GPU-Accelerated Multi-Display Applications for Large CAD Model Visualization on a Commodity Desktop

Chao Peng and Yangzi Dong

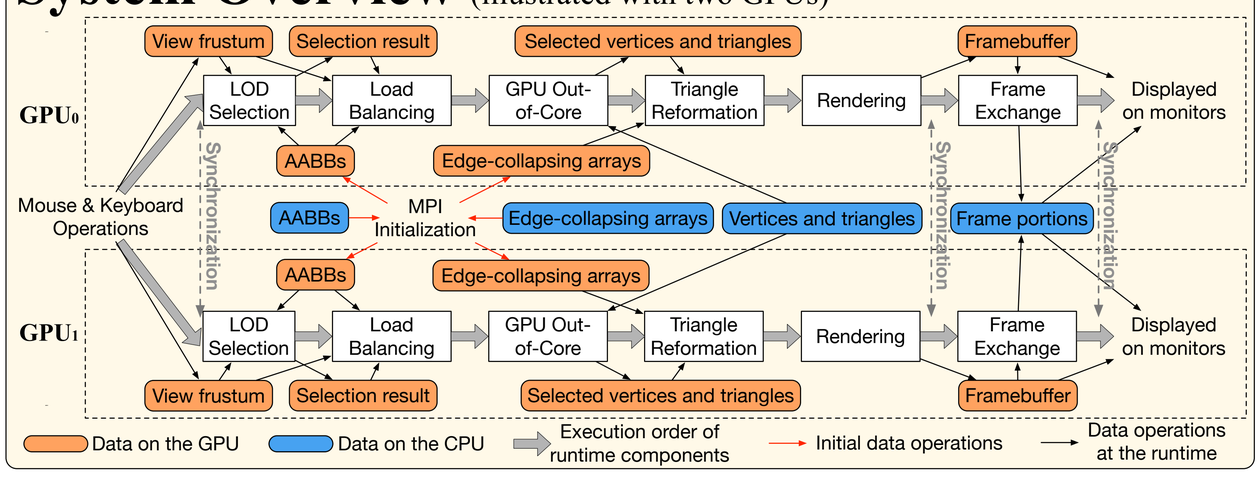
Department of Computer Science, University of Alabama in Huntsville, Huntsville, AL, United States

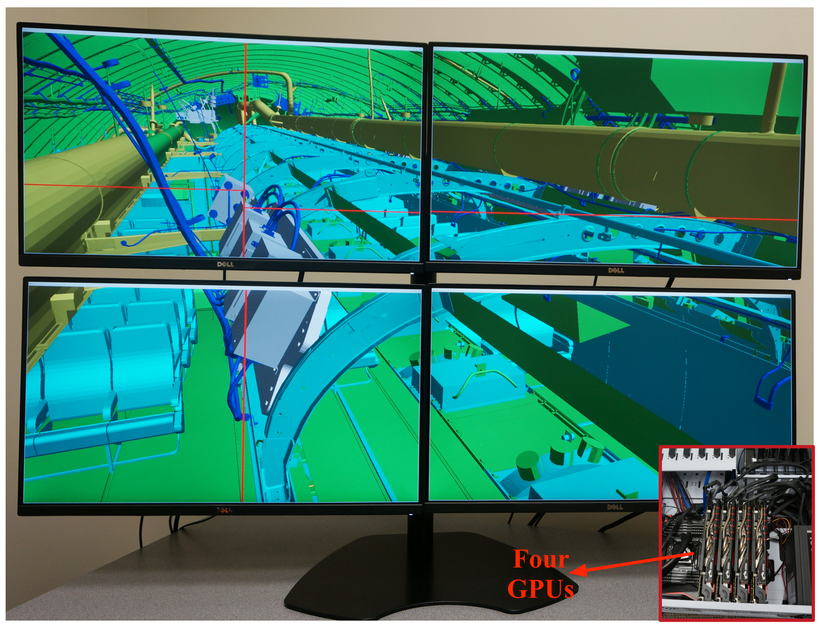
Abstract

Comparing to the desktop with a single GPU, a multi-GPU platform makes more video memory available across the GPUs, and results in a tiled large display by connecting each GPU to a display monitor. However, a multi-GPU platform may not always increase the overall rendering performance due to (1) imbalanced workload distribution among the GPUs, and (2) performance overheads caused by inter-GPU communication and synchronization.

In this work, a novel parallel load balancing approach is developed. It partitions the screen dynamically to determine the amount of vertices and triangles fetched into each GPU. GPUs communicate to each other only once per rendering cycle. The communication overhead is minimized by transferring only a small amount of image pixels rather than chunks of 3D data. Our approach is integrated seamlessly with the state-of-the-art GPU acceleration techniques including parallel mesh simplification, GPU out-of-core, and level-of-detail selection. As a result, our multi-GPU solution achieves real-time rendering performance on multiple displays for the CAD model composed of hundreds of millions of vertices and triangles.

System Overview (illustrated with two GPUs)



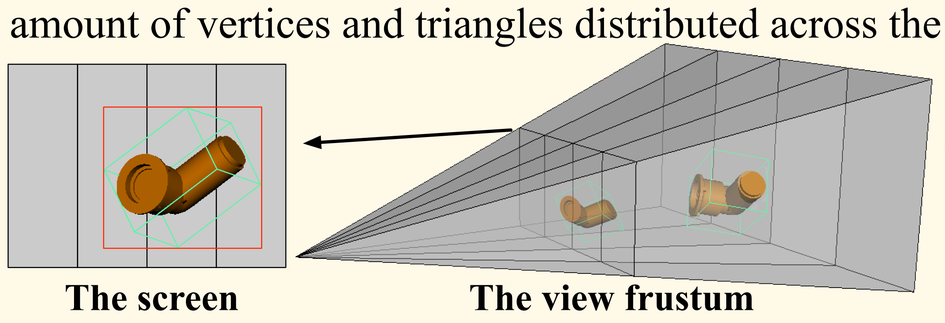


Research Novelty

- **Parallel mesh simplification:** The goal is to select a portion of vertices and triangles that can fit into GPU memory to construct an appropriately simplified version of the CAD model, which will result in a good visual appearance from distance. Our parallel algorithm speeds up the performance by removing the data dependency in edge-collapsing operations that exist in the traditional sequential simplification algorithms.

38.0M / 25.0M **26.6M / 17.5M** **15.1M / 10.0M** **7.4M / 5.0M**
(# of triangles / # of vertices)

- **Parallel load balancing:** A *frustum strip* parallel scheme is developed to partition the screen into regions and balance the amount of vertices and triangles distributed across the GPUs.



The screen **The view frustum**

Four *frustum strips are* generated by evenly splitting the screen.

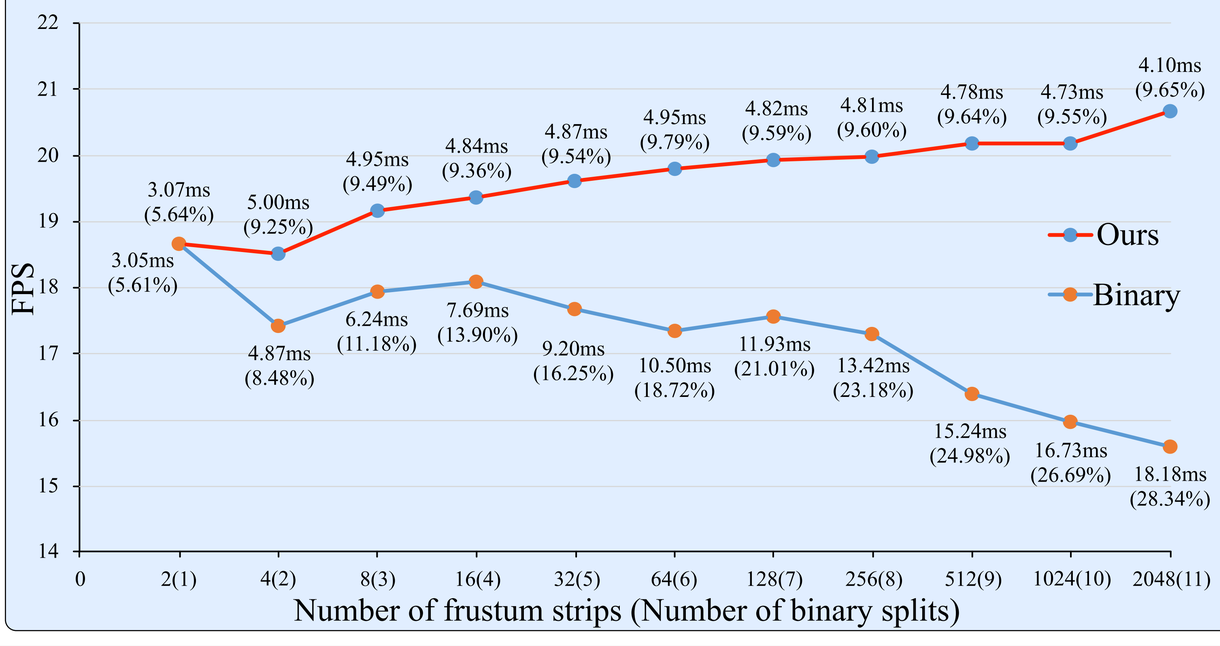
- **Frame exchange:** An efficient inter-GPU communication scheme is developed to exchange the rendered frame among the GPUs.

Implementation

We built a workstation with an Intel 3.50 GHz CPU (12 cores), 64GB RAM, and four Nvidia GTX980 Ti 6GB GPUs. The testing software was implemented on the 64-bit Linux Mint MATE 18.1 system using C++, MPI, CUDA and OpenGL.

Performance Comparison

We compared the FPS between the approach with our parallel load balancing algorithm and the approach with a binary screen partitioning load balancing algorithm. In the figure, each data point is labeled with the algorithm execution time. The value in the parenthesis is the percentage of the algorithm execution time taking out of the total time.



Performance Breakdowns

Configurations		FPS	# of LOD selected triangles (million)	# of transferred triangles to each GPU (million)	Runtime processing components (millisecond)					
Vertex budget (million)	# of GPUs				LOD Selection	Load Balancing	GPU Out-of-Core	Triangle Reformation	Rendering	Frame Exchange
20	1	18.74	27.62	0.45	4.80	—	22.93	2.59	27.14	—
	2	19.99		0.35 0.30	4.08	2.70	16.41	1.51	13.29	11.92
	4	20.67		0.30 0.27 0.27 0.24	5.40	4.10	16.19	0.94	11.40	9.81
40	1	14.86	46.89	0.52	4.84	—	30.81	4.03	32.90	—
	2	16.76		0.45 0.39	3.92	2.73	20.77	2.27	15.79	16.41
	4	17.50		0.41 0.37 0.36 0.34	5.54	4.13	19.44	1.46	13.17	13.94

Conclusion and Future Work

We developed GPU-based parallel algorithms to support the interactive rendering of large CAD models. We demonstrated a load-balanced out-of-core rendering application on a quad-GPU platform. Our approach fully utilizes distributed GPU memory. Advancing the state-of-the-art technologies such as SLI and Equalizer, our approach does not require data to be replicated across the GPUs. In the future, we will evaluate our approach on a cluster system which contains more than four GPUs.

NSF

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE