Out-Of-Core Sort-First Parallel Rendering for Cluster-Based Tiled Displays

Wagner T. Corrêa	Princeton/AT&T
James T. Klosowski	IBM
Cláudio T. Silva	OHSU/AT&T

EG PGV, Germany

September 10, 2002

Goals

- Render large models
- At interactive frame rates
- Using inexpensive hardware
- In high resolution

Applications

- Data visualization
- Medicine
- Engineering
- Weather forecasting
- Entertainment

Approach

- Out-of-core preprocessing
  - build an on-disk octree for the model
- Out-of-core rendering
  - load on demand the visible octree nodes
- Out-of-core parallel rendering
  - use a PC cluster to drive a multi-projector display wall (high resolution, inexpensive)

Why Use a Cluster of PCs?

- Explosive growth of PC graphics cards
- Availability of high-speed networks
- Better than high-end machines
  - better price/performance
  - can be upgraded more often
  - can use different kinds of machines
  - can be used for tasks other than rendering
  - aggregate power scales with number of PCs

Related Work

- Samanta 01
  - assumes model fits in memory of each PC
  - client runs load balancing schemes
  - client may become a bottleneck
- Humphreys 01: WireGL
  - assumes model fits in client's memory
  - client sends geometry to servers every frame
  - client may become a bottleneck

Related Work

- Wald 01
  - ray tracing (less hardware support)
  - slower preprocessing step (2.5h vs. 17min)
  - low resolution (640x480 vs. 4096x3072)

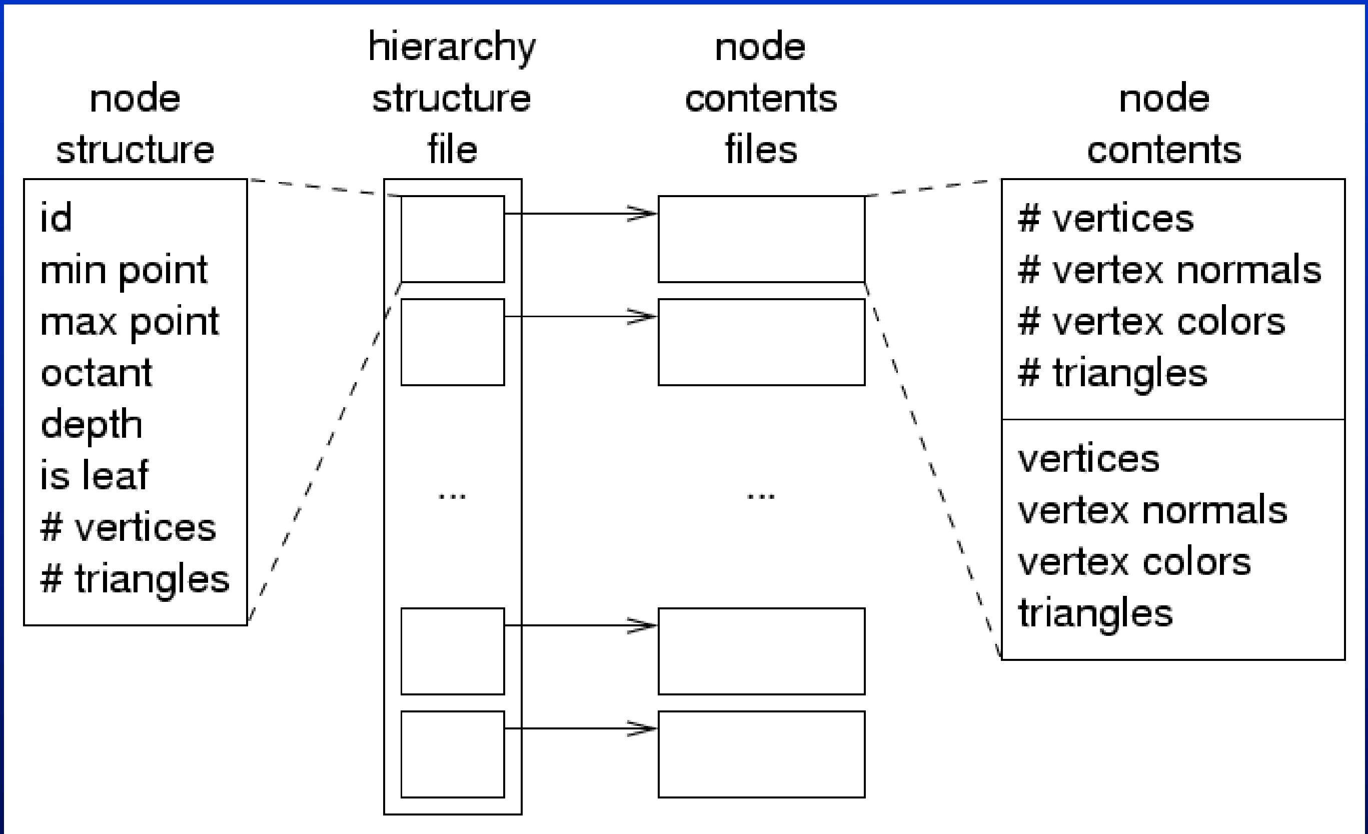
Talk Outline

- Out-of-core preprocessing
- Out-of-core rendering
- Out-of-core parallel rendering
- Results

Talk Outline

- Out-of-core preprocessing
- Out-of-core rendering
- Out-of-core parallel rendering
- Results

The Out-Of-Core Octree Format



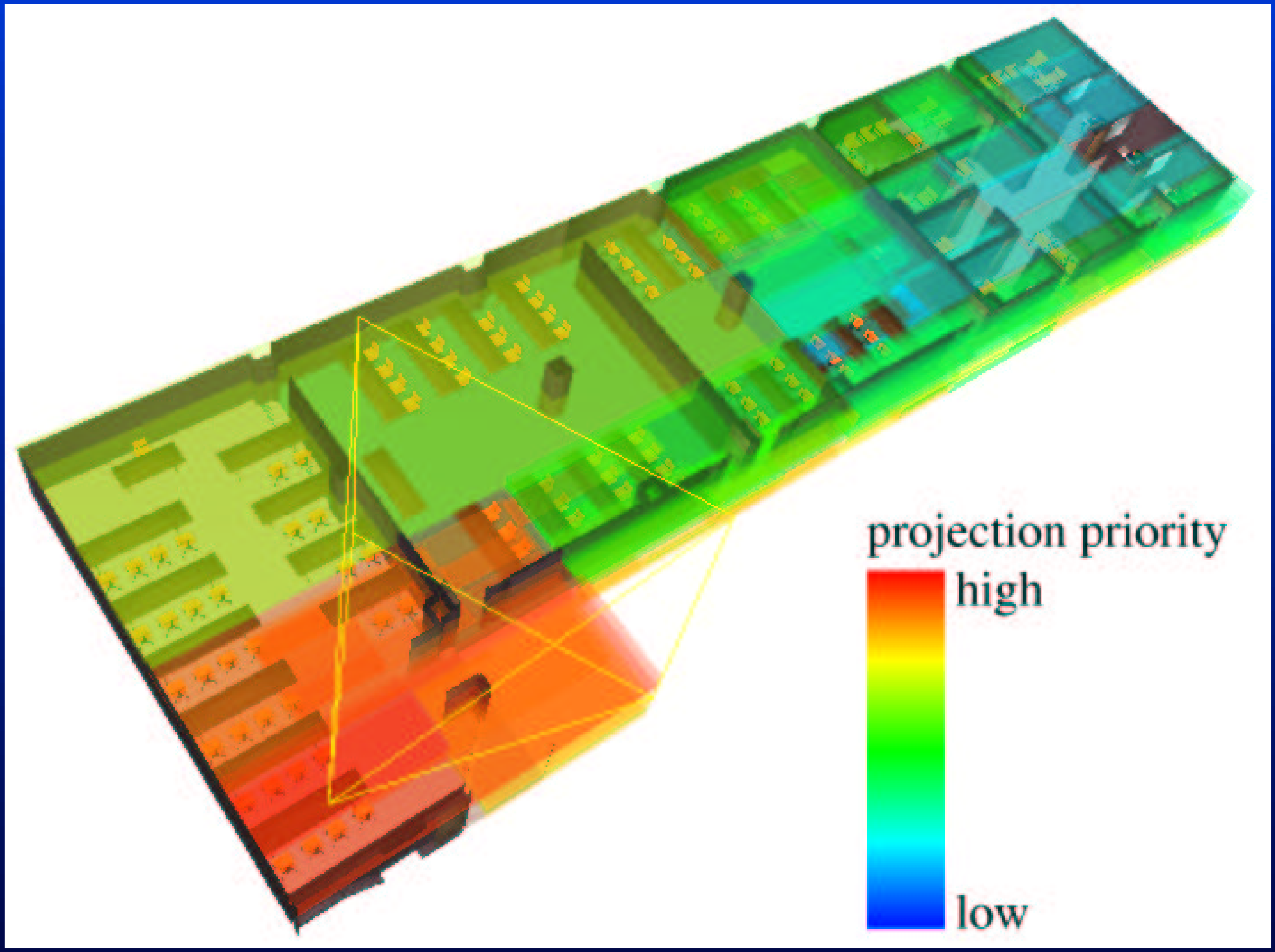
Building the Out-Of-Core Octree

- Break model in sections that fit in memory
- For each section
  - read hierarchy structure (HS) file
  - perform fake insertions
  - for each touched node
    - read old contents
    - reinsert old contents
    - update contents on disk
  - update HS file on disk

The PLP Algorithm

- Approximate volumetric visibility
- Keeps the octree nodes in a priority queue called *front*
- First visits nodes most likely to be visible
- Stops when a budget is reached
- Doesn't need to read the geometry
  - estimates the visible set from the hierarchy structure (HS) file

The PLP Algorithm



The cPLP Algorithm

- Conservative extension of PLP
- Uses PLP to compute initial guess
- Adds nodes to guarantee correct images
- Unlike PLP, needs to read geometry
  - can't determine visible set from HS file only
- Our implementation uses an item buffer
  - can be optimized by using visibility extensions of the graphics hardware

Visibility Preprocessing

- For each node
  - for each sample viewing direction
    - compute *solidity* (estimate how much light is blocked by the node)
  - save solidities on disk
- At runtime, projection priorities are computed by accumulating solidities from node to node using ray tracing

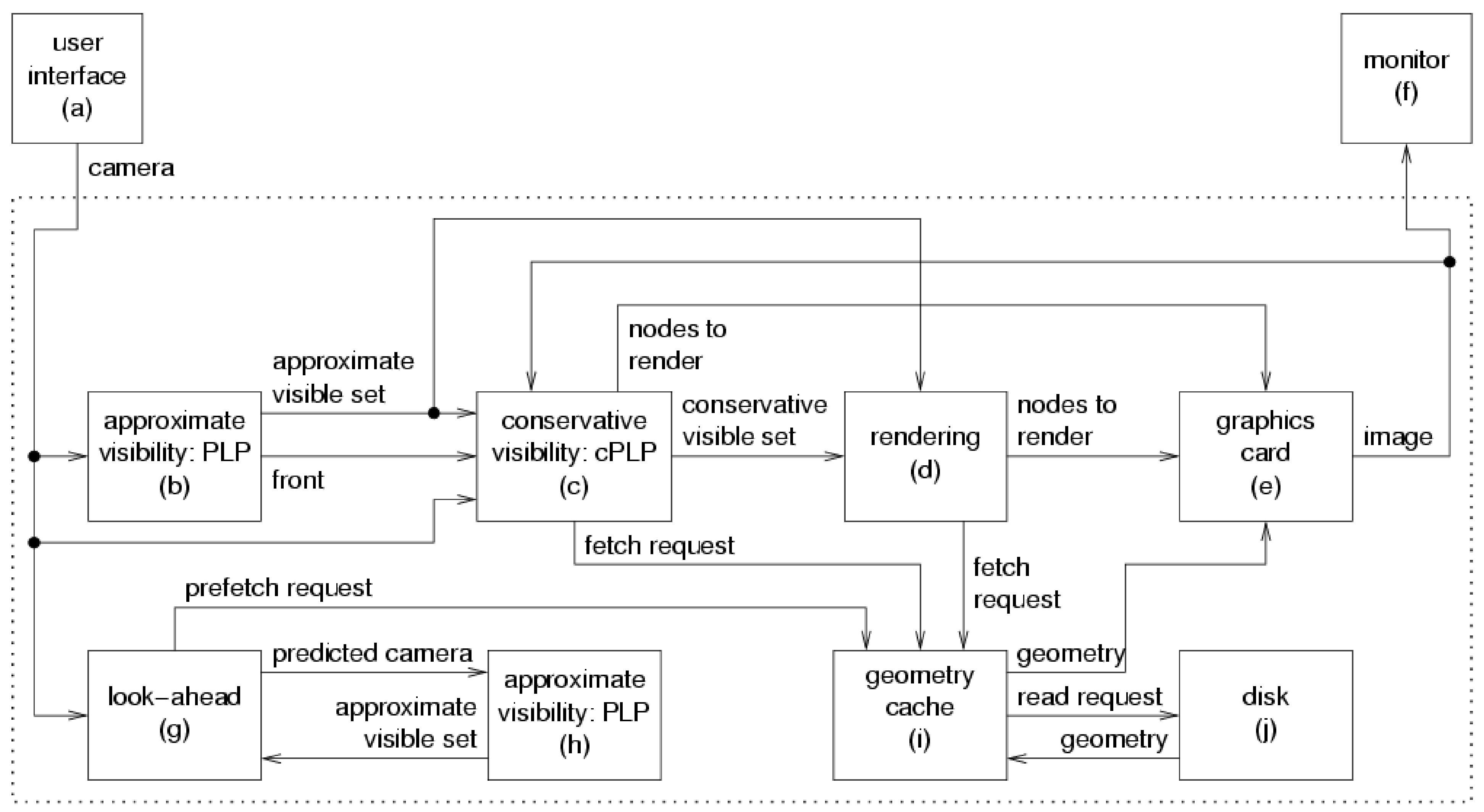
Talk Outline

- Out-of-core preprocessing
- Out-of-core rendering
- Out-of-core parallel rendering
- Results

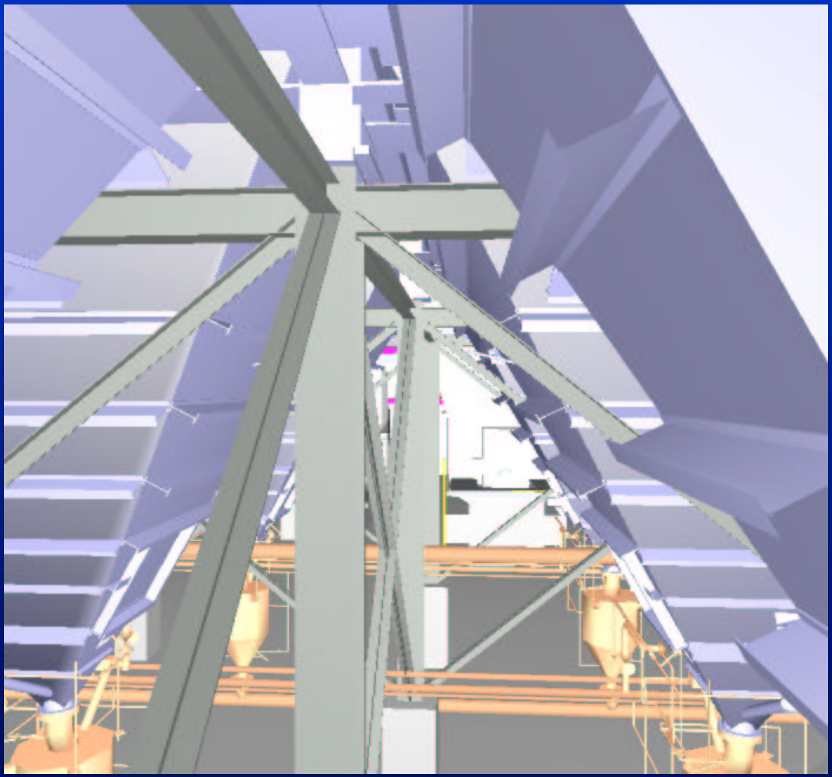
Out-Of-Core Rendering

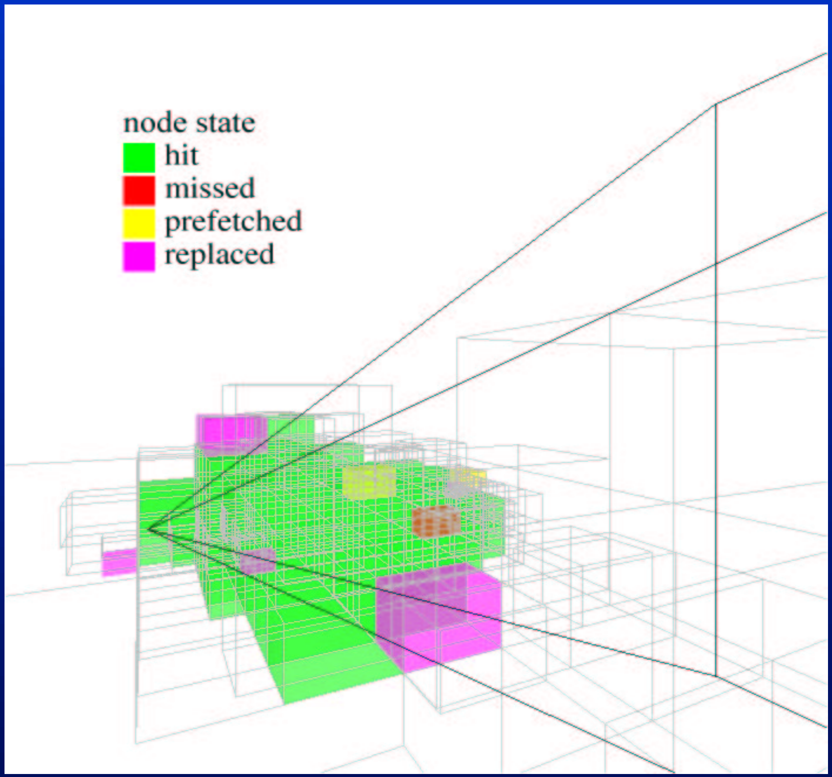
- Load on demand the visible nodes
- Use multiple threads on a single PC
- Overlap
  - rendering
  - visibility computations
  - fetching
  - prefetching

Overview of the Rendering Approach

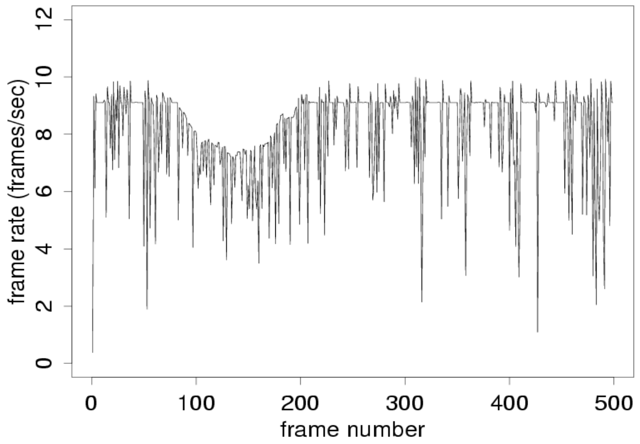


Snapshot of the Geometry Cache

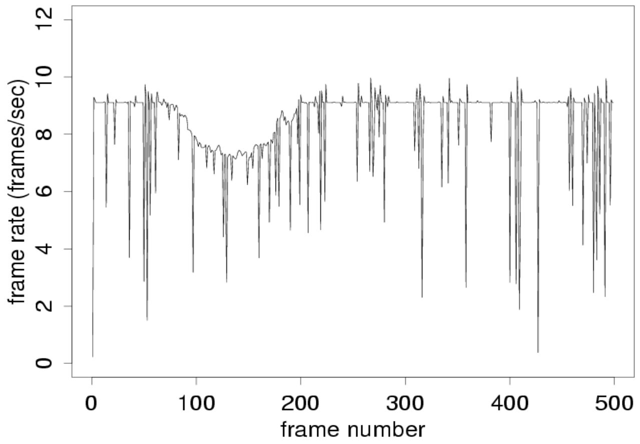




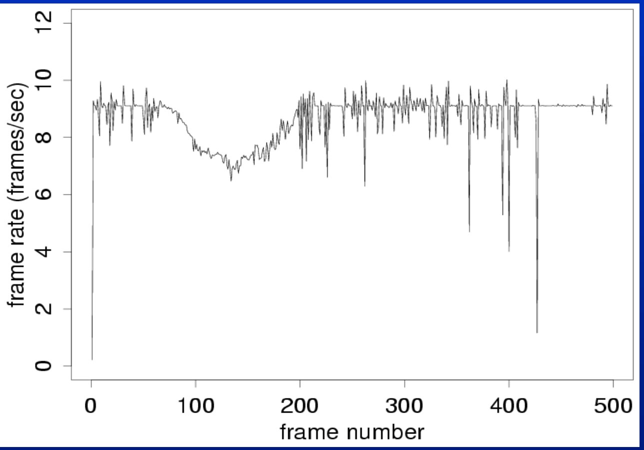
Using Multiple Threads to Improve Frame Rates



sequential fetching and rendering

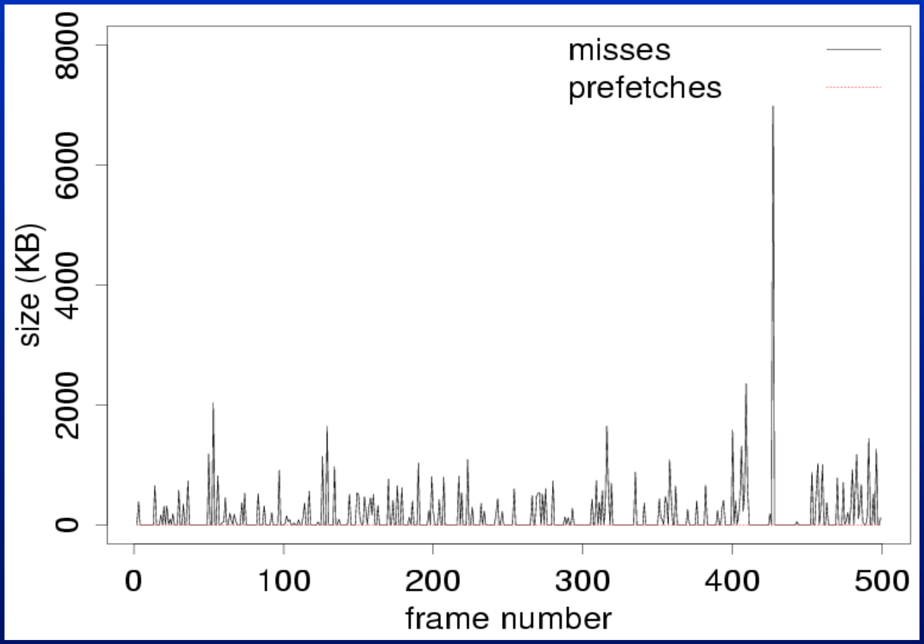


concurrent fetching and rendering

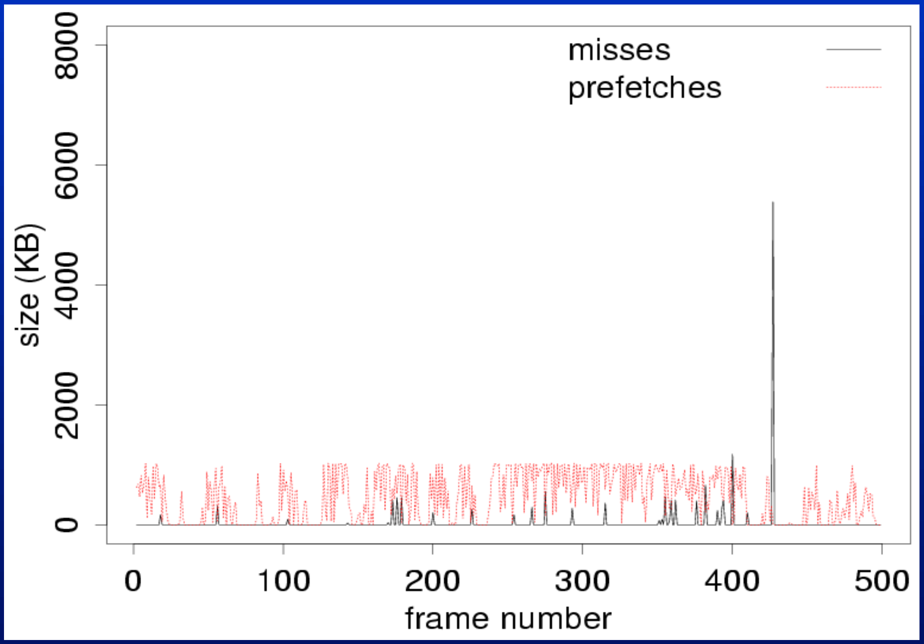


concurrent fetching, rendering, and prefetching

Using Prefetching to Amortize the Cost of Disk Operations



without prefetching



with prefetching

Advantages of the Rendering Approach

- **Out-of-core**
- **Exploits frame-to-frame coherence**
- **Uses from-point prefetching**
  - **less preprocessing than from-region**
- **Uses threads in a single processor to exploit parallelism opportunities**
- **Handles tens of millions of triangles on a single PC at interactive frame-rates**

Talk Outline

- Out-of-core preprocessing
- Out-of-core rendering
- Out-of-core parallel rendering
- Results

Out-Of-Core Parallel Rendering

- So far
  - single PC
  - low resolution images (1024x768)
  - interactive frame rates
- Now
  - display wall driven by a cluster of PCs
  - high resolution images (4096x3072)
  - same or faster frame rates

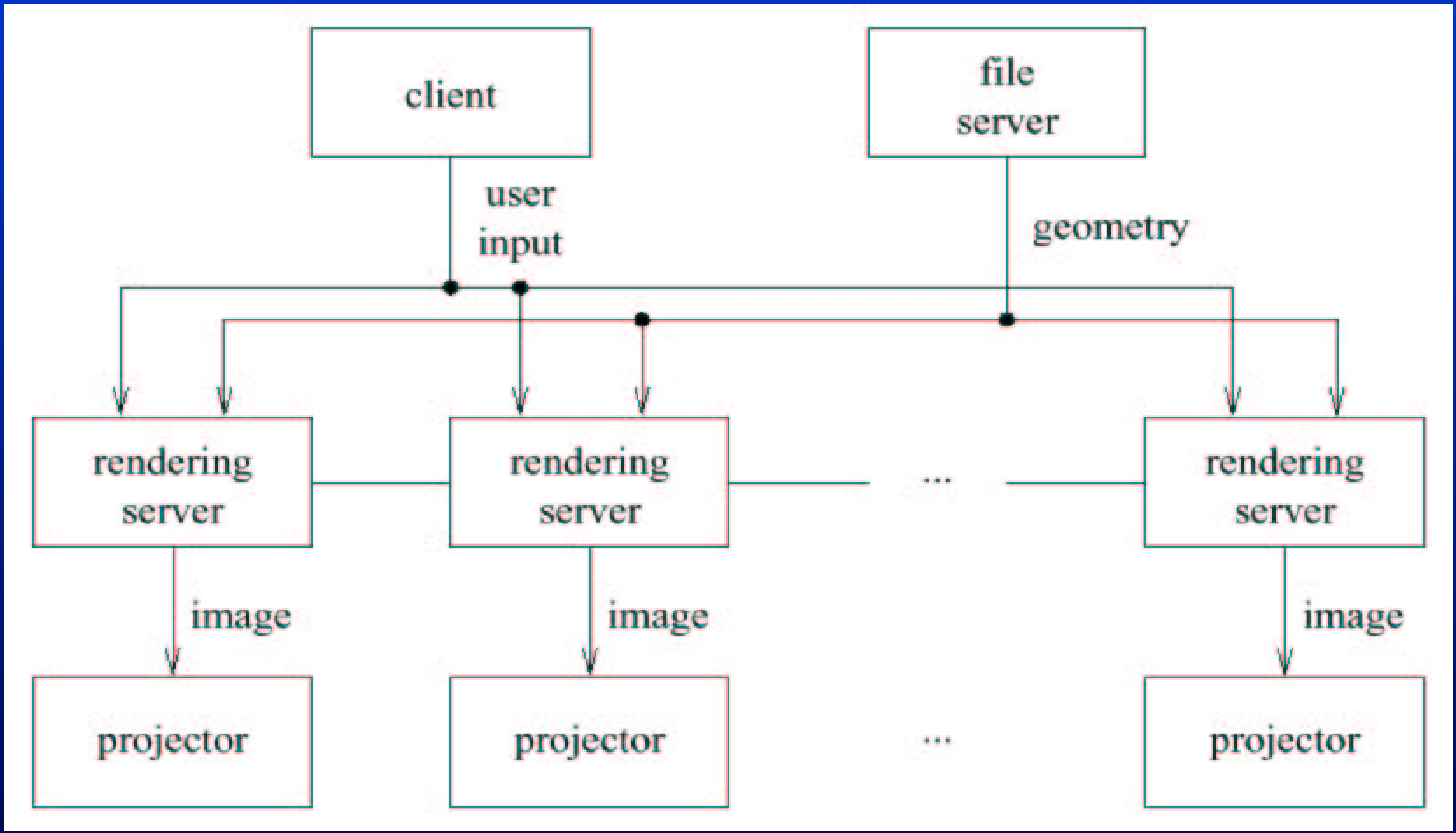
Choosing the Parallelization Strategy

- **Sort-first**
  - distribute object-space primitives
  - each processor is assigned a screen tile
- **Sort-middle**
  - distribute image-space primitives
  - geometry processors and rasterizers
- **Sort-last**
  - distribute pixels
  - rendering and compositing processors

Choosing the Parallelization Strategy

- Why sort-first?
  - each processor runs entire pipeline for a tile
    - that's what PC graphics cards are optimized for
  - exploits frame-to-frame coherence well
- Why *not* sort-middle?
  - needs tight integration between geometry processing and rasterization
- Why *not* sort-last?
  - needs high pixel bandwidth

The Out-Of-Core Sort-First Parallel Architecture



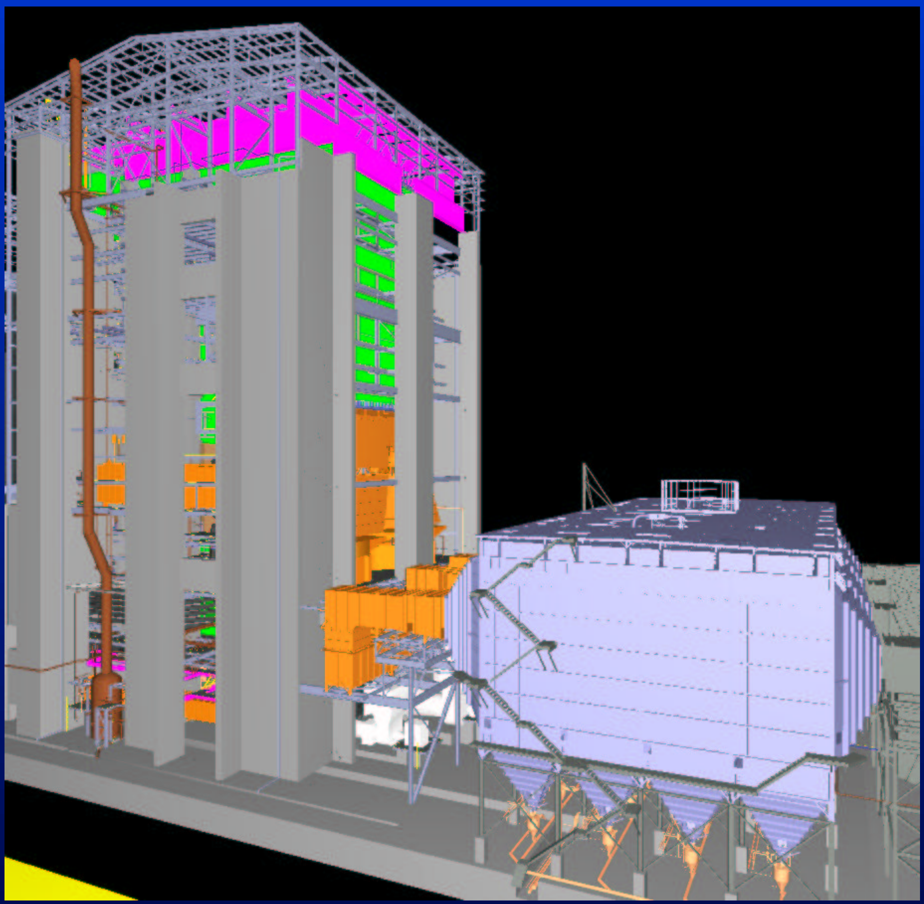
The Out-Of-Core Sort-First Parallel Architecture

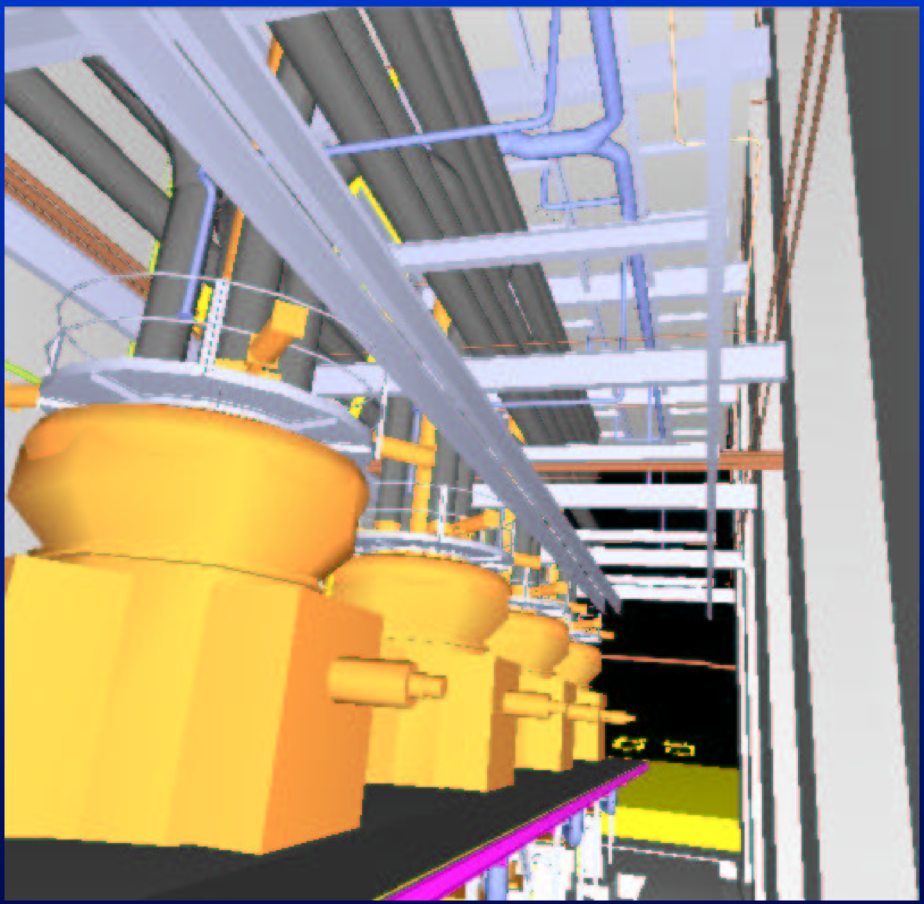
- Given sequential approach, parallel extension is trivial
- MPI is only used to start and synchronize the servers
- Client does almost no work, and can be as lightweight as a handheld computer
- Very different from Samanta 01 and Humphreys 01 (WireGL)

Talk Outline

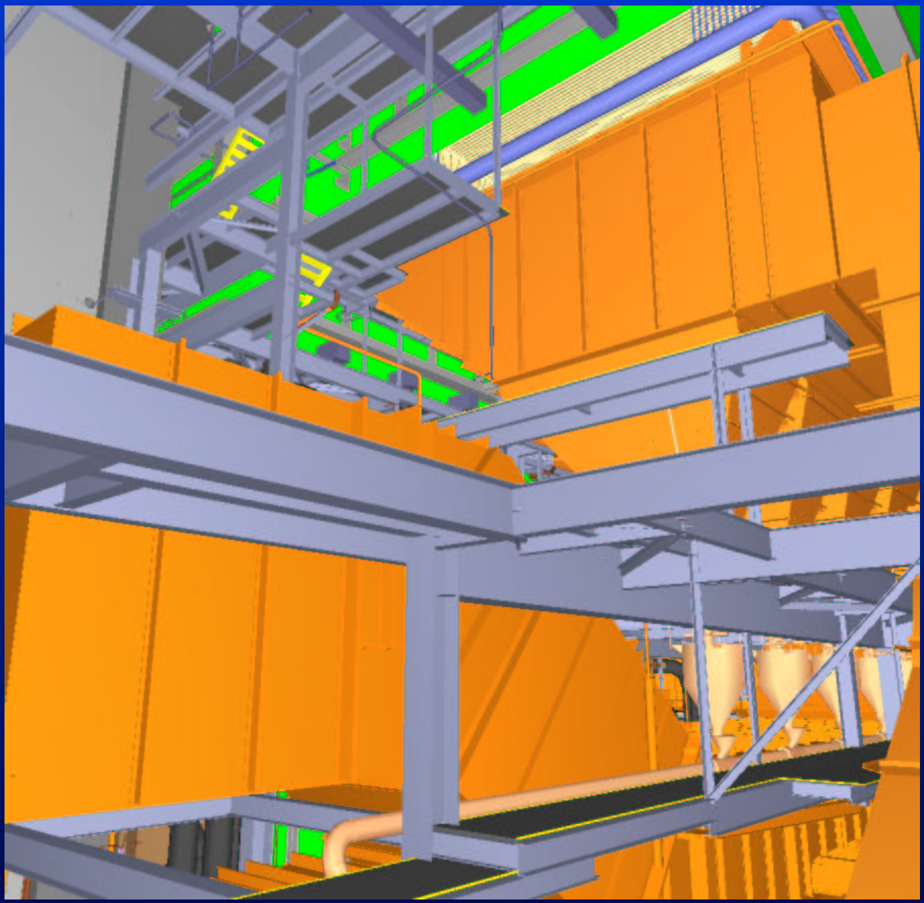
- Out-of-core preprocessing
- Out-of-core rendering
- Out-of-core parallel rendering
- Results

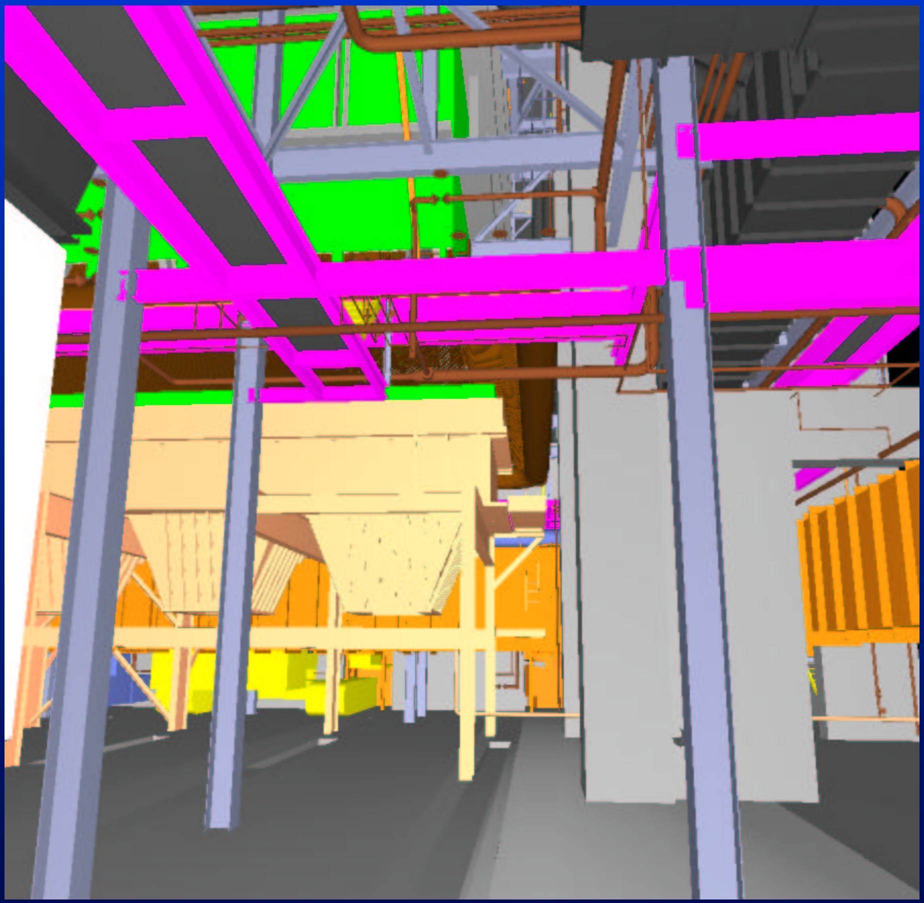
Test Model: UNC Power Plant





Test Model: UNC Power Plant

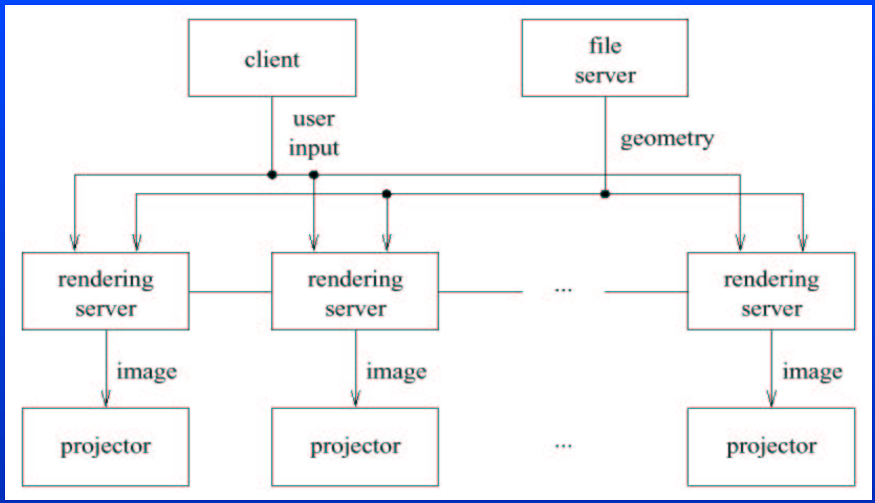




Tests

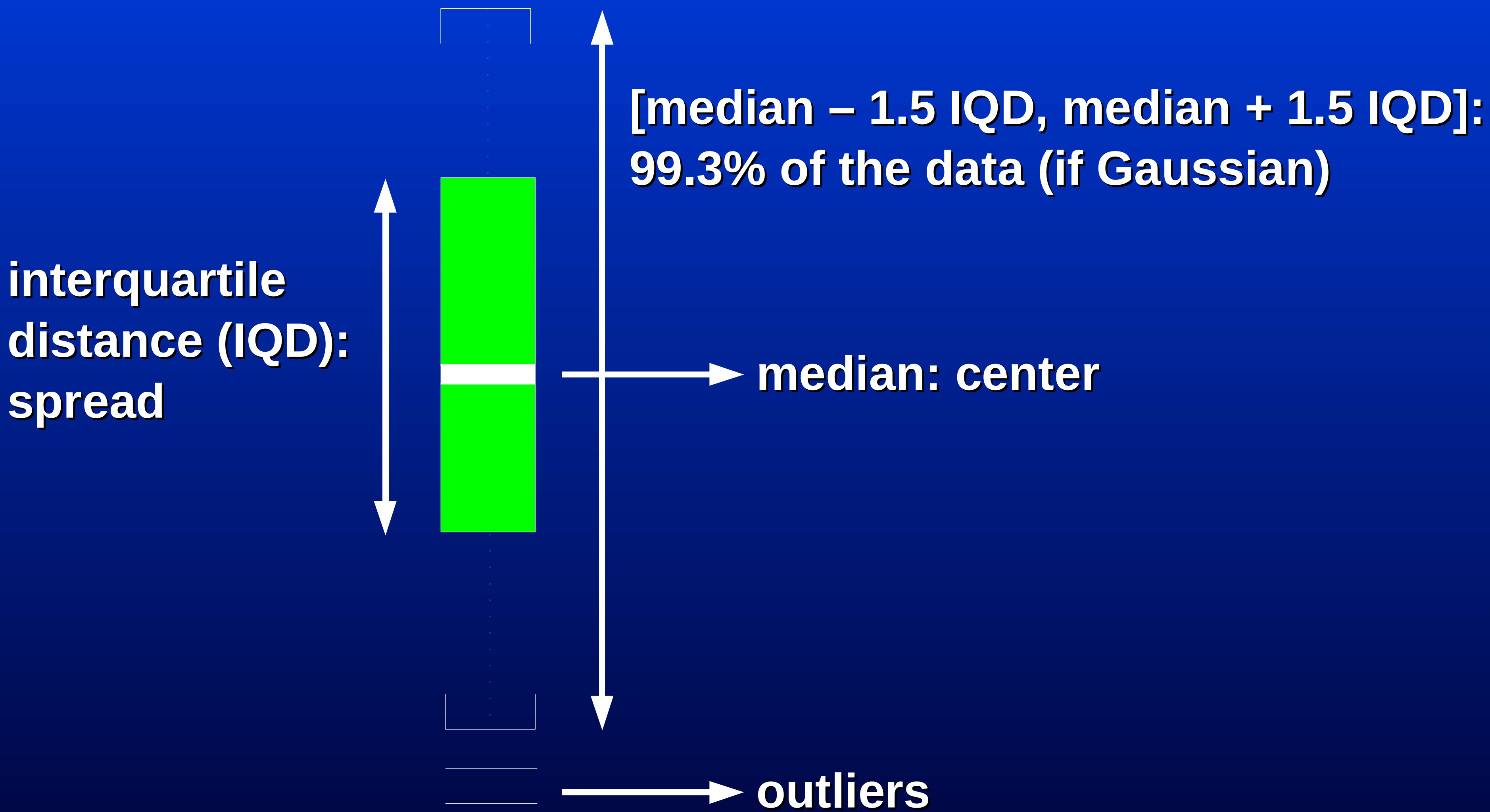
- Pre-recorded 500-frame camera path
- Visibility mode
  - approximate (using PLP)
  - conservative (using cPLP)
- Cluster sizes
  - 1, 2, 4, 8, and 16
- Disk type
  - local and network

Testing Environment

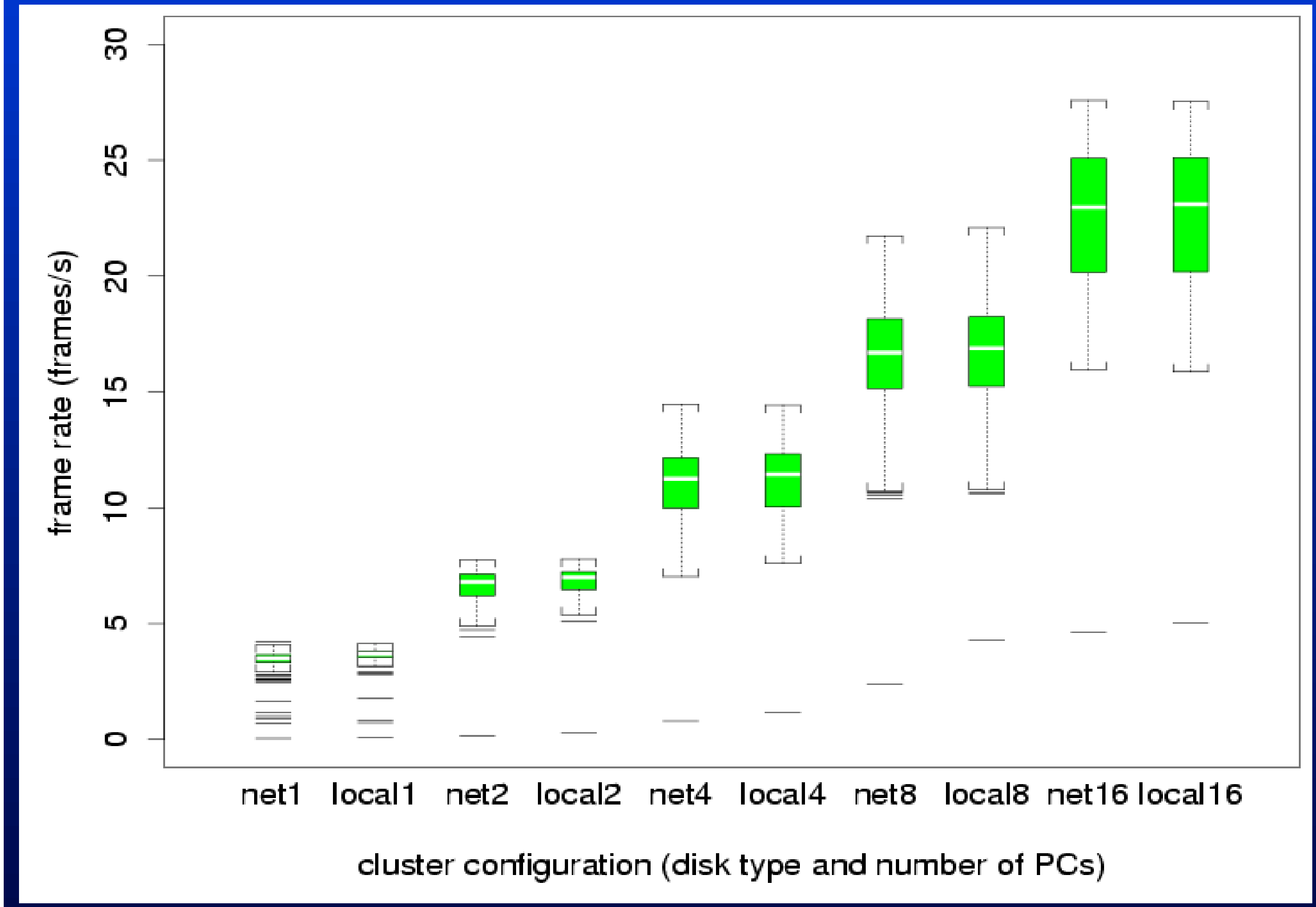


- Rendering servers
  - 900 MHz Athlon, 512 MB of RAM
  - GeForce2, IDE disk
- Client: 700 MHz Pentium III
- File server: 400 GB SCSI disk array
- Network: gigabit Ethernet
- Software: Red Hat Linux 7.2, MPI/Pro 1.6.3

Box Plots


[median – 1.5 IQD, median + 1.5 IQD]:
99.3% of the data (if Gaussian)
interquartile
distance (IQD):
spread
median: center
outliers

Results for PLP (Approximate Mode)

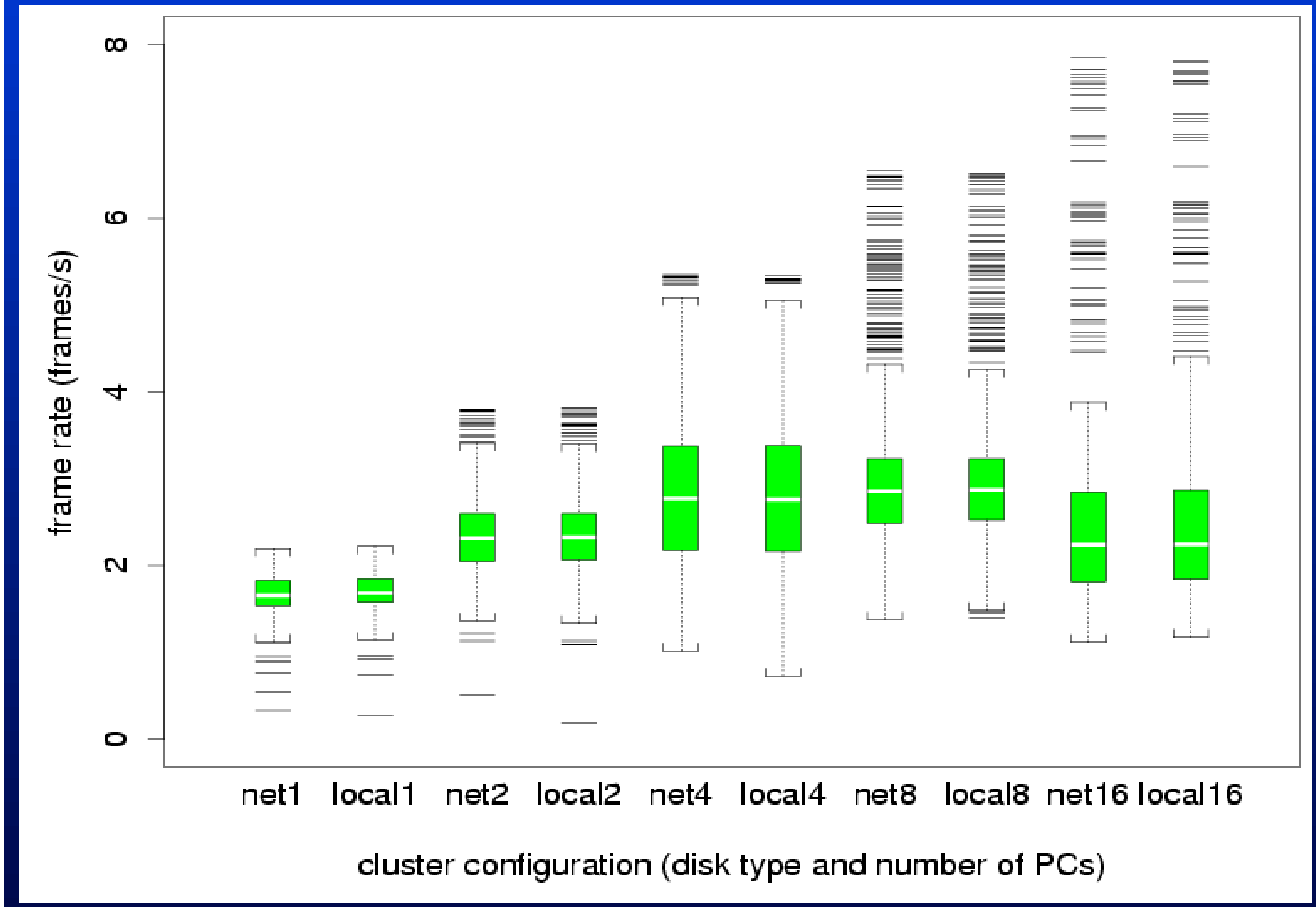


- Total budget of 400K tri/frame
- Median frame rates improve with cluster size
- Disk type makes no difference

Obstacles for Perfect Scalability

- Duplication of effort
  - primitives may overlap multiple tiles
- Communication overhead
  - barrier at the end of each frame
- Load imbalance
  - primitives may cluster into regions

Results for cPLP (Conservative Mode)



- Median frame rates remain almost constant
- Disk type makes no difference
- Additional obstacle: visible geometry may increase with resolution

Summary of Best Results

- Model size: 13 million triangles
- Preprocessing time: 17 minutes
- 1 PC (1024x768 images, 70K tri/frame)
  - median accuracy: 98.1%
  - median frame rate: 9.1 frames per second
- 16 PCs (4096x3072 images, 25K tri/frame)
  - median accuracy: 99.3%
  - median frame rate: 10.8 frames per second

Conclusions

- System for interactive, high-resolution rendering of large models on cluster-based tiled displays
- Advantages
  - simple
  - inexpensive
  - scalable
  - better than expensive high-end systems

Future Work

- Add level-of-detail management
- Add load balancing schemes
- Improve heuristic to estimate visibility
- Handle dynamic scenes

Thanks

- Funding
  - AT&T, CNPq (Brazil), Princeton
- Models
  - UNC Chapel Hill
- Motivators
  - Daniel Aliaga, David Dobkin, Jeff Korn, Kai Li, Wagner Meira, Emil Praun