Distributed Multi-Level Out-of-Core Volume Rendering

Johanna Beyer[1], Markus Hadwiger[1], Won-Ki Jeong[2], Hanspeter Pfister[2]

[1]Geometric Modeling and Scientific Visualization Center, King Abdullah University of Science and Technology

[2]School of Engineering and Applied Sciences, Harvard University

Introduction

In neuroscience, a very promising bottom-up approach to understanding how the brain works is built on acquiring and analyzing high-resolution scans of brain tissue using electron microscopy (EM). This results in volume data of nanometer per pixel resolution and data sizes of many terabytes [1]. To support the work of neurobiologists, interactive exploration of such volumes requires new approaches for distributed out-of-core volume rendering, since these data are much larger than those in current systems [2, 3, 4].

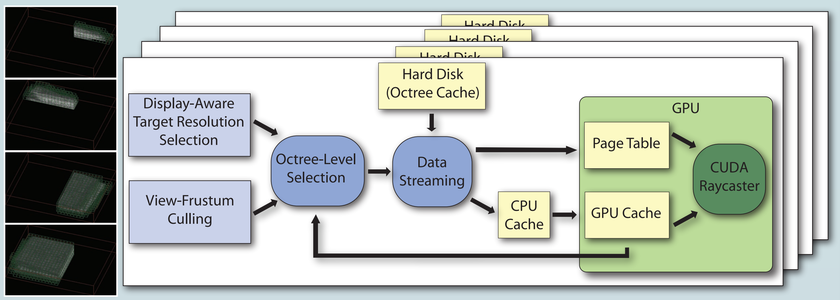
A major goal of our distributed GPU volume rendering system is to try to sustain a screen pixel-to-voxel ratio of about 1:1, which achieves "full" rendering resolution with respect to a given display resolution. This display-aware approach effectively bounds the working set size required for ray-casting. The total working set memory can be increased by using multiple GPU nodes, each rendering a partial volume.

Currently, we achieve largely interactive volume rendering of a 43 GB and a 92 GB EM volume on 1 to 8 Tesla nodes. More nodes increase the working set and thus allow full resolution rendering. Fewer (or one) nodes can still render the entire volume, but with reduced resolution. The required target resolution is directly determined by the output display and is independent of the volume resolution.

Display-Aware Multi-Resolution Volume Streaming and Ray-Casting

To achieve the display-aware ("full resolution") target resolution, volume data streaming and ray-casting in CUDA are guided by two main characteristics:

- Display-awareness selects a target resolution and the corresponding octree level.
  The size of the top-level page table (page directory) used for ray-casting is set to match this resolution.
- Each node performs view frustum culling of blocks required by the target resolution and checks whether all blocks fit into its physical texture cache. If not, a coarser octree level is used.

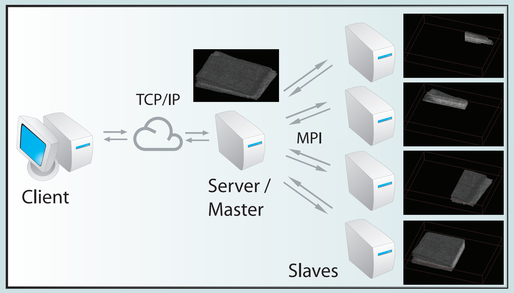


Each GPU node independently streams in voxel data on demand, guided by the render resolution. While streaming in higher-resolution data, lower-resolution data can be freely mixed in by the ray-caster. Given a large enough number of GPU nodes with a large enough total physical texture memory, the volume can always be rendered at "full" target resolution, i.e., with a roughly 1:1 pixel:voxel mapping.

We thank Jeff Lichtman, Clay Reid, the Harvard Center for Brain Science, Dave Luebke and NVIDIA, the Vienna Science and Technology Fund WWTF, Adi Kriegisch and the VRVis Research Center.

Distributed Volume Rendering

Neurobiologists use a thin client that handles the user interface and displays volume-rendered images that it receives from the server. The server is the head node of a GPU cluster, which performs the actual volume rendering in a distributed way. We assume a fast network connection within the cluster, but can handle a potentially slow connection between server and client, allowing neurobiologists to work remotely.

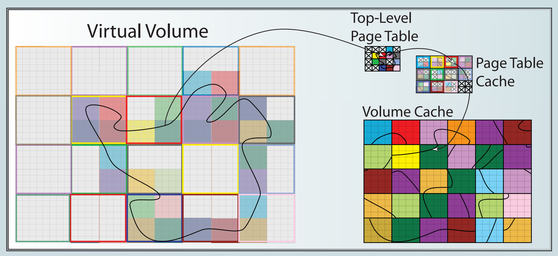


The server distributes rendering on the GPU cluster using MPI. However, MPI is not used to distribute any actual volume data. Each cluster node is only assigned a part of the volume, and is independently responsible for streaming and rendering the corresponding data using its own hierarchy of multiple cache levels. Volume rendering is performed by ray-casting using CUDA. After compositing the partial images produced by the cluster nodes, the final image is compressed and then transmitted to the client.

Virtual Volume Texturing with Multi-Level Paging

Each GPU node renders the volume assigned to it using virtual 3D texturing with single-pass ray-casting. Sample positions are computed in virtual volume space. Physical data voxels are fetched from one large 3D cache texture. Virtual to physical address translation is done using a two-level hierarchy of 3D page tables. Only the top-level *page directory* is fully resident in GPU memory. Each page directory entry refers to a block of 32^3 2nd-level page table entries. Only a subset of these blocks is resident in a page table cache. Each 2nd-level page table entry refers to a small block of 32^3 physical voxels. For any view, only a very small subset is required to be resident in the 3D volume cache texture for ray-casting.

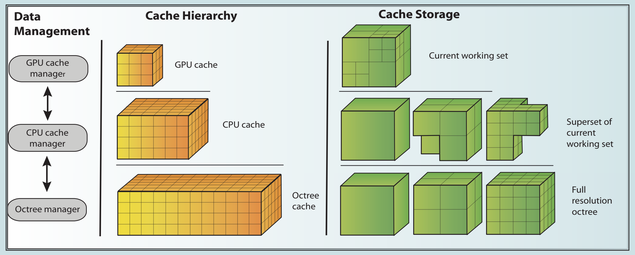
Using small blocks of 32^3 physical voxels achieves very good culling and update granularity, while multi-level paging enables very large volumes.



Our 18,000 x 18,000 x 304 EM volume (92 GB) uses a 32 x 32 x 1 page directory texture, a virtual page table of 563 x 563 x 10 entries stored in a page table cache texture of 64^3, and a physical 3D cache texture of $1{,}024^3$ voxels. This uses 1 GB texture memory / GPU node. Even a $128{,}000^3$ volume (2048 Teravoxels) would require only 128^3 texels in the page directory texture.

Multi-Level Out-Of-Core Memory Hierarchy

Each GPU node employs a multi-level hierarchy of caches and data streaming. Each node operates with minimum synchronization by the master node. For data streaming and ray-casting, this hierarchy is employed autonomously, which minimizes the communication overhead between nodes.



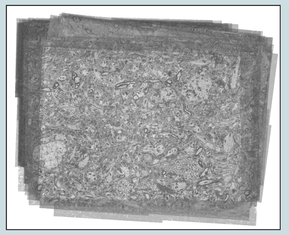
The list of volume blocks required for the current view by the ray-caster is determined via view frustum culling. All needed blocks are streamed into a single 3D cache texture in GPU RAM. However, a larger subset is kept in CPU RAM to minimize overall disk traffic when blocks are reused in a future frame.

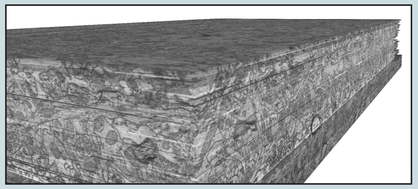
Results

We currently render two large EM volumes of a mouse hippocampus acquired slice by slice the Harvard Center for Brain Science, and pre-registered into aligned volume slice stacks:

- #1: 14,176 x 10,592 x 308 voxels (43 GB).
  pixel size 3-5 nm, slice thickness 29 nm.
- #2: 18,000 x 18,000 x 304 (92 GB),
  pixel size 3-5 nm, slice thickness 29 nm.

Dataset	Number of Nodes	Octree Level	Render time
#1 (43GB)	1	2	1.28 sec
	1	3	0.34 sec
	1	4	0.27 sec
	2	2	0.70 sec
	2	3	0.28 sec
	2	4	0.18 sec
	4	2	0.19 sec
	4	3	0.15 sec
	4	4	0.12 sec
	8	2	0.7 sec
	8	3	0.33 sec
	8	4	0.34 sec





References

[1] Jeong et al. SSECRETT and NeuroTrace: Interactive Visualization and Analysis Tools for Large-Scale Neuroscience Datasets. IEEE Computer Graphics and Applications, pages 58–70, 2010.

[2] Fogal et al. Large Data Visualization on Distributed Memory Multi-GPU Clusters. High Performance Graphics, pages 57-66, 2010.

[3] Gobbetti et al. A Single-Pass GPU Ray Casting Framework for Interactive Out-of-Core Rendering of Massive Volumetric Datasets. The Visual Computer, 24(7), pages 797-806, 2008.

[4] Müller et al. Optimized Volume Raycasting for Graphics-Hardware-based Cluster Systems. Eurographics Symposium on Parallel Graphics and Visualization (EGPGV'06), pages 59-66, 2006.