Commodity Clusters for Immersive Projection Environments

Course Organizers:
Marcelo Knörich Zuffo
Hank Kaczmar ski
Speakers

Hank Kaczmarski  Benjamin Schaeffer  Camille Goudeseune
{hank | schaeffr | camilleg}@isl.uiuc.edu
Integrated System Laboratory - Beckman Institute
University of Illinois at Urbana-Champaign

Marcelo Knorich Zuffo  Luciano Soares  Paulo Bressan
{mkzuffo | lsoares | pbressan}@lsi.usp.br
Laboratório de Sistemas Integráveis - Escola Politénica
Universidade de São Paulo

Bruno Raffin  Philippe Augerat
{bruno.raffin | philipe.augerat}@imag.fr
Laboratoire Informatique et Distribution
Institut d'Informatique et Mathématiques Appliquées de
Grenoble CNRS-INPG-INRIA-UJF

Course Organization

This course is organized in 4 modules:

• Module I: The Anatomy of a Commodity VR Cluster, from Application to Technology Choices
• Module II: VR with Commodity Clusters: Hints and Tricks
• Module III: Cluster Programming Environments for VR
• Module IV: Hands-On Laboratory
Module I: The Anatomy of a Commodity VR Cluster, from Application to Technology Choices

Speakers:
Benjamin Schaeffer
Camille Goudeseune
Hank Kaczmarski
Luciano Soares
Marcelo Knörich Zuffo

Module I Summary

• Architecture Overview
• Hardware Overview
• Software Overview
• I/O Device Integration
• Administration Overview
Module I Objectives

• In this module, architecture and technology issues will be discussed.
• The goal is to let VR cluster developers design a system matching final application requirements.
• We will discuss various design options for clusters, particularly capability/cost tradeoffs.

Immersive Projective Technology (IPT)
Commodity computing

• Building complex computational infrastructure with low cost pieces that you find on the nearest supermarket:

• A system that effectively runs a large class of applications and can be built for almost no cost using surplus parts ("the hundred-dollar cluster").

Previous experiences on cluster computing:

• Numerical computing
• Database computing
• Graphics render farms
• Multimedia delivery
• Web hosting and internet providers
• Real time visualization
The VR Cluster

- **What is on the spot in this Course:**
  - REAL TIME IMMERSIVE PROJECTION
  - VIRTUAL REALITY
  - How to aggregate computing commodities to support our Virtual Reality needs

Commodity parts

**Industrial Age Commodities**
- Grocery, food, beverage
- Iron, plastic
- Civil construction materials
- Clothes

**Info Age Commodities**
- Computers
- Content (information)
- Consumer electronics
**Commodity parts**

![Graph showing the total number of computers in the world from 1980 to 2005. The x-axis represents the years from 1980 to 2005, and the y-axis represents millions of computers. The graph shows a steady increase in the number of computers over time.]

**Figure 1 – Total Number of Computers in World**

---

**Why VR Clusters?**

- The rapid evolution of consumer electronics and computing technology (and its intrinsic short obsolescence cycles) leads to a constant renewal of low cost commodity parts that are available everywhere, to anyone, anytime.

- A recent opportunity for final users, developers, researchers and anyone else in society interested in VR.
Why VR Clusters?

• **Cost**
  - Large scale manufacturing

• **Performance**
  - Moore’s law on PC’s

• **Flexibility**
  - Diversity of integration options

• **Access**
  - Anywhere, to everyone, anytime

Why VR Clusters?

• **Upgradeability**
  - System can be constantly upgraded according to release of new commodity parts

• **Availability**
  - The availability of a broad range of commodity parts that could be attached to the cluster
Putting all together

Building a cluster yourself

**VR Cluster can be integrated in several ways, and architectural tradeoffs should be based on:**

- Performance demanded by final applications
- Graphics and Numerical Scalability
- Budget
- Obsolescence cycle
- Future Upgrades
Architecture Overview

- **Overall requirements**
- **Software architecture**
  - Master/slave
  - Client/server
- **Cluster architecture**
  - Networking
  - Render node power
  - Application node present?
- **Application architecture**
  - Static vs. dynamic
Overall Requirements

- All systems need to fulfill 3 requirements:
  - Genlock
  - Swap-lock
  - Data-lock
- Different mixtures of hardware and software can be used

Genlock

- The cluster render nodes each produce a sequence of video frames.
- For a coherent image across the multiple displays, these frames must be produced in phase.
- Otherwise, you'll see tearing in animated objects. Critical for active stereo.
- Either pure hardware or software/hardware solutions (like SoftGenLock).
Swap-Lock

- Different views of a scene take different times to render.
- Consequently, frame buffer rendering and swaps need to be synchronized.
- Otherwise, you'll see tearing with animation. The view that renders slowest will appear to lag.
- Software (network sync) or hardware specific schemes (Wildcat boards) exist.

Data-Lock

- Each render node draws its frames using locally held information.
- The information held on the render nodes must be consistent for the views to remain consistent across nodes.
- Software only.
Software Architecture

• Client/Server

• Master/Slave

Master/Slave

• Identical copies of the application run on all cluster nodes.

• Small amounts of information transfer between nodes. Often just information from input devices or timestamps. A custom protocol also possible.
Master/Slave

- One node is a master, handling changing the application’s state, and the others are slaves.

- Can either be based on a wrapper built on the communications paths in a well-known library or specific to a given application.
Client/Server

• One cluster node (server) serves data to the render nodes (clients).

• Built on a general protocol.

Client/Server

• Can be more flexible than master/slave. Many applications can embed a server that works with the same render clients.

• Probably uses more communications bandwidth than master/slave. Tradeoff with flexibility
Client/Server

Types of Client/Server

- **What kind of data is being communicated in the protocol?**
  - Pixels
  - Graphics Primitives
  - High-level information, such as scene graph data

- **Data type impacts bandwidth requirements**
Cluster Architecture: Networking

- Client/server software that sends pixels or even graphics primitives may require the best available networks to run well. Relying on higher-level information, like scene graph data, tends to reduce this requirement.

- Master/slave software often requires very little network bandwidth.

Cluster Architecture: Networking

- Assuming that latency is small with regard to frame times (< 5 ms) and genlock does not occur across the network, latency is less important than bandwidth and broadcast capability.

- Myrinet networks have low latency but are point-to-point and cannot do multicasting.

- Gigabit Ethernet is the winning choice for VR Clusters!
Render Node Power

• For master/slave software, the render nodes are directly running copies of the program. Hence, they must be able to perform complex computational tasks.

• For client/server software, the render nodes are nearly dumb. Hence, they simply need to concentrate on networking/graphics ability.

Application Architecture

• Maintaining coherent views of the world on the render nodes is the main problem (data-lock).

• Coherency is more difficult to maintain if the world is highly animated.
Application Architecture

Application classification (a continuum)

- Static (viewer applications)
  - Navigating through a static scene graph
  - Maybe animation based on time stamps
- Dynamic
  - Highly animated, possibly based on internal computations, file I/O, or by real-time data from remote sensors

Application Node

Present?

- Master/slave software does not require an additional application node. All nodes in the cluster are render nodes.

- Client/server software is designed for a cluster with a non-rendering application node, which serves data to the render nodes.
Software Architecture: Applications

• Master/Slave architecture is most efficient for static applications. May be difficult to fit a given dynamic application into this framework.

• Client/server architecture has advantages for writing dynamic applications. It forces state synchronization at the server.
Hardware Overview

I. Computer hardware

II. Networking

III. Video projectors and graphics cards:
   * a. Stereo technologies (passive vs. active)
   * b. Refresh rate
   * c. Graphics card fill rate
   * d. Volume rendering accelerating boards

IV. Overall display device:
   * a. Ferrous metal interference with magnetic tracking
   * b. Space requirements

V. High-end VR solutions

VI. Cheap/ portable VR

In the beginning...
we only had clusters

In the 1990’s...
we spent $250k/graphics pipe

In the 21st century...
clusters return
Hardware Overview

Networking

- Scramnet
- Hippi
- FDDI
- Myrinet
- Ethernet
  - 10Gbit
  - 1Gbit
  - 100Mbit
- Serial/parallel ports

Video projectors and graphics cards:

Stereo technologies (passive vs. active):
High framerate needed for frame sequential active stereo (ideally 120 – 144 vertical hertz)
Twice the number of (much less expensive) graphics outputs/projectors for passive stereo
Hardware Overview

Graphics cards:

Hardware genlock:

High end gaming:
Hardware Overview

**Overall display device:**

- a. Interference with magnetic tracking
  - Keep ferrous metal completely away
  - Keep all metallic arrays away
  - Keep the space electrically quiet
- b. Space requirements
  - 35’ x 35’ x 35’ volume needed for a 3 meter cube six-surface virtual reality environment

Hardware Overview

**High-end VR solution**

**Active stereo:**

- **CRT/DLP projector** $25-90k
- **Hardware genlocked graphics** $4k
- **Liquid crystal shutter glasses** $1200 ea
  (with IR glass trigger emitter)
Hardware Overview

Low-end VR solution

Passive stereo:

- 2 commodity DLP projectors $5k ea
- High end gaming graphics $300
- Polarized glasses $5 ea

A VR Cluster Taxonomy

- **Node configuration in the cluster**
  - Homogeneous vs. heterogeneous
  - Application node
  - Render node
  - I/O node
A VR Cluster Taxonomy

• **Related to how nodes are configured regarding its:**
  • Cost
  • Performance
  • Manageability
  • Upgradeability

A VR Cluster Taxonomy

**Device mapping**

• Symmetric vs. asymmetric

**Related to how information is routed through the various subsystems:**

• Data (networking)
• Control (all devices)
• User (keyboard, mouse, IO Devices)
• Video (projectors and displays)
A VR Cluster Taxonomy

System Integration

- Centralized vs. Distributed

Related how to integrated nodes in the environment

- Ex. Distributed PCs in the lab
- Building everything on a rack
- PC wall

A VR Cluster Taxonomy

Asymmetric Distributed Heterogeneous VR Cluster
A VR Cluster Taxonomy

Symmetric Centralized Homogeneous VR Cluster

Homogeneity: All nodes are identical
- Easy to physically replicate
- Easy to clone software

Centralized
- Easy to assemble on a single rack

Symmetry:
- Manageability: Data (networking switches), Video (video switch), Control (MultiSerial interface), User (KVM)
- Better programming environment
A Symmetric Homogeneous Centralized VR Cluster

**Advantages:**
- Easy application porting and developing
- Performance
- Flexibility
- Scalability
- Better manageability

**Disadvantages**
- Cost in long term (since you change something you need to change for all nodes).
- Complexity & Logistics (plugging it all together)

A VR Cluster Taxonomy

Asymmetric Centralized Heterogeneous VR Cluster

- Network Switch (Cluster & Backbone)
- CPU 1
- CPU 2
- CPU 3
- CPU 4
- CPU 5
- CPU 6
- I/O Node
- Stereo Sync
- Multiprojection Immersive Environment
- VGA Signal
- PCI
- TCP/IP
- AGP
Asymmetric Homogeneous Centralized VR Cluster

Advantages:
- Cheap
- Straightforward according to your needs
- Easier to build and maintain

Disadvantages
- Smaller obsolescence cycles
- Performance limitations
- Poor load balancing
- Dedicated software and applications
Software Overview

• Porting and Developing to a VR Cluster
  • Choosing the correct strategy based on what commodities you have available (cost/performance)
  • How easily modifiable to suit your VR needs
  • How easily modifiable to fit in your cluster
Software Overview

- **Need to consider the software legacy**
  - 10 years of software developing upon high-end workstations

- **Need to consider the convergence**

High End:
- OpenGL
- Applications
- High end App

VR Cluster

PC:
- Direct3D
- Games
- Low end App

Graphics Programming

**Graphics Libraries**
- OpenGL, Direct3D

**Graphics Packages**
- Performer, Game Engines, VTK

**Graphics Middleware**
- VR Juggler, CaveLib, Diverse, MultiGen-Paradigm
Cluster Programming

Network Protocols
• TCP/IP, VIA

Cluster Packages
• MPI, OpenPBS, LFS

Communication Middleware
• CORBA, VNC

VR Cluster Programming

VR Cluster Libraries
• WireGL, SoftGenLock, DICELib

VR Cluster Packages
• OpenPBS

VR Cluster Middleware
• Syzygy, NetJuggler
### Syzygy

<table>
<thead>
<tr>
<th>General Description</th>
<th>A programming toolkit for distributed graphics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>IRIX, Windows, Linux</td>
</tr>
<tr>
<td>Programming Interface</td>
<td>C, C++</td>
</tr>
<tr>
<td>VR Device Support</td>
<td>Limited</td>
</tr>
<tr>
<td>Public Domain</td>
<td>Yes</td>
</tr>
</tbody>
</table>

[http://www.isl.uiuc.edu/ClusteredVR/ClusteredVR.htm](http://www.isl.uiuc.edu/ClusteredVR/ClusteredVR.htm)

### WireGL

<table>
<thead>
<tr>
<th>General Description</th>
<th>OpenGL driver minimally modified applications to render to cluster environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>Unix, Windows</td>
</tr>
<tr>
<td>Programming Interface</td>
<td>OpenGL + cluster extensions</td>
</tr>
<tr>
<td>VR Device Support</td>
<td>No</td>
</tr>
<tr>
<td>Public Domain</td>
<td>Yes</td>
</tr>
</tbody>
</table>

WireGL

- Stanford University Computer Graphics Lab
- Standard client/server network model.
- Uses OpenGL primitives.
- Produces tiled images; treats a physically disjoint set of display as a single logical display.
- Runs on TCP/IP and Myrinet GM.

VTK

<table>
<thead>
<tr>
<th>General Description</th>
<th>The VTK (Visualization Toolkit) is a open source object oriented visualization package.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>Windows, Linux, UNIX</td>
</tr>
<tr>
<td>Programming Interface</td>
<td>C, C++, Java</td>
</tr>
<tr>
<td>VR Device Support</td>
<td>No</td>
</tr>
<tr>
<td>Public Domain</td>
<td>Yes</td>
</tr>
</tbody>
</table>

http://www.kitware.com
## SoftGenLock

<table>
<thead>
<tr>
<th>General Description</th>
<th>SoftGenLock enables active stereo and genlock on Linux clusters equipped with VGA compatible graphics cards.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>Linux</td>
</tr>
<tr>
<td>Programming Interface</td>
<td>No</td>
</tr>
<tr>
<td>VR Device Support</td>
<td>Not applicable</td>
</tr>
<tr>
<td>Public Domain</td>
<td>Yes</td>
</tr>
</tbody>
</table>

http://softgenlock.sourceforge.net

## DICELib

<table>
<thead>
<tr>
<th>General Description</th>
<th>Synchronization and data Sharing data library upon TCP/IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>Unix, Windows</td>
</tr>
<tr>
<td>Programming Interface</td>
<td>C, C++</td>
</tr>
<tr>
<td>VR Device Support</td>
<td>No</td>
</tr>
<tr>
<td>Public Domain</td>
<td>Yes</td>
</tr>
</tbody>
</table>

http://www.lsi.usp.br/~brunobg/dicelib/
NetJuggler

<table>
<thead>
<tr>
<th>General Description</th>
<th>Cluster support for VR Juggler</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>Linux, Windows</td>
</tr>
<tr>
<td>Programming Interface</td>
<td>C++</td>
</tr>
<tr>
<td>VR Support</td>
<td>Trackers, gloves, IO/Devices</td>
</tr>
<tr>
<td>Public Domain</td>
<td>Yes</td>
</tr>
</tbody>
</table>

http://netjuggler.sourceforge.net

VR Juggler

- Iowa State University’s Virtual Reality Applications Center.
- Open Source VR software development environment.
- VR virtual platform and application framework.
- Modular extensible micro-kernel system
- Open source: Irix, Linux, Win32.
- Separates system dependent and system independent components.
**OpenPBS**

<table>
<thead>
<tr>
<th>General Description</th>
<th>The OpenPBS (Open Portable Batch System) is a flexible batch queueing and workload management system for heterogeneous clusters.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>Unix</td>
</tr>
<tr>
<td>Programming Interface</td>
<td>Not applicable</td>
</tr>
<tr>
<td>VR Device Support</td>
<td>Not applicable</td>
</tr>
<tr>
<td>Public Domain</td>
<td>Yes</td>
</tr>
</tbody>
</table>

[http://www.openpbs.org](http://www.openpbs.org)

- **Developed by Viridian System for NASA.**
- **Operates on networked, multi-platform Unix environments (POSIX standards).**
- **Supports dynamics distribution of workloads across wide-area networks.**
- **Provides a GUI for job submission, tracking and administration.**
## VNC

<table>
<thead>
<tr>
<th>General Description</th>
<th>VNC (Virtual Network Computing) is a remote display system for a wide variety of machine architectures.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>Unix, Windows</td>
</tr>
<tr>
<td>Programming Interface</td>
<td>Not applicable</td>
</tr>
<tr>
<td>VR Device Support</td>
<td>Keyboard, mouse</td>
</tr>
<tr>
<td>Public Domain</td>
<td>Yes</td>
</tr>
</tbody>
</table>

http://www.uk.research.att.com/vnc/index.html

## MPI

<table>
<thead>
<tr>
<th>General Description</th>
<th>MPI (Message Passing Interface) is the standard for multicomputer and cluster message passing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>Unix, Windows</td>
</tr>
<tr>
<td>Programming Interface</td>
<td>C, C++, Fortran (77, 90 and 95)</td>
</tr>
<tr>
<td>VR Device Support</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>Public Domain</td>
<td>Yes</td>
</tr>
</tbody>
</table>

http://www.mpi-forum.org/
MPI

- Proposed as standard by vendors, implementers, and users.
- Designed for high performance on both massively parallel machines and on workstation clusters.
- Many MPI-1 specification implementations in Windows and Unix.
- MPI-2 specification is not completely implemented on tools:
  - Dynamic Process Management
  - One-Sided Communication
  - Parallel I/O

VIA

<table>
<thead>
<tr>
<th>General Description</th>
<th>The VIA (Virtual Interface Architecture) specification defines a industry standard cluster communication library</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>Unix, Windows</td>
</tr>
<tr>
<td>Programming Interface</td>
<td>C, C++,</td>
</tr>
<tr>
<td>VR Device Support</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>Public Domain</td>
<td>Yes</td>
</tr>
</tbody>
</table>

http://www.nerc.gov/research/FTG/via
Section Bibliography


Software Issues

A few terms

- **Latency and Jitter**
- **Update rate**
  - Analogous to frame rate for video.
- **Noise and Resolution**
- **Input “sensor”, output “effector”**
  - Sensors: from buttons to motion-tracking systems.
  - Effectors: sound, haptics, LEDs, robots.

Latency

How much lag from a sensor to a node?

- **Inherent latency of sensor**
  - E.g., RS232.
- **Software latency**
  - Polling, interrupts.
- **Network latency**
- **OS latency at each node**
Latency

How much lag from a node to an effector?

- Same as for sensors, but reversed!
- OS latency at each node
- Network latency
- Software latency
- Inherent latency of effector
  - Physical inertia.

Latency

How much lag from a sensor, through nodes, to an effector?

- Just add up the individual latencies.
- Push a button ➔ node ➔ ... ➔ node ➔ hear a beep
- Move a wand ➔ ... ➔ see a pointer move
Jitter

Jitter is irregularity of latency

- **Worse than raw latency**
  - User adapts to latency, but cannot predict jitter.
- **Sources:** stressed network, wireless network, task-switching PC, etc.
- **Extreme jitter can overwhelm a node**
  - No data for a while, then many frames at once.

Update rate

A “frame” of data is a snapshot of the state of the sensor or effector

- **Update rate is independent of latency**
  - GPS sensor might have 1 Hz rate and 50 msec latency.
  - Motion tracker might have 200 Hz rate and 150 msec latency.
  - Similar to bandwidth vs. latency on a network.
Noise and Resolution

- Internal noise of a sensor or effector
- External noise: interference
  - Stray magnetic fields, for magnetic motion tracker
  - Radio interference, for wireless network.
  - Hiss, hum, distortion in audio I/O.
- Noise reduces effective resolution
- Illusory resolution
  - Joystick’s reported range is ±32K, but with only 256 distinct values

Tradeoffs

Better performance where it counts!

- Interpolating between frames increases update rate, increases noise
- Filtering reduces sensor noise, increases latency
  - Extended Kalman Filter can reduce noise without increasing latency.
- Domain-specific filtering
  - Know practical limits on a tracked head’s position, orientation, and speed.
Multi-Rate systems

Example: VR wand with magnetic tracker and buttons

- Total latency is maximum of individual ones
- Total jitter is sum of individual ones!
- Prefer increased latency to “beating”
  - One sensor has 25 Hz update rate, another has 30 Hz: 5 Hz throbbing “beat” between them.
  - So resample the faster 30 Hz sensor at 25 Hz.
  - Analogous to frame lock in a multi-screen display.

Motion Tracking

- Magnetic: Ascension, Polhemus
- Hybrid inertial/ultrasonic: InterSense
- From $1K to $100K. More $ buys you:
  - more sensors (position and orientation)
  - lower latency, higher update rate
  - higher resolution, higher accuracy
  - longer range (wireless)
- Wireless (802.11) adds some latency and jitter
Motion Tracking

**Other technologies**

- **Optical (retroreflective balls)**
  - Designed for noninteractive use
  - Sub-millimeter accuracy!

- **Ultrasonic**
  - Interference from echoes and other sounds.

- **Multiple video cameras**
  - Still experimental
  - 30 frames per second: slow update rate, high latency.

Audio and multimedia adapters

"Multimedia" = audio + joystick

- Many sound cards include a "game port" for joystick or gamepad
  - But most recent joysticks use USB port instead.

- **“HID” standard for USB input devices**
  - Which axis does what, from the program’s point of view.

  [www.usb.org/developers/hidpage.html](http://www.usb.org/developers/hidpage.html)
Joysticks and Gamepads

- Joysticks need a table; gamepads don’t.
- How many buttons?
  How many axes (degrees of freedom)?
- Too many buttons is confusing: might need different gamepads for different applications.
- Does it have a driver for your preferred OS?
- Wireless: prefer RF to infrared.

Gamepad vs. Motion tracking

- A motion tracker has more degrees of freedom (“DOF”).
- But the springs of a gamepad’s thumb-joysticks have rudimentary force-feedback.
- Using a 6DOF motion-tracked gamepad, one forwards-backwards DOF suffices for navigation.
Audio (sound cards)

Comparing sound cards
- On-motherboard sound is usually cheap
  - If you need sound quality, buy a sound card anyways.
- PCI is much faster than ISA
- “SNR” signal to noise ratio
  - >75 dB is good: if this is good, so is the whole card.
- Frequency response
  - Good if “flat” within 3 dB, from 80 Hz to 10 kHz.
- Again: does it have a driver for your OS?

Comparing sound cards, flashy features
- Ignore the hype aimed at the game market
- Vendor API’s in hardware
  - Aureal A3D 1 and 2, DirectSound 3D, Creative EAX 1 and 2, RSX, Dolby surround, Qsound, ...
  - Rapidly changing: don’t rely on one for a five-year project
Section Bibliography

- **Motion-tracking manufacturers**
  - InterSense, www.isense.com
  - Ascension Technologies, www.ascension-tech.com
  - Polhemus, www.polhemus.com

- **Kalman Filters.**
Driver availability

Graphics Cards drivers should take full advantage of the graphics capabilities such as the OpenGl support

Hardware

- **PC Computers**
  - Intel Processor
  - Powerful Motherboard (chip-set)
  - Commodity network cards
  - Commodity graphic card
Linux vs. Windows

What I should use?

<table>
<thead>
<tr>
<th></th>
<th>Windows</th>
<th>Linux</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PROS</strong></td>
<td>Many applications</td>
<td>Virtually no cost</td>
</tr>
<tr>
<td></td>
<td>Driver availability</td>
<td>Usually better performance</td>
</tr>
<tr>
<td></td>
<td>Game oriented</td>
<td>Robustness &amp; Reliability</td>
</tr>
<tr>
<td></td>
<td>Easy configuration</td>
<td>Unix Legacy (ex. OpenGL Performer)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Open Source</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Very flexible</td>
</tr>
<tr>
<td><strong>CONS</strong></td>
<td>Not open source</td>
<td>Complex configuration</td>
</tr>
<tr>
<td></td>
<td>Not flexible</td>
<td>Driver unavailability</td>
</tr>
</tbody>
</table>

System manageability

The Video Signal Switch

Routing video signal across the multiple projection systems

VR Cluster  \(\rightarrow\) VIDEO SWITCH  \(\rightarrow\) Displays, Domes, Cave
System manageability
The Video Signal Switch

• **How to select it:**
  • Analog Video Bandwidth
  • Controllability (remote and local)
  • Number of video ports (4, 8, 16)
  • Scalability
  • Video signal Interface (VGA, RGBHV, DVI)

System manageability
Multiserial

Controlling multiple devices
System manageability
Multiserial

• **How to select it?**
  - Total number of ports
  - Serial interfaces (RS232, RS422, USB)

System configuration

Software cloning across nodes
Software installation and configuration

• Local Installation

• Network Installation
System configuration
Windows

Installation steps:
• OpenGL
• GLUT (OpenGL Utilities)
• Graphics card drivers (set resolution and active stereo support)

System configuration
LINUX

• Installation steps:
  • Install X11 (GLX)
  • Install OpenGL (Mesa)
  • Install GLUT (OpenGL Utilities)
  • Install Graphics Card Driver
  • KickStart (Linux RedHat)
# Video configuration

<table>
<thead>
<tr>
<th>Digital</th>
<th>Analog</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLP and LCD projectors</td>
<td>CRT projectors</td>
</tr>
<tr>
<td>No conversion</td>
<td>Conversion</td>
</tr>
<tr>
<td>No blanking interval time</td>
<td>Blanking interval time</td>
</tr>
<tr>
<td>Higher resolutions</td>
<td>Crosstalk and noise</td>
</tr>
<tr>
<td>Noise rejection</td>
<td></td>
</tr>
</tbody>
</table>

**Digital Connectors**: DVI, HD15

**Analog Connectors**: HD15 VGA Connector

---

### Video configuration

**Compositing hardware**

- **Multi Projection**
- **Depth Channel** (Z depth)
Video Gen-Lock

To phase-lock the timing of one video display with another

![Electron ray](image)

CRT

Two CRTs without Genlock

![Two CRTs](image)
Two CRTs with Genlock

Synchronizing frame-buffer renderings in all nodes, usually named FRAME-LOCK in industry

Frame 1  Frame 2

CORRECT!
The “ready” signal is transmitted to the previous node as soon as the graphic cards complete the scene.

When the master node receives the ready signal, it sends the “done” signal to each node swap the frame.
Rate-Lock

- Whether there is a single computer or many, for a better immersion the frame swapping should be done at a constant rate.
- Usually, many applications are dependent of the number of polygons to set the swap speed.

Module II: VR with Commodity Clusters: Hints and Tricks

Speakers:
- Bruno Raffin
- Camille Goudeseune
- Hank Kaczmarski
- Marcelo Knörich Zuffo
- Phillipe Augerat
Module II Summary

• Basic Hardware Set-Up Issues
• Advanced Issues and Solutions
• Useful devices
• Systems administration
• Software development methodologies

Module II Objectives

• Discuss low-level practical details of constructing a VR cluster
• Compare technology
Building a VR Cluster

**Going to Supermarket:**

- Have Plan!
- Logistics is fundamental!

**Going to the Supermarket**

**Preliminaries**
- Check your refrigerator
- Plan your meals
- Build a shopping list

**Go shopping!**
- Only buy what you will need, avoid impulse buying
- Amount
- Cost, don’t buy something just because it is cheap
- Choosing Quality Brands
- Validity
Going to the Supermarket

- **Back to home**
  - Unpacking boxes
  - Organize the mess
- **Again in a month**

Building a VR Cluster

- **Preliminaries**
  - Understand your application
  - Define your architecture
  - Build your shopping list
Building a VR Cluster

- **Go shopping!**
  - Performance
  - Obsolescence
  - Cost
  - Choosing technology
  - Upgradeability
- **Back to the lab**
  - Unpacking boxes
  - Putting it all together

---

**Module II: Section A**

Basic Hardware Setup Issues

Bruno Raffin
bruno.raffin@imag.fr
HVAC

• Some motherboards are equipped with a thermal sensor

• Extra thermal sensors may be added in the room

• PCs and video-projectors are the most important heat sources to take into consideration for air conditioning calibration

Cables

• Generally a lot (power lines, networks, keyboard, mouse, video, …)

• Label the different cables (save a lot of time)

• Check cables do not exceed maximal sizes (signal degradation)
Power Lines

- Apply to each PC the usual rules
- A master switch to turn off all units at once
- Do not connect on one wall-socket more than it can support

Video

- VGA: analog video signal
  - Today’s standard
  - Low cost
  - Sensible to signal degradation
- DVI Digital Video Interface
  - More expensive
  - A standard but not yet widely available
  - Better resistance to signal degradation
  - Easier to handle (e.g. video compositing)
KVM
Keyboard, Video, Mouse

Switch between PCs with one Keyboard, Display and Mouse
- Mechanical or electronics, VGA or DVI
- May not support high resolutions
- Convenient for cluster administration
- Not suitable for large clusters: too many cables

Video Switchers

- DVI or VGA, check resolutions supported
- Some devices may introduce extra latency

Simple video splitter:
- 1-in 2-out video distribution Amplifier
  Input: PC video signal,
  Outputs: video-projector + monitor

Video switcher:
- Extensive video switching capabilities
Genlock, Stereo Signal

Genlock signal:
- Propagated through dedicated network
- Sync on external genlock signal (Wildcat)

Stereo signal (for active stereo):
- VESA standard connector: mini-din 3
- Direct connector on some graphics cards (Wildcat)
- Derivation from the VGA connector (StereoGraphics)
- Retrieved from parallel port (SoftGenLock)

Graphics Cards

Performance: hardware + driver issue

Active Stereo Support:
- Hardware + driver approach (Wildcat)
- Software approach (SoftGenLock)

Genlock Support:
- Hardware approach (Wildcat)
- Mixed hardware + software approach (Artabel)
- Software approach (SoftGenLock)
Graphics Cards

Performance: hardware + driver issue

PCI:
- Several slots per mother board (are the drivers able to accelerate 3D graphics with multiple boards?)
- Some graphics cards not available for PCI
- Support more electrical power demand from graphics card

AGP:
- One slot per mother board
- Higher bandwidth than PCI, more watts

VGA versus DVI
- DVI a better choice for video quality but be sure all components along the video path support it
Advanced Issues and Solutions

Screen material

**Gain:** same as an RF antenna: you don't really “gain” light, you redirect light straight forward, so...

higher gain screens appear brighter straight on, but off-axis viewing suffers

---

Advanced Issues and Solutions

Screen material

**White screen/Black screen??**

In an immersive environment with adjacent screens, images can “bleed” onto adjacent screens, so black screen material is superior, but...

black screens somewhat attenuate projector light relative to white screens and really attenuate IR used for remote control, so actual “correct” solution depends on application
Advanced Issues and Solutions

Screen material
if application is active stereo:
“blast” IR through a black screen
with vast numbers of IR sources or
one amplified source or...
forget IR and let radio waves do the job

Screen material
if application is passive stereo,
select a screen that:
maintains either the circular or
linear polarization you are using and
helps with the “no real black level”
problem of LCD or DLP projectors
Advanced Issues and Solutions

**Identifying and eliminating magnetic-field and RF interference**

**Why?**

Magnetic tracking technology VERY sensitive to resident near-line-frequency AC fields

Even if your wireless gear adapts to frequency congestion, performance suffers

---

**Advanced Issues and Solutions**

**Identifying and eliminating magnetic-field and RF interference**

**Where?**

Variable-frequency motor drives operate from 30-400 hz and spread harmonics well into the ultrasonic range

Cordless phones and microwave ovens can overwhelm frequency bands used for wireless ethernet, joysticks and headphones
Useful devices

- Wireless devices including 802.11b, 900Mhz, infrared control, wireless motion tracking.

- Serial interface controllers: projectors, switches, lights, trackers, audio, video switching, etc.

- Sound systems: Tracked head phones versus environment speakers.
Wireless channels

- 900 MHz
- 2.4 GHz
- 802.11, 802.11b Ethernet
- Infrared (“IR”)
- Custom radio
- Bluetooth: rapidly evolving

900 MHz

**Consumer gamepads, headphones, telephones**

- up to 115 Kbps bandwidth, <100 μsec latency.
- Multiple devices may coexist (~80 channels)
  - Linx Technologies “HP Series-II”: add 900 MHz to your own analog or digital devices!
2.4 GHz

“Better” than 900 MHz, costs a bit more
• Longer range, better penetration
• Much more bandwidth: 2–11 Mbps
  • Can even stream video (e.g., www.x10.com cameras).

802.11

Ethernet over 2.4GHz-band radio link
• 1–2 Mbps bandwidth; 802.11b is 5–8 Mbps.
• Latency is a few msec (“ping”)
  • Higher on a loaded network.
• $60 per PCMCIA card
• Multiple devices can coexist
  • But latency and bandwidth degrade.
• Used by wireless motion trackers
Custom radio

Serial data transmitter/receiver

• **Linx Technologies LC series Tx/Rx**
  - Self-contained, 300-400 MHz, tiny, about $20.
  - 5 Kbps bandwidth, <10 μsec latency.
  - Nondestructively changes CrystalEyes sync from IR to RF.

• [www.linxtechnologies.com](http://www.linxtechnologies.com)
• [https://courses.ece.uiuc.edu/ece345/cgi-bin/view_project.pl?fall2001_22](https://courses.ece.uiuc.edu/ece345/cgi-bin/view_project.pl?fall2001_22)

Infrared

**Inexpensive (built-in IrDA is free)!**

• **9.6 Kbps – 4 Mbps bandwidth; 0.1 – 10 μsec latency**
  - Many manufacturers, many specifications.

• **Multiple devices can’t coexist**
  - Unless they send commands sporadically, like a TV remote control.

• **Incompatible with CrystalEyes IR sync**
Serial interface controllers

- Video Projectors
- Room lights (www.lutron.com)
- Motion trackers (Ascension, Polhemus)
- Audio (MIDI control of mixers, etc.)
- Video matrix switchers (www.extron.com)

- Convert any of these to wireless with a “radio modem.”

Sound

Headphones or speakers?

- Plain old headphones
- Motion-tracked headphones
- Multiple motion-tracked headphones
- Loudspeakers
- Many loudspeakers
Motion-tracked headphones

Accurate 3D position
- Turn your head and the sound stays in place
- What if someone else doesn’t turn their head?
  - Sound world and visual world become decoupled.
  - Tracking only orientation of extra headphones solves most of this problem, e.g. flux-gate compass.

Speakers

- Speakers are to Headphones what Screens are to Head-Mounted Displays
  - 1 device for $n$ users, vs. $n$ devices for $n$ users.
- Can’t make a sound closer than the closest speaker.
- Easier for users to talk to each other
- Mechanical difficulties
  - Avoid echoes from large video screens.
  - Screens and speakers shouldn’t block each other.
References

- **Flux-gate compass**
  - TCM2 compass module from Precision Navigation, Inc.

HW Management Tools

- **VR Cluster:**
- Even more complex environment
  - Many more interesting pieces to plug
  - Intrinsic heterogeneous distributed environment

What to manage?
- The cluster itself and ...
- Everything else plugged in the cluster
  - Video routing, Cluster nodes, Networking, I/O Devices, projectors, trackers, Environment (air cooling, lights)
HW Management Tools

- **Pandemonium**
  - A multiprojection console
  - 6 21” stereo monitors
  - Control Table

- **Zeus**
  - Management server
  - C++ and Linux

- **Sirius**
  - Cross platform visual interface (Java)
Cluster computing

- Cost effective solutions for high performance technical computing
- Applications: weather forecast, bioinformatics, oil production, etc
- Thousands of small clusters worldwide
- Clusters up to 10,000 nodes
Cluster components and software

Service node

- NFS (Network File System)
- NIS (Network Information System)
- NTP (Network Time Protocol)
- Cluster Management Software: other slide
- PBS (Resource management and scheduling)
- Accounting, licensing, storage, etc
- GFS (Global File System)
Frontal node

- **OpenPBS + patches** ([http://www.openpbs.org](http://www.openpbs.org))
  - Job submission
  - for non interactive clusters

- **OpenSSH/OpenSSL**
  - Secure Transactions

- **Monitoring tools**
  - for large clusters
  - hardware failure detection

- **File staging - other slide**

Compute nodes: standard environment

- **Language:** C, C++, Fortran, Java
- **Compilers:** GNU, Intel, PGI, NAG, Absoft, Futjitsu, Code warrior
- **Make, CVS**
- **Debugger:** gdb, totalview ([http://www.ETNUS.COM](http://www.ETNUS.COM))
Compute nodes: cluster middleware

- **MPI, the standard message passing library**
  - **mpich** high portability for UNIX, NT/Win, Globus
    (http://www-unix.mcs.anl.gov/mpi/index.html)
  - **lam** high availability, performances
    (http://www.lam-mpi.org/)
- **PVM parallel virtual machine**
  (http://www.epm.ornl.gov/pvm/)
- **Distributed Objects (CORBA/DCOM/RMI), RPC, sockets, Shared memory over network**
- **Profilers:** MPE (MPICH), Paradyn, Vampir

Compute nodes: math library

- **MKL** Linear algebra, fast Fourier transforms and vector math functions optimized by Intel itself.
  (http://developer.intel.com/software/products/mkl/)
- **Atlas** adaptive software architecture that provides Linear Algebra libraries.
  (http://www.netlib.org/atlas/)
- **FFTW** Fast Fourier transforms
  (http://www.fftw.org)
- **SCALAPACK** Linear Algebra
  (http://www.netlib.org/scalapack/)
- **PETSc** data structures and routines for both sequential and parallel scientific computing.
  (http://www-fp.mcs.anl.gov/petsc)
- **Commercial solutions:** NAG, IMSL
Compute nodes: parallel math library

- SCALAPACK: Scalable Linear Algebra Package
- PBLAS: Parallel Basic Linear Algebra Subprograms
- LAPACK: Linear Algebra Package
- BLAS: Basic Linear Algebra Subprograms
- BLACS: Basic Linear Algebra Communication Subprograms

Message Passing Interface (e.g., MPI, PVM)

Cluster installation

Hardware issues: choose PC boxes, HVAC, network

Software issues: choose OS and cluster distribution

Install one node with OS, development tools and programming library

Clone nodes
Hardware issues

- Individual components:
  - Chipset
  - Memory
  - CPU
  - AGP bus
  - PCI bus
  - Network
  - Disks
- Application and budget?
- Trade-offs

Hints

- Heat sink for high-end CPUs and GPUs
- Extra hardware helps cluster management: KVM switches, multiserial, etc
- Dual cpu boards save space and money but contention
- Check PXE compliance and performance
- Node and cluster benchmark suites (http://www.xtreme-machines.com/bps/)
Software

- **Linux for cluster computing:**
  - de facto standard
  - Middleware and management tools
  - open source

- **Linux/Windows for graphics:**
  - depends on drivers/middleware/hardware availability (ex: Myrinet, Wildcat)

Cluster Distribution

- **Linux Red Hat (or Mandrake, etc) +**
  - OSCAR * open cluster group * initiative (http://oscar.sourceforge.net)
  - NPACI kickstart based solution, (http://rocks.npaci.edu/index.php)
  - Score Myrinet based distribution (http://pdswww.rwcp.or.jp/dist/score/)
  - Ka scalable cloning tool (http://ka-tools.sourceforge.net)
  - Commercial solutions: Scyld (http://www.scyld.com/)

- **Windows 2000**
  - Remote Installation Service (RIS), Norton Ghost, Ka
Installation (1)

- Gather all information describing the cluster:
  - Names
  - IP addresses (private addresses)
  - Partition layout
- **Standard Linux install on one node**
- **Install the cluster management software on the node**

Installation (2)

- **Fill cluster database**
  - Build image per client type (partition layout, HD type)
  - Define clients (network info, image binding)
  - Setup networking (collect MAC addresses, configure DHCP, build boot floppy)
- **Start automatic installation**
  - Boot clients / build
Installation (3)

- **Complete setup with post install:**
  - Should be reboot for Linux
  - tricky for Windows
- **Install benchmark suites:**
  - bps, povray, linpack
- **Upgrade operations:**
  - Clone/install nodes again
  - Software distribution tools

Software/data distribution

- **C3** parallel rsh, rcp and other utilities
  (http://www.csm.ornl.gov/torc/C3/)
- **Ka-tools** scalable rsh and rcp
  (http://ka-tools.sourceforge.net)
- **NFS, prsh**
  for small clusters
Monitoring

• **System manager:**
  - PCP seamless access to all system metrics (http://www.sgi.com/software/co-pilot)
  - ganglia cluster global view (http://ganglia.sourceforge.net)
  - **hardware monitoring tools** (http://www.netroedge.com/~lm78/index.html)

• **User:**
  - PCP
  - Graphical PBS interface
Software porting

• Needs:
  • Sockets interface
  • Posix interface
  • Process control and management
  • System V IPC
  • Pathname mapping from UNIX to Windows
  • Memory mapping and shared memory
  • etc

Software porting

Cygwin

A UNIX environment for Windows.

• A DLL which acts as a UNIX emulation layer providing substantial UNIX API (threads, sockets, shmem, etc)

• A collection of tools, ported from UNIX, which provide UNIX/Linux look and feel (X11, shell, emacs, etc)

• Links with win32 libraries

http://www.cygwin.com/
Software porting

Win32 : native windows environment

• ANSI C libraries
• Win32 threads + wrapper for personal needs
• Winsock2 + wrapper for personal needs
• Cross-platform parallel programming: MPI, PVM
• Remote access tools: ssh, VNC

Software porting

Abstraction layers

• Red hat posix threads for win32, POSIX 1003.1 2001 Standard
  (http://sources.redhat.com/pthreads-win32/)
• Commercial : Uwin, Microsoft Interix
  BSD sockets interface, posix implementation, etc.
Software maintenance

• **Compilers:**
  • Intel, PGI, Code warrior
  • Cygwin gcc

• **Makefiles:**
  • tmake maintain makefiles for multiple platforms
    (http://www.trolltech.com/developer/download/tmake.html)
  • Cygwin make

• **CVS** windows client (http://www.cvshome.org/)

Software maintenance

• **Cross-platform libraries**

  • **Math libraries:**
    • Intel MKL
    • ATLAS

  • **Parallel libraries:**
    • MPI, PVM
    • CORBA, JAVA
Module III Summary

- Software Architecture Overview
- WireGL Case Study
- Net Juggler Case Study
- CORBA-based Application Distribution Case Study
- Syzygy Case Study
- Examples of implementing applications
Module III Objectives

- Understanding many ways to implement distributed VR software
- Deeper analysis of specific packages
- Some examples of applications
Software Architecture Overview

- Enumeration of cluster-based graphics software technologies
- Classification of existing technologies under the framework presented in Module I
- Advantages and disadvantages of each for software development

Software Classification

- **Master/slave**
  - Identical copies of the code run on each render node.
  - Low bandwidth requirements.

- **Client/server**
  - Generic clients run on the render nodes and are fed by data from a server
  - Further differentiated by the kind of data transmitted: Pixels, graphics primitives, high-level data (like scene graph)
Existing Free Software Technologies

- WireGL (Chromium)
- Princeton Omnimedia Group
  - DGL (distributed OpenGL)
  - Application synchronization API
- VNC
- Net Juggler
- Syzygy
- Cluster Juggler
- DICELib

WireGL (Chromium)

- Replacement OpenGL library that sends OpenGL commands over the network to be rendered
- Optimizations to reduce bandwidth
- Chromium allows processing of OpenGL for various effects
- Client/server: graphics primitives
- Originally developed for tiled display walls. Graphics only.
Princeton Omnimedia: DGL

- Replacement OpenGL library that sends OpenGL commands over the network to be rendered
- Optimizations to reduce bandwidth
- Client/server: graphics primitives
- Originally developed for tiled display walls. Graphics only.

Princeton Omnimedia: Sync API

- Originally designed for tiled display walls
- Several API calls, including:
  - Barrier + broadcast value (allows for a collective function call)
  - Which tile am I?
  - What’s the view frustum?
- Master/slave
- Group has done work with fine-grained synchronization via system API replacement
- Graphics only
VNC

- Can be used to drive a tiled display
- Make a large virtual framebuffer and ship pixels to vncviewer’s on each tile
- Client/server: pixels
- Adaptability to non-rectangular displays as are prevalent in VR?

Net Juggler

- Cluster support for VR Juggler
- MPI based
- All nodes keep the same status
- Data-lock provided by all-gather collective communication
- Can handle more than just graphics because built on top of VR Juggler
- Tiled walls, HMDs, and CAVE-like environments
Syzygy

• Framework for cluster-based multimedia
• Integrated distributed application management
• I/O device management
• Native scene-graph API (client/server: high level) and general framework for building master/slave applications
• Tiled walls, HMDs, and CAVE-like environments

Cluster Juggler

• A cluster-enabled extension to VR Juggler, produced by VRAC
• Works on heterogeneous clusters
• Enables migration path for VR Juggler applications to clusters
• Persistent input managers
DICELib

• **Distributed Cave Engine Library**
  - Master/slave
  - Minimally invasive cluster library
  - Includes functionality for sharing data and efficient synchronization
  - Designed to work over low-bandwidth TCP/IP networks
  - I/O support, for instance joysticks

**Features:**

• Efficient synchronization and data sharing (Data locking)
• Low processing consumption
• Easiness to adapt existing programs to use it
• Multicast support instead of point-to-point
Existing Commercial Software Technologies

- SGI graphics cluster
- Hewlett-Packard Visualization Center
- Multigen-Paradigm
- Artabel

Hewlett-Packard Visualize Center

- Hewlett-Packard Visualize Center
  - 3D SLS (single logical screen)
  - Broadcasts OpenGL over gigabit ethernet from server to display clients
  - Client/Server: graphics primitives
**SGI Graphics Cluster**

- **SGI graphics cluster**
  - Data sync API provided
  - Programming style open-ended

**Multigen Paradigm**

- Extends the Multigen Vega library, a visual simulation toolkit
- Master/slave
- Default configuration is to transmit input events. But this can be disabled to accept data from a simulation host.
- Uses TCP and UDP (via the ACE framework)
Artabel: Fleye Series

- Graphics clusters that are compatible with OpenGL applications
- Client/server: distribution of graphics primitives
- Swap-Lock and genlock provided
- Administrative middleware

Implications for Development

- OpenGL wrappers: most transparent for user. Do they support non-tiled displays and stereo?
- Master/slave based on wrapping a well-known API, for instance Net Juggler or an extension of GLUT. Again transparent, but there are often programming restrictions (no threads, application changes state only in response to wrapped event loop)
- Using a new API (Syzygy or SGI graphics cluster APIs). Not transparent but potentially more flexible.
WireGL/Chromium

- The client acts as a graphics source generating graphics commands.
- The server machines act as graphics sinks, receiving graphics commands.
- The OpenGL application does not need to be modified.
WireGL Project

- Distributed parallel OpenGL interface.
- General tiled OpenGL interface that generates network streams for a distributed display.
- Networking abstraction for supporting different SAN technology/API’s.
- Extensible network server that manages asynchronous reception of work quanta.

Execution

- Each rendering server is connected to an output device, typically a projector, which are arranged to create a tiled display.
- The application will dynamically link with the WireGL client driver, ...
- ... and begin transmitting graphics commands to the rendering servers.
Chromium Project

• Adds additional mechanism, Stream Processing Units (SPU).
• Each SPU implements a complete OpenGL API.

WireGL Features

| PROS                  | Uses unmodified OpenGL applications.  
|                       | Minimizes re-transmission of graphics data. 
|                       | Reduces the communication, but… |
| CONS                  | High network bandwidth. 
|                       | Add some non-standard additional OpenGL calls |
WireGL

Tiled Image

WireGL Client

WireGL Primitives

OpenGL Server

OpenGL Server

Possible Extensions

Stereo Image

WireGL Client

WireGL Primitives

OpenGL Server

OpenGL Server

2 images: left eye and right eye
Possible Extensions

Cubic Image

WireGL Client

OpenGL Server

OpenGL Primitives

OpenGL Server

OpenGL Server

References

- http://sourceforge.net/projects/chromium/


- WireGL: Software for OpenGL Tiled Rendering http://graphics.stanford.edu/software/wiregl
Net Juggler Case Study

- VR Juggler
- Net Juggler: distributed rendering for VR Juggler applications
- SoftGenLock: Software genlock and active stereo for Linux clusters
- Coupling parallel numerical simulations and parallel rendering with Net Juggler
VR Juggler

- Open Source Platform for VR, VRAC, Iowa State University
- A user and developer community getting bigger every day: www.vrjuggler.org
- Abstraction of the I/O devices
- Direct access to graphics API:
  - OpenGL, Performer, Open Inventor, OpenSG
- Linux, Windows, IRIX, Solaris, HP Unix

VR Juggler

- Simulator Mode
- Performance Monitoring
- On-line Configuration
  - Configuration Files
    - Tracker + Displays + ...
  - VJ Application
    - vjUser, vjButton, ...
VR Juggler
ConfigChunks

Configuration data are organized in ConfigChunks:

JugglerUser
  Name "User1"
  headPos { "VJHead" }
  interocular_distance { "0.229" }
end

VR Juggler GUI:
vjControl
Clustering Solutions for VR Juggler

Data Lock at

- Input Event Level:
  - Cluster Juggler
  - Net Juggler
- Scene Graph Level:
  - OpenSG
- Graphics Primitive Level:
  - WireGL

Net Juggler

- Open source: netjuggler.sourceforge.net
- Team:
  - Jérémie Allard, Valérie Gouranton, Loïck Lecointre, Sébastien Limet, Souley Madougou, Emmanuel Melin, LIFO, Université d’Orléans, France
  - Bruno Raffin, Lab ID-Imag, Grenoble, France
Net Juggler

**Performance Monitoring**

**Simulator Mode**

**On-line Configuration**

**Configuration Files** (add each PC role)
- PC1 -> Tracker + left display
- PC2 -> Front display
- PC3 -> Wand + Right display

**Net Juggler**

**VJ Application**

**Net Juggler**

**Tracker event**

**Wand events**

**Configuration events**

**Net Juggler communication layer (data lock)**

**VJ Application**

**VJ Application**

**VJ Application**

**Net Juggler**

**swaplock**
Net Juggler Communications

- Data in transit (device + config events):
  - Small amount
  - Do not dependent on the graphics complexity of the scene

- Communication Layer implementation:
  - Data aggregation
  - Optimized collective communications

  Scalable
  Low bandwidth required
  Fast Ethernet Network sufficient

Running a VR juggler Application on a Net Juggler Cluster

- Add kernel->parseArg(argc, argv) in application main

- Modify the Makefile

- Check the application retrieves all input devices through VR Juggler input devices:
  - Position data, time, random numbers, ....
Net Juggler Cluster Configuration Files

From existing VR Juggler configuration files:

• Add a host attribute to each ConfigChunk
  • host : "All"
    → ConfigChunk applied to each node
  • host: "pc1"
    → ConfigChunk applied to node pc1
• Change Proxy ConfigChunks into SharedProxy ConfigChunk

Application Launching

From existing VR Juggler configuration files:

• Use an extra PC as a console
• Remotely launch X on all nodes (use the Ka-tools):
  • pX –f nodelist
• Launch the application:
  mpirun n1 n2 n3 cubes cluster.base.config\cluster.display.config
  (the syntax may vary depending on the MPI implementation used)
Application Launching

Cave Quake 3 running on a Net Juggler Cluster

Blurry effect: active stereo

vjControl

- **Dynamics Reconfiguration**
  
  Use vjControl to connect to Net Juggler (default port 4451), check the cluster configuration and modify it

- **Performance Monitoring**
  
  Use vjControl to connect to a node of the cluster (default port 4450) and retrieve performance data from that node
Net Juggler: Hardware Requirements

Cluster Hardware

- PCs + graphics cards (check driver quality) + Fast Ethernet network (100 Mbits/s)

Input Devices

- Net Juggler supports any input devices VR Juggler and the OS also support
Net Juggler: Software Requirements

VR Juggler patched
- Patch included in the Net Juggler distribution

Graphics API
- Any Graphics API VR Juggler supports (OpenGL, Performer, OpenSG, Open inventor,...)
- Remote command execution (rsh or ssh)
  - Required by mpirun and the Ka-tools
- MPI (Message Passing Interface)
  - Over TCP/IP we advice to use lam-mpi or mpich
  - MPI is implemented on top of almost any kind of network

Passive Stereo

- Left and right views separated by light polarization
- 1 PC + 1 video-projector per eye and per projection surface
- One SurfaceDisplay ConfigChunk per PC with a “Left eye” or “Right Eye” view
Passive Stereo

• Alternate projection of the right and left eye images synchronized with shutter glasses

• With CRT projectors:
  • Image flipping during video blanking

• With multiple projectors video retraces must be synchronized (genlock)

Active Stereo

• Alternate projection of the right and left eye images synchronized with shutter glasses

• With CRT projectors:
  • Image flipping during video blanking

• With multiple projectors video retraces must be synchronized (genlock)
**SoftGenLock**

Software approach to enable active stereo and genlock on Linux clusters
([netjuggler.sourceforge.net](http://netjuggler.sourceforge.net))

→ Works with potentially any graphics card
  (use VGA registers)

---

**SoftGenLock**

Video blanking:
- Send flip signal to glasses
- Get ready to display the right eye image
SoftGenLock

Video blanking:
  Send flip signal to glasses
  Get ready to display the left eye image

Net Juggler + SoftGenLock

• Launch X with a double horizontal resolution:
  Modes "1024 768"
  Virtual "2048 768"

• Define 2 SurfaceDisplay ConfigChunk per PC
  • A left eye display with origin (0,0)
  • A right eye display with origin (1024,0)

• Launch the application: mpirun ...

• Launch SoftGenLock: insmod softgenlock.o
Installing SoftGenLock

• PC to shutter glasses cable:
  SoftGenLock sends the sync signal through the parallel port
  ➡️ Build a DB25 (male) – mini-din 3 (female) cable

• Real time linux kernel (Rt Linux or RTAI)
  SoftGenLock must be triggered at very precise times (vertical blanking)
  ➡️ SoftGenLock runs as a RT Linux or RTAI real time kernel module

Synchronization Network

• SoftGenLock requirements for synchronization:
  • Fast (few tens of micro-seconds)
  • Sync barrier can be called form a real time module
Synchronization Network

Solution: A parallel port based synchronization network

- PC-to-PC null modem cable (6 μs between two machines
- TTL_PAPERS like sync network (www.aggregate.org)
  - PCs are connected to a central box through their parallel port
  - Boxes are tree assembled for large clusters
  - Very high performance and scalability:
    \( T_{sync} = (7 + 0.1 \log_8 p) \mu s \)

Net Juggler

- Net Juggler duplicates the application:
  Low communication overhead but some data and computations duplicated
  Ok for 80% of the applications
  \( fps \text{ on one PC} = fps \text{ on 8 nodes PC cluster} \)
- In some cases, performance can be improved by distributing data and computations instead of duplicating them
  Beware of the cost of extra communications
Net Juggler

Tracker event

Configuration events

Wand events

Net Juggler communication layer (data lock)

Net Juggler communication layer (data lock)

VJ Application - Distributed part

VJ Application Duplicated part

VJ Application Duplicated part

VJ Application Duplicated part

The Fluid Application
The NjFluid Application

- Fluids simulation: Stam’s Navier-Stokes equation solver
- 2D grid distributed on the nodes + parallel Navier-Stokes equation solver
- Implementation using the PETSc math library (based on MPI)
- Net Juggler: a framework to coordinate distributed rendering and distributed pre-rendering

Results

- 6 dual PIII-800 MHz 2D
- GeForce 2 GTS 64 MB
- 100 Mbits/s Fast Ethernet Network
- 128x128 grid

<table>
<thead>
<tr>
<th></th>
<th>1 display</th>
<th>4 displays</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solver duplicated</td>
<td>8 fps</td>
<td>8 fps</td>
</tr>
<tr>
<td>Solver distributed</td>
<td>21 fps</td>
<td>20 fps</td>
</tr>
<tr>
<td>on 4 nodes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Solver distributed</td>
<td>24 fps</td>
<td>22 fps</td>
</tr>
<tr>
<td>on 6 nodes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CAL’s Demos

- **SoftGenLock:**
  - Active Stereo end genlock with GeForce cards

- **Net Juggler:**
  - The fluid application
    - Coupling a parallel numerical simulation and parallel rendering with Net Juggler
  - Hufo Fur
    - Per pixel lightning for fur rendering using GeForce hardware

HUFO Fur
Module III: Section D
CORBA Case Study

Marcelo Zuffo
mkzuffo@lsi.usp.br

CORBA Case Study

Common Object Request Broker Architecture

- CORBA-based distributed VTK
- CORBA-based distributed OpenGL
- CORBA-based distributed JAVA
CORBA

- Distributed Object Middleware
- Complete specification for object oriented distributed application development
- Many languages: C, C++, JAVA, ADA ...
- Many platforms: Unix, Windows, ...

Why CORBA?

- CORBA is more interoperable than Remote Procedure calls (RPC) and Remote Method Invocation (RMI)
- CORBA is a open standard widely accepted in industry
- Other similar middleware:
  - Microsoft DCOM
  - SUN EJB
When to Use It?

- Heterogeneous cluster environments
- Heterogeneous application environments
- Object oriented programming
- Heterogeneous programming environments

CORBA 3.0

Asynchronous Messaging and Quality of Service Control
- Clients and objects may control ordering
- Priority, deadlines, and time-to-live
- Start time and end time for time-sensitive invocations
- Control routing policy and network routing hop count

Real-Time CORBA
- Priority models to achieve predictable behavior for both hard and statistical realtime environments
Performance Issues

The performance can be improved by using customized and optimized ORB implementations:

- Optimized Request de-multiplexing and dispatching
- Optimized memory management coupled with these techniques
- Optimized communication protocol
- Optimized presentation layer

VR Cluster Middleware

- CORBA does not directly support most of the cluster middleware functionalities.
  - It can be extended to act as a cluster middleware.
- The features of the CORBA that will facilitate such extension are:
  - A large set of communication model
  - Diverse set of services
  - Easy integration with other technologies.
VR Cluster Middleware

• **Process**
  - Support Single Image in terms of process space using the Life Cycle Service to implements a interface.

• **Memory**
  - CORBA offers a large set of synchronization model and support for many programming languages.

• **I/O System**
  - It is possible use Jini, which is a Java Technology, to provide a Single I/O space in a distributed environment.

VR Cluster Middleware

• **Parallel Programming**
  - Implementation of MPI-Based CORBA

• **High Availability**
  - Hot Standby
  - Active Replication
Master/Slave Design

VTK Application
Java / C++
Event Service

VTK Application
Java / C++
Event Service

VTK Application
Java / C++
Event Service

CORBA

Environment Architecture

Event Channel
Node 1
Node 2
...

Interaction 1
Interaction 2
Interaction 3
...
CORBA, C++ and OpenGL

Tiled Image

CORBA, Java and VTK

CAVE Immersive Environment
Section Bibliography


Syzygy

- Design overview
- Distributed Scene Graph
- Master/slave application framework
- I/O Device Integration using Syzygy
- Scaling down: simulators and other development methods using Syzygy

Syzygy Snapshots

http://www.isl.uiuc.edu/
The Problem

- Cluster-based VR leads to a bewildering variety of computers and devices!
- Management: How to quickly move from demo to demo... if each requires an entirely different set of software resources?
- Connections: How to integrate the different devices? The system must be robust, self-assembling, and self-healing!
- Options: Severe communications bottlenecks mean many approaches to cluster-based VR are appropriate.

Syzygy Architecture
Communications Layer

- Message based. Binary format.
- Automatic translation between machine architectures (endianess and byte alignment)
- Server, client objects with extensive connection management functionality.
Communications Layer

• **Distributed barrier objects.**

• **By default, connections can occur in any order between objects, be broken, reformed, etc. Robust.**

• **Includes simple abstraction layer to ease writing single source Unix/Win32 applications.**

Phleet

• **Distributed operating system (minimal)**

• **szgServer controls.**

• **szgd provides remote execution services.**

• **szgServer is merely a connection broker. Most communication occurs directly between programs.**
Phleet

- Parameter database stored in szgServer. Meta-config file that can be managed from the command line of any computer.
- Global locks
- Simple message API routes through szgServer to all managed programs (like Unix signals). Can tell a program to reload its parameters or exit.

Media

- Protocols not dependent on Phleet. Can implement glue code yourself (use only what you want).
- Built using the tools in the communications layer. Based on synchronized message buffer transfers.
Media

- Distributed scene graph
- Master/slave application framework
- Sound
- VR-specific functionality (like calculation of view frusta)
- When integrated using phleet into media objects, one has a powerful self-configuring, fault-tolerant system.

Distributed Scene Graph

- Client/server. One node serves geometry to the render nodes.
- Bandwidth is automatically conserved. Only scene changes are sent from one frame to the next.
- Semantic nature of scene graph helps eases programming.
- Robust. Render clients can disconnect and reconnect while the application is running. Dynamically change your display. Similarly, stop your application and start a new one without bringing down the render clients.
**Master/Slave Application Framework**

- **Master/slave = same application runs synchronized on render nodes.**
- **One node is the master and distributes I/O or other control info.**
- **Framework makes it easy to write a synchronized application, handle input devices, and manage the whole thing using phleet.**
- **A good way to write a custom distributed graphics protocol!**
Master/Slave Application Framework

I/O Device Integration

- **Built on the same communications layer as everything else.**
- **Robust client, server objects make it easy to get device data onto the network and into your program. Put that Windows joystick on the network!**
- **Phleet allows configuration of the multitude of I/O devices and helps them make connections.**
Simulators

• Need a reasonable replacement for the VR device in order to do debugging, etc.
• Random PCs around the office or lab provide multiple screens. A “wand sim” program lets you generate tracker data from the desktop. Really, we only simulate the input device. The rest of the system is unchanged.
• Hard guarantees ensure that if your program runs correctly on one graphics pipe, it will run correctly on 6 or more.

Free Software

• Syzygy is licensed under the GNU LGPL
• It comes with everything you need to start showing demos right away!
• Download the latest release at www.isl.uiuc.edu (follow the software link)
Module III: Section F
Examples of Implementing Applications

Benjamin Schaeffer
schaeffr@isl.uiuc.edu

Application Implementation

- **Entertainment**
  - CAVE Quake 3
  - CARMEN: a shared virtual environment

- **Scientific Visualization**
  - Mathematics: visualization of hyperbolic geometry
  - Supercomputer performance analysis

- **Medical**
  - Realtime volume visualization
  - Isosurface visualization
CAVE Quake 3

- Steve Taylor wrote a GPL viewer for Quake3 levels. Paul Rajlich made the code VR-friendly.
- The view only depends on eye position, view direction, and timestamp.
- Simple matter to use Syzygy’s master/slave framework to create an application that can run synchronized on a cluster.
- Very small amounts of data exchanged per frame. Can run well on a slow network, basically at the speed of a single render box.
- Access to source code critical.

http://www.isl.uiuc.edu/
**CARMEN: shared virtual world**

- Two people interact in a replica of a real outdoor space with a third who actually inhabits that space and perceives the others via augmented reality.
- One participant uses an HMD powered by a small graphics cluster. Visuals provided by Syzygy’s distributed scene graph. (client/server: high-level)
- Infrastructure reuse: the same code base used for the client/server style of distributed graphics can be reused to provide a reliable backbone for maintaining a shared virtual world and serving the world state to the various participants.

[http://www.isl.uiuc.edu/](http://www.isl.uiuc.edu/)
Mathematics: Hyperbolic Geometry

- George Francis and others at the UIUC had developed a CAVE program to visualize and navigate through 3D hyperbolic geometry.
- The scene depends only on a navigation matrix.
- Master/slave framework used to implement the cluster-based version.
- Strip out the CAVE-specific code and implement a new main loop based on the draw code, distributing the nav matrix, and new input device handling routines (simple).

http://www.isl.uiuc.edu/
Mathematics: Hyperbolic Geometry

http://www.isl.uiuc.edu/

Supercomputer Performance Analysis

- Virtue: project of the Pablo group at UIUC (Dan Reed). Performance analysis of Grid applications (in Virtual Reality).
- Included a “time-tunnel” metaphor for visualizing traces of MPI codes.
- Cut the time tunnel module from the rest of the Virtue code. Now had something independent of CAVElib.
- Used Syzygy’s distributed scene graph to display the highly animated MPI trace. Easy to map trace animation events to scene graph changes.
- Just sending needed scene changes (instead of whole scene) is critical in keeping bandwidth usage minimal.
Supercomputer Performance Analysis

http://www.isl.uiuc.edu/

Real-time Volume Visualization

- Want to be able to display real-time volume data instead of polygonal data.
- First look at static data sets. Do a custom renderer (texture slice based) for the volume viz.
- Use the master/slave framework to synchronize the custom render apps.
- Shows that master/slave can add application flexibility.
Real-time Volume Visualization

- Visible Human Dataset: A High Resolution volumetric data from the human body.
- This data is very rich in texture and anatomical information, requiring direct volume rendering techniques
- Using an extra volume rendering board (the VP500 (www.terarecom.com))
- Native C++/OpenGL Windows Application
- Data-lock provided by DICELib

Visible Human Analysis

http://www.lsi.usp.br/~caverna
Isosurface Visualization

- Instead of real-time voxel data, look at isosurfaces generated offline from brain data.
- Client/server programming method, using distributed scene graph.
- Since we're viewing a static object, the only thing changing is the POV (a single matrix in the scene graph).
- Since the distributed scene graph server only sends updates to connected clients, the bandwidth used is small.

Architectural Walkthrough

- A standard application in architecture is walkthrough high resolution models
- We build a 3D-Max – OpenGL converter and a walkthrough navigator
- Native C, OpenGL Windows/Linux
- Data-lock provided by DICELib
- The SIBENIK Cathedral
  - ~90,000 Polygons Active Stereo
  - Model provided by Marko Dabrovic (mdabrov@rna.hr)
Architectural Walkthrough

http://www.lsi.usp.br/~caverna

Oil Mining in Depth Ocean

- Simulation of oil platforms in high sea, considering the ocean dynamics upon risers (oil pipes)
- Native OpenGL Performer implementation in Linux
- Synchronization provided by DICELib
  - 150K polygons
- Data courtesy from Petrobras S/A and Escola Politécnica da USP.
Depth Ocean Oil Extraction Simulations

http://www.lsi.usp.br/~caverna

Depth Ocean Oil Extraction Simulations

http://www.lsi.usp.br/~caverna
Bibliography


Bibliography


Bibliography


Bibliography


• NCSA. Display Wall in a Box. www.ncsa.uiuc.edu/TechFocus/Deployment/DBox


Bibliography


Bibliography


Module IV Objectives

- To have a practical experience with PC cluster VR
- Hands on:
  - Systems Overview
  - Configuration
  - Applications

Lab Hardware

- Low-end cluster: 2 PCs, single CPU
- Mid-range cluster: 4 PCs, single CPU
- High-end cluster: 6 PCs, dual CPU
- Network options: gigabit & 100Mbit ethernet, Myrinet
- Graphics options (hardware/software genlock, active/passive stereo)
Lab Hardware

• Multi-screen VR display: A two-screen projection VR display with active stereo and magnetic tracking, driven by 3 PCs.

• Wireless game controllers, wireless system administration devices (iPaqs with Java interfaces), joysticks, loudspeakers, etc.

• System Management, how to have control over applications, configurations and devices.

Acknowledgements

• Authors wish to thanks the following organizations:
  • Beckman Institute of the University of Illinois at Urbana-Champaign
  • CNRS, INPG, INRIA, UJF
  • Escola Politécnica da Universidade de São Paulo
  • Intel do Brasil
  •FINEP Financiadora de Estudos e Projetos, MCT-Brasil
  • Université d'Orléans