Large Scale Data Visualization

Large Datasets

- Large datasets: D >> 10 M_D
  - D: Hundreds of gigabytes to terabytes and even petabytes
  - M_D: 1 to 4 GB of RAM

- Examples:
  - Single large data set
  - Time-varying data set
  - Multiple data sets

- Interactivity is important
  - 20 – 30 Hz >> animation playback
  - Min 10 Hz >> update rate of hand-eye interaction
  - The slowest is processing user request – 10 sec is the target for the interactive system >> with large datasets it can reach up to minutes

Techniques for Large Scale Datasets

- Data streaming
  - Processing subset of the larger dataset
- Task parallelism
  - Independent modules execute in parallel, user needs to identify the number of independent tasks
- Pipeline parallelism
  - Modules execute in parallel but on independent subsets of data
- Data parallelism
  - Code within each module of the application execute in parallel
- Hybrid methods:
  - Combination of the above techniques can be used

Data Streaming

- Sometimes this is the only approach when data exceeds the available computational resources

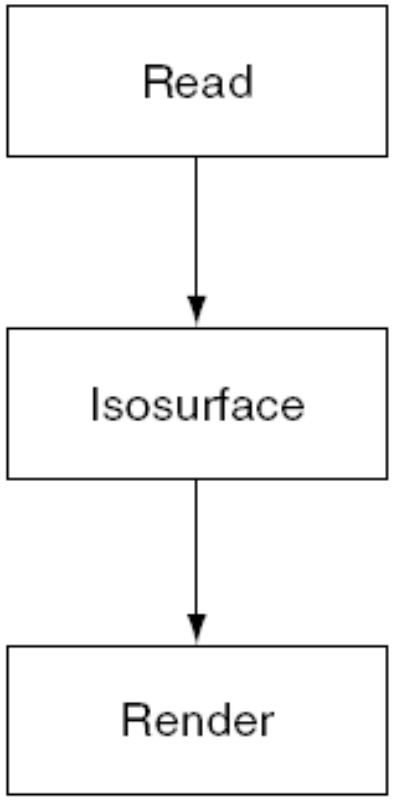


Figure 27.2 An example module network for isosurface computation and rendering.

Task Parallelism

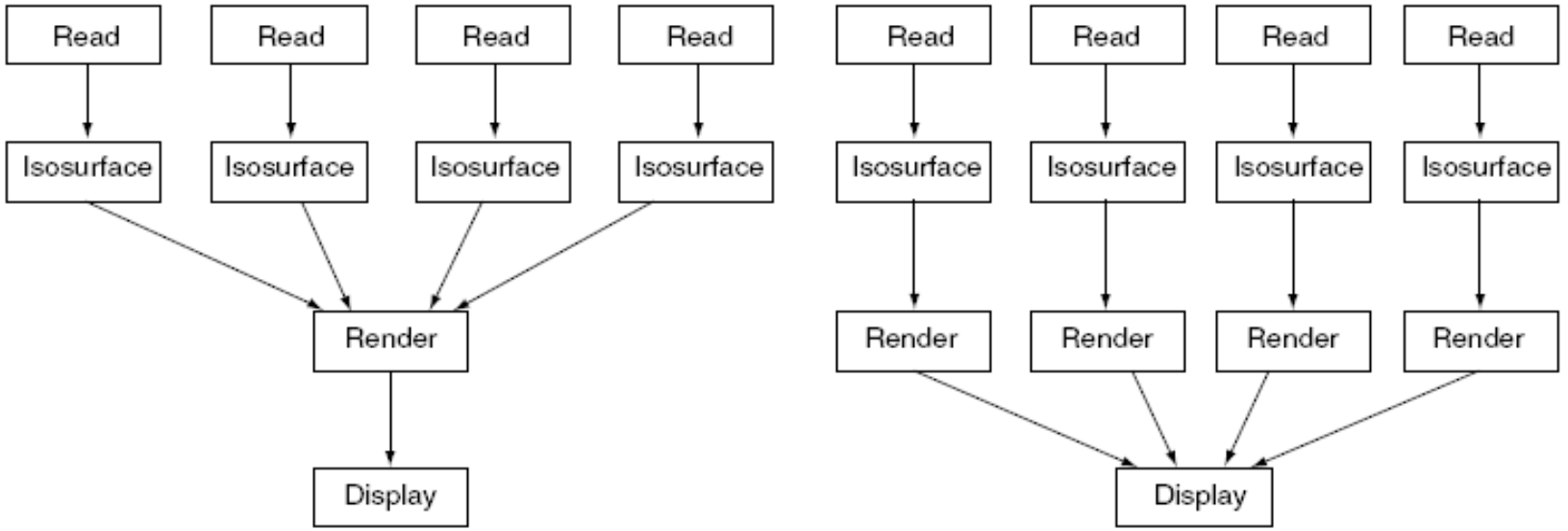


Figure 27.5 The two networks used in the task parallelism examples. The key difference between the two is the delivery of data to the display task. The display task on the left receives geometry, while the one on the right receives image data.

Pipeline Parallelism

- Task A, D and E are all operating on different portions of data
- This approach is best suited for situations where there are multiple, heterogeneous tasks
- The advantage of this approach is that it allows parallel use of the overall computing resources

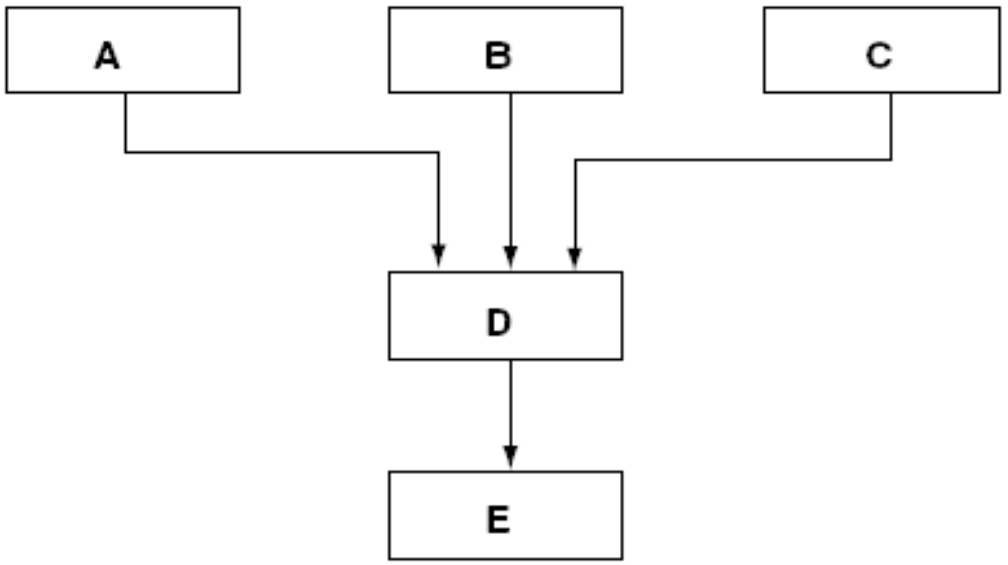


Figure 27.1 This figure shows a module network consisting of five modules (A, B, C, D, and E) and arrows depicting the data connections between modules.

Solution Techniques

Dataset Size	*Tasks*	*Resources*	*Solution*
Large	Any	Single CPU	Data streaming
Large	Homogenous	Multiple CPUs	Data parallelism
Any	Independent	Multiple CPUs	Task parallelism
Any	Sequential	Multiple CPUs	Pipeline parallelism

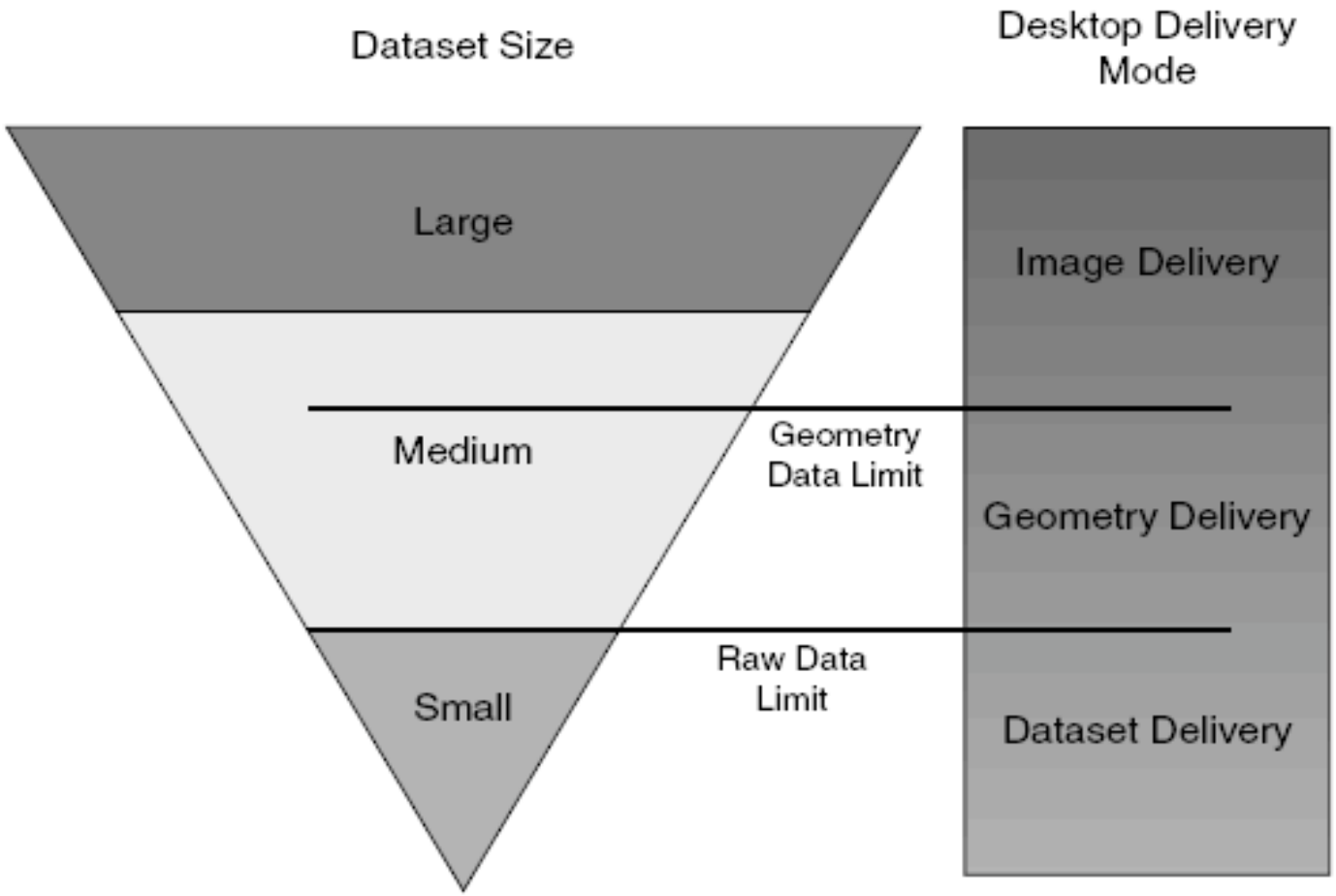
Figure 27.11 Summary table of solution techniques.

Desktop Delivery System

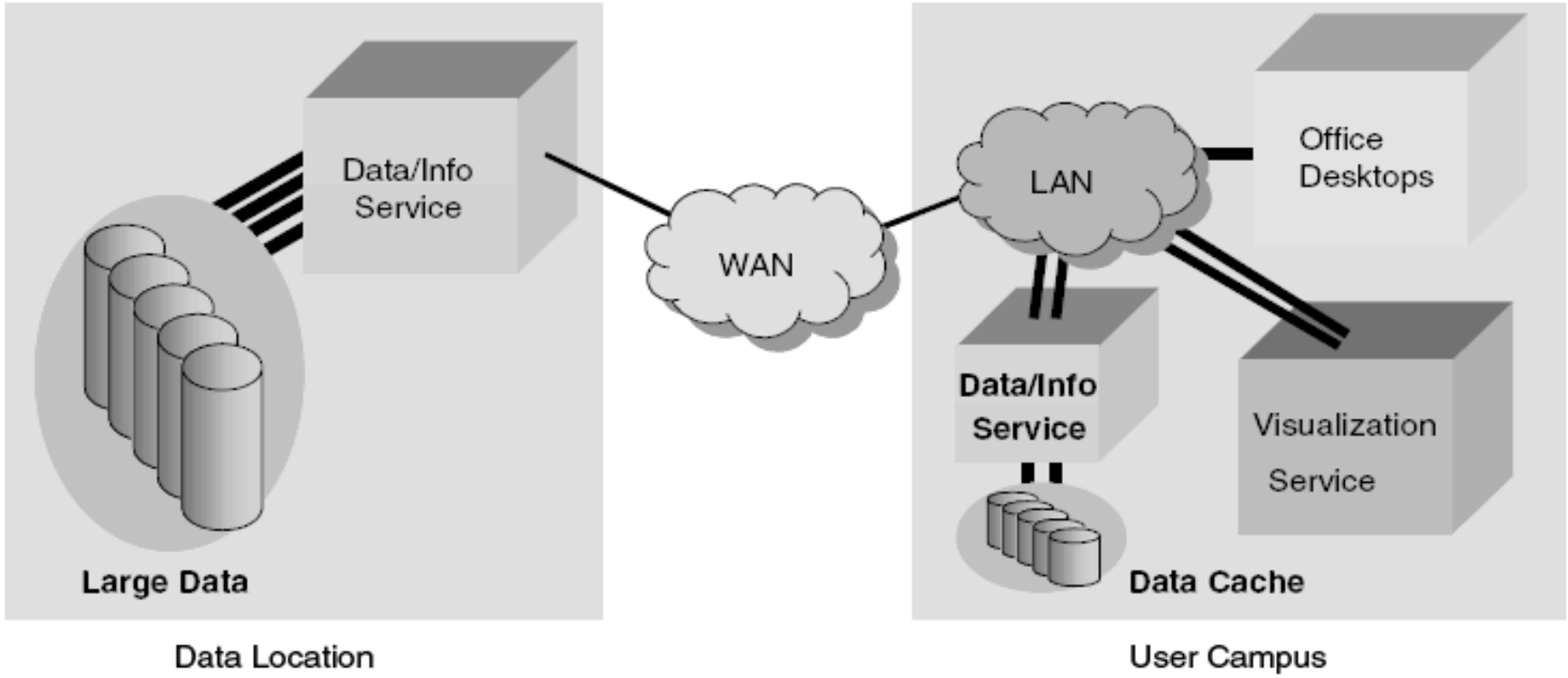
- How to enable the visualization of large scale datasets on the desktop?
  - Desktop delivery is one solution (bringing information to the user)

- Issues
  - Size of the data
  - Desktop capabilities
  - Level of interaction desired
  - Network issues
  - Computing power needed

Effect of Data Size on DDS

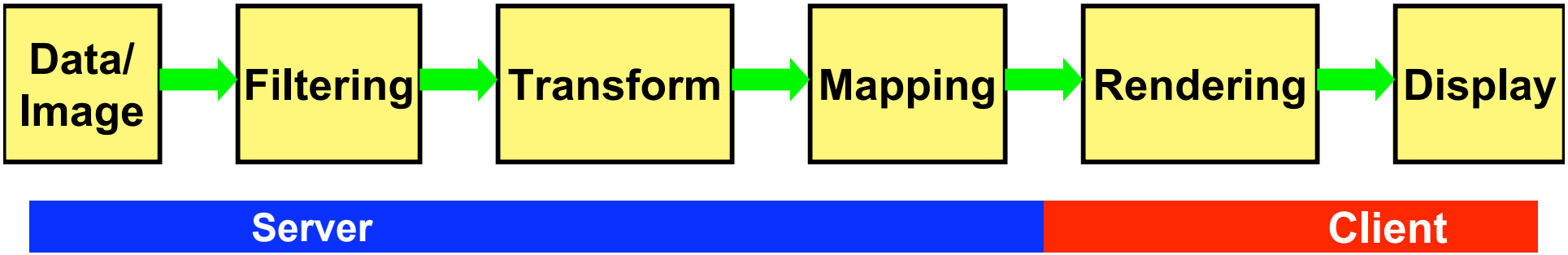


Visualization for DDS



Remote Visualization

- A schematic view of visualization pipeline

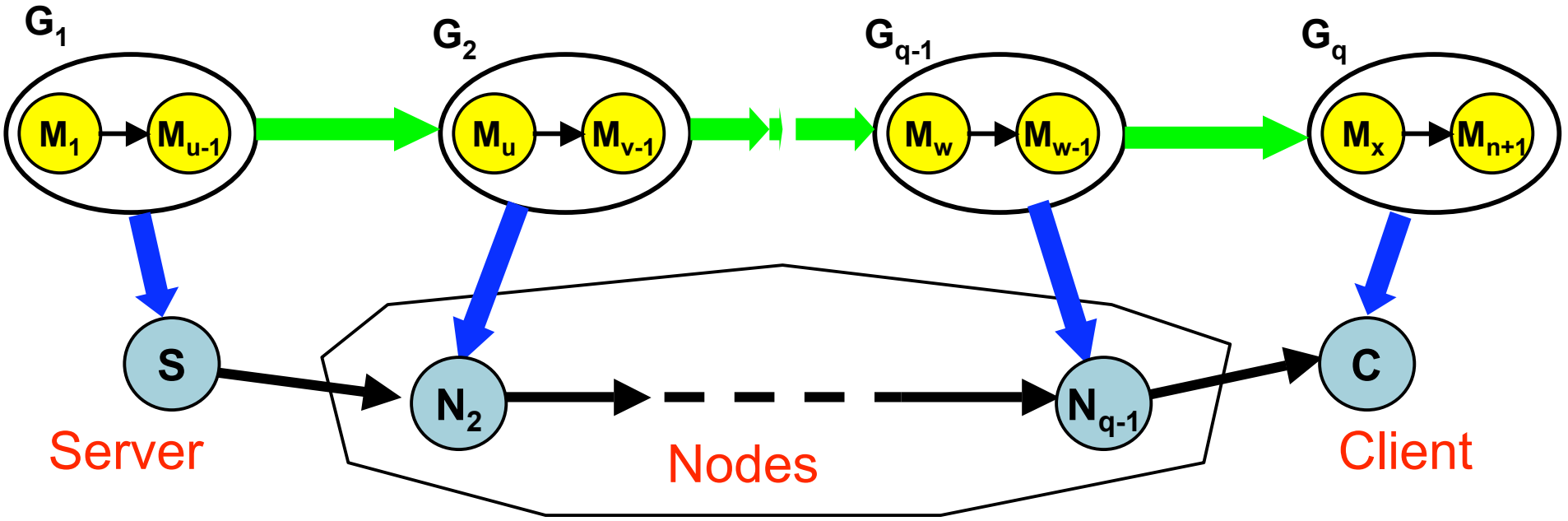


- Fixed type of data exchange mechanism
  - Fat client or fat server model

- Dynamic client-server paradigm:
  - Choose the data types (e.g, geometric primitives, framebuffer or final images) to be sent to the client.

Adaptive Network Mapping

- Optimal visualization pipeline decomposition and adaptive network mapping



- Network conditions: Bandwidth and node characteristics
- Mapping: Organize pipeline modules into groups and dynamically assign them to network nodes

Amira Web-Based Services

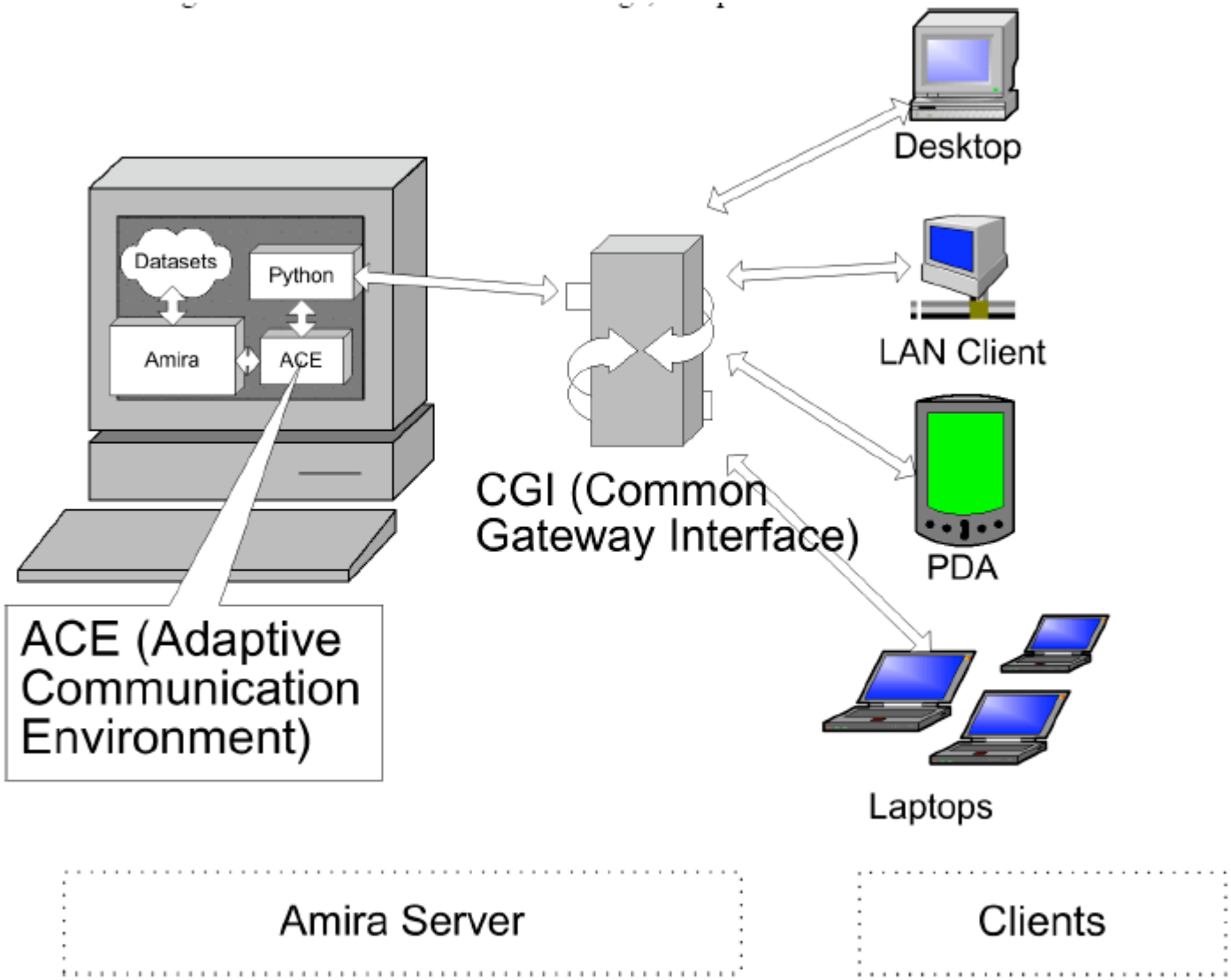


Figure 2 Schematic diagram of the "client – Amira server" setup used for WEB-IS2

HTML Interface

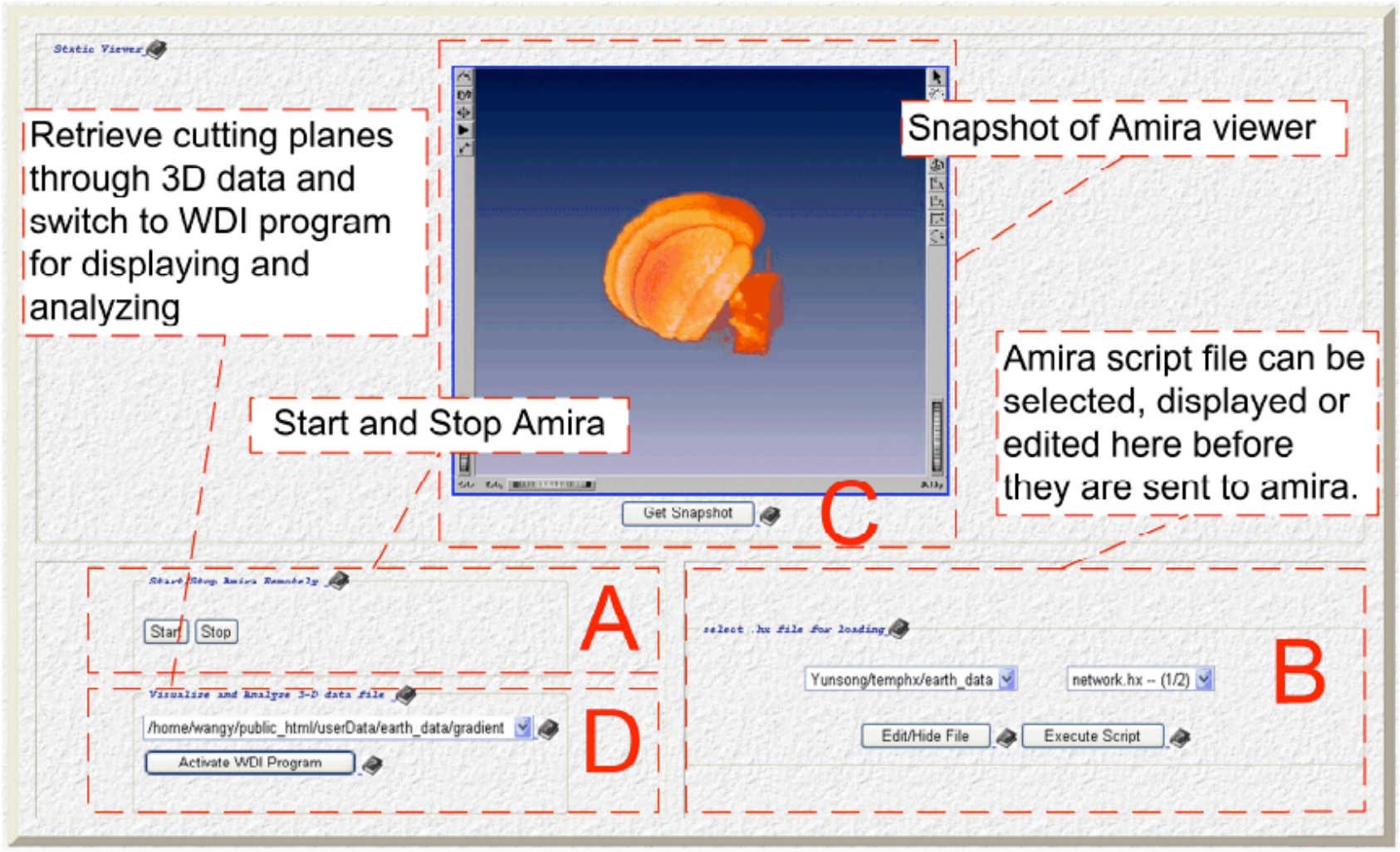
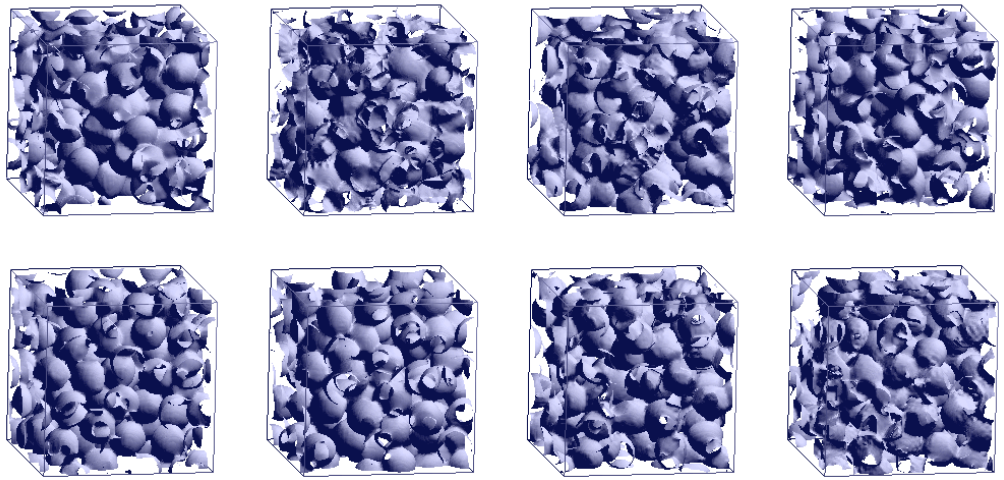


Figure 1 Web-based service for Amira allows user to analyze, render and view (using amira) the large datasets over the internet from a web browser.

Time Varying Data

- Examples:
  - CFD, MD, Neuron excitement, Evolution of a thunderstorm, Seismic reflection from geological strata
- Generally, multiple values are stored at each data point
  - A single dataset can require hundreds of gigabytes to terabytes of storage space
- Rendering of time-varying data:
  - Reading of large files continuously or periodically throughout the course of the visualization process
- Improvements
  - Encoding the data, Hardware decoding of data, Modern GPU, Parallelization of process, Image compression

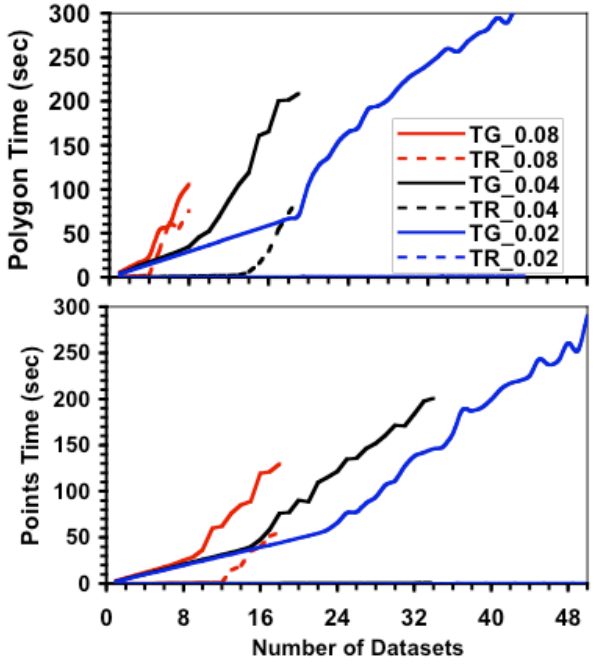
Multiple Datasets Visualization (MDV)

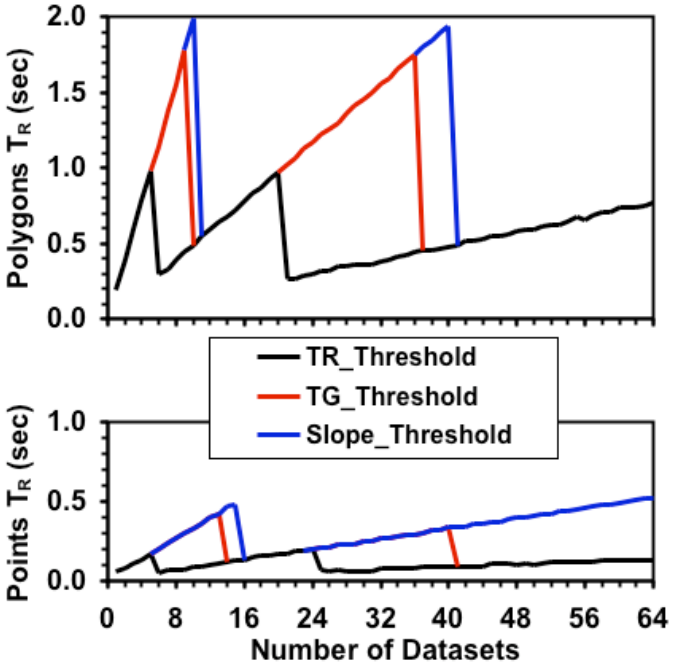


MDV represents simultaneous visualization and analysis of multiple sets of data

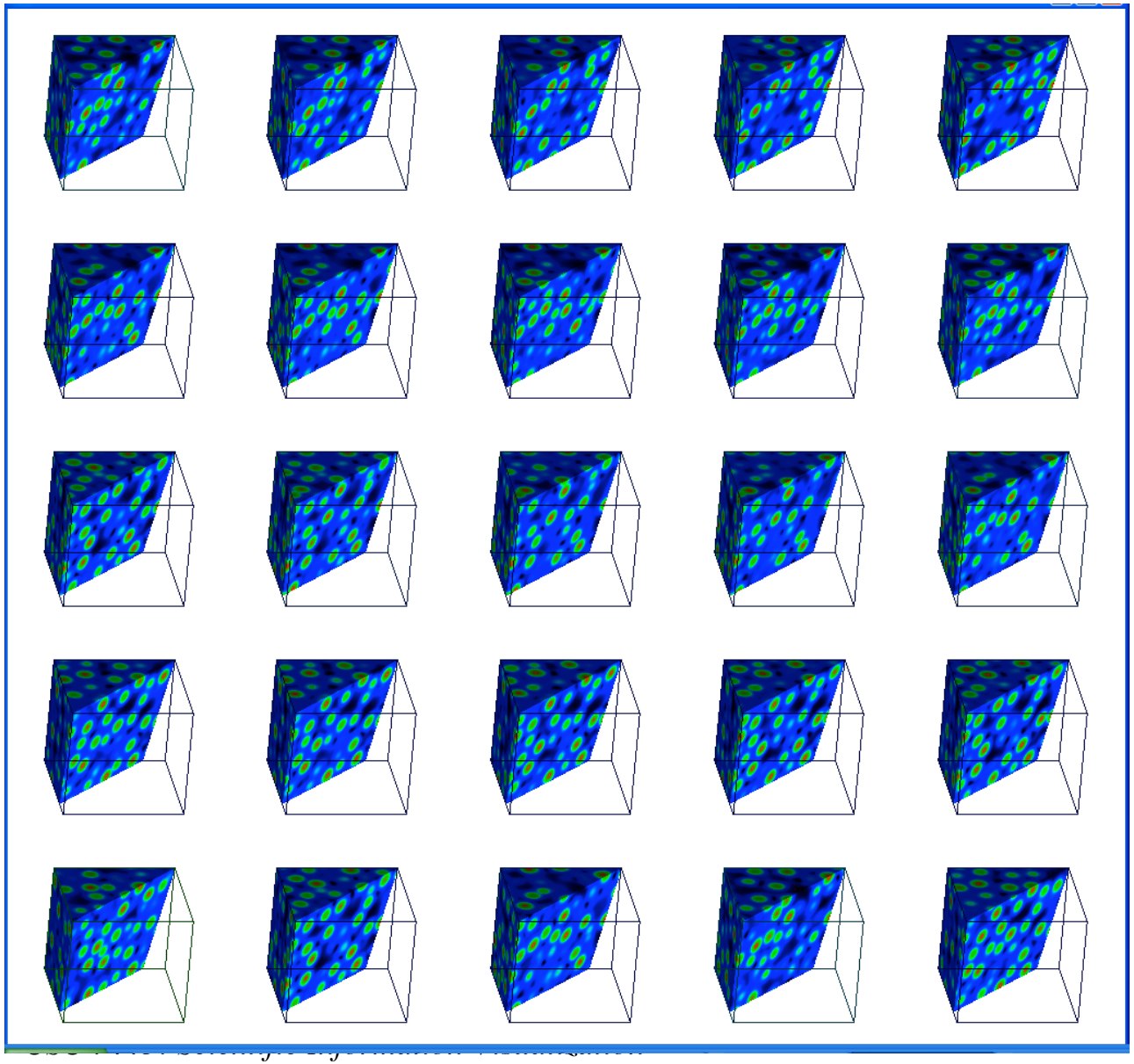
Scalable adaptive isosurface extraction (AIM and OPIM)

64 sets of scalar volume data with size of 256^3 and 512^3





MDV Example



Twenty five sets of the scalar volume data of 256^3 size in a planer clipped mode using 3D surface texture mapping

The data represent the electron density distributions in liquid MgO calculated as a function of the simulation time

Multi-scale color map:

Blue: density from 0 to 0.05

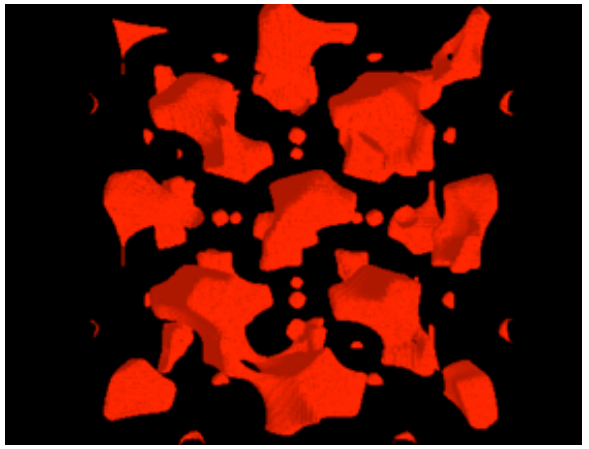
Blue and green: density from 0.05 to 0.5

Red: density above 0.5

B. B. Karki, LSU

MDV: Component-Based Isosurface Extraction

- Components are disjoint geometric parts of isosurfaces
- Process a subset of isosurface components selected based on inter-dataset coherency
  - User-defined thresholds
  - Only dissimilar components from different datasets processed completely
- Two advantages
  - Crack-free isosurfaces
  - Effective in identifying interesting structural differences and suppressing the noises



Isosurface comprised of many spatially disjoint components; Different components correspond to electron distributions around different atomic sites.

Data Coherency Approach

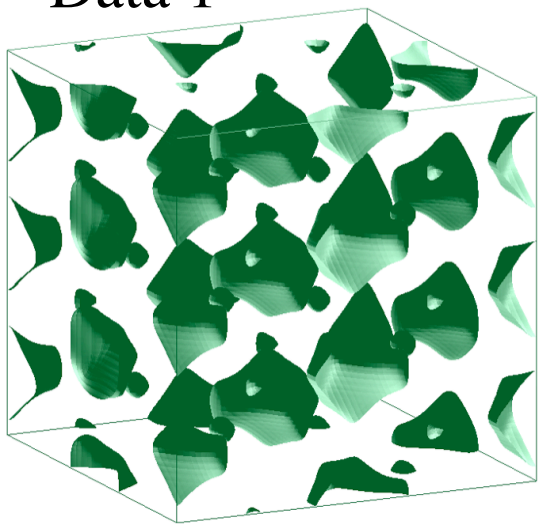
- Data coherency is a measure of similarity between datasets
- Exploit data coherency at the level of individual voxels and components
- Divide datasets into two categories:

  Reference dataset (RDS) and Nonreference dataset (NRDS)
  - Considering two datasets: one RDS and NRDS
  - RDS isosurface extraction is performed and the polygons are used subsequently to also represent parts of the NRDS isosurfaces
  - For NRDS, isosurface extraction is performed only for those voxels which significantly differ from the corresponding RDS voxels; the polygons in other voxels are simply retrieved from RDS

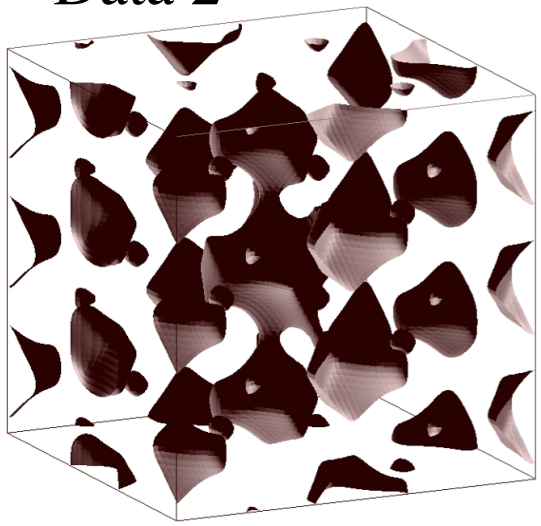
Inter-Dataset Coherency Method

- Compare each data volume with reference volume(s) at some octree level
- Only dissimilar octree nodes are processed for polygon generation
  - Polygon data are stored on per node basis.

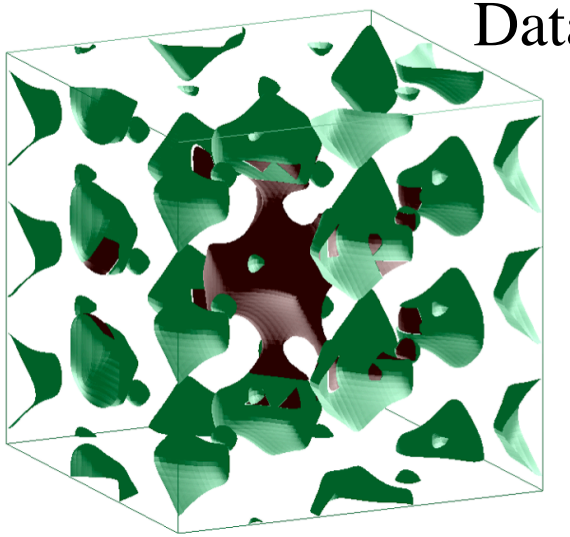
Data 1

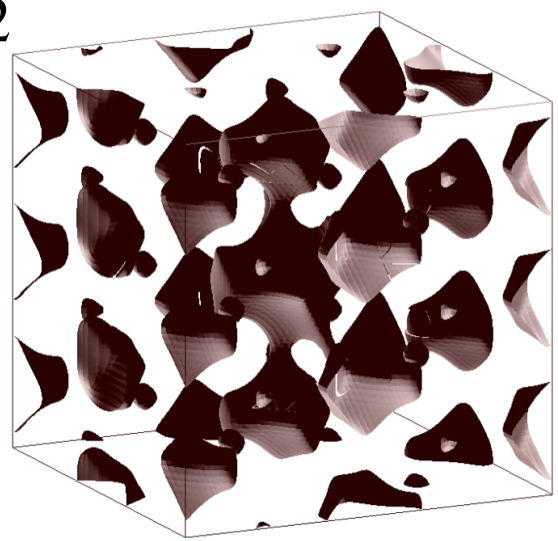


Data 2



Data 2





Processing RDS

- Completely process RDS for isosurface extraction
  - Polygons are subsequently used to also represent parts of NRDS isosurfaces
- Find the components and extract the corresponding polygons
- Construct a map representing the relationship between voxels and the components: $RDS.map[v_i]$

Processing NRDS

- Compute voxel difference: *vox.diff*
- Pick up unprocessed significant voxel and find its component:

  $vox.diff > threshold1$

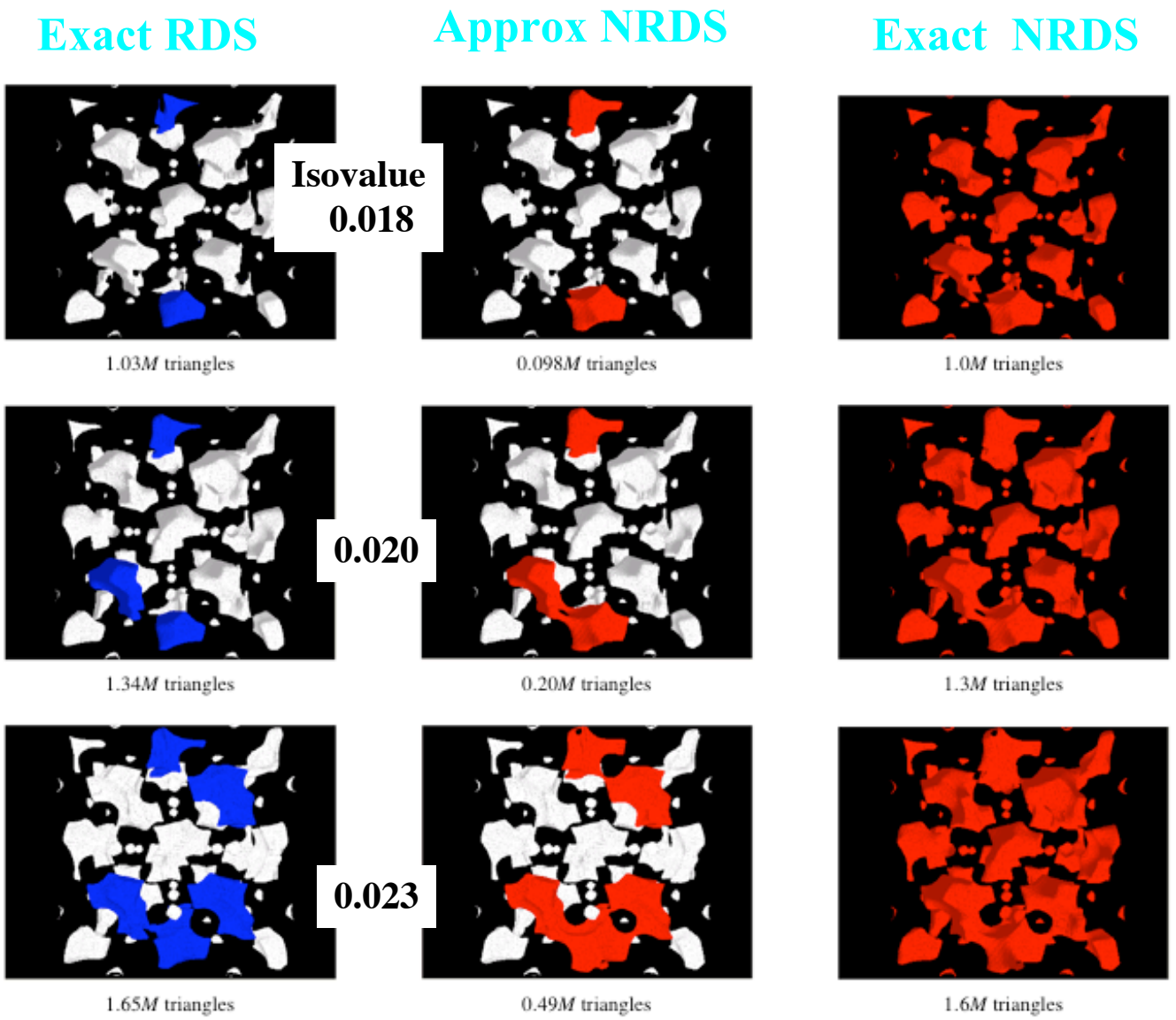
- Extract polygons of the component if it contains sufficiently many significant voxels (*nSV*):

  $nSV > threshold2 \times nV$

  where *nV* is the number of voxels in a given component
- Retrieve the polygons of the component from RDS if no new polygons are extracted in corresponding regions of NRDS
  - Overlap between RDS and NRDS is large (as determined by *threshold3*)
- Repeat until all significant voxels have been visited

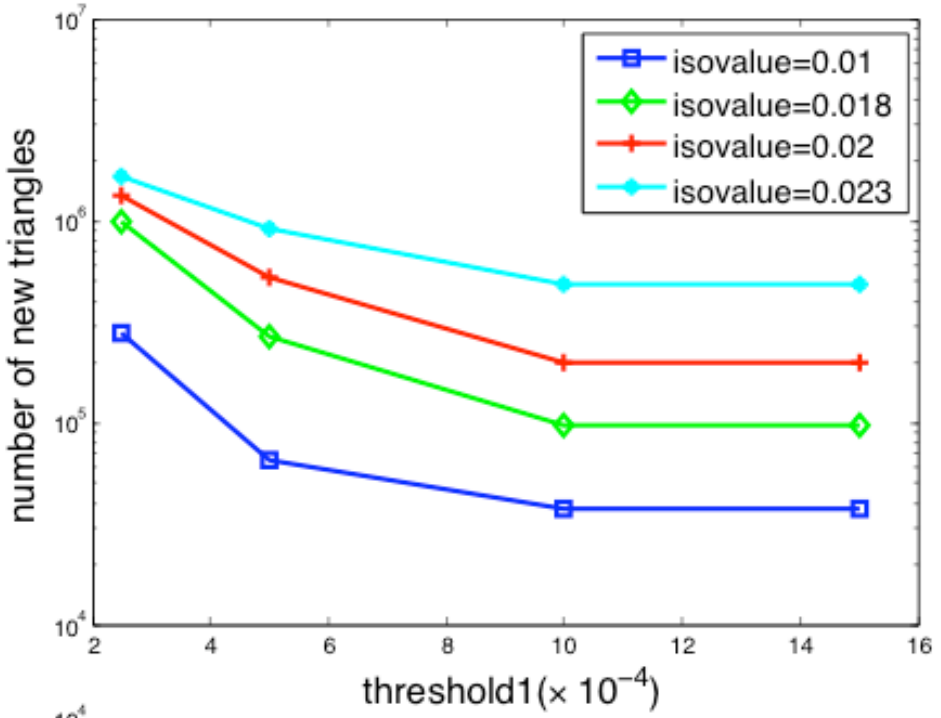
RDS and NRDS Isosurfaces

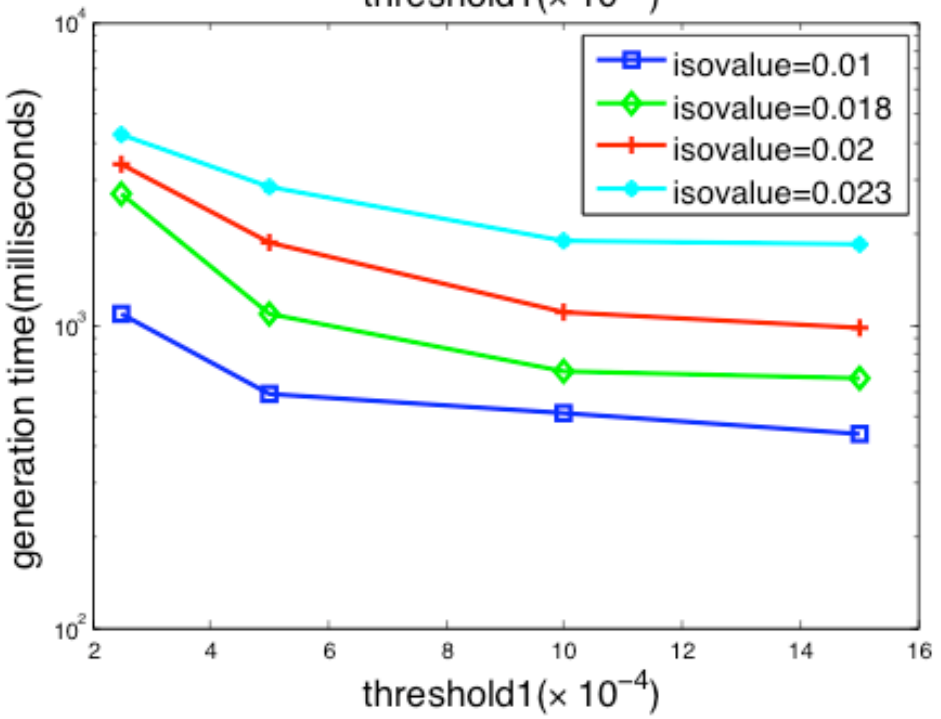
- NRDS isosurfaces approximated using our approach
  - Electron density isosurfaces
- Blue components belong to RDS
- Red components are directly extracted from NRDS and white ones common to both are extracted from RDS
- Some disjoint (blue) components merge to new (red) component in NRDS



Performance Results

- As *threshold1* with *threshold2* and *threshold3* fixed at 0.25 decreases, more voxels in NRDS turn into significant voxels
- If we increase *threshold2*, some components are discarded so less overhead in polygon extraction
- *Threshold3* involves only retrieval of polygon data for rendering
- Electron density isosurfaces vary with isovalue





On the Fly Isosurface Extraction

- Do not assume data coherency
- Use simple shapes (bounding boxes) to represent components from both datasets
- Polygons are extracted once users select regions of interest
- Isosurfaces extracted in large yellow box in the bottom left of the top right window