Visualizing Terascale Datasets with Impostors

Thomas Quinn
University of Washington
Astronomy

G. Lake, T. Quinn, J. Stadel, J. Wadsley,
J. Gardner, G. Stinson, G. Lufkin, R. Roskar

Parallel Programming Laboratory

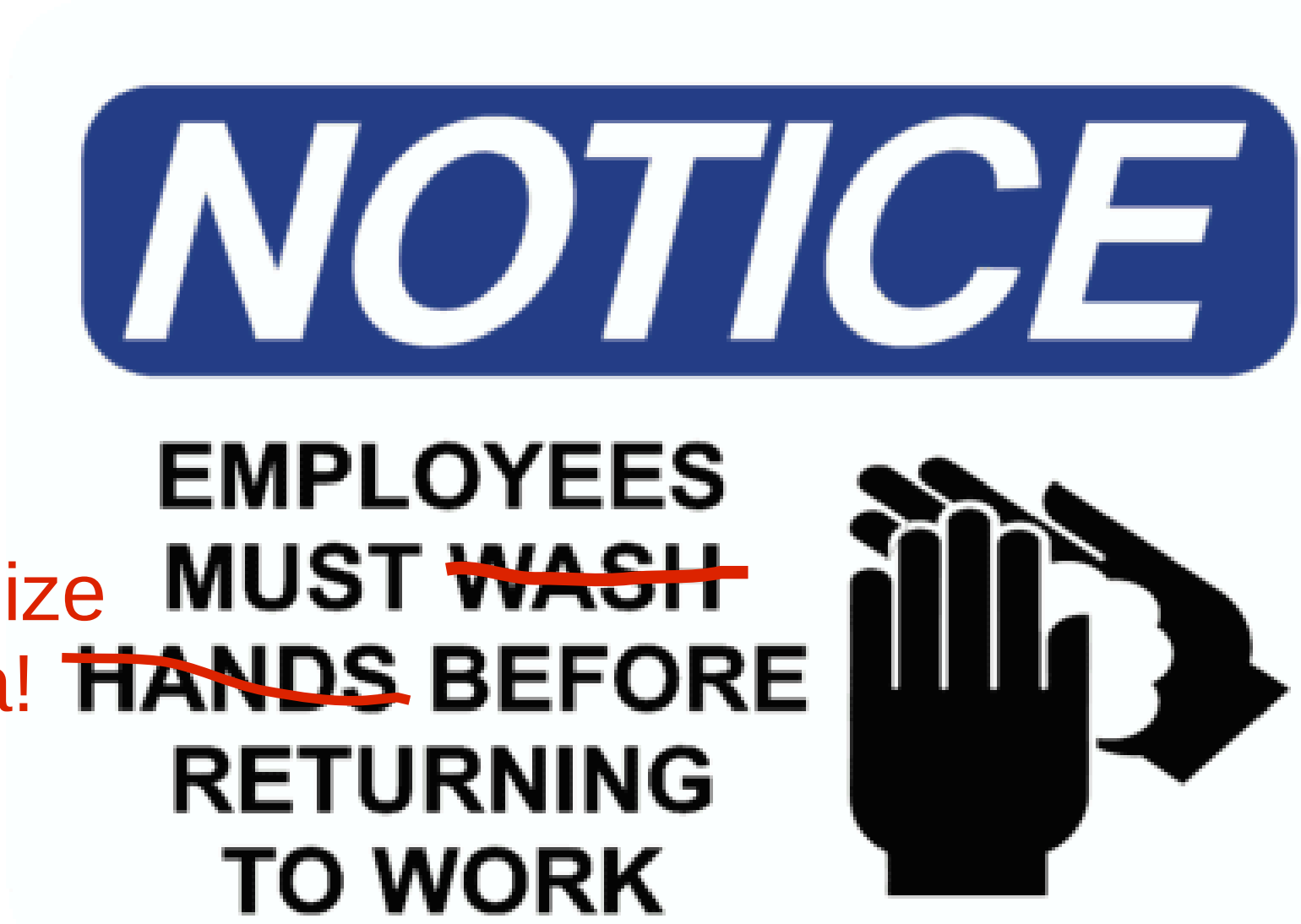
Department of Comuter Science
University of Illinois
Urbana-Champaign

L. V. Kale, O. Lawlor (UAF),
F. Gioachin

Importance of Computer Graphics

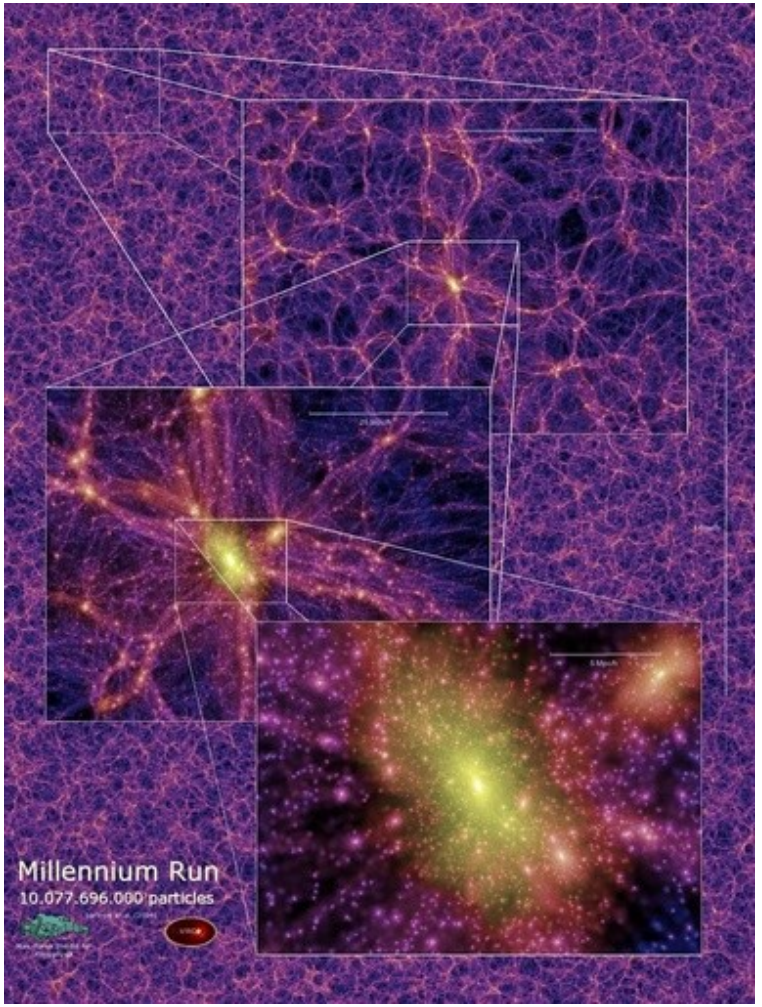
- “The purpose of computing is insight, not numbers!”
  R. Hamming
- Vision is a key tool for analyzing and understanding the world
- Your eyes are your brain’s highest bandwidth input device
  - Vision: >300MB/s
    - 1600x1200 24-bit 60Hz
  - Sound: <1 MB/s
    - 96KHz 24-bit stereo
  - Touch: <100 per second
  - Smell/taste: <10 per second

Visualize data!



Large Scale Structure: Current Simulation

- “Fair sample” (700 Mpc) of Universe with 10 billion particles
- 1000 particles/galaxy
- 1 Teraflop-week to complete
- .3 TB snapshots



Halo Simulations: What's needed

Ghalo simulation, Stadel et al

- Billions of particles in a single halo
- Dark Matter detection experiments
- Influence on disk
- Theories of gravitational collapse (Insight!)

GPU Rendering Drawbacks

- Graphics cards are fast
  - But not at rendering lots of tiny geometry:
    - 1M primitives/frame OK
    - 1G pixels/frame OK
    - 1G primitives/frame not OK
- Problems with billions of primitives do not utilize current graphics hardware well
- Graphics cards only have a few gigabytes of RAM (vs. parallel machine, with terabytes of RAM)

Parallel Rendering Advantages

- Multiple processors can render geometry simultaneously

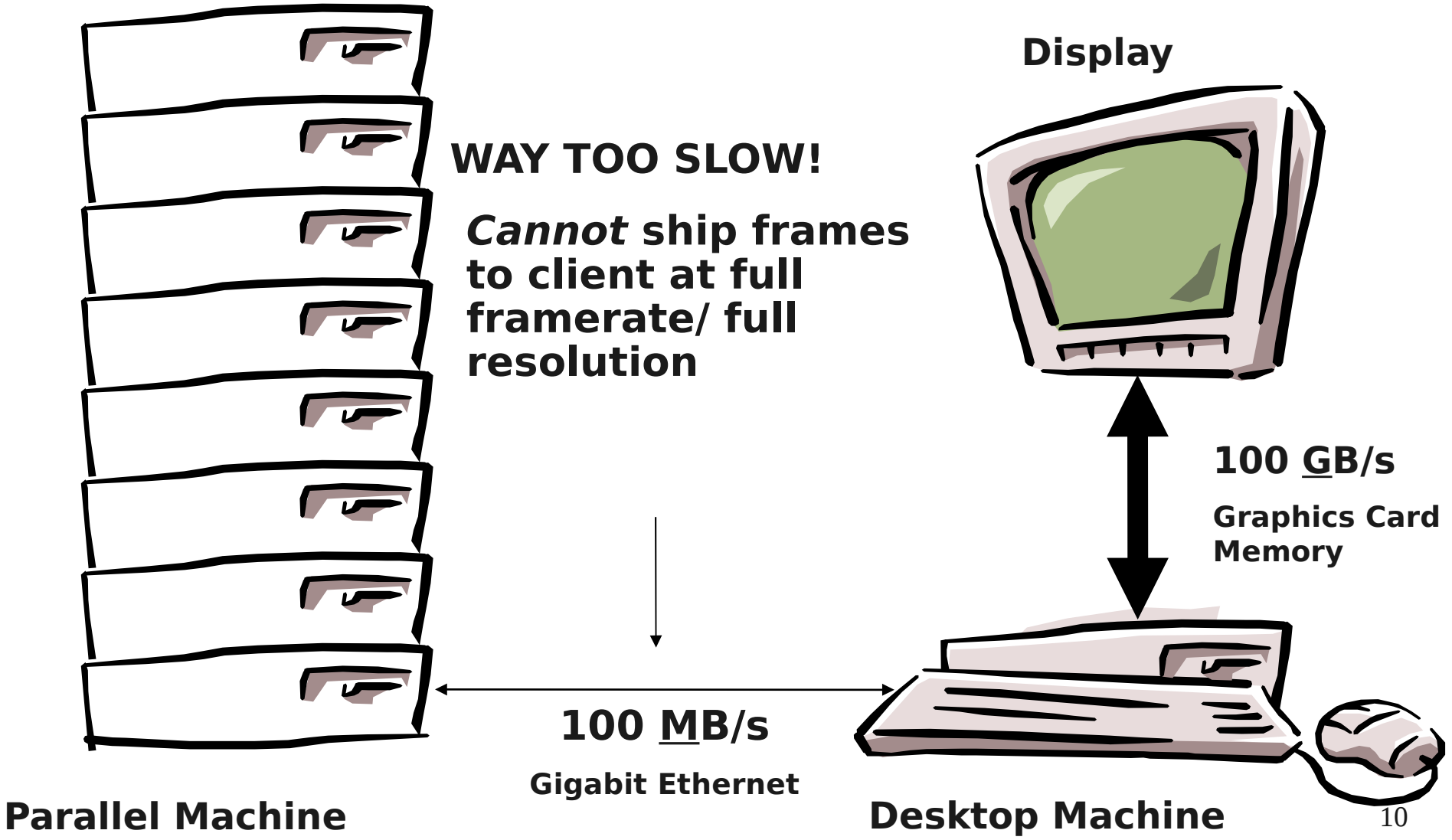
Processors	4	8	16	24	32	48
MParticles/second	7.14	15.71	32.71	49.18	65.49	81.68

48 nodes of Hal cluster: 2-way 550MHz Pentium III nodes connected with fast ethernet

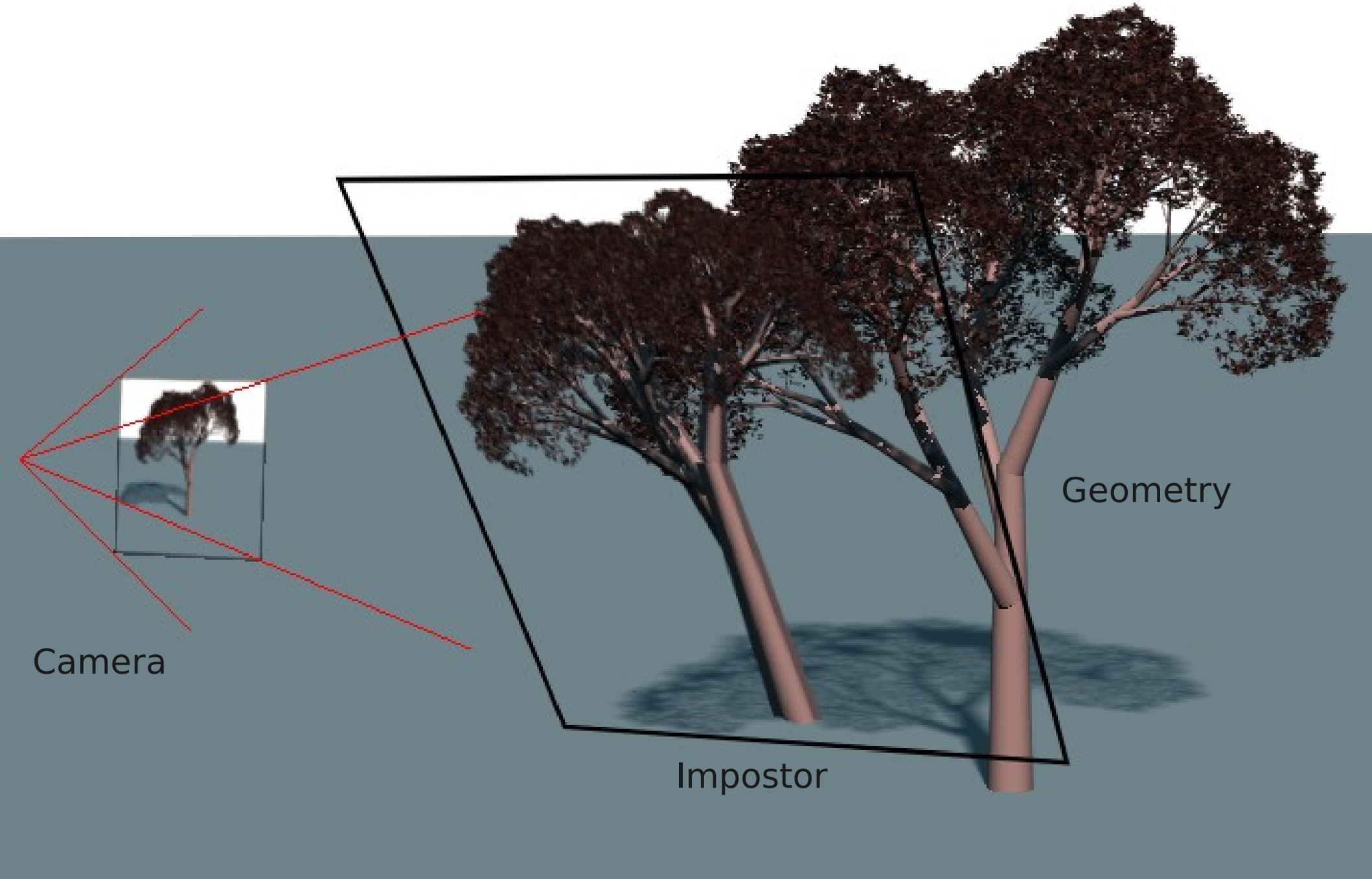
- **Achieved rendering speedup for large particle dataset**
- **Can store huge datasets in memory**
- **BUT: No display on parallel machine!**
- **Ignores cost of shipping images to client**

Parallel Rendering Disadvantage

- Link to client is too slow!



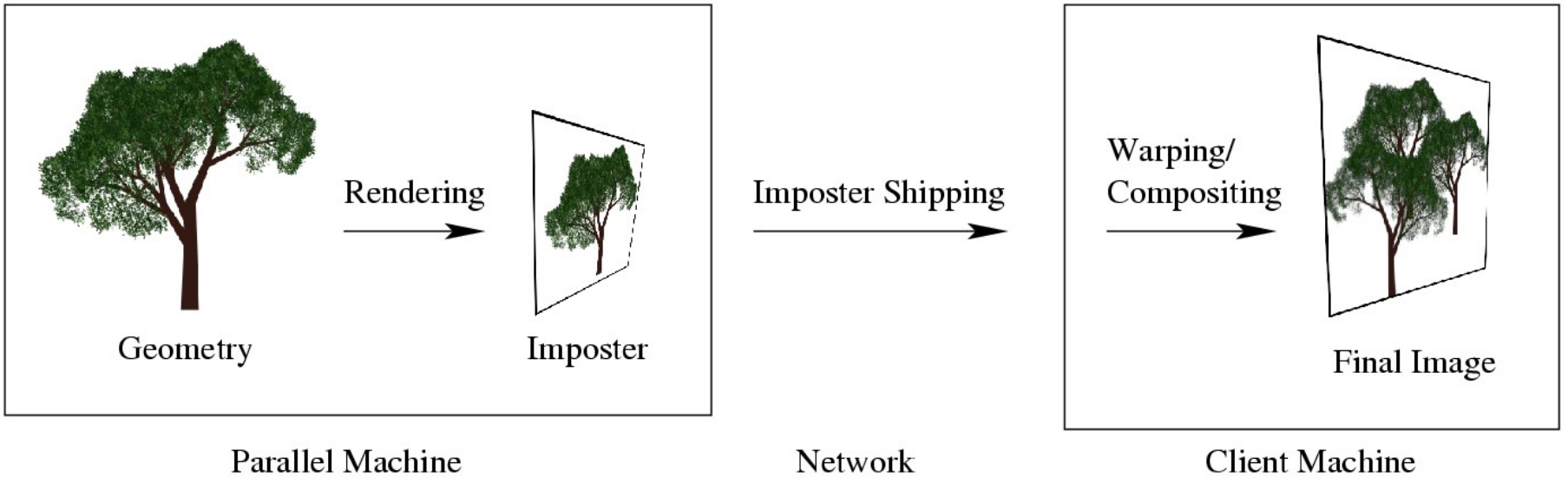
Impostors : Basic Idea



Parallel Impostors Technique

- Key observation: impostor images don’t depend on one another
- So render impostors in parallel!
  - Uses the speed and memory of the parallel machine
    - Fine grained-- lots of potential parallelism
  - Geometry is partitioned by impostors
    - No “shared model” assumption
- Reassemble world on serial client
  - Uses rendering bandwidth of client graphics card
  - Impostor reuse cuts required network bandwidth to client
    - Only update images when necessary
  - Impostors provide latency tolerance

Client/Server Architecture



- **Parallel machine can be anywhere on network**
  - **Keeps the problem geometry**
  - **Renders and ships new impostors as needed**
- **Impostors shipped using TCP/IP sockets**
  - **CCS & PUP protocol [Jyothi and Lawlor 04]**
  - **Works over NAT/firewalled networks**
- **Client sits on user's desk**
  - **Sends server new viewpoints**
  - **Receives and displays new impostors**

Salsa: an interactive visualization/analysis tool

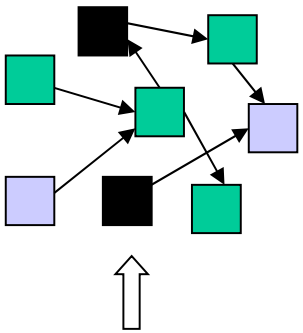
- Analysis/Visualizes particle datasets
- Parallel implementation in Charm++ language
- Interactive injection of analysis code into parallel program
- Interactive visualization using Java/JOGL client
- Map/Reduce features

Charm++: Migratable Objects

Programmer: **[Over] decomposition into virtual processors**

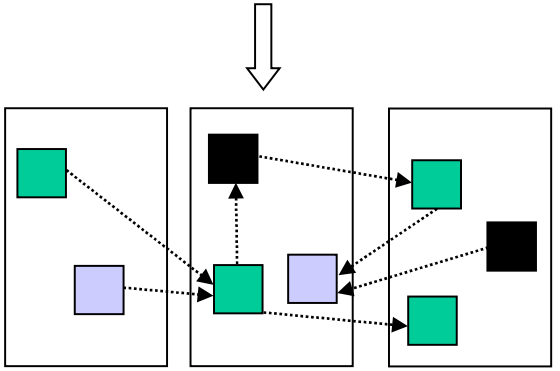
Runtime: **Assigns VPs to processors**

Enables *adaptive runtime strategies*



User View

System implementation

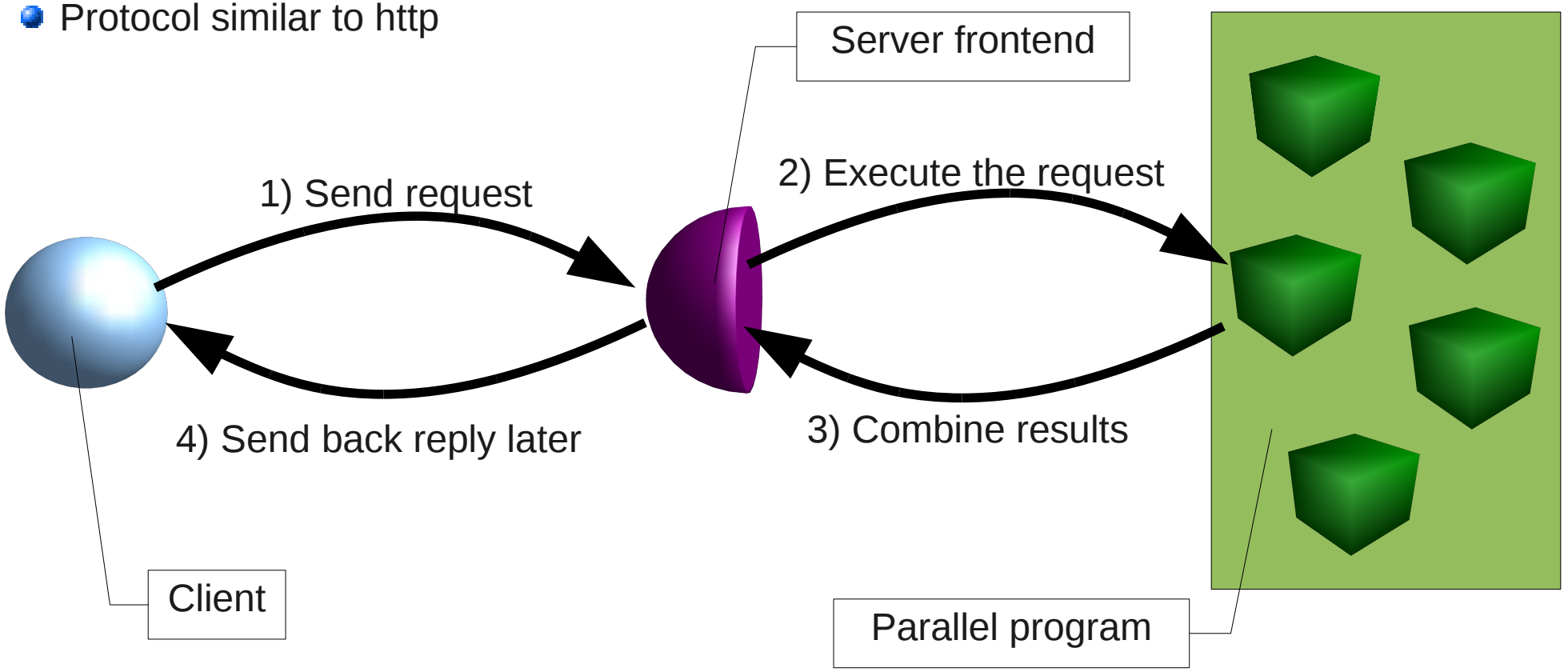


Benefits

- Software engineering
  - Number of virtual processors can be independently controlled
  - Separate VPs for different modules
- Message driven execution
  - Adaptive overlap of communication
- Dynamic mapping
  - Heterogeneous clusters
    - Vacate, adjust to speed, share
  - Automatic checkpointing
  - Change set of processors used
  - Automatic dynamic load balancing
  - Communication optimization

CCS - Converse Client-Server Protocol

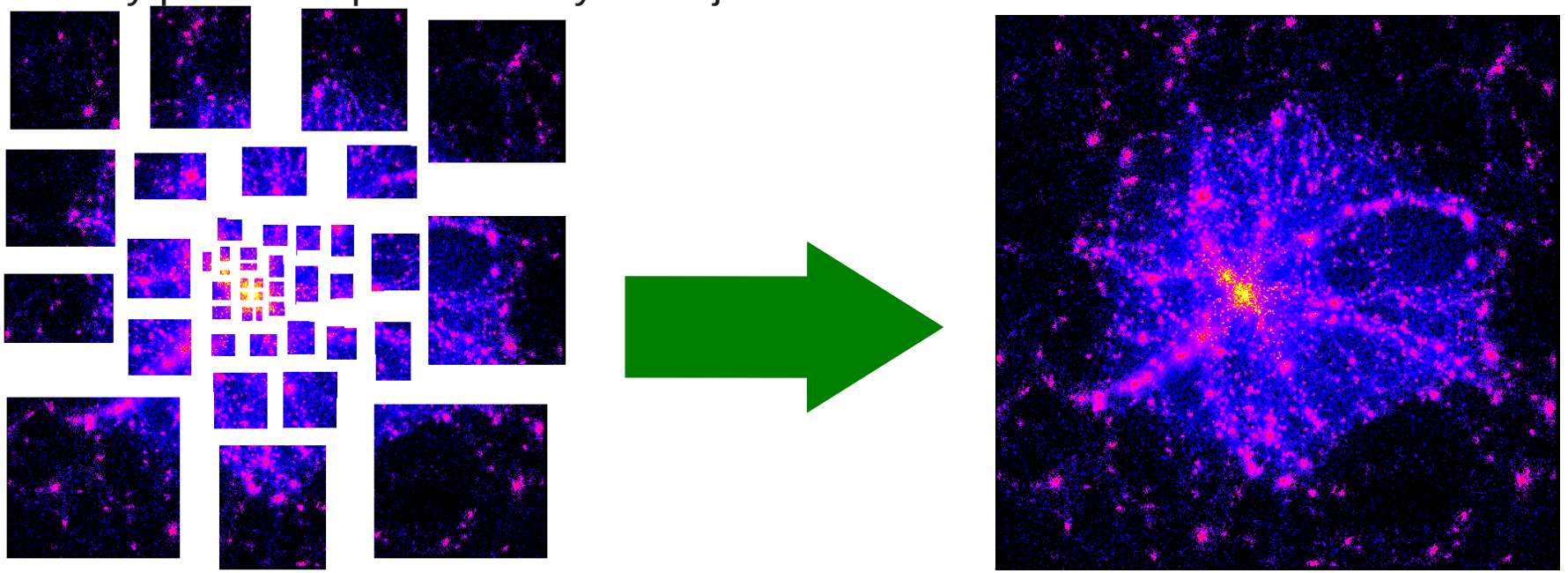
- Protocol similar to http



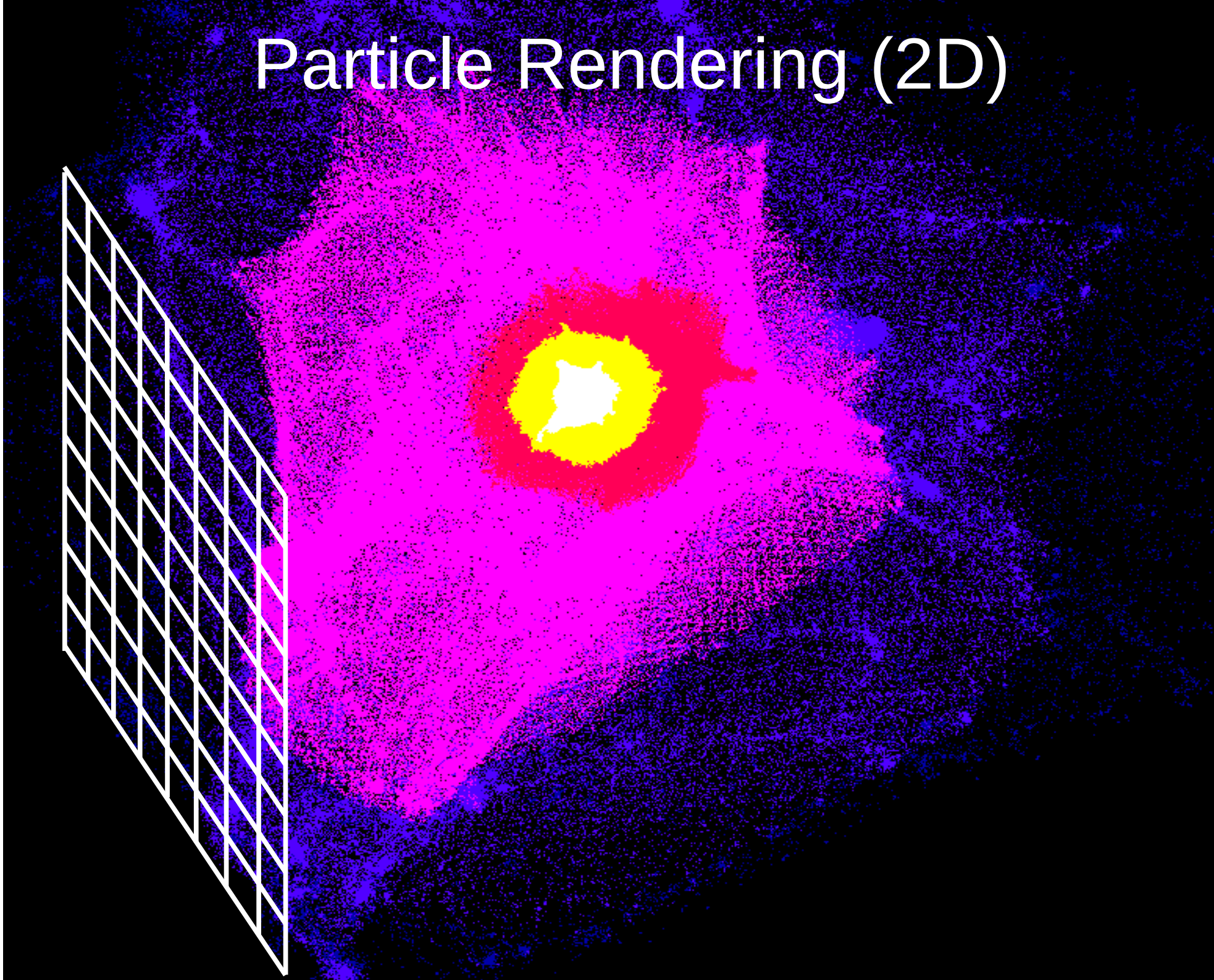
LiveViz

- Uses CCS functionality
- Upon request, every object creates a piece of image
- The image is combined and sent back to the client
- Scales well with number of processors

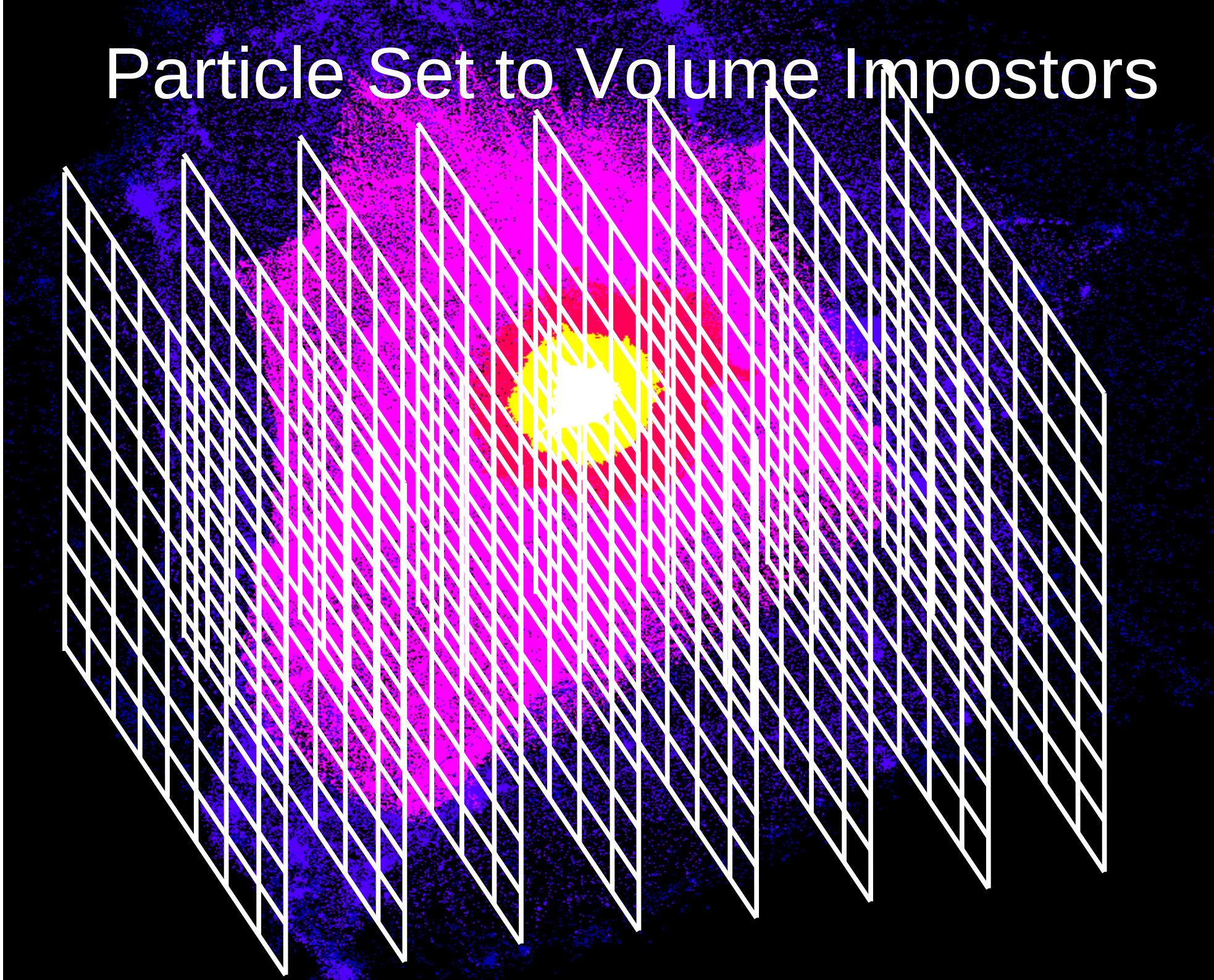
every piece is represented by an object



Particle Rendering (2D)



Particle Set to Volume Impostors



Volume Impostors Technique

- 2D impostors are flat, and can't rotate
- 3D voxel dataset can be rendered from any viewpoint on the client
- Practical problem:
  - Render voxels into a 2D image on the client by drawing slices with OpenGL
  - Store maximum across all slices: glBlendEquation(GL_MAX);
  - To look up (rendered) maximum in color table, render slices to texture and run a programmable shader

Status

- Applicable to large range of astrophysical data
  - SDSS – 40M particle 3D catalog of MW stars
  - LSST – 20B object catalog
- Usable on nVida 7XXX or better GPUs
- Available soon: `hpcc.astro.washington.edu`
  - Also see AISR software library
- Investigating better compression (not JPEG) and performance (GPGPUs) to reduce latency