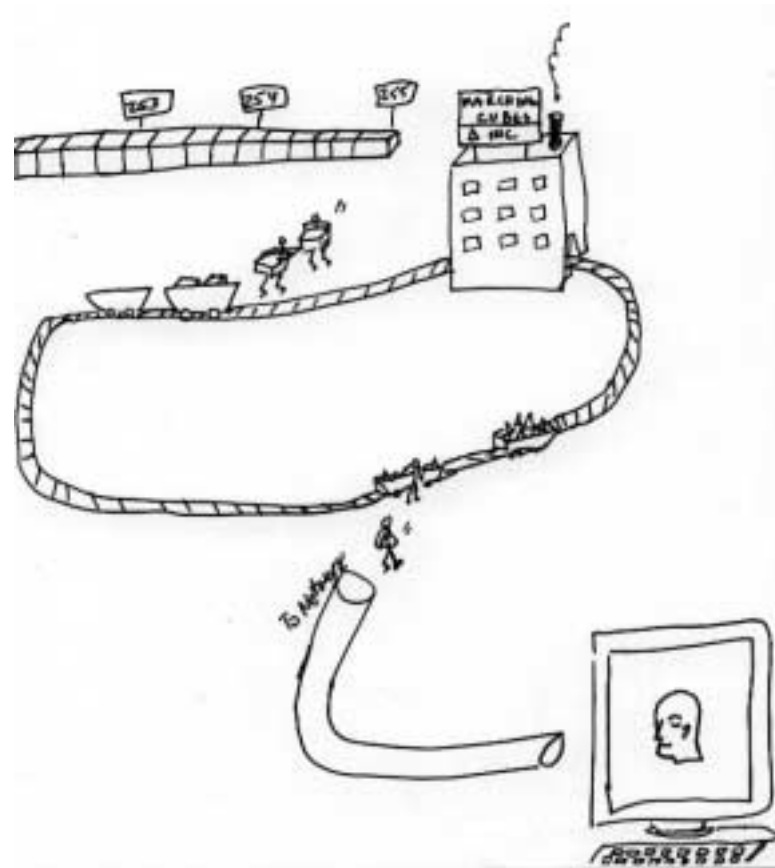


Fast Remote Isosurface Visualization With Chessboarding



Alisa Neeman, Peter Sulatycke,
Kanad Ghose



Remote Isosurface Visualization

- Means to view and understand large scientific data sets over the web
- Gigabit backbones encourage development
- Telemedicine (video and visualization) removes physical distance as a factor in patient care
- End-to-End Performance Issues
 - Bottlenecks at network fringes
 - Distance-induced latency
 - No quality of service guarantees
- Critical to reduce amount of data sent over network
- For medical data, critical to reduce it *losslessly*

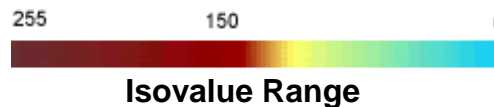
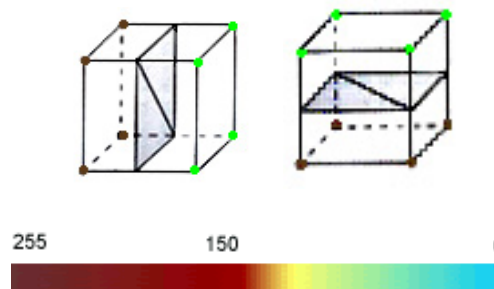
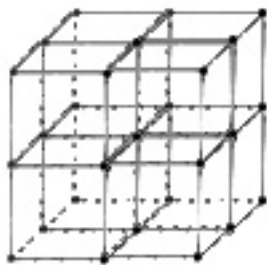


Contributions of This Work

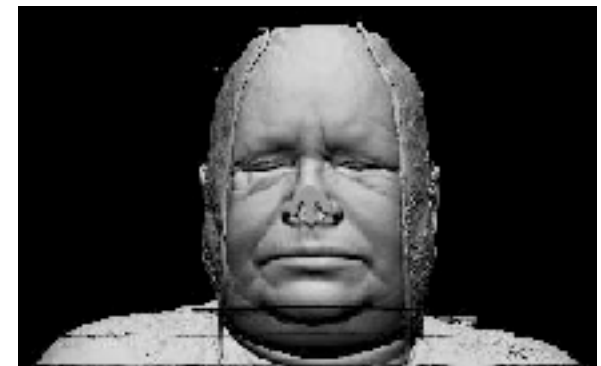
- Client-server visualization featuring:
 - Cross-network pipelining to hide computation time
 - Chessboarding modified for out-of-core use providing lossless compression
- Experiment with tradeoffs between computation at client and server
- Identify improvements leading toward a multithreaded server
- Gather baseline timing data running over standard broadband

Visualization Pipeline

- With visualization there are several clear stages from data extraction to display.
- Often described as a pipeline



Isovalue Range



The Visualization Pipeline

Pipeline
Interval Tree
Chessboarding
Details & Results

Ordered steps from data extraction to display

Data Extraction

Find Cells
Intersecting
Isosurface:
98% of data
ignored!

Data Calculations

Create
Polygons

Rendering Calculations

Transformations/
Texture Mapping/
Lighting

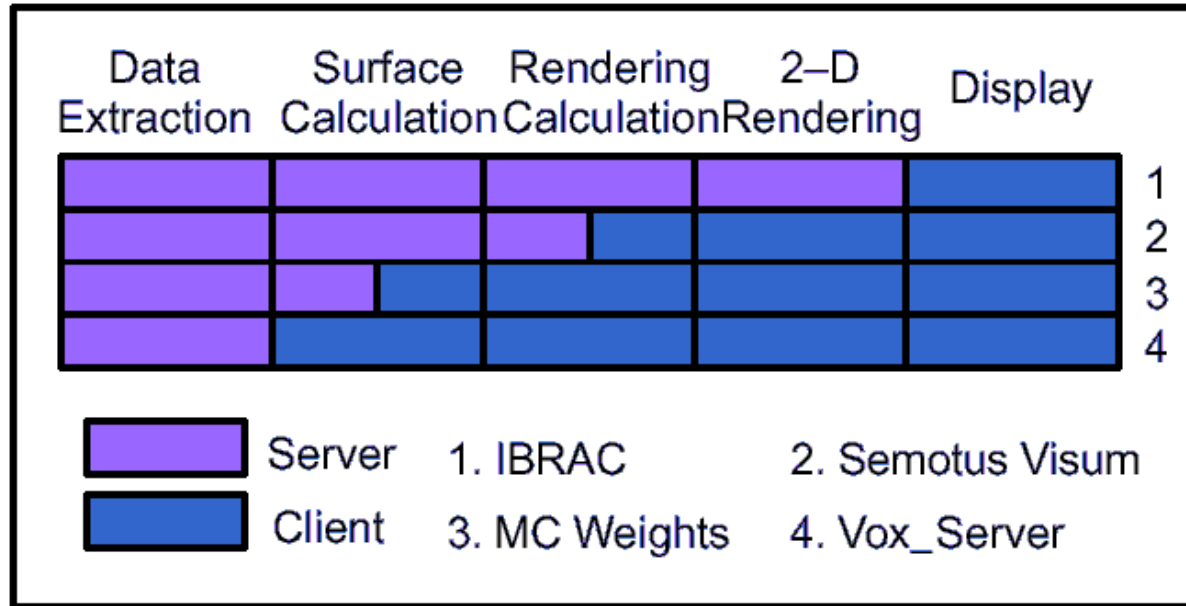
Rendering

Display

Where does the extra stage go?

Data Transport

Related Work: Various Pipeline Decisions



- IBRAC: Image Based Rendering With Acceleration and Compression. MPEG2-like compression of images. Client estimates motion of pixels, server sends pixels that can't be estimated from a ray-traced 3D model.
- Semotus Visum: Server creates triangle mesh based on depth buffer and color buffer. Mesh and image sent. Client texture maps image onto mesh.
- Engel, Westermann and Ertl: Marching cubes interpolation weights sent. 26 bytes per cell in most common case.
- Vox_server: sends chessboarded data cells. Approx. 17 bytes per cell.

Tradeoff Analysis

Majority of Pipeline On Server Side	Majority of Pipeline on Client Side
<p>Positives:</p> <ul style="list-style-type: none">■ No delay to view initial image.■ Reasonable response time.	<p>Positives:</p> <ul style="list-style-type: none">■ Lossless image.■ Model may be viewed from any angle; client may even look <i>inside</i> model.
<p>Tradeoffs:</p> <ul style="list-style-type: none">■ Image may be lossy.■ Single image viewed only from limited angles.■ Continuous, small demand for bandwidth	<p>Tradeoffs:</p> <ul style="list-style-type: none">■ Initial delay to receive data.■ Interactivity slows as data size increases.■ One-time large demand for bandwidth

Overview of Design

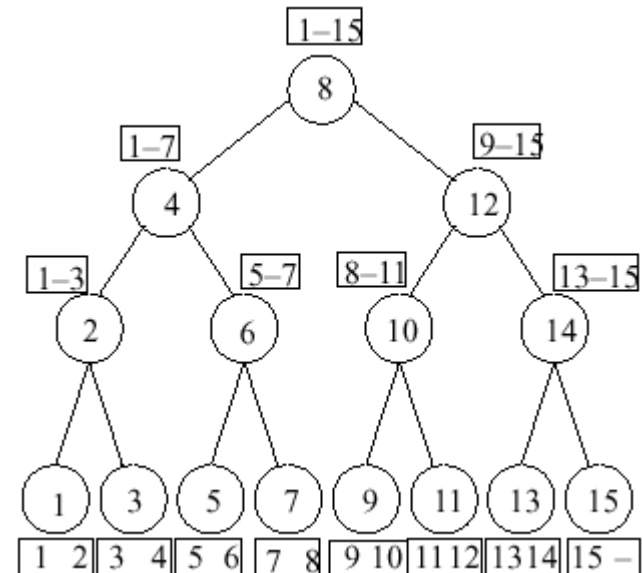
- Server side:
 - Search optimized interval tree on disk
 - Chessboarding to remove redundant data
- Client side
 - Cross-network pipelining of surface construction
- Producer-Consumer With Bounded Buffer
 - Producer extracts cells that intersect isosurface
 - Consumer(s) create the triangle that make up the isosurface

Motivation for Interval Trees

- Data organized by interval (min through max) of a cell's isovalues
- Spatial coherence lost but intersecting cells located in $O(h + K)$
 - h: height of tree
 - K: number of cells' intervals intersecting isosurface

Interval Trees On-Disk

- Nodes written to disk during preorder traversal
- Benefit: Keeps sub-tree contents contiguous
 - reduced seeking
 - unidirectional head movement
- Price paid for data structure:
 - must replicate cell data (left list, right list)
 - store cells rather than pointers
- Small auxiliary files to help with navigation contain distances to nodes, length of nodes



Why Use Chessboarding?

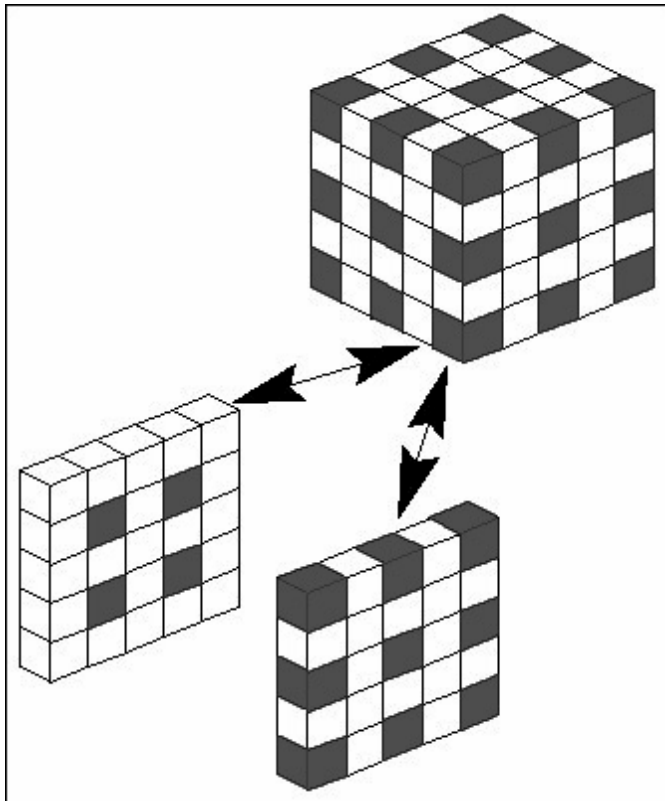
■ Benefits

- Reduce on-disk storage cost
- Reduce seek and read time
- Reduce network transfer time
- No additional cost for creating triangles

■ Tradeoff

- Multithreading no longer improves performance

Chessboarding



***Exterior of Volume Coated In
Black Cells***

Interior Cell Rules:

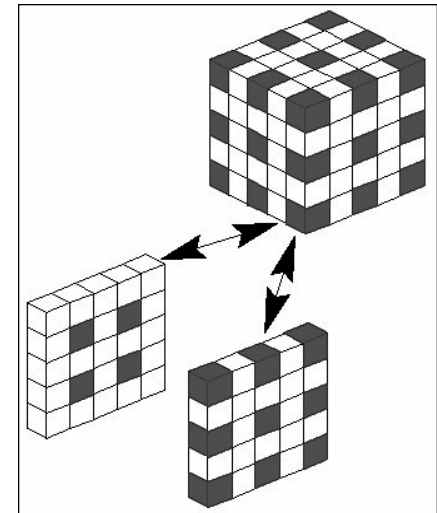
- 1. No two black cells are adjacent.**
- 2. Each white cell vertex is contained within a neighboring black cell**

Once We Remove White Cells..

- If we have 3 slices of data, we can recreate white cell data in middle
- Size of data set **reduced** roughly **by a factor of 4**

Chessboarding Ingredients

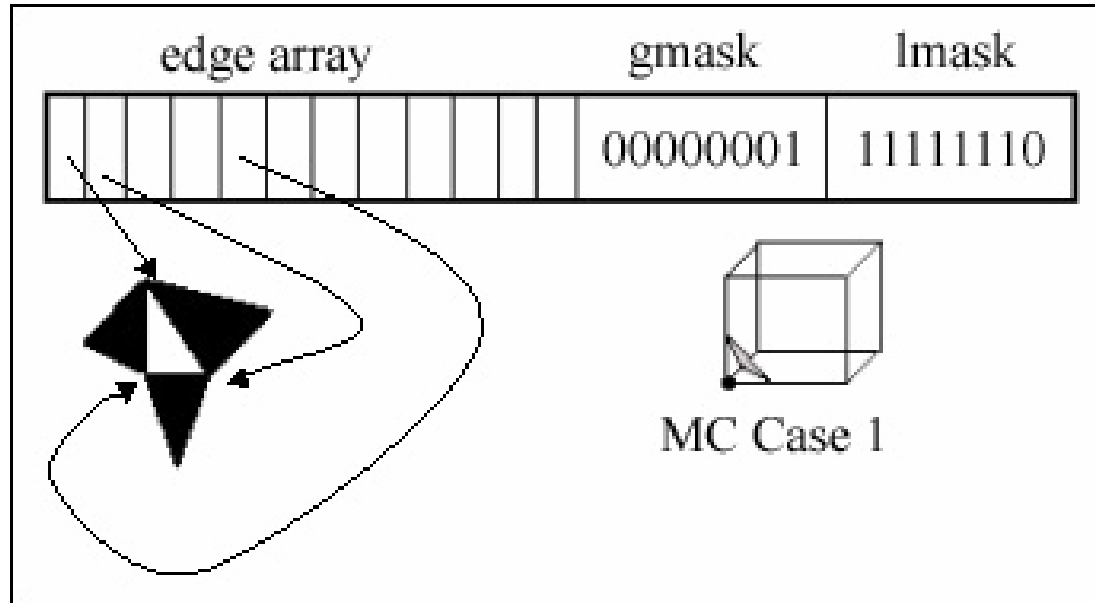
- Array of black triangle *vertices*
- Hash table of white cells
 - Repeatedly accessed by black cell neighbors
- Three slices to pipeline
 - Once black cells processed, we can form triangles for interior slice white cells.
 - *Although we no longer have spatial coherency, we can test that all black cells surrounding a white cell have been processed, and create its triangles.*



How can we do this?

Chessboarding Data Structure

White Cell Hash Table Entry



gmask: vertices greater than isovalue (Marching Cubes index)

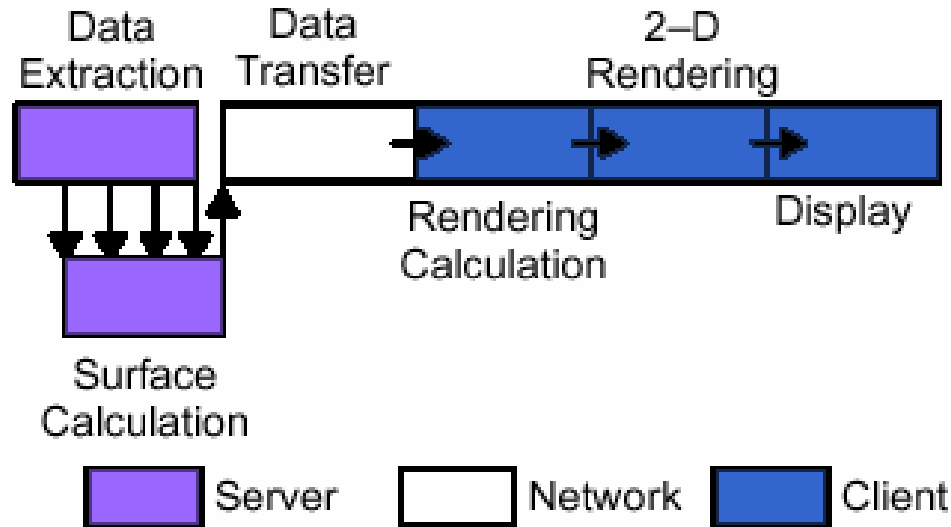
lmask: vertices less than isovalue, confirms all edges found

Lookup tables tie black cell edges to corresponding white cell edges

Edge array entry points to black cell vertex to quickly form white triangles.

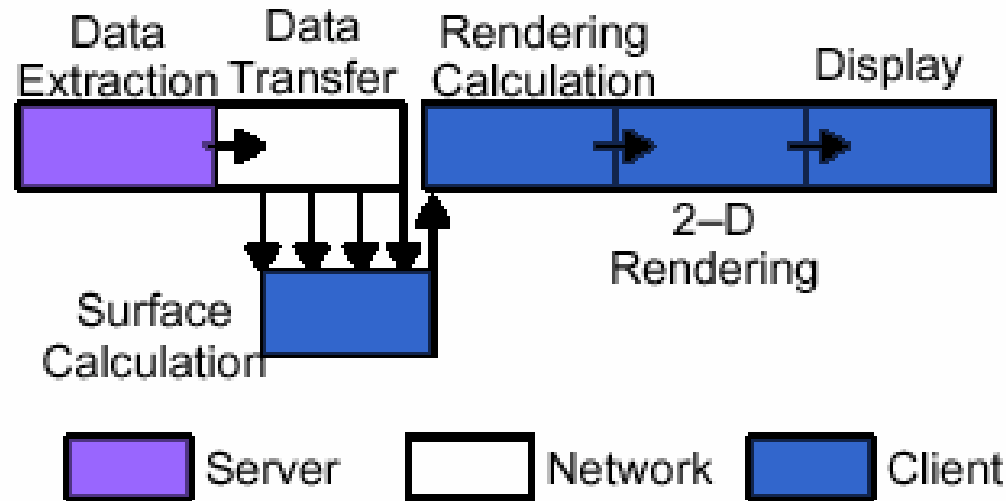
Great! But can we do it remotely?

Triangle_server (Baseline)



- Producer and consumer *both* stationed on server, therefore:
File I/O stage concurrent with triangle calculation
- Server has dual processor; **addition concurrency within triangle calculation stage.**
- **Completely processed triangles are ready for rendering when they arrive at the client.**

Vox_server pipeline



- Server is producer. Client is consumer
- **Data Transfer concurrent with triangle calculation on the client side**
- Network latency hides *single-threaded* triangle calculation

Experiments

- Compare sending processed triangles vs. raw black cells by
 - Amount of data sent
 - Time from extraction to triangles
- Compare with overlap of pipeline stages:
 - Data Extraction and [Multithreaded] Computation
vs.
 - Data Transport and [Single Thread] Computation
- For both a LAN and over broadband

Platform Details

■ Server:

- Cluster node with two Pentium 4, 2.2 GHZ processors
- 4 GB RAM
- 15,000 RPM SCSI drive

■ Client:

- Dell Dimension 4500, 2 GHZ Pentium 4
- 512 MB RAM

■ Data Set:

- First 256 slices of Visible Woman data set (512 x 512 x 256)
 - Each slice 200 sample points thick
- Raw size: 128 MB
- As Interval Tree: 6.7 GB
- As Chessboarded Interval Tree: 1.7 GB

One Hop Via LAN



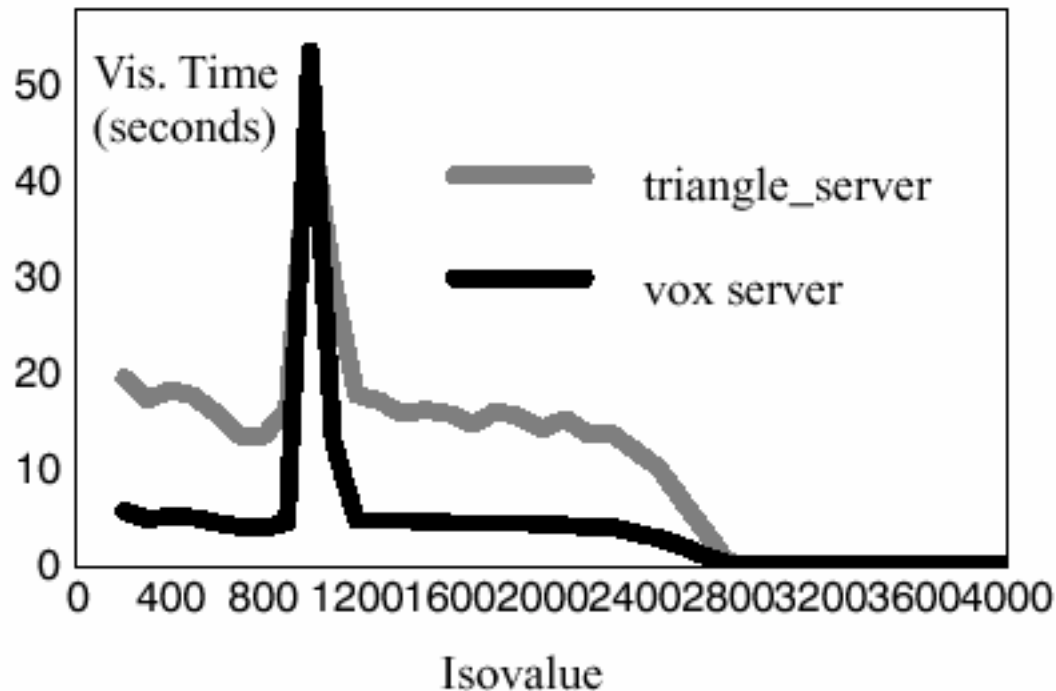
At 100 Megabits Per Second

Data Reduced by 87%

25 Million bytes vs. 200 Million bytes

- Cube and triangle roughly same size (68 bytes vs. 72 bytes)
- Send **1 cube** rather than the **2 or more** triangles that Marching Cubes usually creates
- One fourth the number of cubes with chessboarding
- $100\% * .50 * .25 = 12.5\%$ amount of data

LAN Time Results



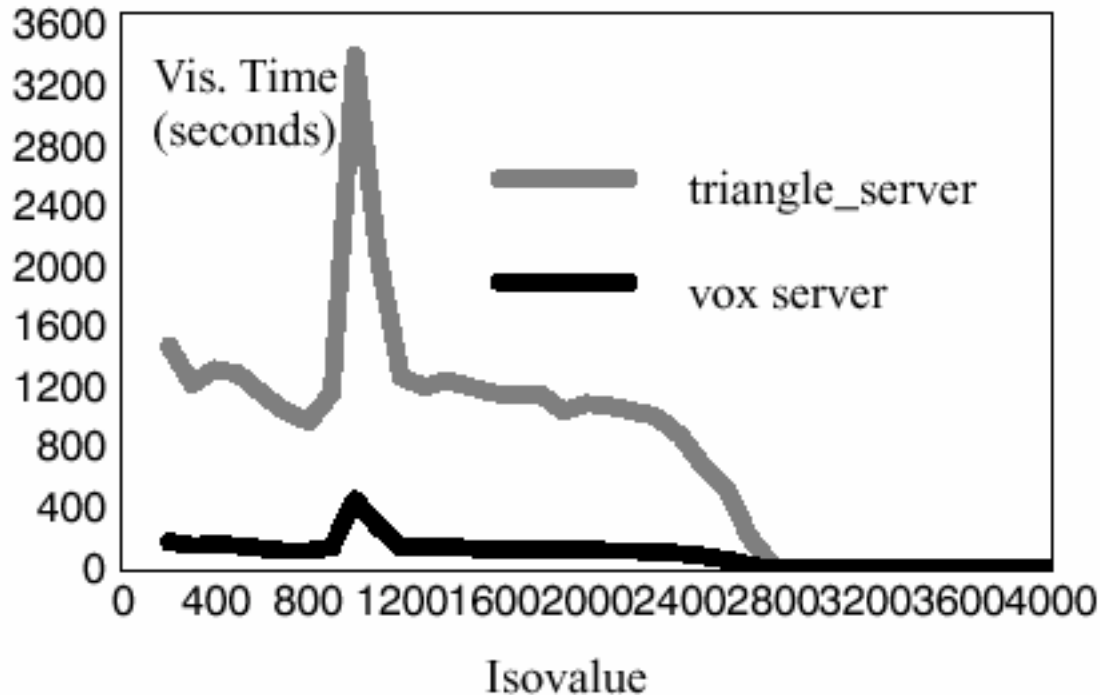
For the majority of isovalue:

- vox_server 68% faster than the triangle_server
- displayed an isosurface on the screen in under 8 seconds.

Via Broadband

5 HOPS At 2 Mbps × 1 HOP Via Lan

Results



- Time to receive data **increased** from on the **order of seconds to minutes**
- Median vox_server time: close to 2 minutes
- Peak time: 7 $\frac{3}{4}$ minutes, but what do we get for it?
- Processing of data totally hidden by network latency
- vox_server 87% faster than the triangle_server, almost exactly the same as amount of data sent

Conclusions/Future Work

- Sending chessboarded cells across the network provides faster access to remote data for isosurface visualization
 - Dominating factor network latency
 - Processing can be hidden by pipelining
- Reduces the burden on the server
- Potential to serve multiple clients simultaneously: caching, multicast.
- May extend to interval volumes
- Chessboarding may be applicable to irregular (tetrahedral) data sets.

More isosurfaces with the vox_server

