Visapult: A Prototype Remote and Distributed Visualization Application and Framework

Wes Bethel
Lawrence Berkeley National Laboratory
and R3vis Corporation

The Problem

Remote visualization is the process of creating pictures from data, where the data and viewer are on different platforms, often separated by a wide area network. Traditional approaches to remote visualization have tended to fall into two broad categories. *Render-remote* approaches are implementations where data storage and access, visualization, rendering and viewer involve a single machine. *Render-local* approaches require transmission of data to the viewer's machine, where it is then visualized and rendered. The former approach has the potential to address large-data problems, but sacrifices interactivity. The latter approach has the potential to preserve interactivity, but assumes that the data will fit entirely on the local workstation, and that it is practical to transmit the data across the network.

Our Approach

We present *Visapult*, a prototype visualization application and framework that strikes a balance between render-local and render-remote approaches. Visapult was designed to address the needs of scientific researchers who produce and need to visualize hundreds of gigabytes, if not terabytes, of time-varying data. With data of this scale, neither transmission across a network nor local storage is practical, yet interaction with the visualization is crucial. We did not want to sacrifice the gains in understanding that result from motion parallax and stereoscopic presentation [1]. To achieve these goals, we have implemented an application that uses *shared rendering*. Shared rendering refers to a cooperative visualization and rendering framework in which some of the rendering is performed close to the data, and some rendering is performed on the local workstation. Given the disparity between data transmission rates and the requirements for rendering interactivity, it is desirable to keep frame rates isolated from, and independent of, the latency characteristic of network-based applications.

How it Works

Visapult consists of two logical components: a local viewer and a remote, back end data engine. The back end retrieves scientific data from some source, such as a network data cache [2], then performs visualization and partial rendering in parallel. Results are then transmitted to a viewer, which is an interactive 3D graphics application. The viewer and back end communicate over the network using a custom TCP-based protocol. The back end volume visualization engine is an implementation of an image-based rendering assisted volume renderer [3]. The IBR-assisted volume rendering model decomposes nicely into a remote, parallel component and a local, serial component. The remote component scales with available computational resources using an object-order, domain decomposition for parallel volume rendering. The local viewer, also a parallel application, provides rendering at interactive frame rates, even on low-cost hardware or in stereo. In the viewer, a scene graph model is updated by parallel listener processes whenever new data arrives from the back end, but is rendered continuously by a dedicated process, effectively separating network latency from rendering frame rates. Arbitrary geometry can be included in the scene graph so that volume visualization is complemented with grid visualization, isocontours or other visualization techniques. The fundamental design paradigm is the combination of parallel network communication, combined with asynchronous scene graph updates.

The Future

Based upon the results of ongoing performance and profiling analysis, future work in Visapult will focus upon performance improvements. From an architectural perspective, aggressive and asynchronous data prefetching will improve overall data throughput. From a "network awareness" perspective, dynamic analysis of the state of the network can lead to parameter changes based upon current bit rate, reliability or other Quality-of-Service attributes. In some cases, bandwidth reservation is needed to allow for visualization of very large (terascale) scientific data, where the scientific data is located on a network cache, or otherwise not local to the back end visualization and rendering engine.

Acknowledgment

This work was supported by the Director, Office of Science, Office of Basic Energy Science, of the US Department of Energy under Contract No. DE-AC03-76SF00098.

References

[1] Colin Ware and Glenn Franck, Evaluating Stereo and Motion Cues for Visualizing Information Nets in Three Dimensions, ACM Transactions on Graphics, Volume 15, Number 2, April 1996.

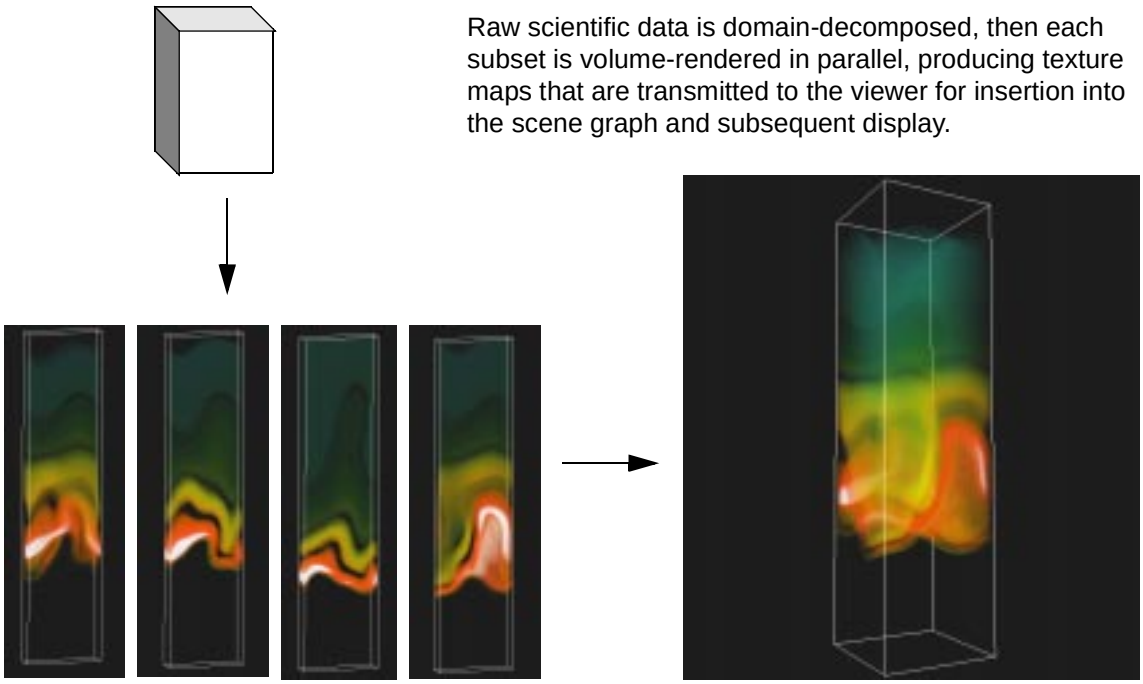
[2] “A Network-Aware Distributed Storage Cache for Data Intensive Environments”, Tierney, B. Lee, J., Crowley, B., Holding, M., Hylton, J., Drake, F., Proceedings of IEEE High Performance Distributed Computing conference, August 1999, LBNL-42896. see: http://www-didc.lbl.gov/DPSS/

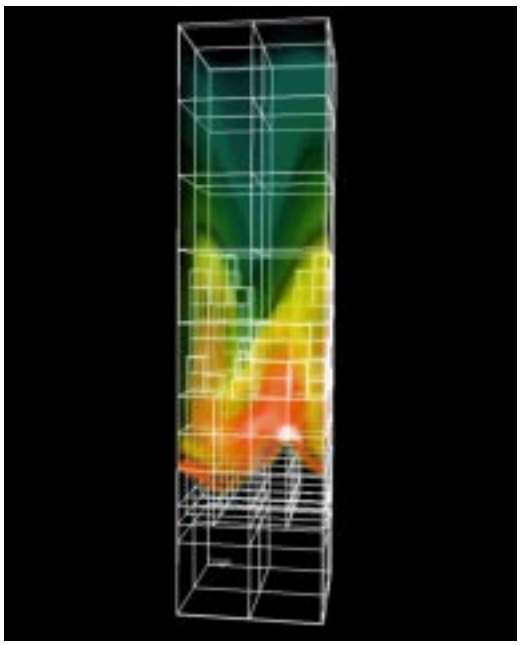
[3] Klaus Meuller, Naeem Shareef, Jian Huang and Roger Crawfis, IBR-Assisted Volume Rendering, in Proceedings of IEEE Visualization 1999, Late Breaking Hot Topics, October 1999.

Wes Bethel
Lawrence Berkeley National Laboratory
and R3vis Corporation

Figure 1.

Raw scientific data is domain-decomposed, then each subset is volume-rendered in parallel, producing texture maps that are transmitted to the viewer for insertion into the scene graph and subsequent display.





Visualization of adaptive, hierarchical grids from the combustion simulation are included with volume rendering.

Figure 2.