Remote Parallel Rendering for High-Resolution Tiled Display Walls

Daniel Nachbaur* Raphael Dumusc* Ahmet Bilgili* Juan Hernando† Stefan Eilemann*

*Blue Brain Project, EPFL; †CeSViMa, UPM

Abstract—We present a complete, robust and simple to use hardware and software stack delivering remote parallel rendering of complex geometrical and volumetric models to high resolution tiled display walls in a production environment. We describe the setup and configuration, present preliminary benchmarks showing interactive framerates, and describe our contributions for a seamless integration of all the software components.

Index Terms—tiled displays, interactive remote rendering, parallel rendering

1 Introduction

Simulations performed on today's high performance computers produce massive amounts of data, which are too expensive to move to another system. On the other hand, tiled display walls have proven to help understanding complex data due to their size, resolution and collaborative usage. Often the two systems are not located in the same facility due to power constraints or other factors.

In this systems poster we present a full production-ready software stack which enables the rendering of large amounts of data on a parallel visualization cluster, streaming of the results over a WAN link to a remote location, where it is displayed at interactive framerates on a tiled display wall alongside other local or remote content. The full software stack, with the exception of one domain-specific application, is available as open source to be reused in similar settings. Figure 1 shows our setup, motivated by the needs of the Blue Brain Project.

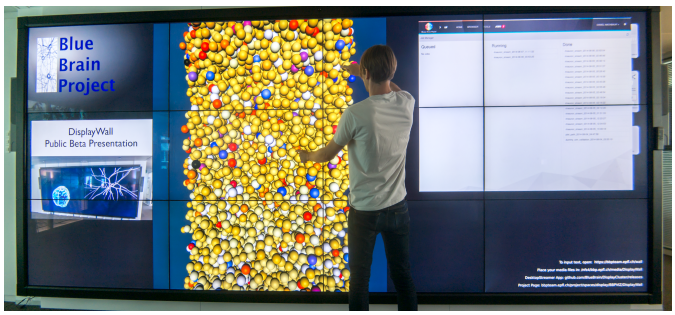

Fig. 1. Multitouch interaction with a remote parallel rendering of a neocortical column simulation (middle) next to the web portal browser (right) used to launch the rendering and a presentation (left)

A remote BlueGene supercomputer with a co-hosted visualization cluster, described in Section 2, is the main producer of simulation data in the project. Interactive 3D visualization applications built on an open source software stack described in Section 3 are used to analyze the results using geometric and volumetric rendering, as described in Section 3.4 and Section 3.5. The results of these visualizations are shown on the local workstation using VirtualGL [1] and on a tiled display wall running DisplayCluster.

*Firstname.Lastname@epfl.ch
†jhernando@fi.upm.es

IEEE Symposium on Large Data Analysis and Visualization 2014
October 9–10, Paris, France
978-1-4799-5215-1/14/$31.00 ©2014 IEEE

2 Hardware

Figure 2 shows a diagram of the hardware setup. At the Lugano facility, a four-rack BlueGene/Q, a parallel GPFS filesystem and a 40-node visualization cluster are hosted, all connected using an InfiniBand switch. The visualization cluster nodes have two NVidia Tesla K20 cards, two Intel Xeon E5-2670 v2 processors (8 cores @ 2.6GHz) and 128 GB of memory. They are also connected to a ten gigabit ethernet switch, which serves the WAN link to Lausanne, located about 250 km awav.

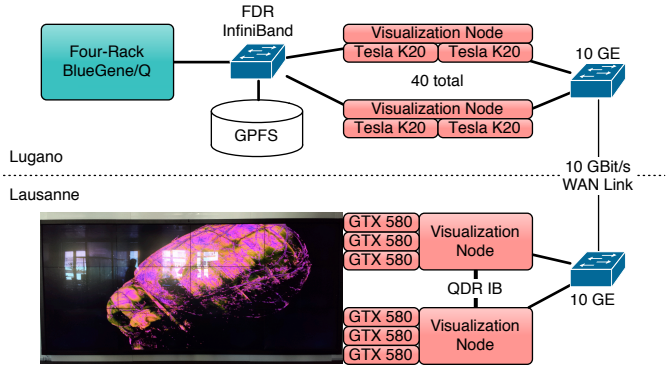

Fig. 2. Hardware Setup

In Lausanne, the link is routed to another ten gigabit ethernet switch connected to the local visualization cluster. For display, a 24 megapixel, 4×3 tiled display wall is used. Two display nodes use three NVidia GTX580 each to drive two Full-HD displays per GPU. Each node has two Xeon X5690 (6 cores @ 3.47GHz) and 24 GB memory. The nodes are connected to the ten gigabit ethernet switch as well as a local QDR InfiniBand network.

3 Software

The hardware resources are managed by the Slurm job scheduling system. The Lugano cluster receives its data from the GPFS and performs parallel rendering using mesh and direct volume rendering. The rendered images are then sent over the WAN link to the Lausanne cluster, where they are displayed on the tiled display wall.

3.1 DisplayCluster

DisplayCluster [5] is an interactive visualization environment for cluster-driven tiled displays. It provides a dynamic, desktop-like windowing system with built-in media viewing capability that supports ultra high-resolution imagery and video content and streaming that allows arbitrary applications from remote sources to be shown.

3.2 Streaming Library

We extended the existing DisplayCluster implementation by reimplementing the streaming using a simple library [2], used by applications

to display their content in a DisplayCluster window. The application usually provides an image buffer, which is compressed and sent asynchronously and in parallel by the stream library. Multiple stream sources from multiple processes can provide content to a single window on the wall, enabling it for cluster-driven parallel rendering. The stream library also implements an event model, where an application can register itself to receive keyboard, mouse and window management events from the display wall.

3.3 Equalizer

Equalizer is a framework to develop and deploy distributed and non-distributed parallel rendering applications. The programming interface is based on a set of C++ classes, modeled closely to the resource hierarchy of a graphics rendering system, described in more detail in [3]. An Equalizer application is configured automatically using local and remote resource discovery, or manually using simple ASCII configuration files.

We integrated the stream library into Equalizer to send the framebuffer of each destination channel to DisplayCluster, using a direct FBO download (if possible) or a texture download. The compression and sending uses the *asyncSend* function, pipelining compression and streaming with rendering. Received events from DisplayCluster are forwarded to Equalizer's event system. This integration allows all Equalizer applications to benefit from streaming without any code changes. It is configured simply by specifying the *DisplayCluster* hostname for all views to be streamed.

3.4 RTNeuron

RTNeuron [4] is a neuroscientific visualization application designed for interactive visualization of detailed cortical circuit simulations. It features several graphics techniques and algorithms tailored to the specific geometrical characteristics of neurons to allow the efficient rendering of neuronal circuits where simulation data is mapped onto the membrane meshes. Some of these features have to do with levels of detail, view frustum culling and correct alpha-blending. To enable the visualization of large circuits at interactive speeds, RTNeuron makes use of Equalizer to implement sort-first parallel rendering with load balancing and sort-last rendering with a static decomposition. We use a static sort-first decomposition as described in Section 4.

3.5 Livre

Livre (Large-scale Interactive Volume Rendering Engine) is an out-of-core direct volume rendering engine. It uses an octree data structure to provide an error based level of detail (LOD) selection, aiming for a constant rendering quality. An asynchronous rendering pipeline separates the rendering algorithm into asynchronous data, upload and render threads. Equalizer is used for sort-first rendering, as shown in Figure 2.

3.6 Web Portal Integration

Within the Human Brain Project, scientists are able to access simulation data from a web-based unified portal. Alongside with several analysis tasks that can be performed on the data, a task to stream the rendering of a precomputed simulation using RTNeuron was implemented. It uses Slurm to allocate rendering resources and launches an RTNeuron instance with an Equalizer configuration to stream the rendering from each GPU to the display wall. User interaction is possible by interacting with the multi-touch system to navigate through the scene. Figure 1 shows the setup of the RTNeuron stream launched from the web portal.

4 Results

Figure 3 shows the performance of streaming a light-weight rendering from the Lugano cluster to our 24 Megapixel wall. We tested three resolutions (1920×1080, 3840×2160 and 7680×3240), two camera positions (full model and close up) and three different tile sizes (256^2, 512^2 and 1024^2). Due to a misconfiguration, the WAN link delivered only 1 GBit/s throughput during the benchmark test.

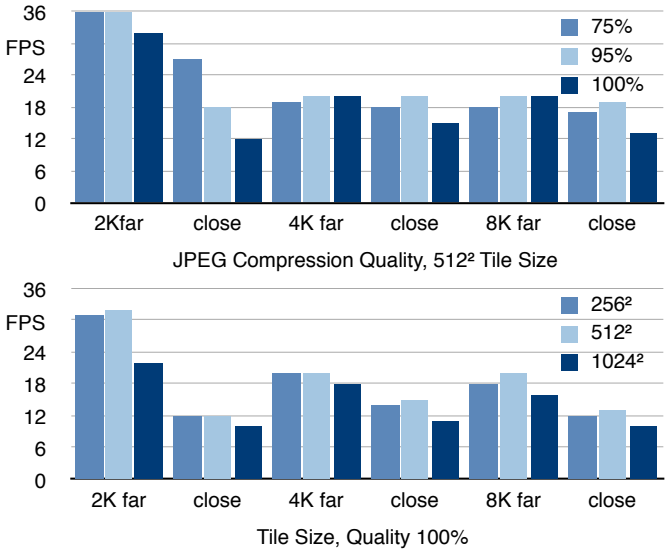

Fig. 3. Performance Benchmarks

The results show that interactive framerates are available even at the full native resolution, that a 512^2 tile size is the best option found, and that 95% compression delivers the best performance in most cases. The far camera has more uniform background pixels and yield better compression ratios. Based on the experiments we settled on a 512^2 tile size and 100% compression quality to avoid artifacts.

5 Conclusion and Future Work

In this poster, we present a full fledged software stack to perform high-resolution remote parallel rendering in a production environment, and describe how to set up and configure the components. This work can be easily reused in other institutions due to its open source implementation.

We intend to improve the streaming performance by employing better compression algorithms, in particular hardware-based encoding on the GPUs and improving the pixel readback performance.

Acknowledgments

This work was supported in part by the Blue Brain Project, the Swiss National Science Foundation under Grant 200020-129525, by the Spanish Ministry of Economy and Competitiveness under the Cajal Blue Brain Project, the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 604102 (Human Brain Project) and the King Abdullah University of Science and Technology (KAUST) through the KAUST-EPFL alliance for Neuro-inspired High Performance Computing. We would also like to thank github for providing an excellent infrastructure hosting our projects at http://github.com/Bluebrain and http://github.com/Eyescale.

References

[1] D. Commander. The VirtualGL Project - 3D without Boundaries. http://virtualgl.org/, 2011.

[2] R. Dumusc and D. Nachbaur. DisplayCluster Stream library. http://bluebrain.github.io/DisplayCluster-0.3, 2014.

[3] S. Eilemann, M. Makhinya, and R. Pajarola. Equalizer: A scalable parallel rendering framework. *IEEE Transactions on Visualization and Computer Graphics*, May/June 2009.

[4] J. B. Hernando, J. Biddiscombe, B. Bohara, S. Eilemann, and F. Schürmann. Practical parallel rendering of detailed neuron simulations. In *Proceedings of the 13th Eurographics Symposium on Parallel Graphics and Visualization*, EGPGV '13.

[5] G. P. Johnson, G. D. Abram, B. Westing, P. Navr'til, and K. Gaither. Displaycluster: An interactive visualization environment for tiled displays. *2013 IEEE International Conference on Cluster Computing (CLUSTER)*, 0:239–247, 2012.