LBNL Visualization Research Program
High Performance Remote Visualization
*Highlights 2005-2007*

E. Wes Bethel
Lawrence Berkeley National Lab
12 Feb 2008

www.vacet.org
Outline

• Remote visualization: definition, approaches.
• MBender: multiresolution remote vis.
• Chromium Renderserver: high performance parallel, hardware-accelerated remote vis.
The Visualization Pipeline

- The Vis pipeline consists of an “assembly line” of components.
- Remote visualization is all about how that pipeline is partitioned between remote and local resources.
Pipeline Partitioning – Send Data

- All visualization, rendering and display happens on “the desktop.”
- Data or data subsets moved between remote and local hosts.
Pipeline Partitioning – Send Geometry

- All rendering and display happens on the desktop.
- Visualization results – e.g., isosurface triangles – moved across the network.
Pipeline Partitioning – Send Images

- All data I/O, visualization, and rendering happen on the remote host.
- Only image data moves across the network.
Remote Visualization

• Which pipeline partitioning works the best?
• Answer(s):
  – It depends on the problem and use model.
  – “Send Images” appears, in general, to be the best option.
• The following slides explain this issue in more detail.
Remote Visualization Performance Experiment

- Three partitions: send data, send geometry, send images.
Absolute Runtime of Three Partitionings

Pipeline execution time (seconds)

- Desktop only
- Cluster isosurface
- Cluster render

Isosurface triangles (millions)

0 0.5 5 50 500

0 0.03 0.51 1.10 2.00 1.01 2.09 0.75 0.32 0.64 19.73 11.00 1.88
Relative Performance of Three Partitionings
Analysis of Strategies

• Send images (Winner): nearly constant performance regardless of data size. Other advantages (to be discussed).

• Send geometry: performance worsens as data grows larger; implementation difficulties.
  – E.g., isosurface produces \( \sim O(N^{2/3}) \) triangles.
  – Not scalable to larger data, more users, more apps.

• Send data: performance worsens as data grows larger; implementation difficulties.
Two Remote Visualization Projects

• Both based on “send images” partitioning.
• MBender (“Media Bender”)
  – First run the application to generate pre-computed imagery.
  – Later, view/explore multidimensional, multiresolution imagery with web browser.
• Chromium Renderserver
  – Harvest/encode/transmit imagery from hardware-accelerated rendering of parallel vis app to remote viewer(s).
MBender – Basic Idea

- **Precompute ordered image sequences**
  - Ordering is sampling through vis or rendering space. E.g., Isocontour level, viewpoint
- **Encoding step**
  - Produce QuickTime VR Object Movies
  - Produce MBender Catalogue/Map files
- **Client-pull, vanilla web server**
  - JavaScript implementation – images, web browser
  - QTVR player for QTVR Object movies
  - MBender client for “the full Monty”
MBender – Industry Standard Clients

- **Apple’s QuickTime VR** *(example, requires network)*
  - “Inside” looking “out”
  - 3D Navigation

- **Apple’s QuickTime VR “Object Movies”**
  - “Outside” looking “in”
  - Two-axis + zoom scientific vis example *(requires network)*
  - One-axis + time scientific vis example *(requires network)*

- **Web Browser/JavaScript example**
  - 1D example *(requires network)*
  - 2D example *(requires network)*

vis.lbl.gov
About QTVR, JS Implementations

• Pro’s:
  – Industry standard “players”: web browser, QT player.
  – Industry standard “formats.”
  – All examples served up by “garden variety” web server – no special server-side config required.
  – Rapid, easy exploration of sci-vis results.

• Con’s:
  – Fixed image resolution.
  – Not friendly use of bandwidth or memory.
  – Some prep work required before use.
  – Fits many, but not all, remote vis use models.
MBender Motivation

• Overcome limitations of previous approaches:
  – Want support for multiresolution imagery.
  – Want more efficient use of client memory.
  – Want more friendly use of bandwidth.
  – Want support for more browsable dimensions.
    • Visualization parameter space.
    • Rendering parameter space.
MBender Demonstration

• **Demo** (requires network)
MBender – Performance Modeling

• Tile Size
  – Server-side storage requirements
  – Client download speed
• Multi-threaded client
  – Overlap I/O, rendering
• Client prefetching algorithm
Tile Size – Server-side Storage
Tile Size – Client Download Rate

- 128x128 overall winner across all networks, degrees of I/O parallelism
Prefetching Algorithm

- Objective: predict which frames a user will want to see and download them ahead of time.
- Client does navigation through $n$-dimensional parameter space.
- Present client supports 4-d navigation:
  - Pan (left/right/up/down)
  - Rotate (azimuth/pitch)
  - Zoom
  - Data browsing
- Different prefetch algorithm depending upon navigation use mode.
Prefetching Performance Improvement

- Test shows “delay time” – time client spends waiting for images to satisfy view.
- Prefetch produces 2x-3x improvement.
MBender – Post-Game Show

- Great idea, works well.
  - Serve up all images through a vanilla web server.
  - No expensive, complex server-side machinery
- Huge potential use: precompute complex, expensive vis, store images for later use.
- Provides illusion of unconstrained navigation
  - But without the bother of manually running the vis application.
- Supports two of three common visualization use modalities.
- Some challenges: lots of image files!
  - 36 horizontal x 18 vertical views: 648 images
  - Multiply by time, by vis parameter, by rendering parameter, pretty soon have *a lot* of images.
Chromium Render Server

- Focus on remote delivery of imagery produced by vis applications.
- Special focus on support for distributed-memory parallel, hardware-accelerated graphics applications.
- Our general solution suitable for use by literally *any* application that uses Xlib/OpenGL (all apps, basically).
- Supports all common visualization use modalities.
Prior Work

• Send-images approaches
  – Virtual Network Computing – VNC (Industry standard, no OpenGL support)
  – SGI’s VizServer (comm., RIP)
  – IBM’s Deep Computing Visualization (comm.)
  – ThinAnywhere (comm.)
  – HP’s Remote Graphics Software (comm.)

• Send-images/send-geom hybrid
  – VirtualGL

• Send geometry
  – CEI’s Ensight Gold Server-of-Servers
CRRS Approach

• Communication protocol: extend VNC
  – Ubiquitous viewer
  – Well-understood protocol
  – Open Source

• Rendering Infrastructure: extend Chromium
  – Solid base for distributed memory, h/w accelerated graphics
  – Open Source
CRRS Architecture
CRRS Optimizations

- **RFB Caching**
  - Avoid re-encoding images at the VNC Proxy
- **Bounding box tracking**
  - Limit RFB Updates to regions of OpenGL window that have changed since the last frame
- **Double-buffering**
  - Overlap rendering with encoding/transmission
- **Frame synchronization**
  - Synchronize parallel rendering streams
CRRS Performance w/RFB Caching

RFB Caching Optimization Improvement

Percent Improvement

City, Tight, 1x1  svPerfGL, Tight, 1x1  City, Hextile, 1x1  svPerfGL, Hextile, 1x1  City, Tight, 3x2  svPerfGL, Tight, 3x2  City, Hextile, 3x2  svPerfGL, Hextile, 3x2

LAN  WAN-R  WAN-H1  WAN-H2
CRRS Performance w/Optimizations

Average Improvement

Percent Improvement

Optimization

RFB Caching | DB | BB | All

LAN | WAN-R | WAN-H1 | WAN-H2

0.00% | 5.00% | 5.00% | 25.00%
Remote visualization capability

- Support for virtually any application
- Including hardware-accelerated rendering
- Supports multiple, collaborative participants (VNC)
- Ubiquitous client application (VNC)
- Completely Open Source (vncproxy.sf.net)
- Deployment activities at LBNL, ORNL.

- Security considerations (next slide)
CRRS Security

- Zone 1 – open internet
- Zone 3 – intracluster fabric
- Zone 2 – intracenter fabric
The CRRS Team

• Tungsten Graphics, Inc.
  – SBIR Small Business
• Lab participants: LLNL, ORNL, LBNL
• Send images holds the most promise
• Two projects that deliver results
  – Both garner peer-reviewed journal pubs in IEEE Transactions on Visualization and Computer Graphics
• Open source software release
• CRRS being deployed for production use at LBNL/NERSC and ORNL/CCS.