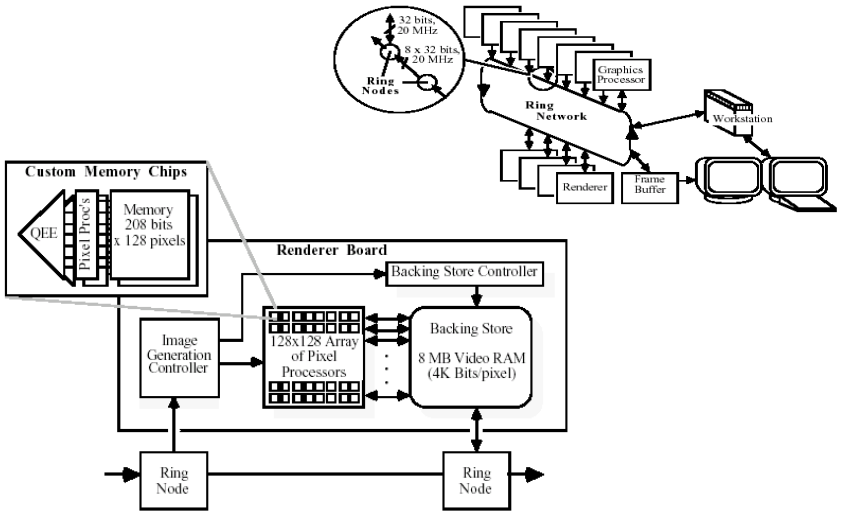


Area(tri):Area(bbox):

On average ~1/6

Average number of pixels inside a tile?

UNC Pixel-Planes5 (1990)



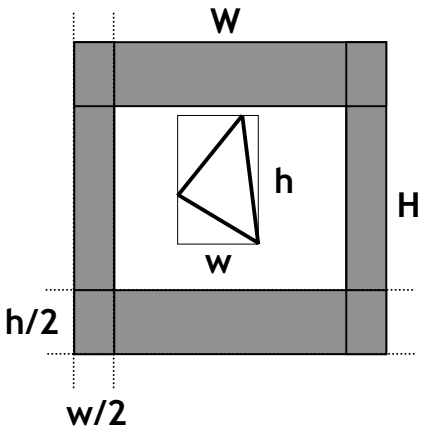
Pixel-Planes 5 Rendering Algorithm

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



Tile

Molnar-Eyles Formula

3 cases

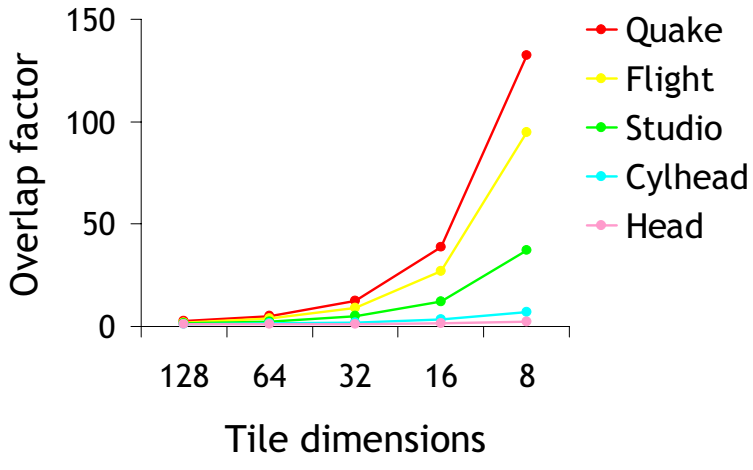
$$4 \times \frac{4(w/2)(h/2)}{WH}$$

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

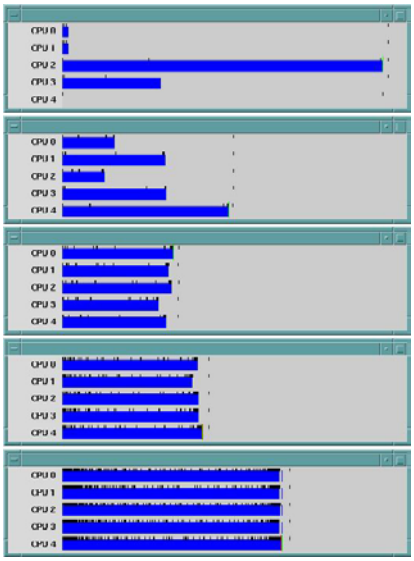
$$1 \times \frac{(W-w)(H-h)}{WH}$$

Total $O = \left(\frac{H+h}{H}\right)\left(\frac{W+w}{W}\right)$

The Overlap Factor



Rasterization Cost



512

256

128

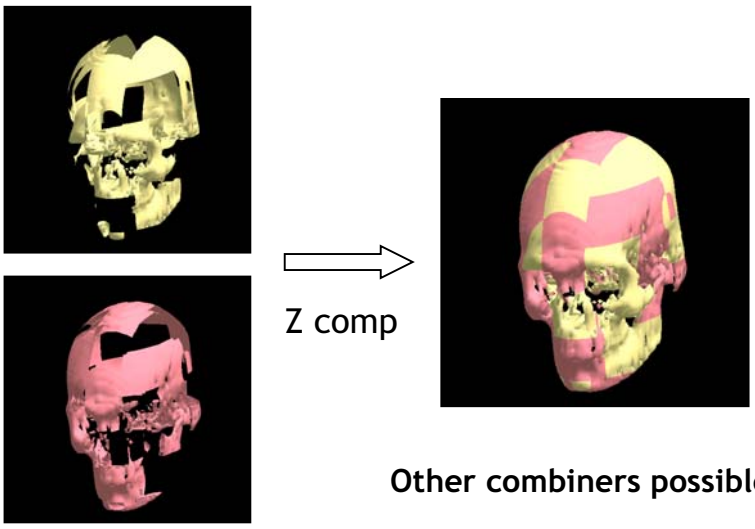
64

32

- 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

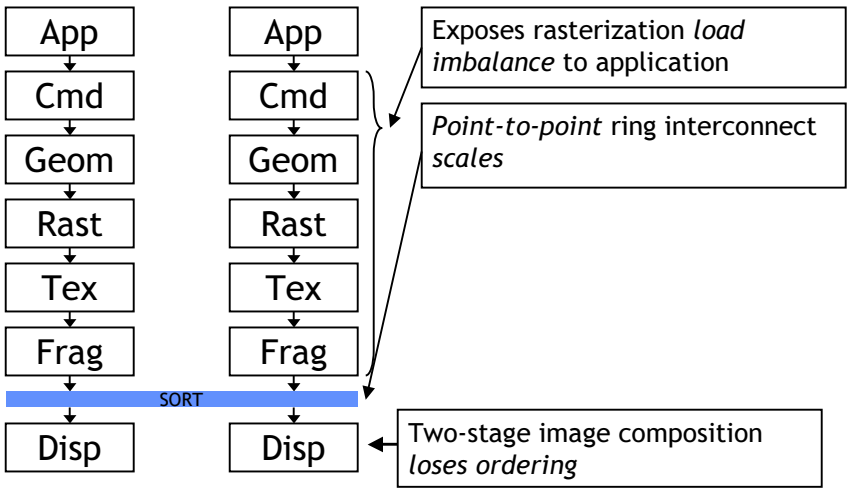
Sort-Last

Z-Composition



Other combiners possible

Sort-Last Image Composition



UNC/HP PixelFlow, Aizu VC-1, Stanford Lightning-2

UNC Pixel Flow

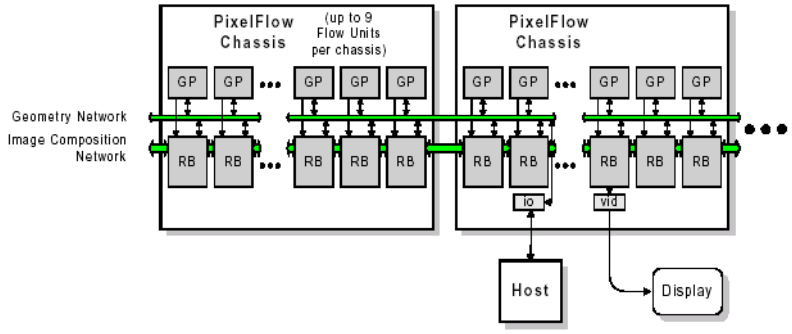


Figure 1: Typical PixelFlow System.

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

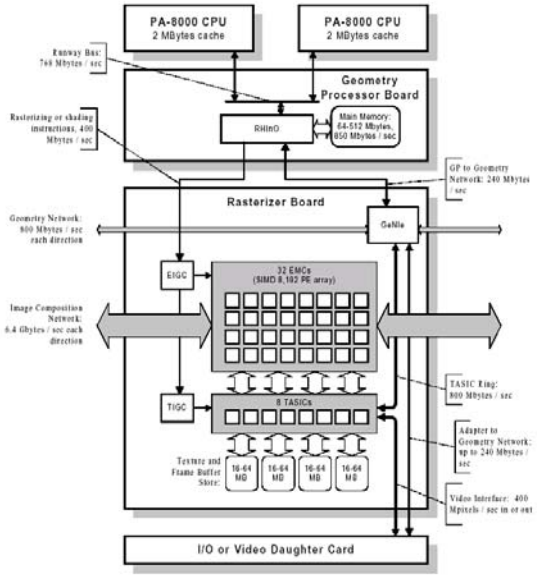


Figure 4: PixelFlow Flow Unit with CPUs and IO/Video adapter.

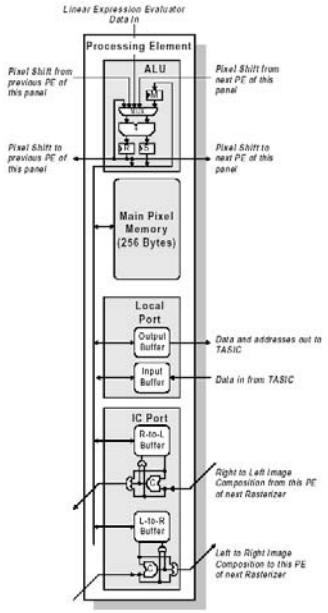
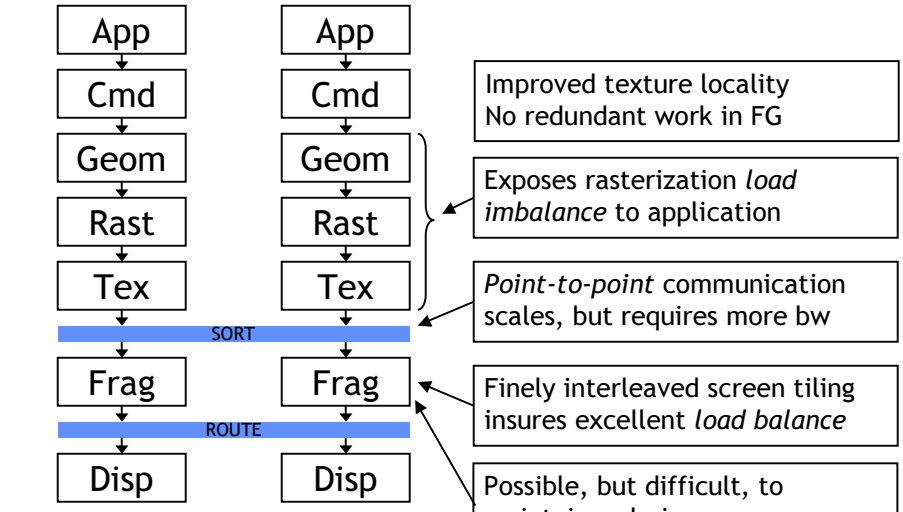


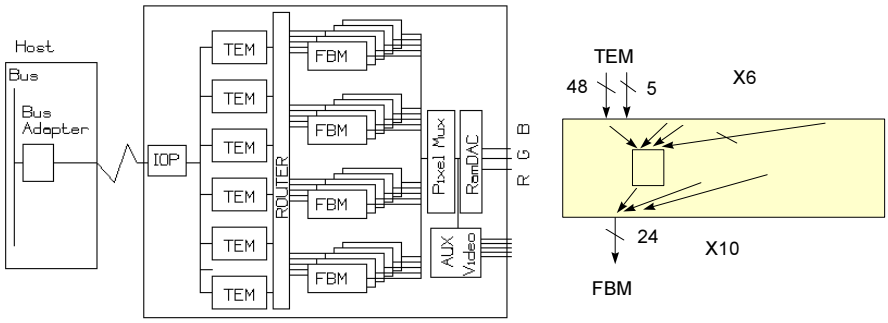
Figure 6: Functional diagram of Processing Element.

Sort-Last Fragment



Kubota Denali, E&S Freedom 3000

Kubota Denali (1993)

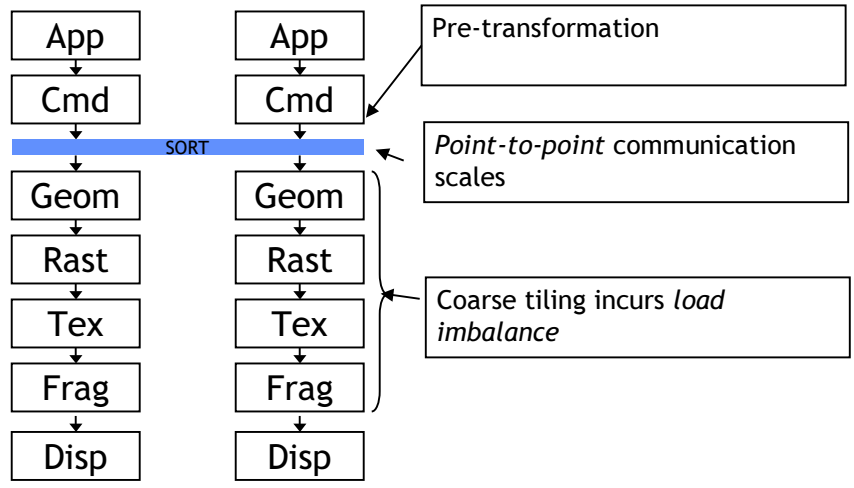


Denali Technical Overview 1.0

Kubota Pacific Computer, 1993

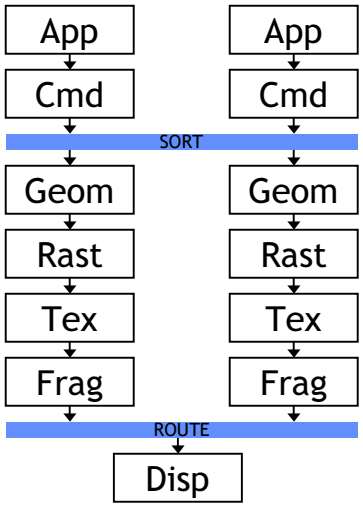
Sort-First

Sort-First

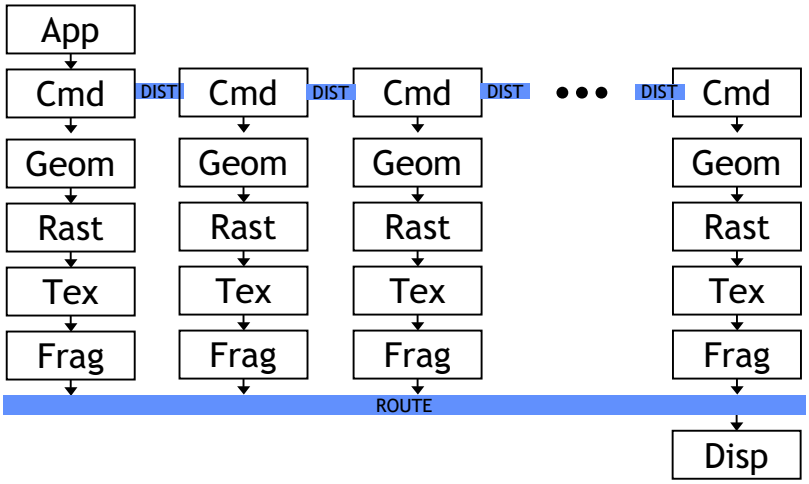


Princeton Display Wall, Stanford WireGL

Sort-First

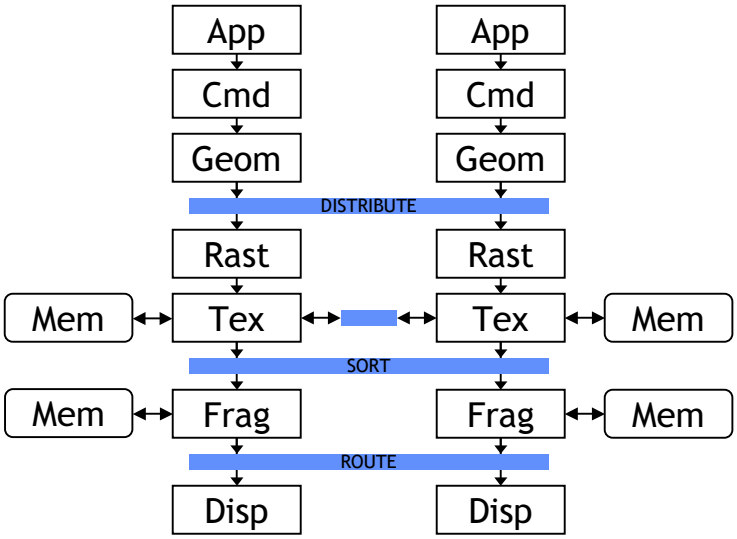


Ring Parallelism



3DLABs

Sort-Everywhere: Pomegranate



Architecture Comparison

	Sort-First	Sort-Middle Tiled	Sort-Middle Interleaved	Sort-Last Fragment	Sort-Last Image Comp.	Pomegranate
Rasterization balanced	✗	✗	✓	✗	✗	✓
Scalable communication	✓	✓	✗	✓	✓	✓
Temporal load balance	✗	✗	✓	✓	✓	✓
Ordered	✓	✓	✓	✓	✗	✓

Trends and Predictions

- Task- vs. Data-parallelism
  - Full cicle? 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?