Massive Dataset Visualization

Aiichiro Nakano

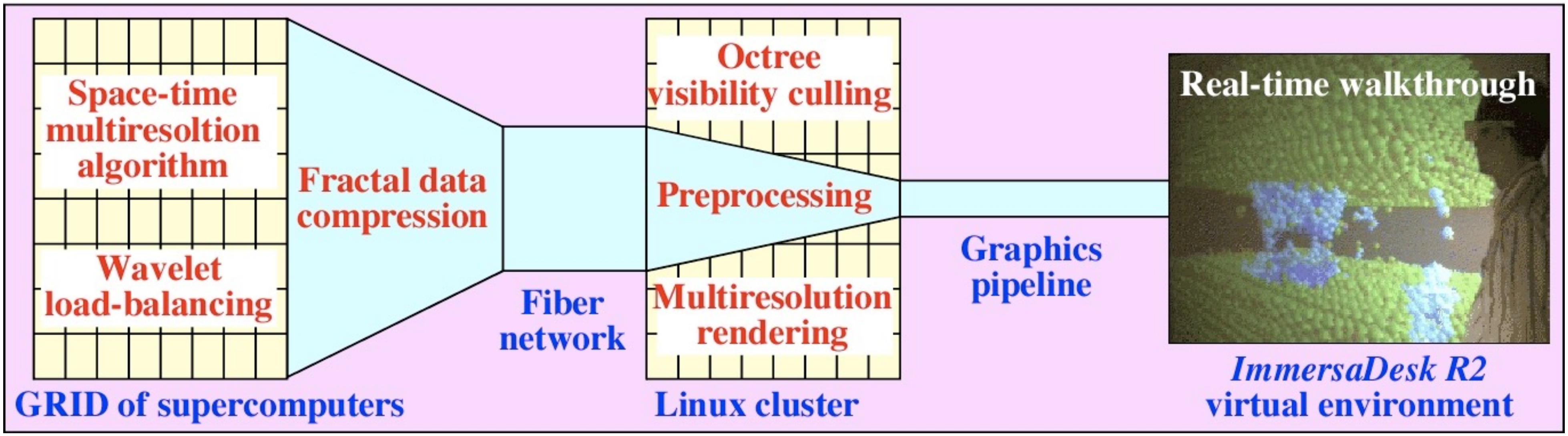
Collaboratory for Advanced Computing & Simulations
Department of Computer Science
Department of Physics & Astronomy
Department of Quantitative & Computational Biology
University of Southern California

Email: anakano@usc.edu

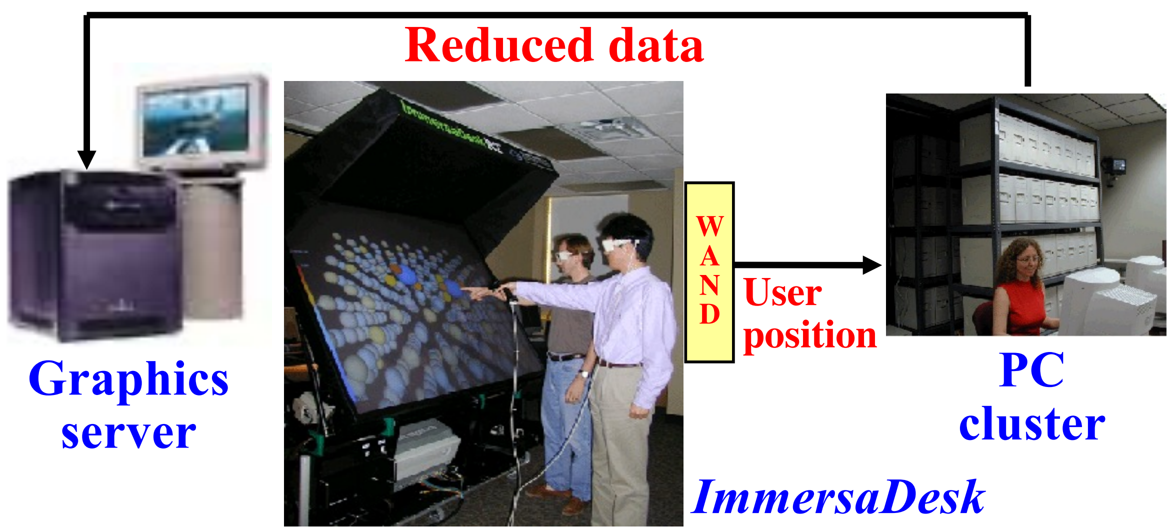
Goal: Visualize billion atoms in real time

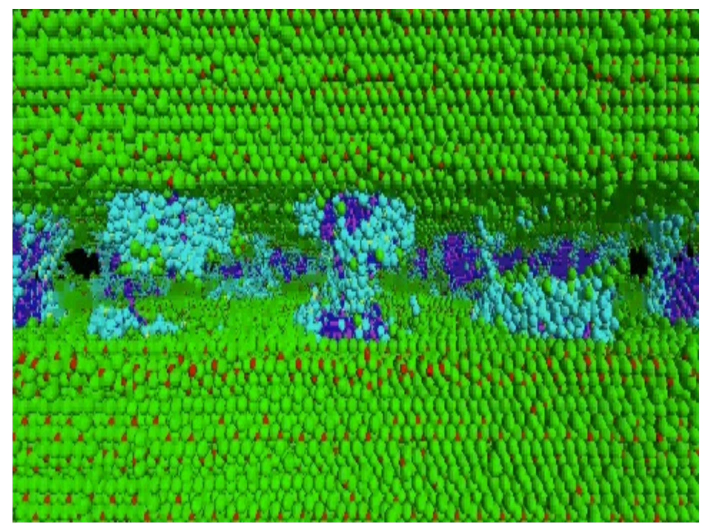
Immersive & Interactive Visualization

Billion-atom walkthrough



Parallel & distributed Atomsviewer





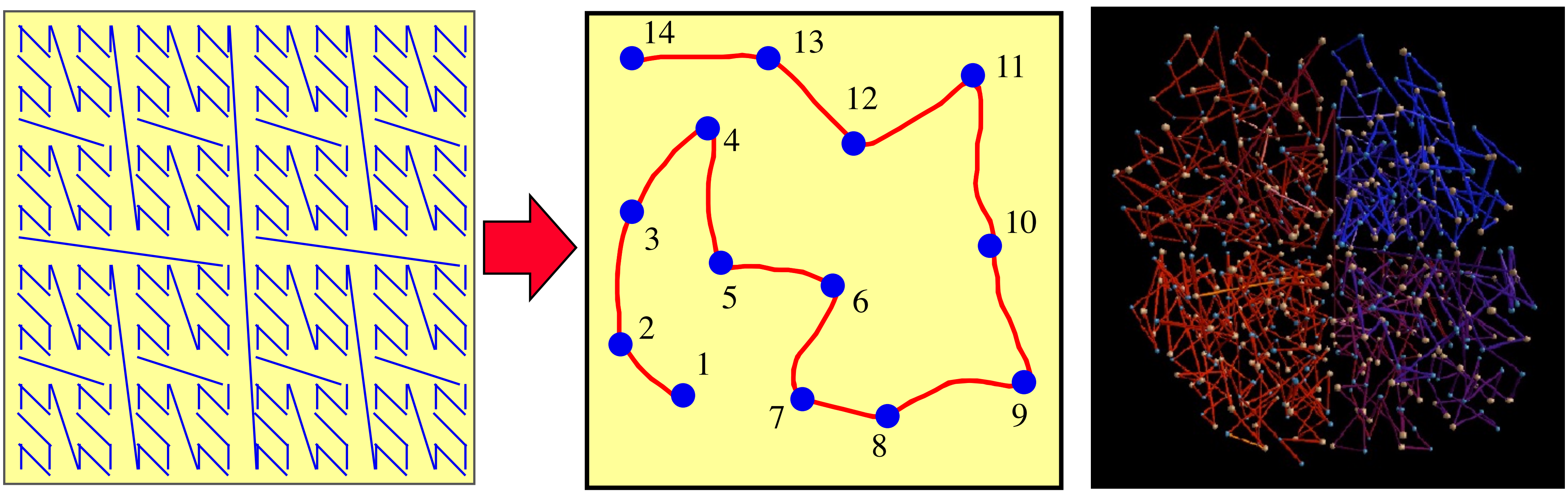
Locality in Data Compression

Challenge: Massive data transfer *via* wide area network:
75GB/step of data for 1.5 billion-atom MD!

→ Solution: Compressed software pipeline

Scalable encoding:
- **Store relative positions on spacefilling curve: $O(N\log N) \rightarrow O(N)$**

Result:
- **Data size, 50 Bytes/atom → 6 Bytes/atom**



Data Compression for Scalable I/O

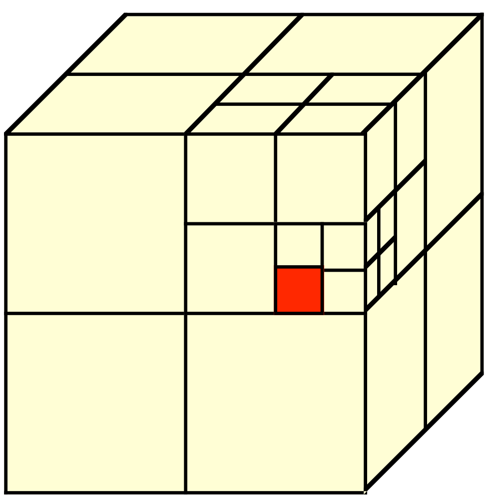
Challenge: Massive data transfer via OC-3 (155 Mbps)

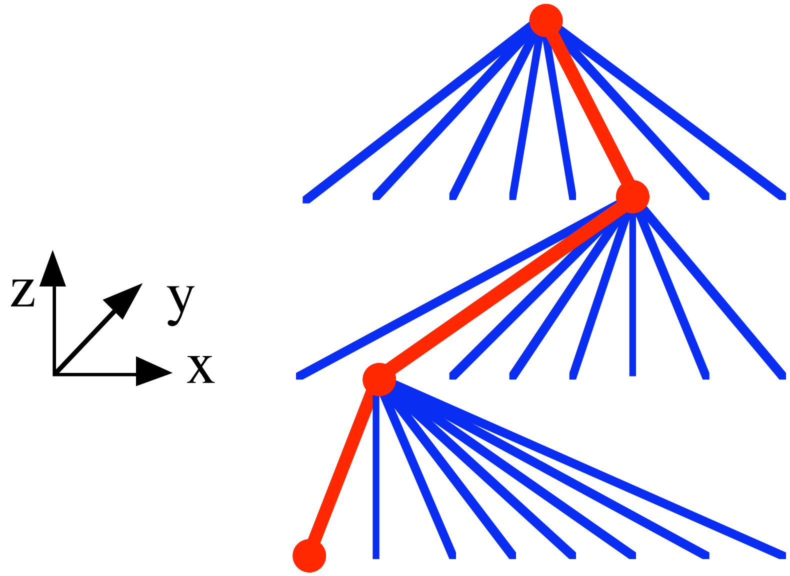
75 GB/frame of data for a 1.5-billion-atom MD!

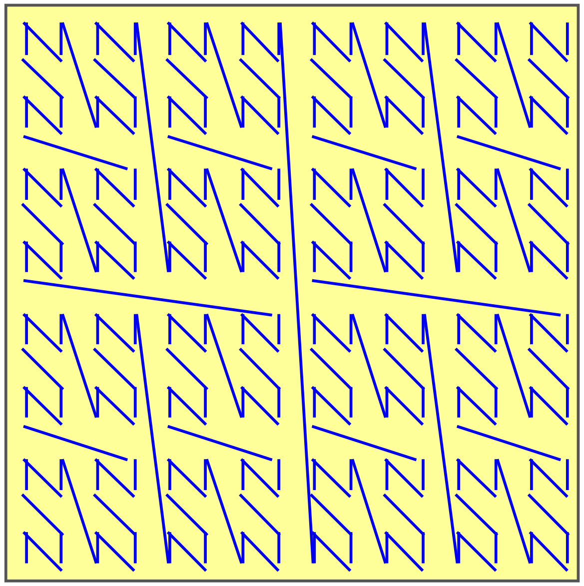
Scalable encoding:

- **Spacefilling curve based on octree index**

```
x =    1    1    0
y =   0    0    0
z =  1    0    0
-----------------
R =  101  001  000
```







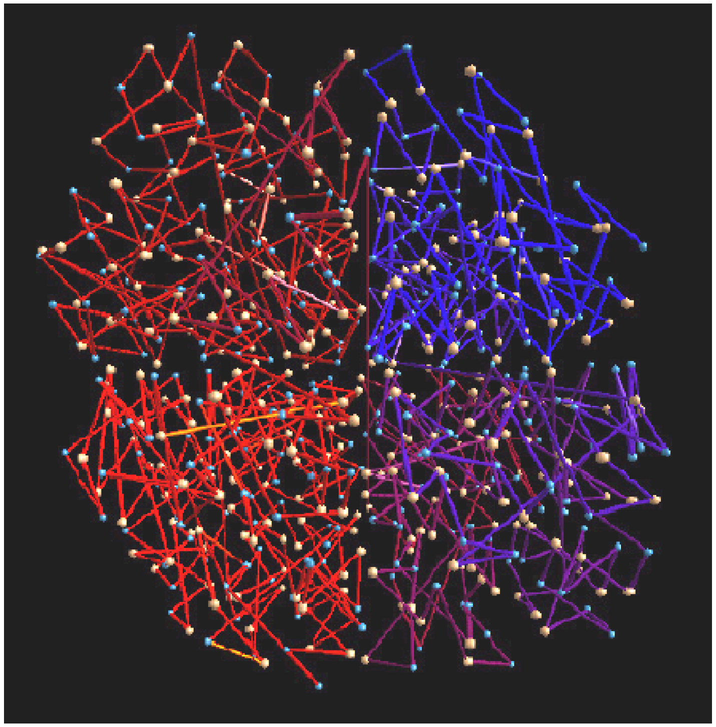
3D → list map preserves spatial proximity

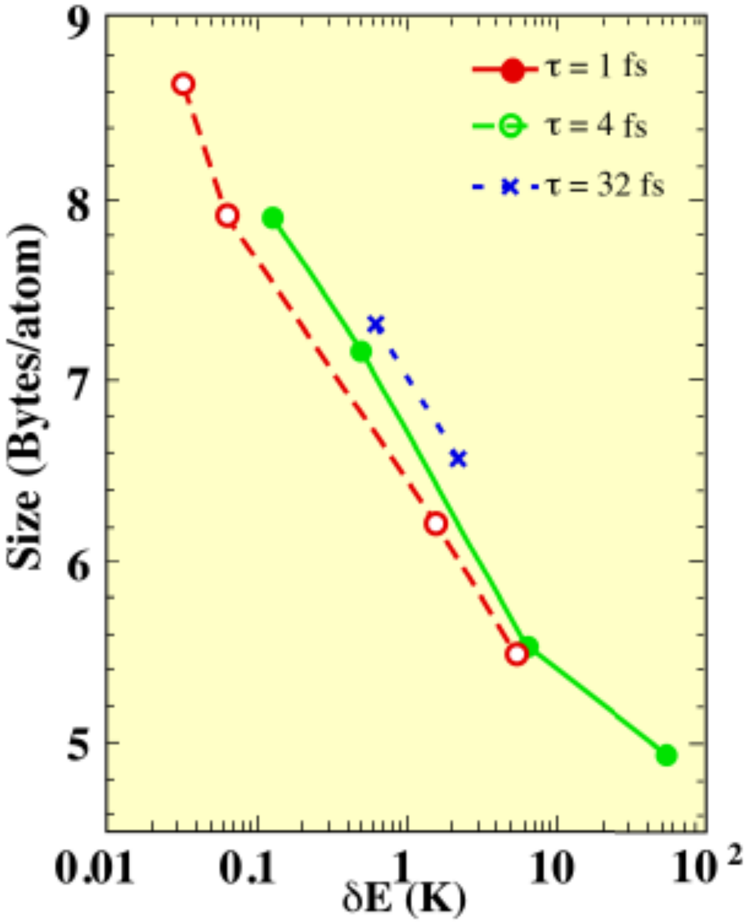
Spacefilling-Curve Data Compression

Algorithm:

1. **Sort particles along the spacefilling curve**
2. **Store relative positions: $O(N\log N) \rightarrow O(N)$**

- **Adaptive variable-length encoding to handle outliers**
- **User-controlled error bound**





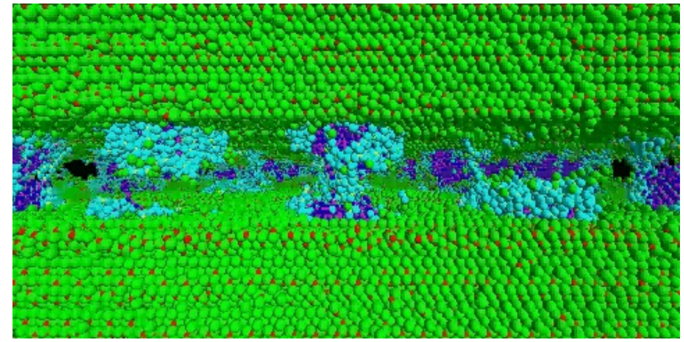
Result:

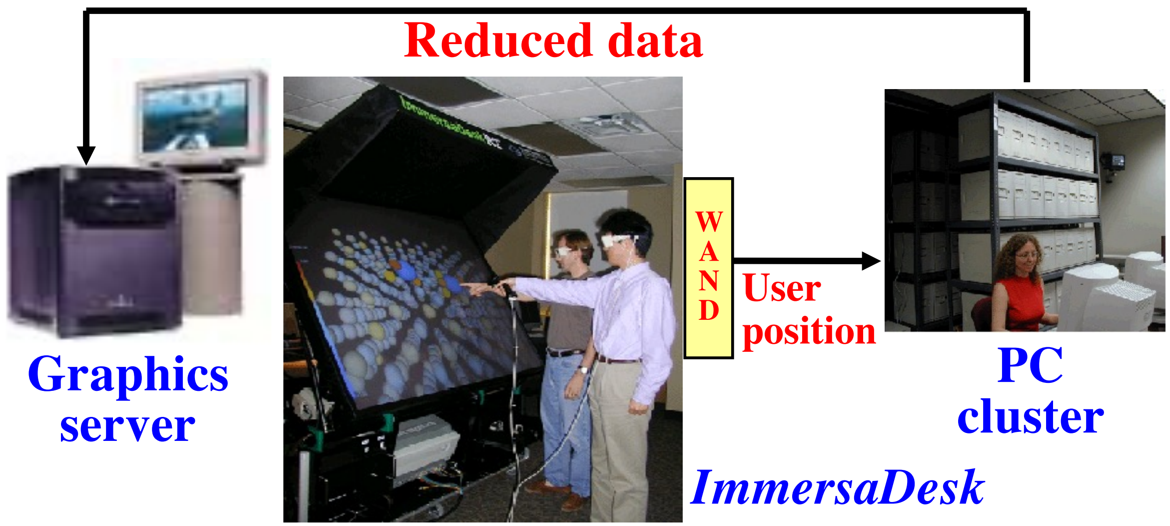
- **An order-of-magnitude reduction of I/O size: 50 → 6 Bytes/atom**

https://aiichironakano.github.io/Omelchenko-DataCmp-CPC00.pdf

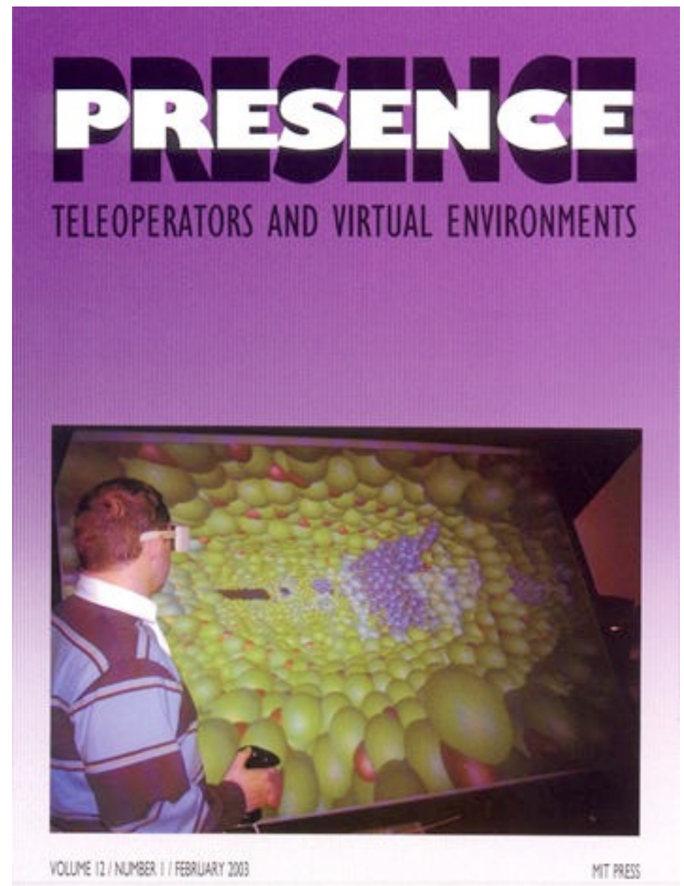
Data Locality in Visualization

- Octree-based fast view-frustum culling
- Probabilistic occlusion culling
- Parallel/distributed processing

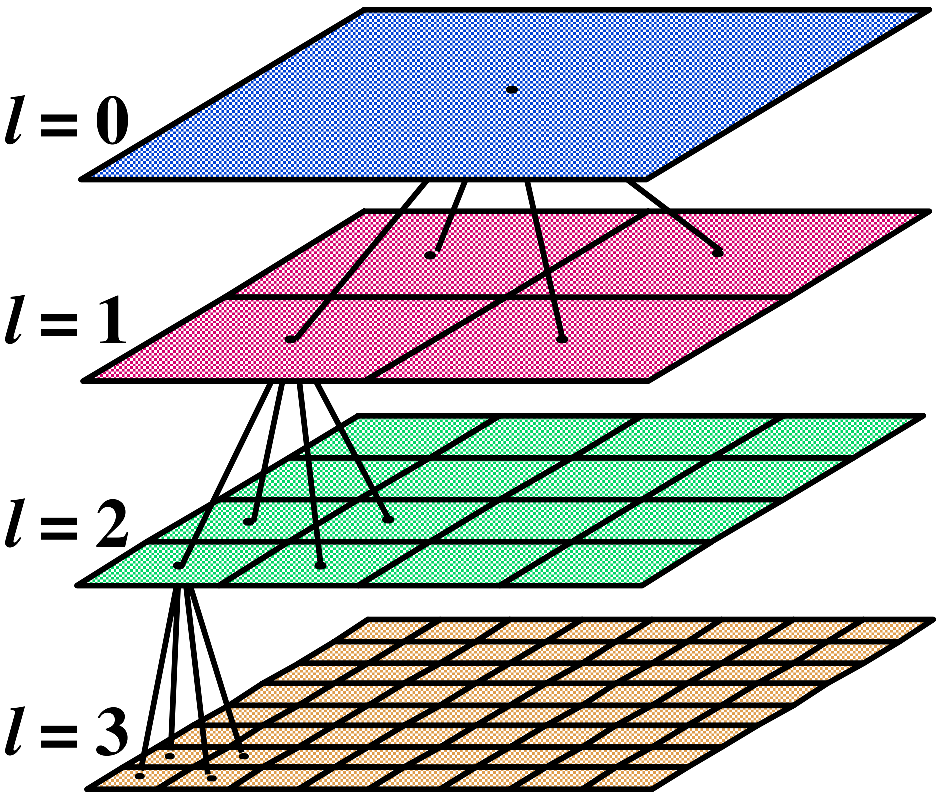




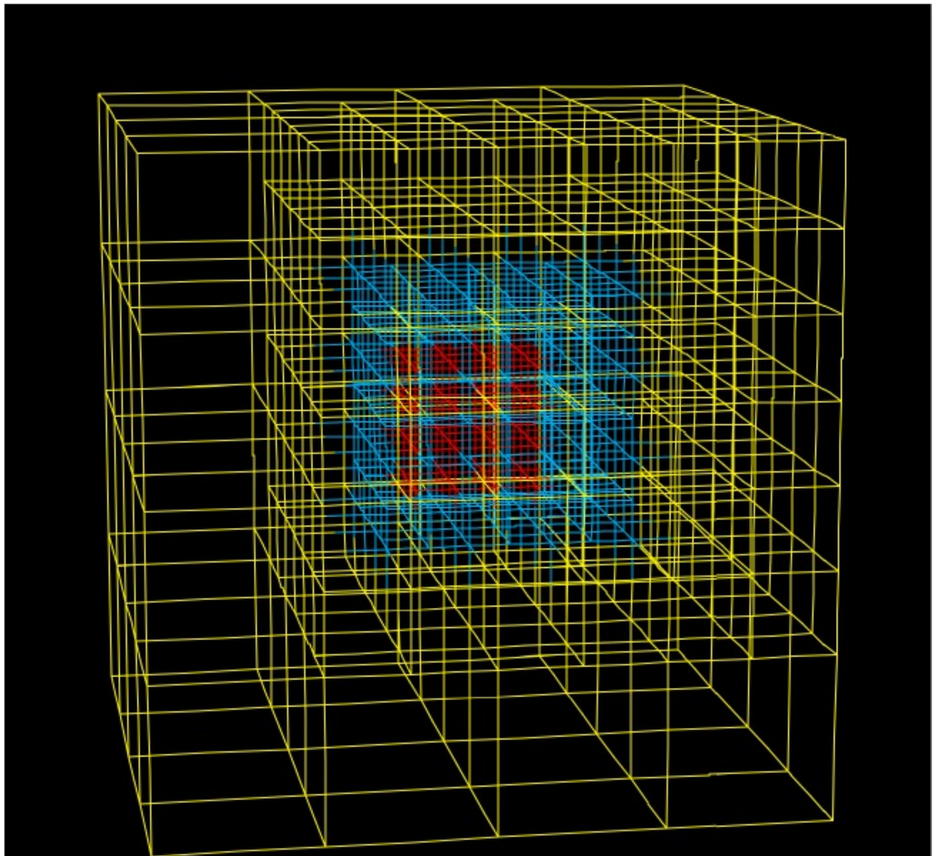
- Interactive visualization of a billion-atom dataset in immersive environment



Hierarchical Abstraction



2D example



Use of an octree

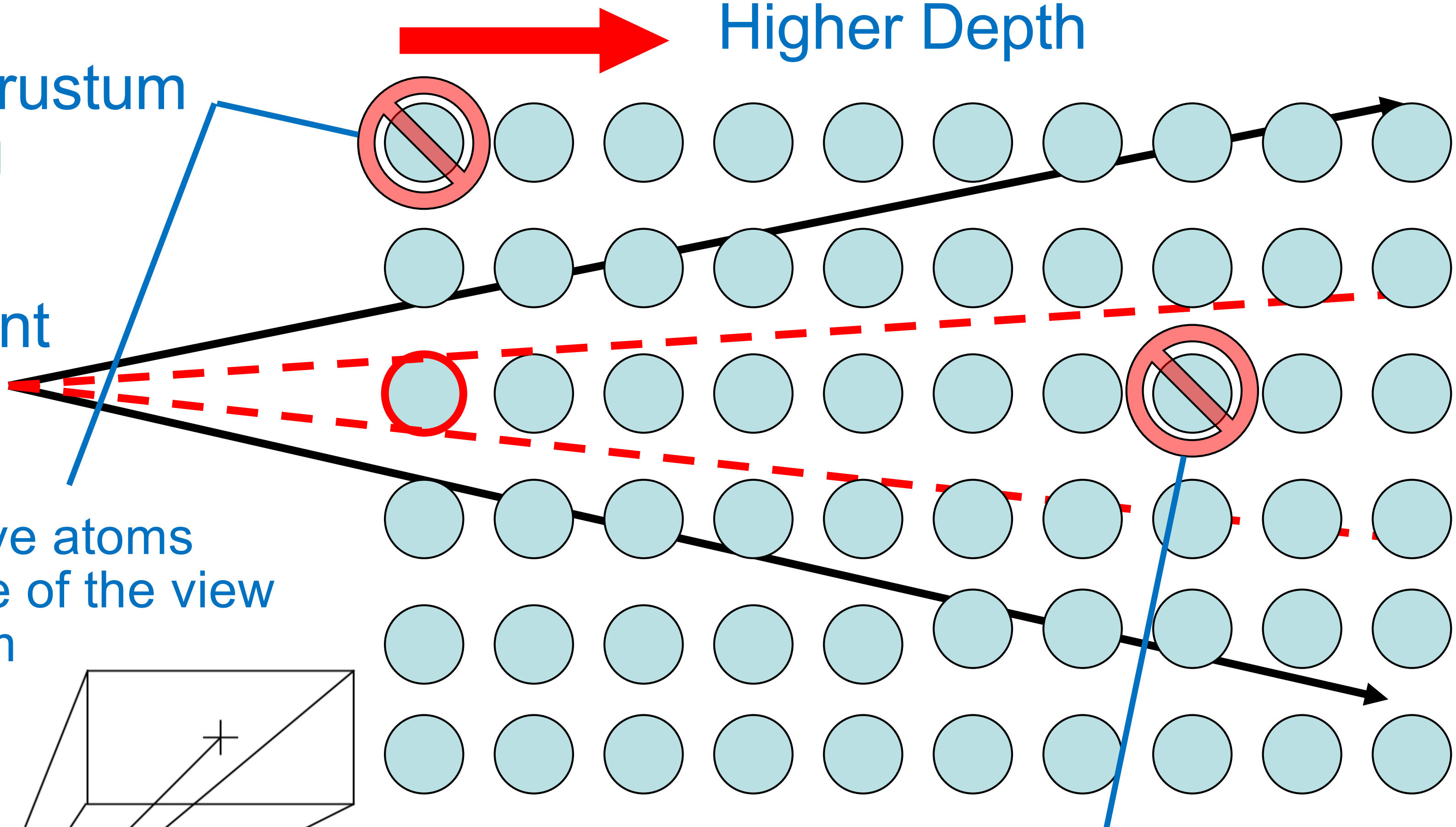
- Larger clusters for longer distances
- Recursively subdivide the 3D space to form an octree

Visibility Culling

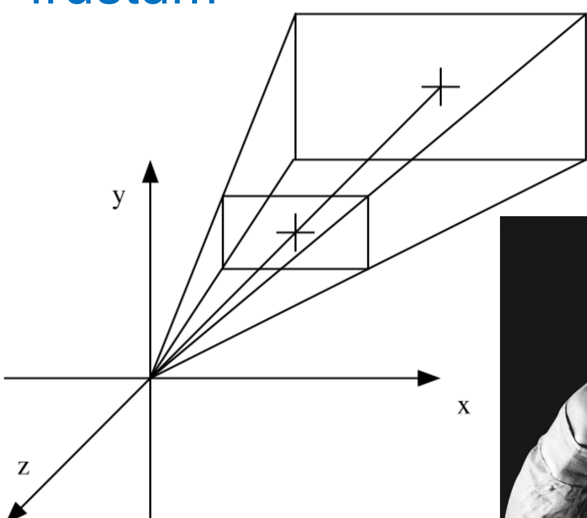
Higher Depth

View frustum culling

Viewpoint



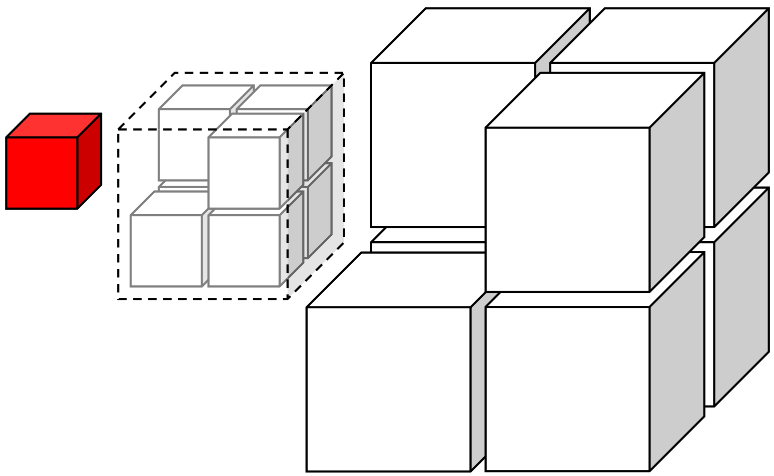
Remove atoms outside of the view frustum



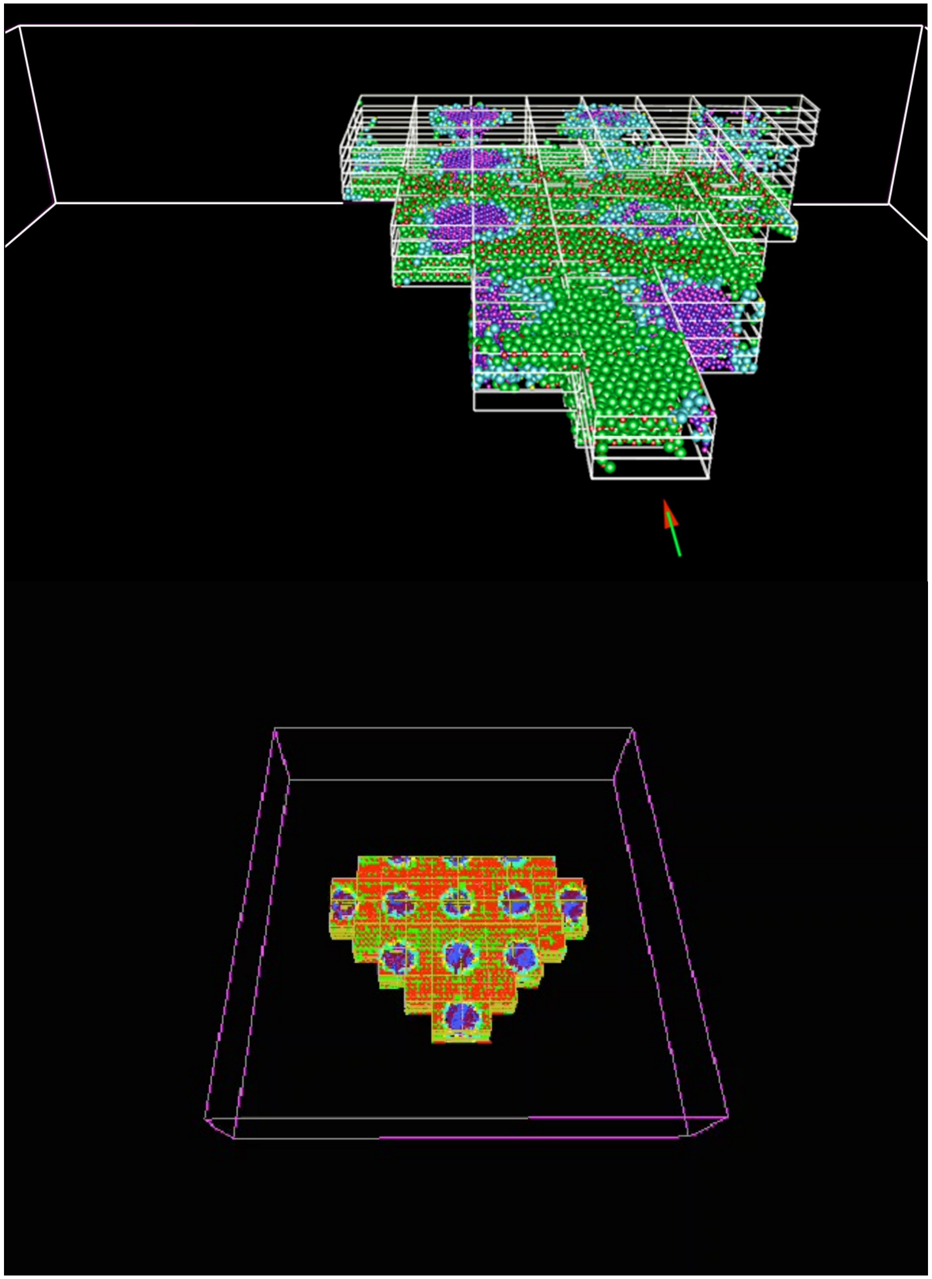
Occlusion culling

Remove atoms hidden by other atoms

Octree-based View-Frustum Culling

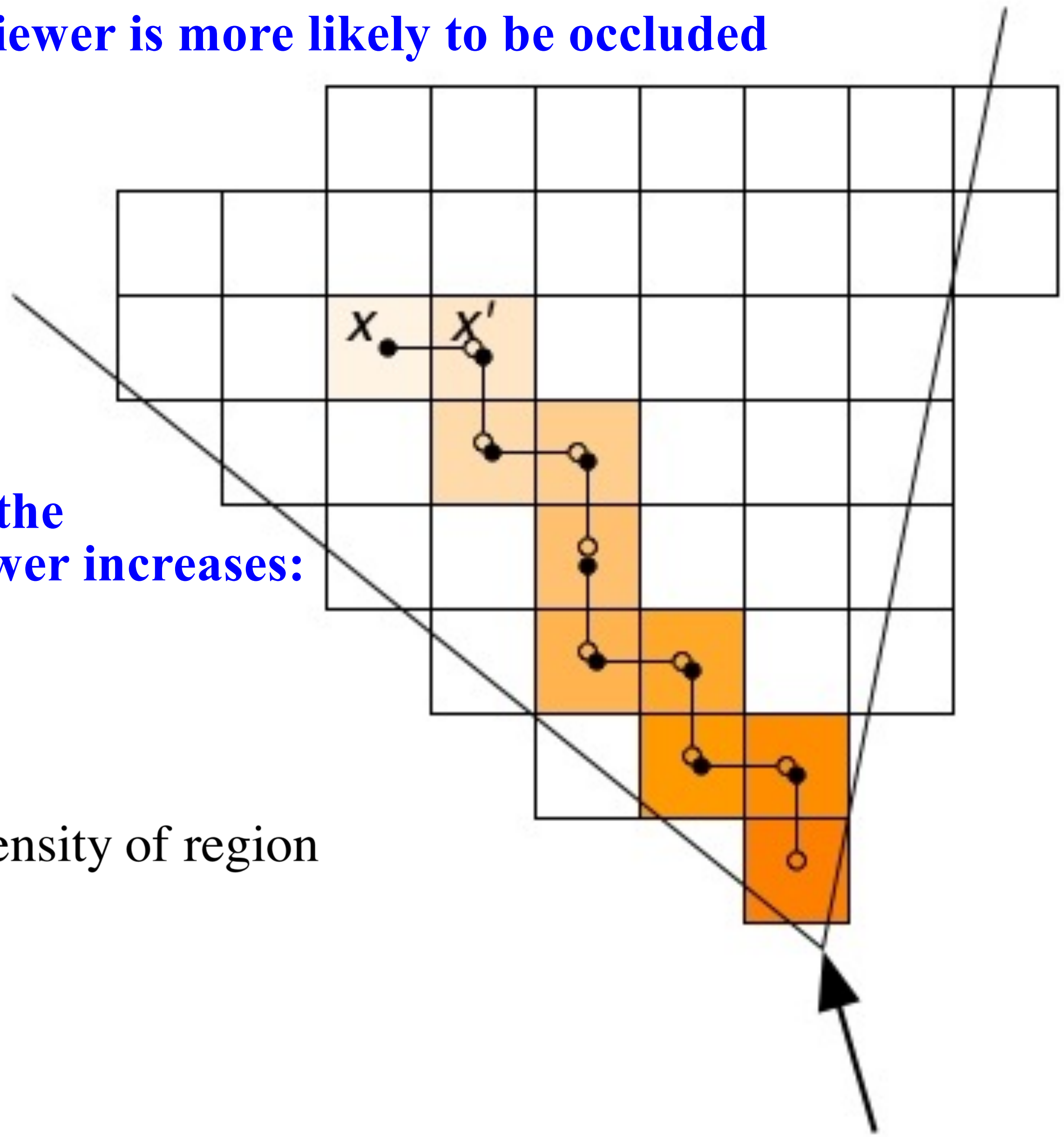


- Use the octree data structure to efficiently select only visible atoms
- Complexity
  Insertion into octree: $O(N)$
  Data extraction: $O(\log N)$



Probabilistic Occlusion Culling

- **Remove atoms that are occluded by other atoms closer to the viewer**
- **Regions farther away from the viewer is more likely to be occluded than one in front of the viewer**



- **Draw fewer atoms per region as the distance of a region from the viewer increases: visibility value *v*(*x*) for region *x***
- **Recurrence along the view line**

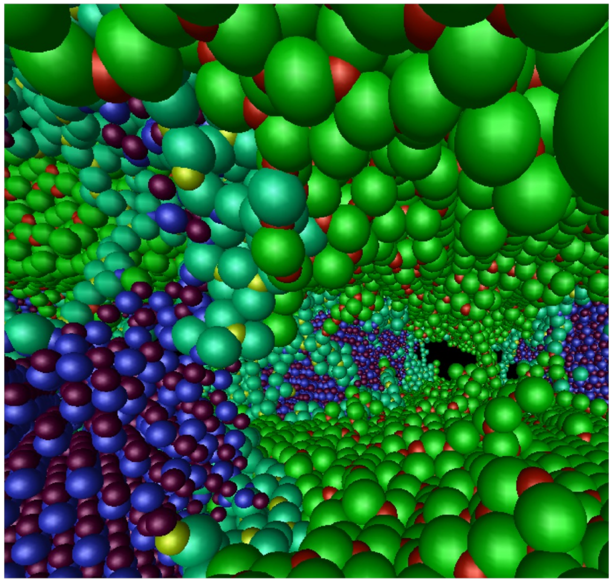
$$v_x = \begin{cases} 1 & x = 0 \\ f(D_{x'}, v_{x'}) & \text{else} \end{cases} \qquad D_x = \text{density of region}$$

- **Run-time adaptation**

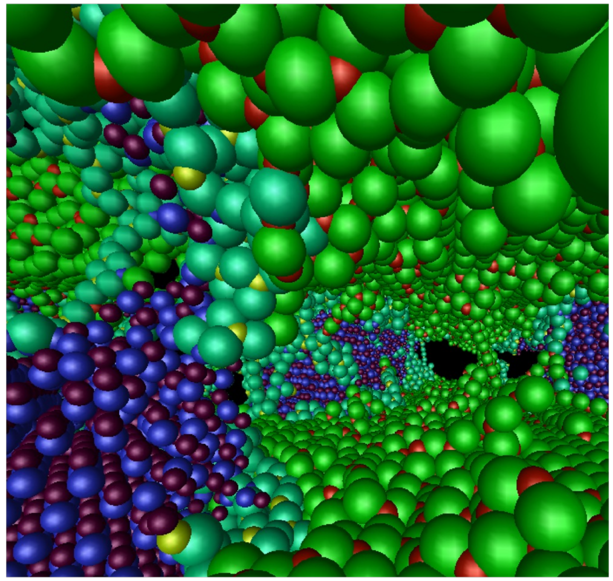
$$v'_x = f(\text{user speed}) \times v_x$$

Results of Probabilistic Occlusion Culling

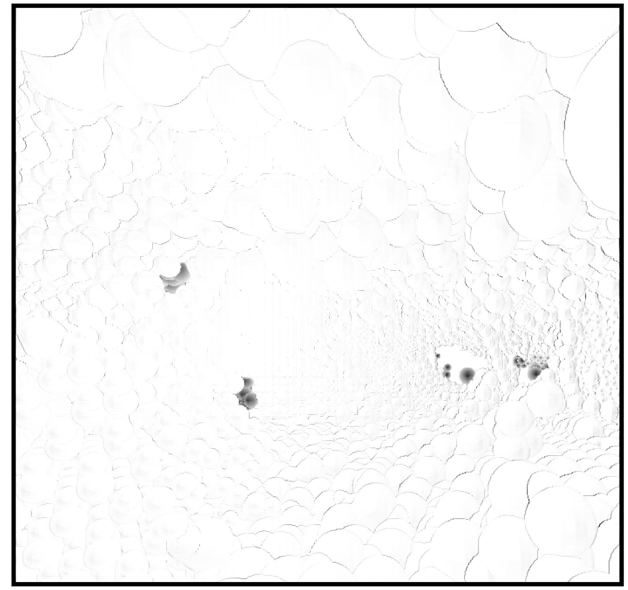
Original



Probabilistic



Difference



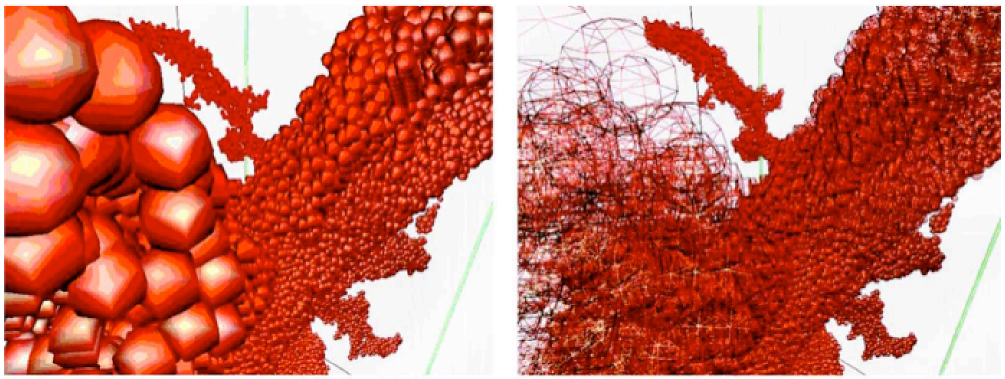
68% fewer objects
3× frame rate

Multiresolution Culling & Rendering

- Per-octree node operations:
  - —Frustum culling
  - —Probabilistic occlusion culling
- Per-atom operations
  - —Multiple levels-of-detail
  - —Occlusion culling (per-object, per-octree node)

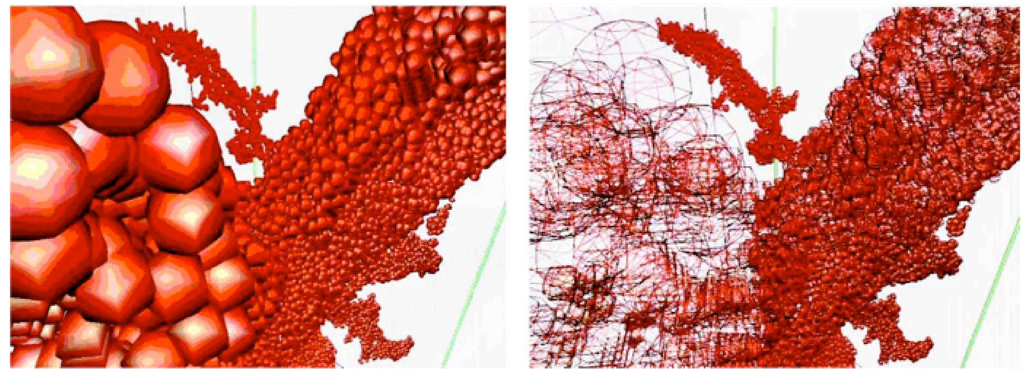
Use less # of polygons for farther atoms

Without multiresolution



.94fps - 90,000 particles

With multiresolution

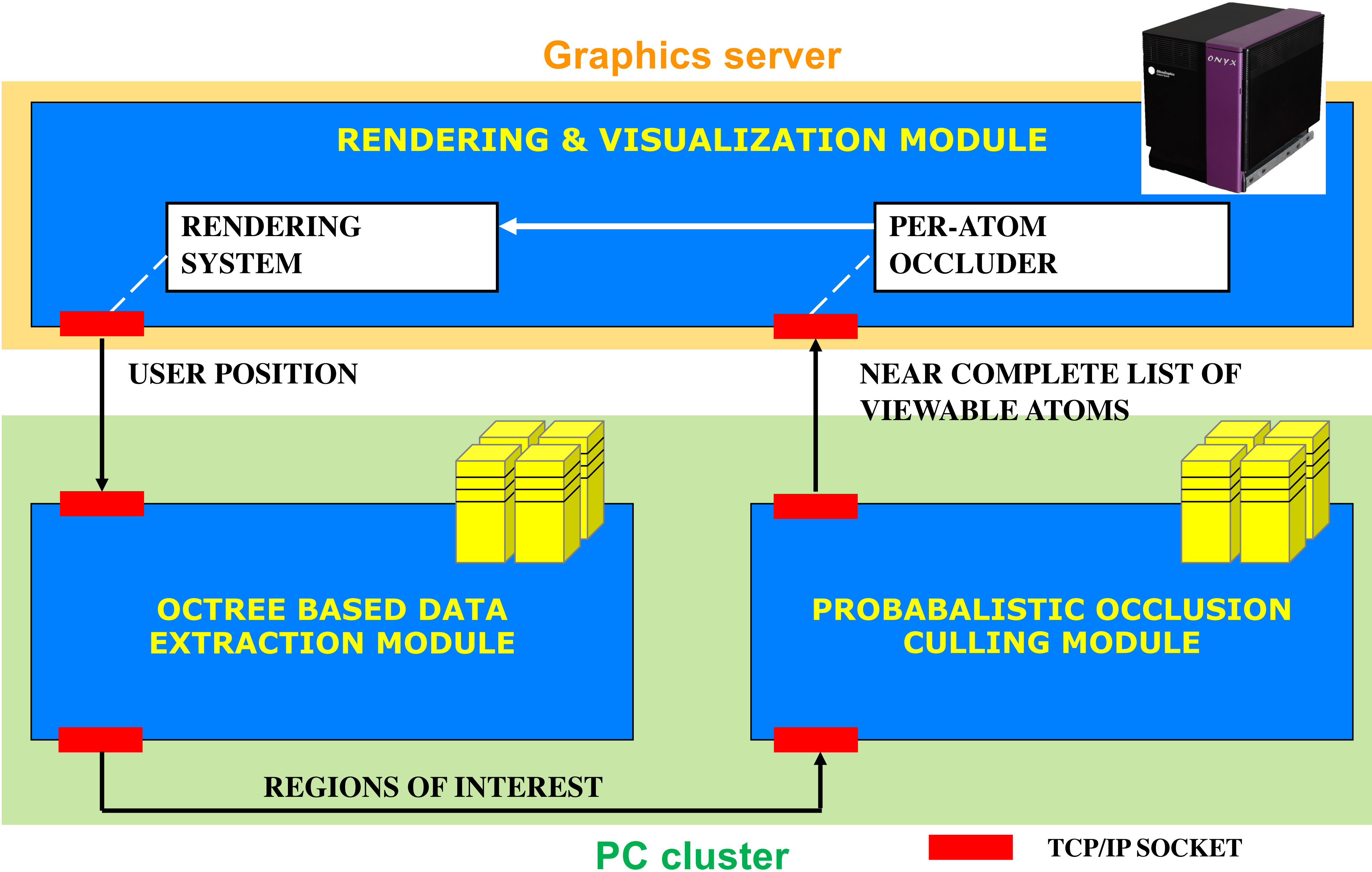


3.2fps - 4,500 particles

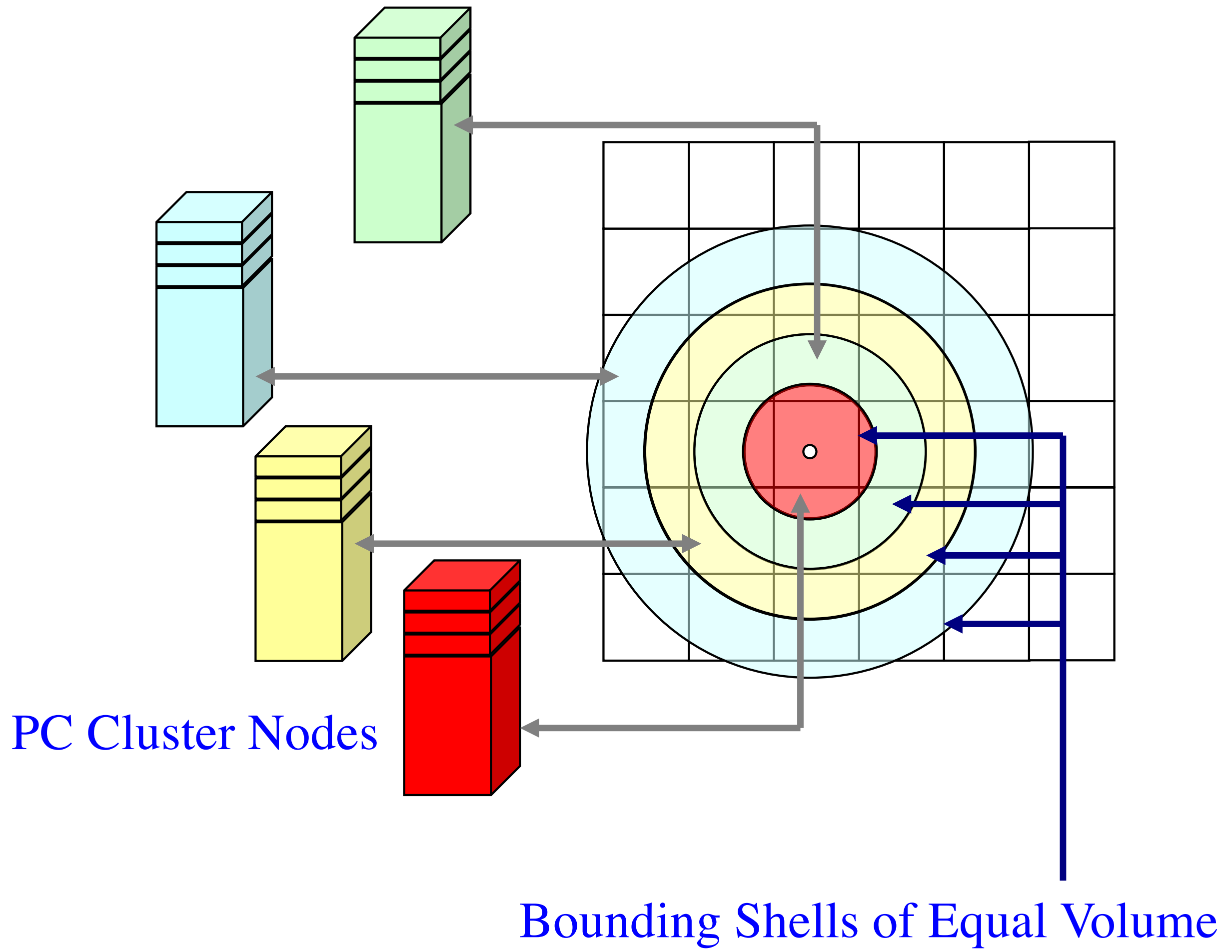
fps: frames rendered per second

Outflow pathways of optic nerves from the retina of a rabbit eye (Experimental data by C. Burgoyne & R. Beuerman, LSU Eye Center)

Distributed Architecture



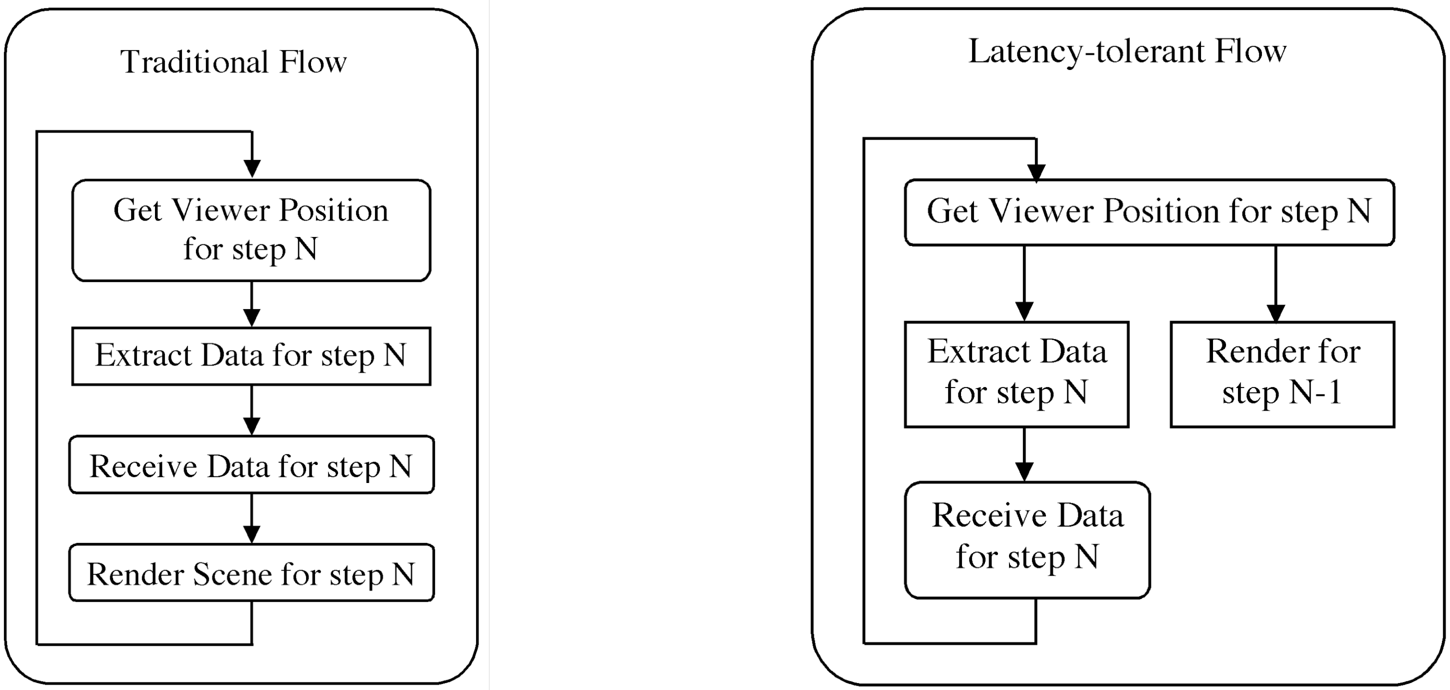
Parallel Octree Extraction



- Individual copies of the octree with each computing node
- Spatial decomposition using concentric shells of equal volume
- Load balancing due to the equal use of each processor for extraction

Latency Hiding

- **Individual modules are multithreaded to reduce network or module latency; *cf.* OpenMP**
- **Minimize latency due to inter-modular dependencies by overlapping the inter-module communication and module computation; *cf.* computation-communication overlap by MPI_Irecv**

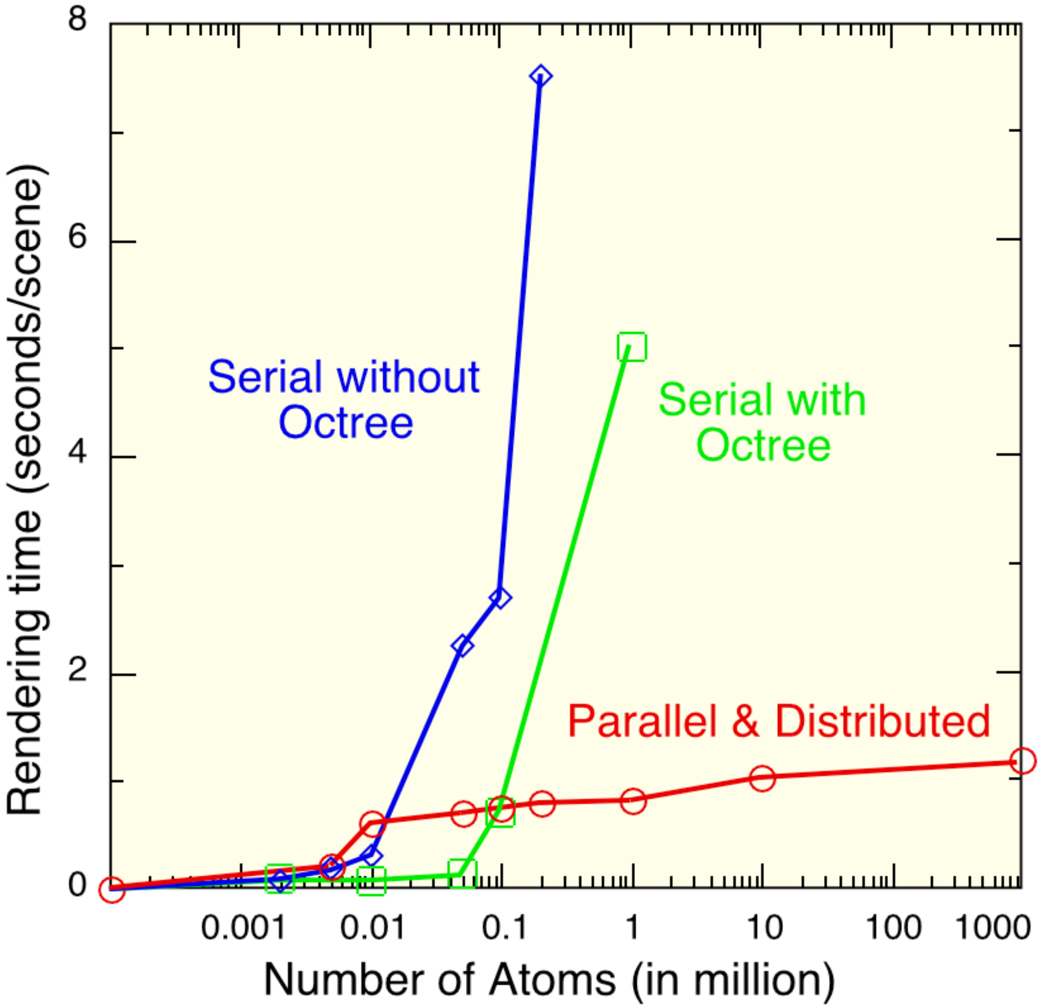


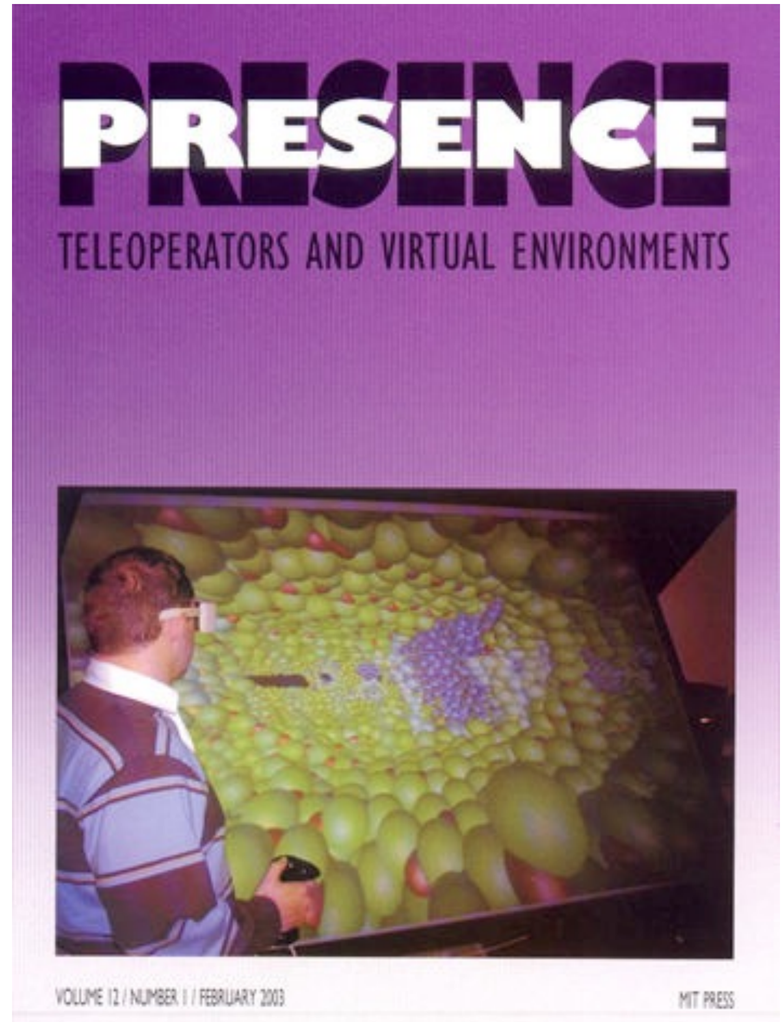
- **Instantaneously trained neural network (CC4** [Tang & Kak, *CSSP'98*]**) predicts the user's next position** [Liu *et al*., *PDPTA'02*]

https://aiichironakano.github.io/cs596/Liu-VizNN-PDPTA02.pdf

Parallel & Distributed Atomsviewer

Real-time walkthrough for a billion atoms on an SGI Onyx2 (2 × MIPS R10K, 4GB RAM) connected to a PC cluster (4 × 800MHz P3)





IEEE Virtual Reality Best Paper

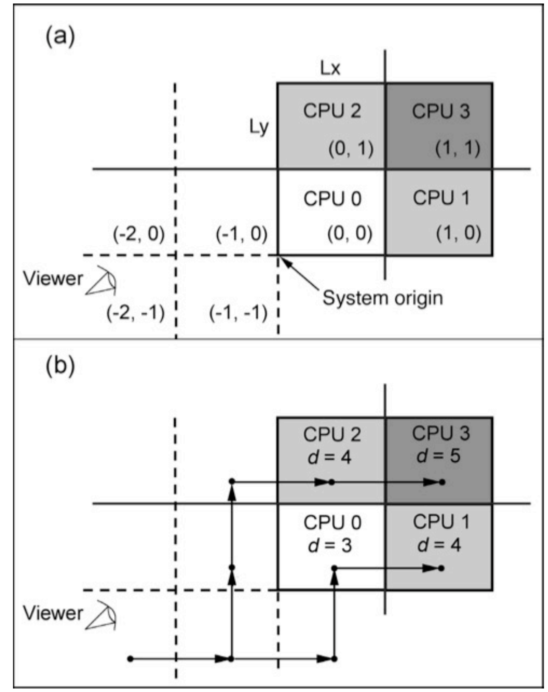
https://aiichironakano.github.io/cs596/Sharma-Viz-Presence03.pdf

In Situ Parallel Rendering

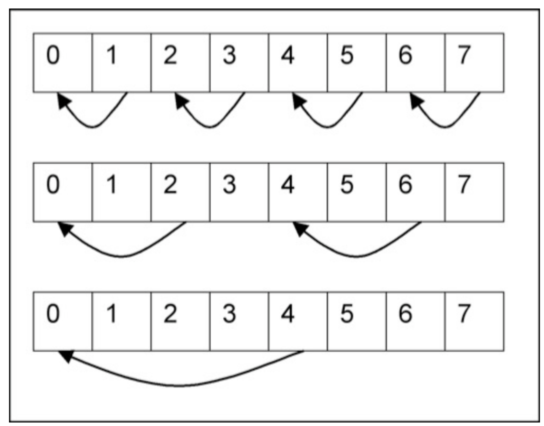
International Journal of Computational Science
1992-6669 (Print) 1992-6677 (Online) © Global Information Publisher
2007, Vol. 1, No. 4, 407-421

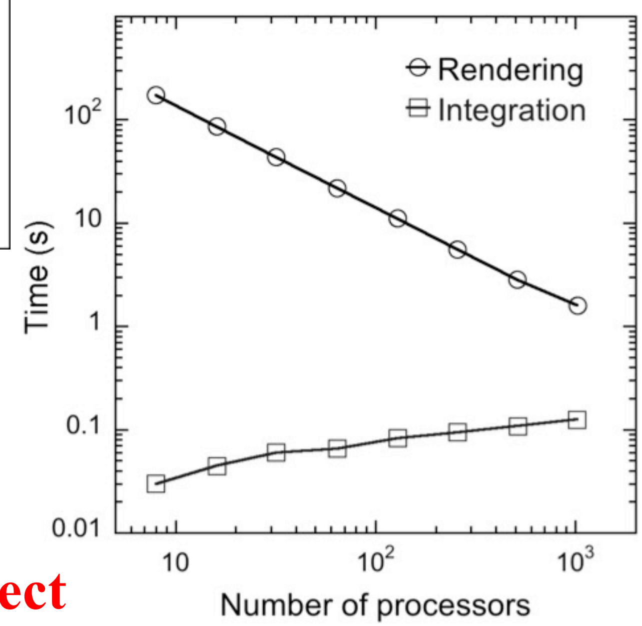
ParaViz: A Spatially Decomposed Parallel Visualization Algorithm Using Hierarchical Visibility Ordering

Cheng Zhang[1], Scott Callaghan[2], Thomas Jordan[2], Rajiv K. Kalia[1],
Aiichiro Nakano[1*], Priya Vashishta[1]



- **Parallel rendering of spatially distributed data: hybrid sort-first/sort-last (who does what)**
- **Scalable depth buffer by domain-level distributed visibility ordering**
- **On-the-fly visualization of parallel simulation without data migration**
- **Parallel efficiency 0.98 on 1,024 processors for 16.8 million-atom molecular-dynamics simulation**



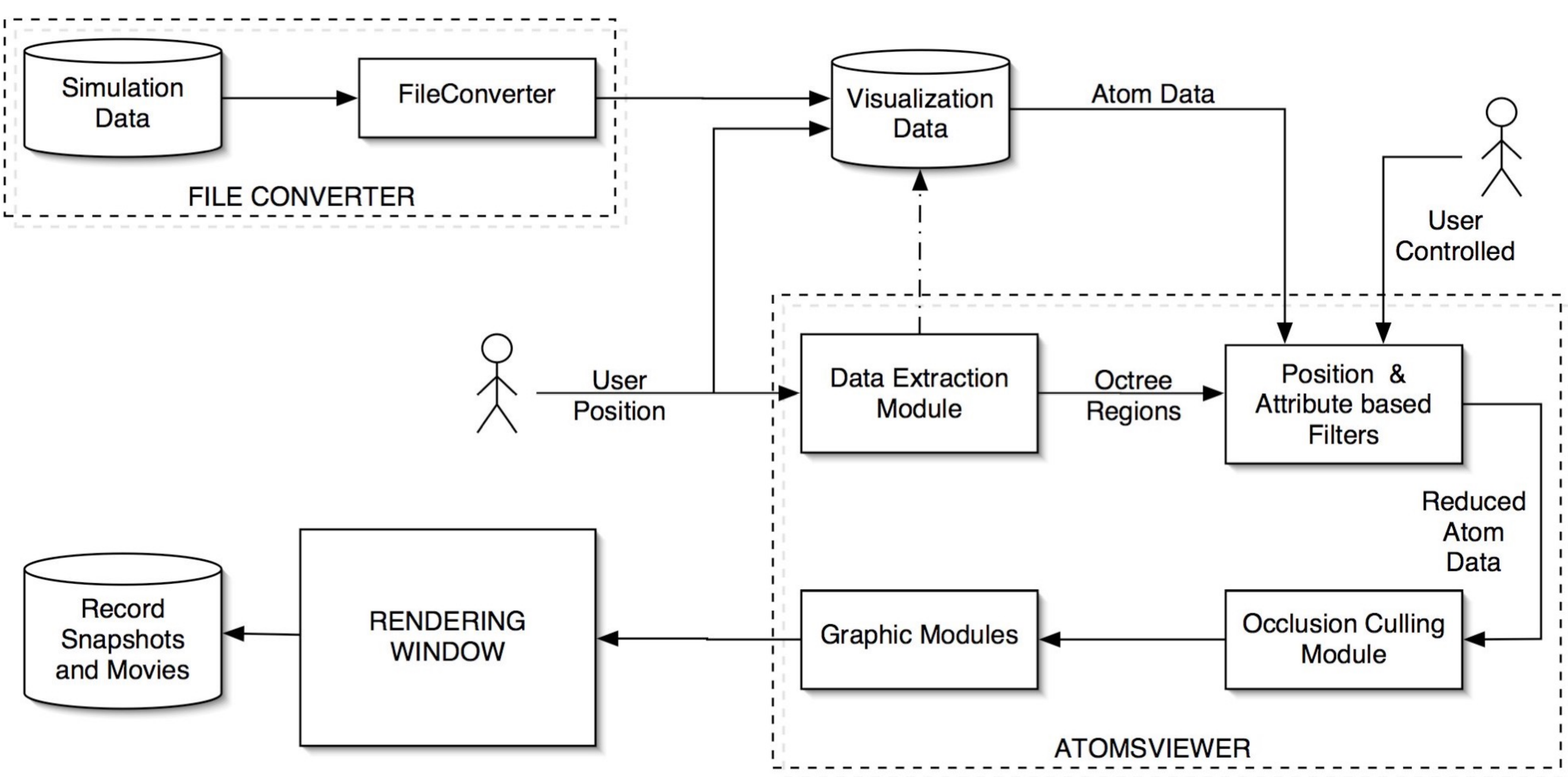


CSCI 596 final project

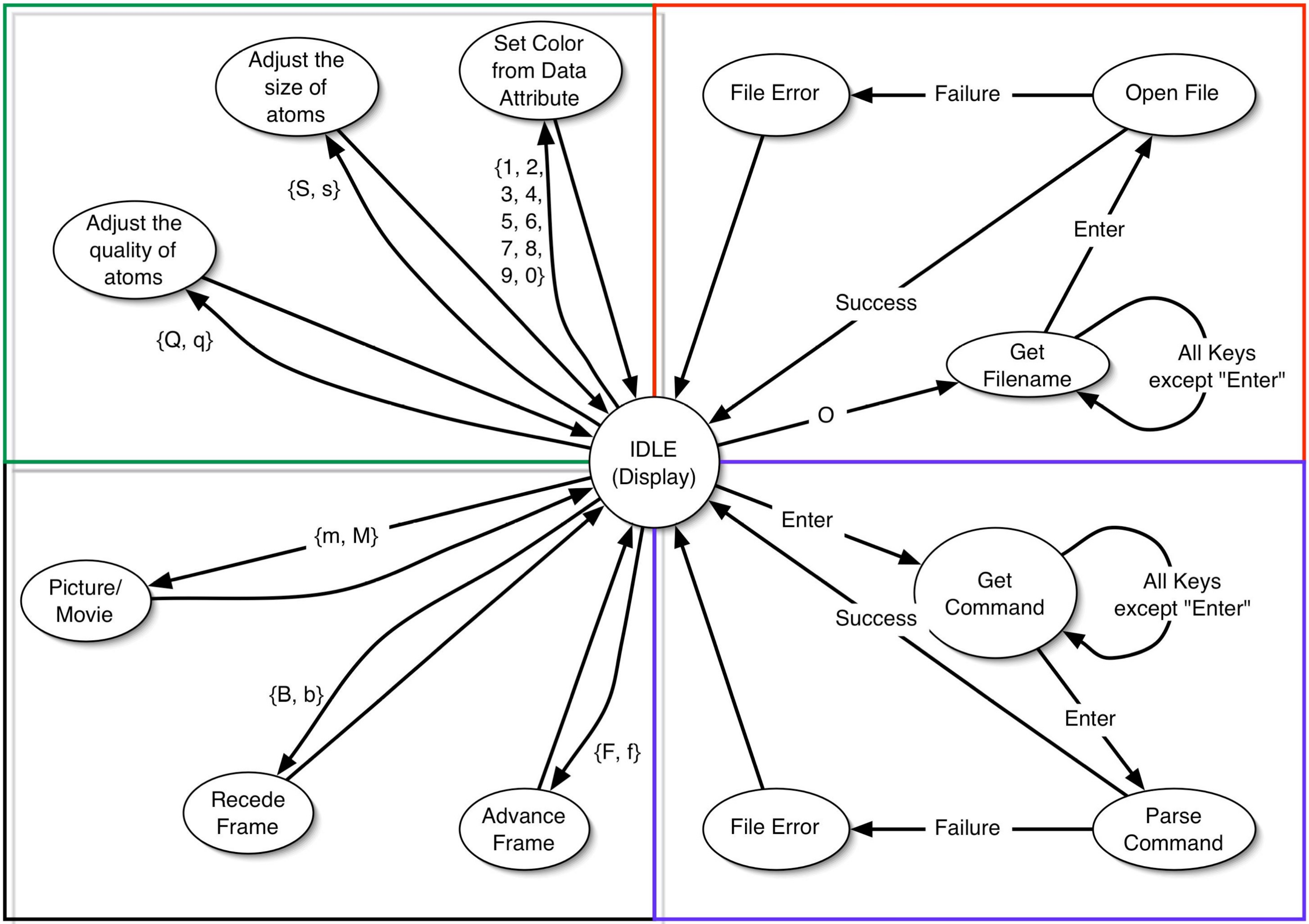
Atomsviewer Code

- **Programming language**
  >C++
- **Graphics**
  >OpenGL
  >CAVE Library (optional)
- **Platforms**
  >Windows
  >Macintosh OS X
  >SGI Irix

Atomsviewer System



Atomsviewer Commands



Atomsviewer Code Dissemination

Computer Physics Communications Program Library

ELSEVIER

Available online at www.sciencedirect.com

SCIENCE @ DIRECT®

Computer Physics Communications 163 (2004) 53–64

Computer Physics Communications

www.elsevier.com/locate/cpc

https://aiichironakano.github.io/cs596/Sharma-viz-CPC04.pdf

Scalable and portable visualization of large atomistic datasets ☆

Ashish Sharma *, Rajiv K. Kalia, Aiichiro Nakano, Priya Vashishta

Collaboratory for Advanced Computing and Simulations, Department of Computer Science, Department of Physics & Astronomy, Department of Material Science & Engineering, University of Southern California, Los Angeles, CA 90089-0242, USA

Received 15 June 2004; accepted 8 July 2004

Available online 16 September 2004

Abstract

A scalable and portable code named Atomsviewer has been developed to interactively visualize a large atomistic dataset consisting of up to a billion atoms. The code uses a hierarchical view frustum-culling algorithm based on the octree data structure to efficiently remove atoms outside of the user's field-of-view. Probabilistic and depth-based occlusion-culling algorithms then select atoms, which have a high probability of being visible. Finally a multiresolution algorithm is used to render the selected subset of visible atoms at varying levels of detail. Atomsviewer is written in C++ and OpenGL, and it has been tested on a number of architectures including Windows, Macintosh, and SGI. Atomsviewer has been used to visualize tens of millions of atoms on a standard desktop computer and, in its parallel version, up to a billion atoms.

Program summary

Title of program: Atomsviewer
Catalogue identifier: ADUM
Program summary URL: http://cpc.cs.qub.ac.uk/summaries/ADUM
Program obtainable from: CPC Program Library, Queen's University of Belfast, N. Ireland
Computer for which the program is designed and others on which it has been tested: 2.4 GHz Pentium 4/Xeon processor, professional graphics card; Apple G4 (867 MHz)/G5, professional graphics card
Operating systems under which the program has been tested: Windows 2000/XP, Mac OS 10.2/10.3, SGI IRIX 6.5
Programming languages used: C++, C and OpenGL
Memory required to execute with typical data: 1 gigabyte of RAM
High speed storage required: 60 gigabytes
No. of lines in the distributed program including test data, etc.: 550 241
No. of bytes in the distributed program including test data, etc.: 6 258 245
Number of bits in a word: Arbitrary

Submit your code/paper to *CPC*!

☆ This paper and its associated computer program are available via the Computer Physics Communications homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding author.
E-mail address: anakano@usc.edu (A. Sharma).

doi:10.1016/j.cpc.2004.07.008