Multi-Display Ray Tracing Framework

Luciano A. Romero Calla[1], Bipul Mohanto[2], Renato Pajarola[1], and Oliver Staadt[2]

[1]University of Zurich, Switzerland
[2]University of Rostock, Germany

Summary

We have developed a framework for multi-display rendering using advanced technologies such as MPI (Message Passing Interface), CUDA (Compute Unified Device Architecture), CUDA IPC (Inter-Process Communication), OptiX 7.6, and the C++ programming language.

Related Work

We can divide the related work into rasterization and ray-tracing based approaches. Among the latest rasterization works, we find [1], which extend [2] to handle load balancing and LOD compared to *Equalizer* [3]. *Equalizer* [3] is a framework for scalable, parallel rendering and data distribution for large scale visualizations. Another relevant work is [4], which extends OpenGL to implement a distributed framework for high-performance visualization systems.

Our framework belongs to the second group of ray-tracing-based approaches. In this group, we find [5] that presents a framework for rendering large tiled display walls as a display service. [6] proposed a distributed frame buffer approach and extended the API from OSPRay [7]. Finally, not related to multi-display rendering but in the scope of distributed rendering, in [8] a data-distributed solution to path-tracing massive scenes across multiple GPUs has been proposed.

Overview

Figures 1, 2, and 3 shows our framework running in a 2×3 display wall.

Fig. 1: Ray Tracer: primary and shadows rays, Sponza scene.

Fig. 2: Ray Tracer: primary and shadows rays, San Miguel scene.

Fig. 3: Ray Tracer: Variable Rate Path Tracer, custom scene.

Framework

The Display processes run the multi-display module. Its implementation extends the OpenGL-based viewer from the ***gproshan*** framework [9] to handle multi-display using MPI and CUDA IPC. An RT & Display process initialize and run a ray-tracer implementation per GPU. It handles all the render tasks for the process running on the same GPU and their respective displays.

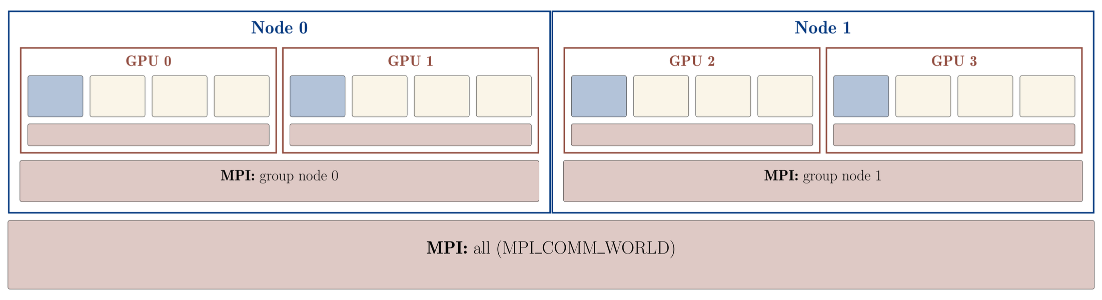

Fig. 4: Framework architecture

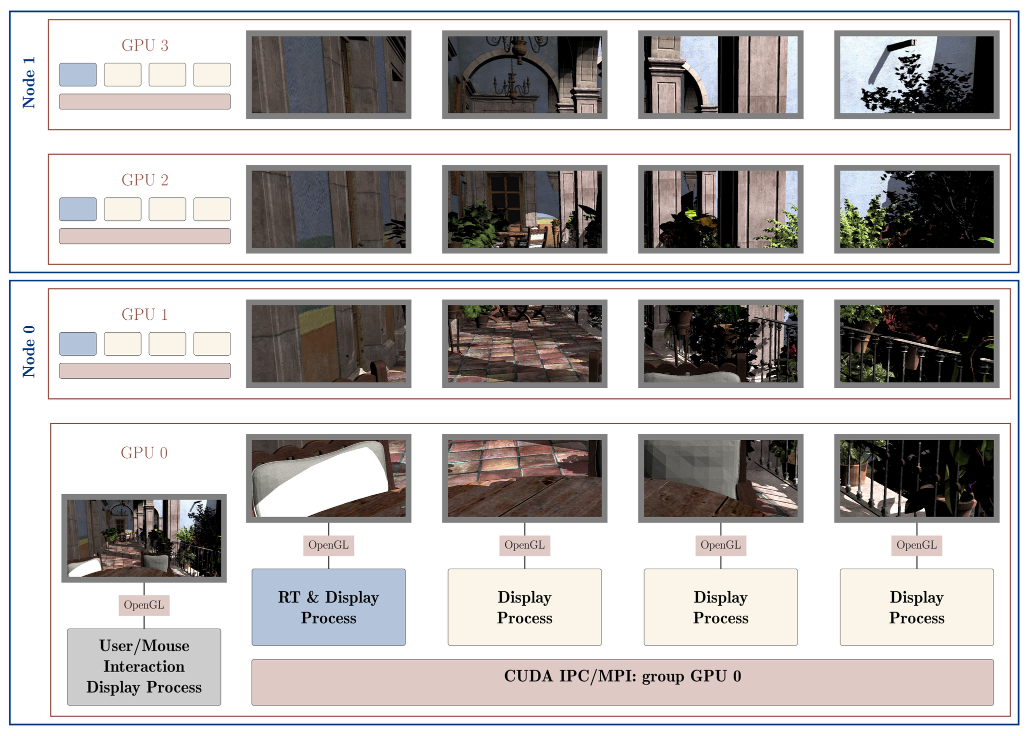

Fig. 5: Framework architecture

The setup to run our experiments for the general framework consists of two nodes with an Intel Core i7-10700K processor, 32GB of RAM, and NVIDIA GeForce RTX 3090 with 24GB of memory and a GeForce RTX 3080 with 10GB of memory, respectively in each node. The setup includes four monitors on the first node and three on the second one, all with a resolution of 2160×1440 pixels. Table in Figure 6 shows basics results.

Scene	Triangles	Monitors	GPU Memory	GPU usage %	FPS	Rendering
San Miguel	9980699	4	5719 MiB	72 %	68	per gpu
San Miguel	9980699	4	18171 MiB	70 %	73	per process
Sponza	262267	4	2410 MiB	66 %	67	per gpu
Sponza	262267	4	4933 MiB	67 %	72	per process
San Miguel	9980699	4, 3	3968 MiB, 3382 MiB	69 %, 55 %	74	per gpu
San Miguel	9980699	4, 3	**exceeds memory on second node**			per process
Sponza	262267	4, 3	2411 MiB, 1825 MiB	64 %, 50 %	64	per gpu
Sponza	262267	4, 3	4939 MiB, 3375 MiB	66 %, 49 %	76	per process

Fig. 6: FPS for a ray tracer with primary and shadow rays.

Variable Rate Path Tracer

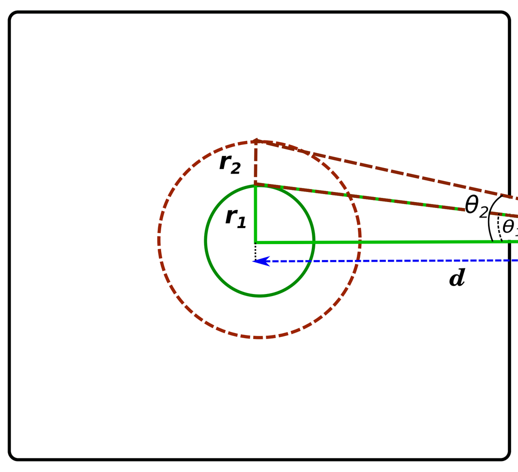

Fig. 7: Foveated, intermediate, and peripheral regions.

We have used 4 SPP (sample-per-pixel) for the unbiased path tracer and 4 SPP in the foveated, 2 SPP in the intermediate, and 1 SPP in the peripheral region for the variable rate path tracer. The ray bounces are limited to three. In addition, we used Reinhard Tone Mapping [10] for post-processing. Figure 8 illustrates the results. The average framerate for uniform sample path tracer is *3.72fps*, whereas the variable rate path tracing achieved framerate on average *16.62fps*; that is $4.45\times$ faster.

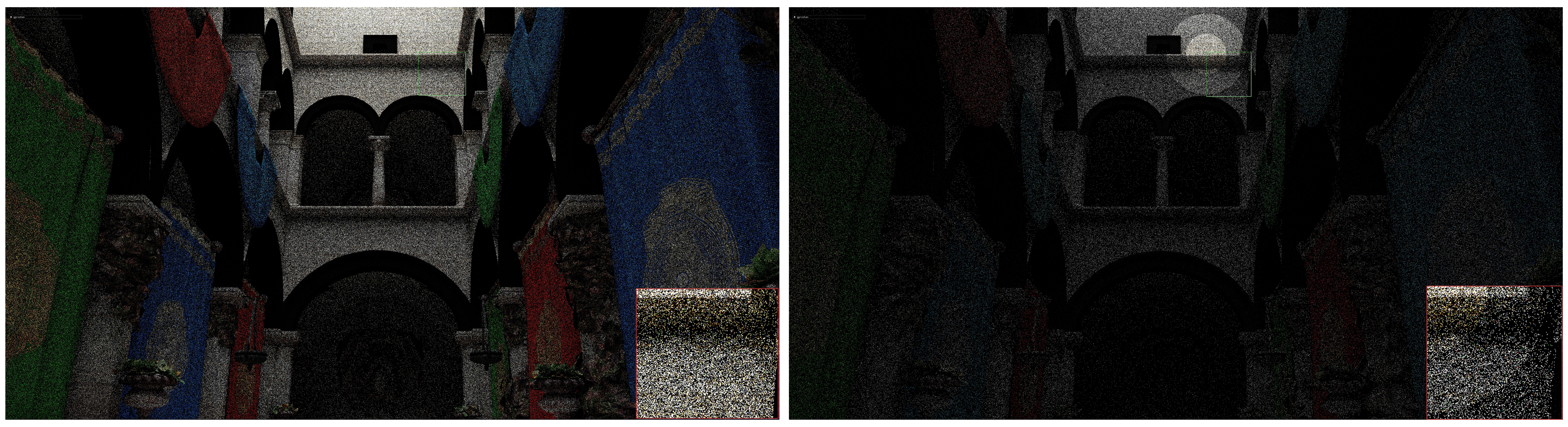

Fig. 8: The rendering results compare uniform (left) and our variable (right) numbers of radiance rays. The green box (right) marks the foveated region. In the lower right corner, a $3\times$ zoom inset view of the area is displayed.

References

[1] Yangzi Dong and Chao Peng. "Multi-GPU multi-display rendering of extremely large 3D environments". In: *The Visual Computer* (Dec. 2022). ISSN: 1432-2315. DOI: 10.1007/s00371-022-02740-7.

[2] Yangzi Dong and Chao Peng. "Screen Partitioning Load Balancing for Parallel Rendering on a Multi-GPU Multi-Display Workstation". In: *Eurographics Symposium on Parallel Graphics and Visualization*. Ed. by Hank Childs and Steffen Frey. The Eurographics Association, 2019. ISBN: 978-3-03868-079-6. DOI: 10.2312/pgv.20191111.

[3] Stefan Eilemann, David Steiner, and Renato Pajarola. "Equalizer 2.0 - Convergence of a Parallel Rendering Framework". In: *IEEE Transactions on Visualization and Computer Graphics* 26.2 (Feb. 2020), pp. 1292–1307. DOI: 10.1109/TVCG.2018.2870822.

[4] Kai-Uwe Doerr and Falko Kuester. "CGLX: A Scalable, High-Performance Visualization Framework for Networked Display Environments". In: *IEEE Transactions on Visualization and Computer Graphics* 17.2 (Mar. 2011), pp. 320–332.

[5] Mengjiao Han et al. "A Virtual Frame Buffer Abstraction for Parallel Rendering of Large Tiled Display Walls". In: *IEEE Visualization Conference (VIS)*. 2020, pp. 11–15. DOI: 10.1109/VIS47514.2020.00009.

[6] Will Usher et al. "Scalable Ray Tracing Using the Distributed FrameBuffer". In: *Computer Graphics Forum* 38.3 (2019), pp. 455–466. DOI: 10.1111/cgf.13702.

[7] I Wald et al. "OSPRay - A CPU Ray Tracing Framework for Scientific Visualization". In: *IEEE Transactions on Visualization and Computer Graphics* 23.1 (2017), pp. 931–940. DOI: 10.1109/TVCG.2016.2599041.

[8] Milan Jaroš et al. "GPU Accelerated Path Tracing of Massive Scenes". In: *ACM Transactions on Graphics* 40.2 (Apr. 2021). ISSN: 0730-0301. DOI: 10.1145/3447807.

[9] Luciano Arnaldo Romero Calla and Lizeth Joseline Fuentes Perez. *gproshan: geometry processing and shape analysis framework*. URL: https://github.com/larc/gproshan.

[10] Yasir Salih et al. "Tone mapping of HDR images: A review". In: *2012 4th International Conference on Intelligent and Advanced Systems (ICIAS2012)*. Vol. 1. 2012, pp. 368–373. DOI: 10.1109/ICIAS.2012.6306220.

EG23

Saarland Informatics Campus

University of Zurich[UZH]

Universität Rostock

Traditio et Innovatio

Funded by the European Union

This project has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No 813170.

EG23

LaTeX TikZposter