SAND2008-3743C

Remote rendering for ultrascale data

Kenneth Moreland[1] **and Greg Humphreys**[2]

[1]Sandia National Laboratories, P.O. Box 5800 MS 1323, Albuquerque, NM 87185-1323
[2]Department of Computer Science, School of Engineering and Applied Science, University of Virginia, 151 Engineer's Way, P.O. Box 400740, Charlottesville, VA 22904-4740

E-mail: kmorel@sandia.gov

Abstract. The mission of the SciDAC Institute for Ultrascale Visualization is to address the upcoming petascale visualization challenges. As we move to petascale computation, we are seeing a trend not only in the growth but also in the consolidation of computing resources. As the distances between user and petascale visualization resources grow, the expected performance of the network degrades, especially with respect to latency. In this paper we will explore a technique for remote visualization that leverages unstructured lumigraph rendering. This technique will provide an interactive rendering experience regardless of the network performance to the remote visualization resource. The unstructured lumigraph rendering can also replace many of the other level-of-detail techniques currently used that have problems that are exasperated by petascale data.

1. Introduction

Over the past two decades, we have seen an extraordinary amount of change in both the hardware and software used for the visualization of simulation results. Throughout the 1980's, graphics technology was still too slow to provide interactive rendering of any reasonably sized 3D mesh. Visualization of this type of data was typically script-driven and off-line.

By the end of the 1980's, specialized graphics hardware became available [1]. In the early 1990's, this specialized graphics hardware became common, although expensive. Visualization of large simulations (for the time) necessitated the use of big SMP computers with lots of memory and several rendering pipelines.

In the late 1990's and early 2000's, large scale rendering and visualization moved from the specialized SMP computers to clusters comprising commodity components with distributed memory [2, 3]. The reasons for this change were two-fold. First, it became clear that a visualization platform based on an SMP architecture was not going to scale. Second, the entertainment industry was driving commodity graphics hardware to improve at a much faster rate than the specialized visualization hardware.

As visualization and simulation platforms continue to grow and change, so to do the challenges and bottlenecks. Although image rendering was once the limiting factor in visualization, rendering speed is now often a secondary concern. In fact, it is now common to perform interactive visualization on clusters without specialized rendering hardware. The new focus on visualization is in loading, managing, and processing data. One consequence of this is the practice of co-locating a visualization resource with the supercomputer for faster access to the simulation output [4, 5].

Most of us are not fortunate enough to have a supercomputer or associated visualization resource located in our office, yet we still require access to these resources. A remote visualization capability allows us to provide the widest access to our resources; anyone with network capabilities can perform large scale visualization from their desktop using remote visualization.

The distance between scientist and computing/visualization resources, as a trend, is growing. The US Office of Science, through programs like SciDAC, is encouraging scientists from different disciplines and different organizations to collaborate; the US Department of Defense and Department of Energy are consolidating resources like supercomputers; universities are reaching out to researchers around the world. As the distance between user and visualization resource becomes larger, remote visualization becomes more important and more challenging. One of the goals of the SciDAC Institute for Ultrascale Visualization is to ensure that remote visualization remains an interactive experience, even when accessing petascale data on the far side of the world.

2. Remote Rendering

Unlike most large-scale parallel processing, visualization tends to be an interactive process by nature. The parallel visualization must be controlled remotely to make its use convenient and, in some cases, possible. Thus, most large-scale turn-key visualization systems, such as ParaView, VisIt, and EnSight, provide some way to access the parallel visualization remotely. The architecture of these systems varies, but all comprise a client, run on a local desktop or laptop, that connects, via a socket, to one or more servers running on a remote parallel machine.

All data processing occurs at the server. It is assumed that the data may be too large to load and certainly too large to efficiently process on the client, for that is the whole purpose of leveraging a remote parallel machine. Rendering, however, can occur at either the client or the server.

User interaction requires the rendering to be very responsive. Rendering data in the client can be convenient as the locally rendered image is immediately available. However, rendering at the client requires all applicable geometry to be shipped from server to client. Sometimes this is feasible as extracted surfaces are typically much smaller than the volumes from which they are derived. Even so, surfaces extracted from simulations run at capacity on large supercomputers will still be too large for clients. Furthermore, some rendering techniques, such as volume rendering, require the full mesh geometry, which cannot be effectively loaded on the client. For these reasons, most large scale visualization systems provide server-side rendering.

A server side rendering uses a parallel rendering algorithm on the server and ships images back to the client. The server generally uses a "sort-last" image-based algorithms, which tend to be scalable because their overhead is set by the number of pixels in the output image and therefore limited by the resolution of the physical display [3, 6–9].

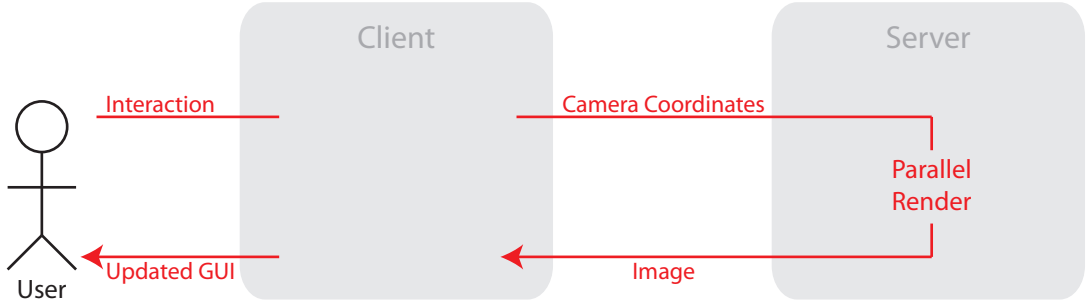

Figure 1. Update cycle for remote rendering.

Figure 1 displays the typical update cycle for remote rendering. A user interaction in the client will spawn the need for a new image, which will be forwarded to the server along with new rendering parameters such as camera coordinates. The server will perform the render using a sort-last parallel algorithm. That image is sent back to the client where it is used to update the user interface.

This remote rendering cycle maximizes the rendering speed in terms of primitives per second of a parallel visualization system. However, there is no guarantee that it will provide interactive frame rates: a minimum of 10 Hz. The physical limitations of the server's rendering hardware may not be able to process all the geometry of petascale in less than a tenth of a second. Furthermore, the network connection between client and server is unlikely to be able to stream full resolution images at an acceptable rate.

To accommodate situations where rendering a full resolution image interactively is not possible, many rendering systems employ at least two modes of rendering. A lower-quality "interactive" render occurs while the user is manipulating the view. In this mode quality and accuracy can be sacrificed for speed. A high-quality "still" render occurs when the user stops interacting, the system is ready to idle, and the user is ready to study the visualization image. In this mode the entire detail of the visualization is presented even if the user must wait a few moments to get it.

ParaView uses multiple level-of-detail techniques to make the full remote render cycle interactive [10]. A geometric level of detail uses a quadric clustering algorithm [11] to provide a decimated, but representative, version of the geometry. An image level of detail reduces the resolution of the images generated, which reduces the amount of work for the parallel rendering algorithm and the image transfer back to the client. A lossy compression based on the amount of quantification levels for the color provides yet another level-of-detail parameter.

Although these techniques work well to provide an interactive remote rendering loop on a local area network, it sometimes fails on a wider area network. The largest problem imposed by a wide area network is the latency that can occur in the connection between client and server. This adds a delay to the remote rendering loop that cannot be accommodated by any level-of-detail technique.

The only possible way around the network latency is to perform all interactive rendering local to the client. Such an approach is not uncommon. The only caveat is that this local rendering does not have access to the full geometry, and the local rendering becomes a compromise between the detail that can be seen and the data that must be retrieved. VisIt performs this local rendering by drawing only the bounding box of the data during interaction. The bounding box can provide orientation, but it is often difficult to conceptualize the final rendering result, especially when zoomed far in the data. ParaView has the ability to bring in the decimated geometry to the client and render that locally, but even the decimated geometry can saturate the client when it comes from a large server.

We propose a new technique for approximate interactive rendering in remote visualization using image based techniques. As in the previous examples, the approximate rendering will happen completely locally. The difference is that the approximate rendering will be based on images rendered with the full geometry. As shown in Figure 2, high-quality still images generated by the server are cached on the client. This cache of images is used to synthesize images from new viewpoints during interactive rendering.

Using images rendered from the server to synthesize other approximate images has two distinct advantages. First, it is scalable with respect to the size of the geometry being represented. Like the sort-last parallel rendering employed on the server, the overhead of the technique is independent of geometry size. Second, the image base rendering can leverage the rendering already being done on the server. Lengthy preprocessing of the data is not necessary, as is the case with geometric decimation.

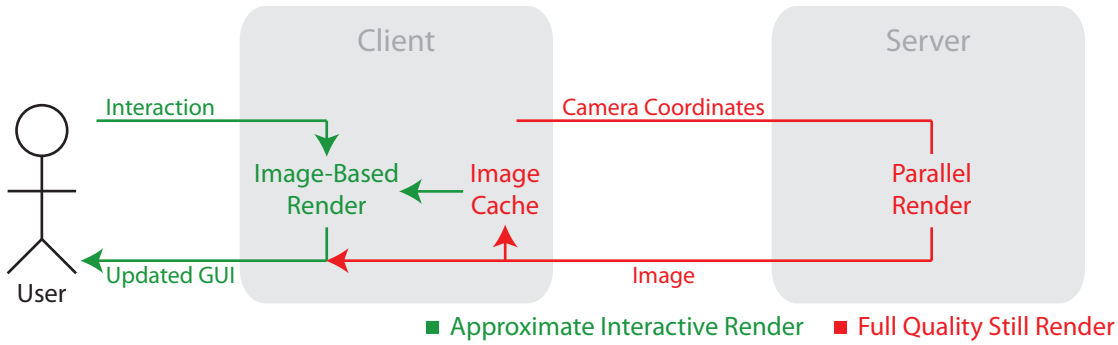

Figure 2. Update cycles for remote rendering with a local image-based approximate interactive rendering in the client.

3. Unstructured Lumigraph Rendering

To be contributed by University of Virginia folks.

4. Conclusions

Remote rendering is a vital part of large scale visualization in the field. It makes a limited visualization resource accessible to a wider circle of users. However, most of the techniques used today are limited by the amount of data they can render, limited by the latency of the network connections (and thus the physical distance), or limited by the amount of interactivity afforded. By leveraging an image based technique like lumigraph rendering allows us to circumvent these technical issues and provide a fully interactive scalable visualization environment in any network environment.

Acknowledgments

This work was done in part at Sandia National Laboratories. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

References

[1] Akeley K and Jermoluk T 1988 High-performance polygon rendering *Computer Graphics (SIGGRAPH '88 Proceedings)* **22** 239–246

[2] Ahrens J, Brislawn K, Martin K, Geveci B, Law C C and Papka M E 2001 Large-scale data visualization using parallel data streaming *IEEE Computer Graphics and Applications* **21** 34–41

[3] Wylie B, Pavlakos C, Lewis V and Moreland K 2001 Scalable rendering on PC clusters *IEEE Computer Graphics and Applications* **21** 62–70

[4] Ahern S 2007 Petascale visual data analysis in a production computing environment *Journal of Physics: Conference Series (Proceedings of SciDAC 2007)* vol 78

[5] Cedilnik A, Geveci B, Moreland K, Ahrens J and Farve J 2006 Remote large data visualization in the ParaView framework *Eurographics Parallel Graphics and Visualization 2006* pp 163–170

[6] Ma K L 1994 Parallel volume rendering using binary-swap image composition *IEEE Computer Graphics and Applications* **14** 59–68

[7] Moreland K, Avila L and Fisk L A 2007 Parallel unstructured volume rendering in paraview *Visualization and Data Analysis 2007, Proceedings of SPIE-IS&T Electronic Imaging* pp 64950F–1–12

[8] Moreland K and Thompson D 2003 From cluster to wall with VTK *Proceedings of IEEE Symposium on Parallel and Large-Data Visualization and Graphics* pp 25–31

[9] Moreland K, Wylie B and Pavlakos C 2001 Sort-last parallel rendering for viewing extremely large data sets on tile displays *Proceedings of the IEEE 2001 Symposium on Parallel and Large-Data Visualization and Graphics* pp 85–92

[10] Squillacote A H 2007 *The ParaView Guide* ParaView 3 ed (Kitware, Inc.) ISBN-13: 978-1-930934-21-4
[11] Lindstrom P 2000 Out-of-core simplification of large polygonal models *Computer Graphics (Proceedings of SIGGRAPH 2000)* 259–262