Massive Dataset Visualization

Aiichiro Nakano

Collaboratory for Advanced Computing & Simulations
Dept. of Computer Science, Dept. of Physics & Astronomy,
Dept. of Chemical Engineering & Materials Science,
Dept. of Biological Sciences
University of Southern California

Email: anakano@usc.edu
Challenge: billion-atom walkthrough

Solution: parallel & distributed Atomviewer

Immersive & Interactive Visualization
Locality in Data Compression

Massive data transfer via wide area network:
75 GB/step of data for 1.5 billion-atom MD!
→ Compressed software pipeline

Scalable encoding:
• Store relative positions on spacefilling curve: $O(N \log N) \to O(N)$

Result:
• Data size, 50Bytes/atom → 6 Bytes/atom
Data Compression for Scalable I/O

Challenge: Massive data transfer via OC-12 (622 Mbps)
75 GB/frame of data for a 1.5-billion-atom MD!

Scalable encoding:
• Spacefilling curve based on octree index

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: $50 \rightarrow 6$ Bytes/atom
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

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

View frustum culling

Higher Depth

Viewpoint

Occlusion culling
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
  $$v_x = \begin{cases} 
  1 & x = 0 \\
  f(D_{x'}, v_{x'}) & \text{else} \\
  \end{cases}$$
  $D_x =$ 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)

Without multiresolution: 0.94fps - 90,000 particles

With multiresolution: 3.2fps - 4,500 particles

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

- OCTREE BASED DATA EXTRACTION MODULE
- PROBABALISTIC OCCLUSION CULLING MODULE
- RENDERING & VISUALIZATION MODULE
  - RENDERING SYSTEM
  - PER-ATOM OCCLUDER
  - NEAR COMPLETE LIST OF VIEWABLE ATOMS
- TCP/IP SOCKET
- USER POSITION
- REGIONS OF INTEREST
Parallel Octree Extraction

- Individual copies of the octree with each node
- Spherical extraction by the use of 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
- Minimize latency due to inter-modular dependencies by overlapping the inter-module communication and module computation

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

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

Cheng Zhang¹, Scott Callaghan², Thomas Jordan², Rajiv K. Kalia¹,
Aiichiro Nakano¹*, Priya Vashishta¹

- Parallel (software) rendering of spatially distributed data: hybrid sort-first/sort-last
- 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

Soft rendering
http://www.mesa3d.org
Atomsvviewer Code

• Programming language
  > C++

• Graphics
  > OpenGL
  > CAVE Library (optional)

• Platforms
  > Windows
  > Macintosh OS X
  > SGI Irix
Atomviewer System

FILE CONVERTER

Simulation Data → FileConverter → Visualization Data → Atom Data

Data Extraction Module

User Controlled

User Position → Data Extraction Module → Octree Regions → Position & Attribute based Filters

Reduced Atom Data

Record Snapshots and Movies

RENDERING WINDOW

Graphic Modules → Occlusion Culling Module → ATOMSVIEWER
Atomviewer Commands

Diagram showing commands and their interactions.
Scalable and portable visualization of large atomistic datasets

Ashish Sharma*, Rajiv K. Kalia, Aichihiro 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 Atomviewer has been developed to interactively visualizing a large atomistic dataset consisting of up to a billion atoms. The code uses a hierarchical view frustum-culling algorithm based on the viewer 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 multiscale algorithm is used to render the selected subset of visible atoms at varying levels of detail. Atomviewer is written in C++, and OpenGl and it has been tested on a number of architectures including Windows, Macintosh, and SGI. Atomviewer 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: Atomviewer
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 was designed and others on which it has been tested: 2.4 GHz Pentium 4/Linux processor, professional graphics card: Apple G4 (867 MHz)/CS, professional graphics card

Operating systems under which the program has been tested: Windows 2000/XP, Mac OS 10.2/10.3, SGIIRIX 6.5

Programming language 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

* 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 addresses: msharma@usc.edu (A. Sharma).

0010-4655/ – see front matter © 2004 Elsevier B.V. All rights reserved.

http://www.cpc.cs.qub.ac.uk/cpc
Other Visualization Tools

- **VisIT** visualization tool at Lawrence Livermore National Laboratory
  
  http://www.llnl.gov/visit/

- **ParaView** parallel visualization application at Los Alamos National Laboratory
  
  http://www.paraview.org/New/index.html

- **VMD** visual molecular dynamics at University of Illinois at Urbana-Champaign
  
  http://www.ks.uiuc.edu/Research/vmd/