

# Image Compositing Hardware

(Another exciting presentation)

© 2002 Brenden Schubert

The Metabuffer: A Scalable Multiresolution Multidisplay 3-D Graphics System Using Commodity Rendering Engines

Lightning-2: A High-Performance Display Subsystem for PC Clusters

Scalable Interactive Volume Rendering Using Off-the-Shelf Components

Figure 1: The Title Slide

# The Metabuffer

- Sort-last parallel realtime rendering using COTS hardware
- Independently scalable in renderers and displays
- Any renderer viewport can map to any area of final composited image

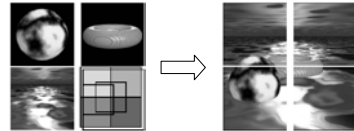


Figure 2: The Metabuffer Overview Slide

# System Architecture

- [A] = COTS renderer
- [B] = onboard framebuffer
- [C] = compositing unit
- Not shown: compositing framebuffers used to drive displays

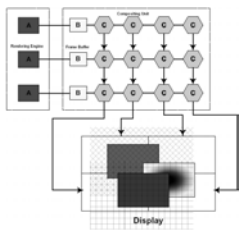


Figure 3: The Metabuffer System Architecture Slide

# Data Flow

- Viewport configuration kept track of on rendering nodes' CPUs
- Routing/Compositing info encoded in image
- Total data is proportional to input size

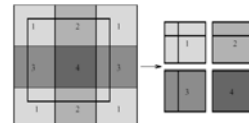


Figure 4: The Metabuffer Data Flow Slide

# Bus Scheduling

- IRSA round robin
  - Idle recovery slot allocation
  - Allows for best overall use of bus among "composers"
- Composer vs. Compositor?

Figure 5: The Bus Scheduling Slide

# Metabuffer Operation

- Frame Transition
- Waiting for PIPEREADY
- Buffers are filled
- Output framebuffers signal completion
- Compositor relays finish signal
- Master compositor signals start of frame
- Composers in pipe begin frame
- Input framebuffer streams out data

Figure 6: The Metabuffer Operation Slide

## Do we have interns at a major hardware manufacturer?

- Nope. Guess we'll just have to simulate/emulate this
- Simulator
  - Tries to accurately implement the architecture
  - C++, multithreaded by object
- Emulator
  - Tries to produce the same output that the Metabuffer would as fast as possible
  - 128 node (PIII-800 GeForce2) Beowulf cluster

Figure 7: The Slide with the humorous title

## Other Things

- Software simulation uses raytracer to generate images and depthmap
- Antialiasing can be done with supersampling and filtering at the output framebuffer
- Foveated vision: multiple user view-tracking?
- Stanford views pi meeting
- High speed active network switch

Figure 8: The Other Things Slide

## F-22 Lightning-2 by NOVALOGIC.

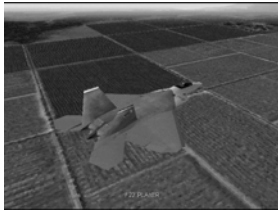


Figure 9: The Joke Lightning-2 Overview Slide

## Lightning-2

- Hardware compositing switch for clusters
- DVI-to-DVI interface
- Scales independently in inputs and outputs
- Actually fabricated in hardware
- Costs a lot

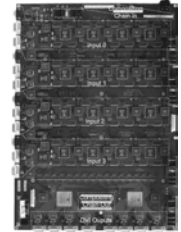


Figure 10: The Lightning-2 Overview Slide

## Lightning-2 Architecture

- Connects to DVI-out from commodity graphics cards in cluster
- DVI captured by first column of modules and propagated across rows
- Compositing for each display output occurs down each column

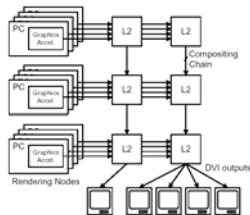


Figure 11: The First Lightning-2 Architecture Slide

## L-2 Module Architecture

- DVI inputs
- RS-232 back-channel
- DVI Compositing in/out
- Memory controller
- Double-buffered framebuffer (SDRAM)

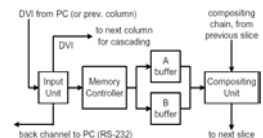


Figure 12: The Second Lightning-2 Architecture Slide

## Pixel Reorganization

- Input frames arrive in input-space raster order
- Compositors operate in output-space raster order
- Must map input order to output order
  - Forward-Mapping:
    - Pixels reordered on write
    - Pixels are stored in output-space raster order
  - Reverse-Mapping:
    - Pixels reordered on read
    - Pixels are stored in input-space raster order

Figure 13: The Pixel Reorganization Slide

## Input

- Three ways to get data rendered by cluster video cards:
  - Read pixels back into system memory
  - Use scan-converters and capturer
  - Use DVI interface
- We'll take #3
- Thus need to transfer pixel mapping information via DVI
  - Specified by encoding strip header in first two pixels of each scanline

Figure 14: The Input Slide

## Frame/Depth Transfer

- Buffer swap on horizontal blanking decreases latency
  - All information encoded in scanline header – no frame-specific dependencies
- Serial back-channel used to tell rendering nodes when previous frame has been received
- Introduces additional frame of latency since rendering nodes could be up to 1 frame out of sync
- Depth information must be read back to main memory and then copied to color buffer to send via DVI
  - Introduces another frame of latency

Figure 15: The Frame Transfer Slide

## Results

- Frame Transfer Protocol
  - Rendering node operates at 70Hz, Lightning-2 operates at 60Hz
  - Allows 2.3ms for back-channel frame transfer signal
  - Achieved >90% of the time (pesky os interference!)
- Depth Compositing
  - Depth copy has constant cost of appx. 17ms
  - At >4 nodes this takes up more than half of the frame time and begins to drop speedup below ideal

Figure 16: The Lightning-2 Results Slide

## Sepia-2

- Custom PCI-card for parallel interactive volume rendering on a cluster
- Single-stage crossbar
- Supports blending and non-commutative compositing operations



Figure 17: The Sepia-2 Title Slide

## Sepia Architecture

- Rendering accelerators (VolumePro 500 raycasters)
- Display device (standard OpenGL renderer)
- PCI Card - 1 for each rendering node and display device
  - 3 FPGAs
  - 2 ServerNet-2 gigabit network ports
  - RAM buffers
- High-speed network switches

Figure 18: The Sepia Architecture Slide

## Sepia PCI Card

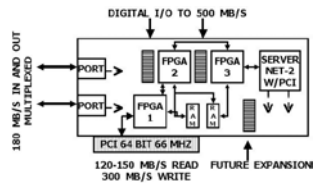


Figure 19: The Sepia PCI Card Slide

## Comparison

- Sepia-2 uses parallel blending operations
  - Lightning-2 uses scan-line based pixel mapping
  - MetaBuffer uses viewport transformations and depth compositing
- Sepia-2 scales linearly (inputs + outputs) using Clos topology networking
  - Lightning-2 and MetaBuffer scale ~quadratically (inputs x outputs) using mesh topology

Figure 20: The Sepia Comparison Slide

## TeraVoxel Operation

- VolumePro generates viewpoint-dependent base plane (sub-volume image)
  - Writes it to main memory
- BP is copied to larger-resolution SBP
- Sepia-2 cards each read SBP from local main memories
  - Transmit over network to display node
  - Display node performs compositing operations
- Display node textures result onto polygon to display using OpenGL

Figure 21: The TeraVoxel Operation Slide

## Sepia Firmware

- Need an associative blending operator to do raycasting in parallel
- "F" is proven to be associative and equivalent to the basic subvoxel blending operation
- Proof is elementary

Figure 22: The Sepia Firmware Slide

## Clos Topology

- Sepia-2 depends on full crossbar networking with linear scaling
- Recursive Clos topology is proven to provide this
- Read the 1953 Clos paper if you want to know more

Figure 23: The Clos Topology Slide

## Sepia Results

- Time spent in VolumePro calculation: 36-42ms
  - Rest of calculation pipelined with no stage >36ms
  - So optimal refresh rates are 24-28fps
- Copying subimage to memory: 5ms
- Transfer over network and blending calculation: 34ms
- Maximum wait for vertical refresh to display new frame: 17ms
- Worst-case combined latency: 102ms

Figure 24: The Sepia Results Slide