Poster Abstracts at Eurographics/ IEEE-VGTC Symposium on Visualization (2010)
D. Auber, G. Melançon, T. Munzner, and D. Weiskopf (Chairs)

Equalizer - Parallel Rendering Framework

S. Eilemann[†1,2], R. Pajarola[‡1]

[1]Visualization and MultiMedia Lab, University of Zurich [2]Eyescale Software GmbH

Abstract

Continuing improvements in CPU and GPU performances as well as increasing multi-core processor and cluster-based parallelism demand for flexible and scalable parallel rendering solutions that can exploit multipipe hardware accelerated graphics. In fact, to achieve interactive visualization, scalable rendering systems are essential to cope with the rapid growth of data sets.

In this poster we present Equalizer*, a toolkit for scalable parallel rendering based on OpenGL which provides an application programming interface (API) to develop scalable graphics applications for a wide range of systems ranging from large distributed visualization clusters and multi-processor multi-GPU graphics systems to single-processor single-GPU desktop machines. We present and demonstrate the most important use cases and recent advances of Equalizer.*

Categories and Subject Descriptors (according to ACM CCS): I.3.2 [Computer Graphics]: Graphics Systems/Distributed Graphics—Parallel Rendering; I.3.m [Computer Graphics]: Miscellaneous/Rendering Clusters—Scalable Rendering; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism/Virtual Reality—Immersive Environments

1. Introduction

The continuing improvements in hardware integration lead to ever faster CPUs and GPUs, as well as higher resolution sensor and display devices. Moreover, increased hardware parallelism is applied in form of multi-core CPU workstations, massive parallel super computers, or cluster systems. Hand in hand goes the rapid growth in complexity of data sets from numerical simulations, high-resolution 3D scanning systems or bio-medical imaging, which causes interactive exploration and visualization of such large data sets to become a serious challenge. It is thus crucial for a visualization solution to take advantage of hardware accelerated scalable parallel rendering. In this systems poster we present a scalable parallel rendering framework called *Equalizer* that is aimed primarily at cluster-parallel rendering, but works as well in a shared-memory system. Figure 1 shows the main use cases for parallel rendering.

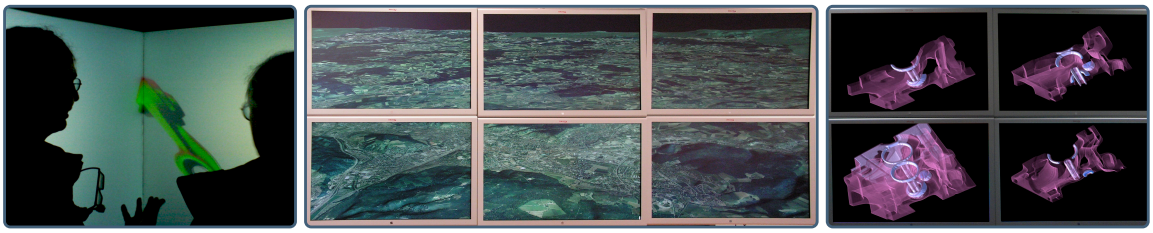
In addition to the basic framework described in [EMP09], we will present the latest research results around Equalizer, namely fast compositing for cluster-parallel rendering [MEP10], an Equalizer-based application for scalable parallel out-of-core terrain rendering [GMBP10], cross-segment load-balancing and reliable multicast for data distribution.

2. Parallel Rendering Framework

Equalizer provides a framework to facilitate the development of 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. The application subclasses these objects and overrides C++ task methods, similar to C callbacks. These task methods will be called in parallel by the framework, depending on the current configuration. A wrapper interface could be written to provide C bindings. This parallel rendering interface is significantly different from Chromium [HHN*02] and more similar to VRJuggler [BJH*01] or MPK [BRE05]. The class framework and in particular its use is described in more detail in [EMP09].

An Equalizer application does not have to select a par-

† eilemann@gmail.com
‡ pajarola@acm.org



(a) (b) (c)

Figure 1: *Various Equalizer use cases: (a) immersive CAVE, (b) display wall and (c) scalable volume rendering.*

ticular rendering configuration itself; it is configured by a system-wide configuration server. The application is written only against a client library, which communicates with the server that does not have to be touched by the developer. The parallel rendering configuration is initialized by the server based on guidelines from the application or a user supplied configuration file. The server also launches and controls the distributed rendering clients provided by the application. Thus the application itself can run unmodified on any configuration which has been initialized by the server, and if none is given the application will run as a stand-alone process on the node it has been started.

3. Fast Compositing

The image compositing stages in cluster-parallel rendering for gathering and combining partial rendering results into a final display frame are fundamentally limited by node-to-node image throughput. Therefore, efficient image coding, compression and transmission must be considered to minimize that bottleneck. We present the different performance limiting factors such as image representation, region-of-interest detection and fast image compression. Additionally, we show improved compositing performance using lossy YUV subsampling and a novel fast region-of-interest detection algorithm that can improve in particular sort-last parallel rendering, as described in [MEP10]

4. Scalable Out-of-core Terrain Rendering

We present a novel out-of-core parallel and scalable technique for rendering massive terrain datasets [GMBP10]. The parallel rendering task decomposition is implemented on top of an existing terrain renderer using Equalizer for cluster-parallel rendering. Our approach achieves parallel rendering by division of the rendering task either in sort-last (database) or sort-first (screen domain) manner and presents an optimal method for implicit load balancing in the former mode. The effciency of our approach is validated using massive elevation models.

5. Cross-Segment Load-Balancing

Cross-segment load-balancing provides a novel approach to scale the display size and rendering performance concurrently using a limited set of rendering resources. Our implementation is completely application-transparent, and decouples the display resources from the rendering tasks. Therefore it provides performance benefits for multi-display environment over the traditional one GPU per display approach.

6. Conclusion

In this poster, we present a state-of-the art distributed parallel rendering framework, which is designed to be minimally invasive in order to facilitate the porting and development of real-world visualization applications. Equalizer is as generic as possible to support development of parallel rendering applications for different data types. Current advances in Equalizer have been mostly transparent to existing applications, that is, they can benefit from new fast compositing algorithms and cross-segment load-balancing without any code modifiations.

References

[BJH*01] BIERBAUM A., JUST C., HARTLING P., MEINERT K., BAKER A., CRUZ-NEIRA C.: VR Juggler: A virtual platform for virtual reality application development. In *Proceedings of IEEE Virtual Reality* (2001), pp. 89–96.

[BRE05] BHANIRAMKA P., ROBERT P. C. D., EILEMANN S.: OpenGL Multipipe SDK: A toolkit for scalable parallel rendering. In *Proceedings IEEE Visualization* (2005), pp. 119–126.

[EMP09] EILEMANN S., MAKHINYA M., PAJAROLA R.: Equalizer: A scalable parallel rendering framework. *IEEE Transactions on Visualization and Computer Graphics* (May/June 2009).

[GMBP10] GOSWAMI P., MAKHINYA M., BÖSCH J., PAJAROLA R.: Scalable parallel out-of-core terrain rendering. In *Proceedings Eurographics Symposium on Parallel Graphics and Visualization* (to appear 2010).

[HHN*02] HUMPHREYS G., HOUSTON M., NG R., FRANK R., AHERN S., KIRCHNER P. D., KLOSOWSKI J. T.: Chromium: A stream-processing framework for interactive rendering on clusters. *ACM Transactions on Graphics 21*, 3 (2002), 693–702.

[MEP10] MAKHINYA M., EILEMANN S., PAJAROLA R.: Fast compositing for cluster-parallel rendering. In *Proceedings Eurographics Symposium on Parallel Graphics and Visualization* (to appear 2010).