Sort-First Parallel Rendering with a Cluster of PCs

Rudrajit Samanta, Thomas Funkhouser, Kai Li, and Jaswinder Pal Singh
Princeton University

The objective of our research is to investigate whether it is possible to construct a *fast* and *inexpensive* parallel rendering system leveraging the aggregate performance of multiple commodity graphics accelerators in PCs connected by a system area network.

We investigate a sort-first approach [1, 2, 3]. As shown in Figure 1, our system comprises of a *client PC*, P *servers* PCs, and a *display* PC arranged in a three-stage pipeline. During the first stage, the client partitions the screen into P tiles and assigns each of them to a different server. During the next stage, every server renders all the graphics primitives at least partially overlapping its assigned tile from a replicated scene database to form a subimage and sends the resulting pixels to the display PC. Finally, the display PC composites all the subimages in its frame buffer for display.

The main challenge is to develop a dynamic screen partitioning algorithm that balances the rendering load among the servers, minimizes overheads by reducing primitive-tile overlaps, and executes at interactive rates.

Our method is based on the mesh-based adaptive decomposition (MAHD) algorithm described by Mueller [2]. Since it is not possible for the client PC to transform and sort every graphics primitive among the tiles, we group primitives into coarse-grained *objects*. The client partitions the screen based on the predicted rendering costs of these objects, while the servers perform the detailed sorting for each image using the bounding volume hierarchy as they render objects. The key idea is that the client and servers share the costs of sorting graphics primitives among tiles, with the major portion of the sorting done in parallel by the servers.

To test the effectiveness of our sort-first approach, we have implemented a prototype system

using a cluster of 18 Pentium III PCs equipped with $250 Hercules 3D Prophet graphics accelerator cards. The PCs communicate over a Myrinet network, which delivers approximately 100MB/s peak bandwidth with 13μs one-way latency. Results from our experiments show that it is possible achieve good speedups (8.1) on a cluster with up to 16 server PCs (See Figure 2). Surprisingly, the limited communication performance of the PC cluster is not the major bottleneck, but rather traditional overheads of parallel rendering limit the scalability of our system. In particular, client processing time grows linearly, server overheads grow linearly due to increasing primitive-tile overlaps, display PC bandwidth becomes a bottleneck beyond 25 frames per second, and the memory capacity of each server limits the size of the 3D model. Alleviating these limitations is a focus for future research.

Overall, we conclude that PC clusters provide an attractive, lower-price alternative for high-performance rendering. By building a working prototype system, we have demonstrated that a sort-first algorithm can work for moderate numbers of processors, and that a PC cluster architecture can be an effective way to combine multiple graphics accelerators together.

References

[1] Steve Molnar, Michael Cox, David Ellsworth, and Henry Fuchs. A sorting classification of parallel rendering. *IEEE Computer Graphics and Applications*, 14(4):23–32, 1994.

[2] Carl Mueller. The sort-first rendering architecture for high-performance graphics. *ACM SIGGRAPH Computer Graphics (Special Issue on 1995 Symposium on Interactive 3-D Graphics)*, 1995.

[3] Rudrajit Samanta, Jiannan Zheng, Thomas Funkhouser, Kai Li, and Jaswinder Pal Singh. Load balancing for multi-projector rendering systems. *SIGGRAPH/Eurographics Workshop on Graphics Hardware*, August 1999.

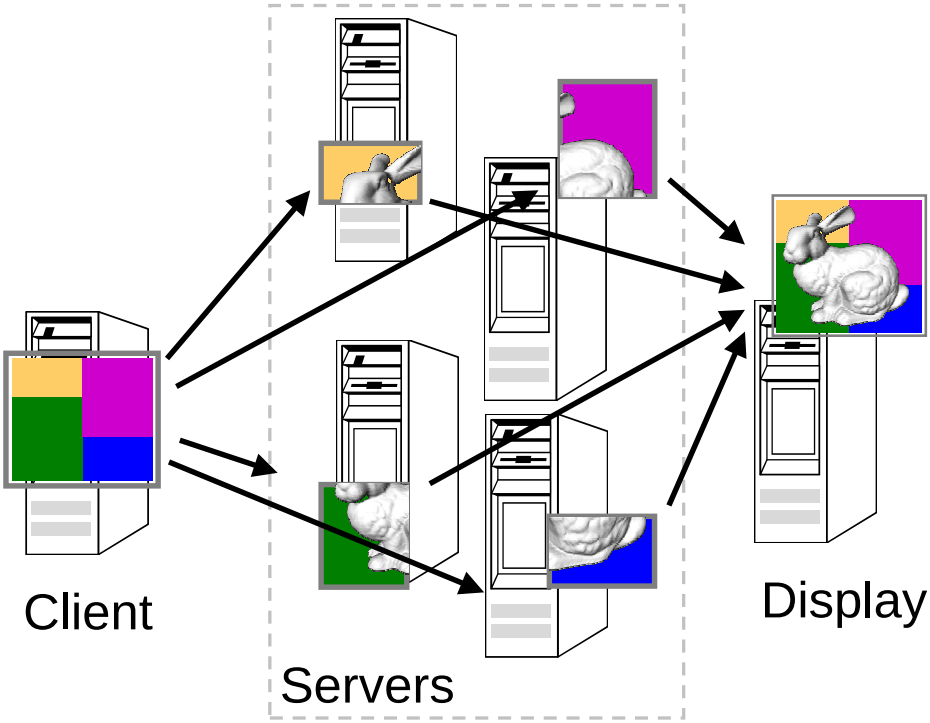

Figure 1: System architecture.

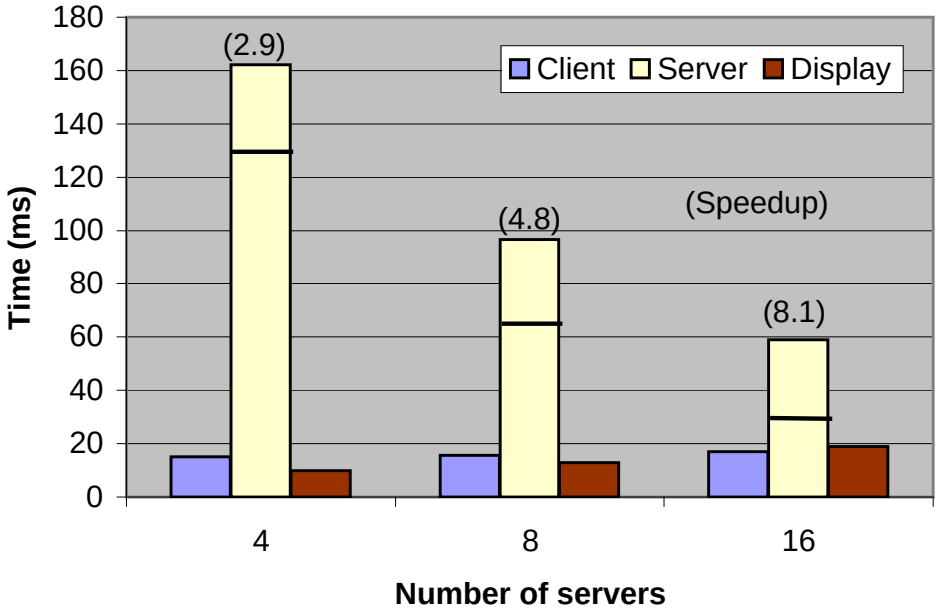

Figure 2: Timing results gathered during tests with a polygonal model representing a skull from the Visible Human Project (2,111 objects and 355,511 polygons). The height of each bar represents the time (in milliseconds) taken by each of our three pipeline stages. Server bars are split into two parts : productive rendering (below) and overheads (above). Speedups are indicated above each bar.