LARGE DISPLAYS FOR COLLABORATIVE VISUALIZATION

Michael Shafae, College of Engineering, California State University, Fullerton, 800 N. State College Blvd., Fullerton, CA 92831, 714-278-3291, mshafae@fullerton.edu

ABSTRACT

Large displays are considered to be multi-megapixel computer displays that are often built from commodity information technology hardware. This class of display technology has a great number of applications due to the relative low cost, high pixel resolution, and the associated computing power attached to such displays. There are palpable benefits when pairing interactive visualization on large displays with collaborative decision-making. The problem with such systems lie in designing software that can take advantage of the unique architecture of this type of computer system. This paper first discusses the architecture and application of these graphics clusters and then proposes an avenue of research for creating interactive visualization applications and analyzing their performance.

INTRODUCTION

Networking the common personal computer lets individuals digitally communicate with one another over a local area network; exchanging data and coordinating work. Linking people together over the network lets individuals work together within an enterprise to tackle tasks too large for any one given individual. Similarly, the same personal computers can be organized into a large parallel computer that uses the aggregate power of each individual processor to solve computational tasks that are too daunting for any single computer. This type of computer architecture is commonly referred to as a cluster.

A cluster is a collection of computers linked together over a data network, which enables the computers to divide up and coordinate their work. Traditionally, clusters are treated as batch processing computers. Gordian tasks are given to a cluster which then slices off manageable pieces and delegates that piece to a node within the cluster. As a node within the cluster solves its piece, the partial result is stored and a new task is delegated to that node. In the end, the partial results are combined to compose the solution to the initially large task. A general overview of cluster architecture and programming can be found in [1] and [2].

Computer clusters were designed to turn a collection of modest performance computers into a single high-performance computer by using a high-speed network to link the individual computers together. Over the past few years, there has been a revolution in the computational power of desktop graphics processors. The graphics processing unit in desktop computers, commonly referred to as GPUs or graphics cards, have more internal memory bandwidth than the host computer they reside in and are increasing in performance at a rate greater than that of the central processing units. The high-speed networking and the GPUs can be put to use to create a high-performance graphics computer that can overcome typical graphics bottlenecks such as pixel-fill rate and geometry processing leading to rendering highly detailed geometric models on large multi-megapixel display at interactive frame rates.

PREVIOUS WORK

The previous work can be divided into two categories. The first is the work with respect to the design of the graphics rendering architecture. In other words, the design of the computer system and how the input is partitioned, rendered and displayed. The second is the interface between the system and the users.

Since the computer is a composition of many stand alone computers, the traditional desktop and mouse motif breaks down.

There has been a rich history of research that investigated how to use a cluster to batch process graphics, but relatively little work has been done investigating interactive rendering on clusters. With interactive rendering on a graphics cluster, the differences in the system's architecture must be respected in order to gain optimal performance. Unlike traditional computer system architecture, the 'read-back' performance is hindered by the relatively slow network connection linking the computers together. In a traditional computer system, reading back a value from memory takes nanoseconds, whereas on a cluster the network latency is in the microseconds. This difference in speed motivates a style of computation called stream processing. Stream processing is based on defining a series of operations to be applied for each element in the stream; in this case the stream is the graphics instructions. The programmer defines a directed acyclic graph of stream processing units that can inject, draw, or filter the contents of the stream. WireGL [3] and Chromium [4] are two systems, which take stream processing approach, however no attempt is made to equitably balance the work-load across the nodes of the cluster.

What is needed is an analysis of how the workload should be partitioned to give some limited guarantee of all elements within the cluster being used effectively. In [5], [6], and [7] a simulated and real graphics cluster are used and the authors devise several load balancing and work-load partitioning schemes which attempt to address the types of situations which severely impact the performance of a graphics cluster. However the work completed was limited to a sub-display aligned task partitioning. This limits the flexibility of the task decomposition since the granularity decreases or increases depending on the size of the display.

A more complete survey of graphics cluster rendering is [8]. The authors look at a number of different software platforms for graphics clusters and compare them according to their performance and ease of use on a four-way graphics cluster.

Interacting with a large display is a separate challenge all together. Some have adapted the traditional desktop and mouse motif to varying degrees of success. Problems faced with the desktop and mouse motif is that the mouse's movement scales poorly to large displays. Furthermore, the mouse and desktop motif works well when used by a single user, but has limited usability in a multi-user setting. A number of pen and touch based user interfaces intended for multiple users are being experimented with in both industry and academia. However it remains a challenge in all interactive systems to deliver the user driven events to the appropriate node within the cluster.

METHODOLOGY

The objective is to have a graphics cluster attached to a large-scale display which can render its display at such a rate that would enable users to interactively visualize data or collaborate over extremely high resolution images. In addition, the objective is to have the task decomposition to be decoupled from the physical sub-displays unlike the previous work.

The problem faced is how coarsely to divide the workload over the nodes of the cluster. On the one hand, very fine task decomposition enables fine control regarding how much work each node undertakes. However this ultimately leads to more time spent decomposing the task into fine kernels rather than time spent solving the larger task. On the other hand, very coarse task decomposition can be done where there will be errors in assigning the work as well as duplication of effort amongst the nodes.

The benefit is that this sort of task assignment can be done quickly. The best course of action is a middle ground between these two extremes that accounts for and scales according to the size of the display (in pixels), the number of processors in the cluster, and the workload overlap between processors.

The first step in this research is to develop a simulator that will provide a test bed to help find that middle ground. Although simulations will need to be validated in a real-world cluster, simulations will give insight into how well the proposed load balancer will scale. Secondly, it is more economical to simulate a large cluster than it is to build one. This is an approach similar to the one taken in [5] but specifically geared to load balancing where the task decomposition is not aligned to the display's sub-displays.

Once the relationship between the size of the display, number of processors and workload overlap is better understood in simulation, the next step is to validate the observations on a prototype graphics cluster. Rendering software similar to [4] will be adapted for this project's rendering.

Beyond these two objectives, future research may delve into studying the usability of different input methods given the distributed nature of a graphics cluster. An additional avenue of research is to look at how to map general-purpose computation onto the GPUs of a graphics cluster.

REFERENCES

[1] R. Buyya, editor. *High Performance Cluster Computing: Architectures and Systems*, volume 1. Prentice Hall, 1999.

[2] R. Buyya, editor. *High Performance Cluster Computing: Programming and Applications*, volume 2. Prentice Hall, 1999.

[3] G. Humphreys, M. Eldridge, I. Buck, G. Stoll, M. Everett, and P. Hanrahan. WireGL: a scalable graphics system for clusters. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 129–140. ACM Press, 2001.

[4] G. Humphreys, M. Houston, R. Ng, R. Frank, S. Ahern, P. D. Kirchner, and J. T. Klosowski. Chromium: A stream-processing framework for interactive rendering on clusters. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 693–702. SIGGRAPH, ACM Press, July 2002.

[5] R. Samanta, J. Zheng, T. Funkhouser, K. Li, and J. P. Singh. Load balancing for multi-projector rendering systems. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 107–116. ACM Press, 1999.

[6] R. Samanta, T. Funkhouser, K. Li, and J. P. Singh. Hybrid sort-first and sort-last parallel rendering with a cluster of PCs. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 97–108, Interlaken, Switzerland, 2000. ACM Press.

[7] R. Samanta, T. Funkhouser, and K. Li. Parallel rendering with k-way replication. In *Proceedings of the IEEE 2001 symposium on parallel and large-data visualization and graphics*, pages 75–84. ACM Press, 2001.

[8] O. G. Staadt, J. Walker, C. Nuber, and B. Hamann. A survey and performance analysis of software platforms for interactive cluster-based multi-screen rendering. In *Proceedings of the workshop on Virtual environments 2003*. ACM Press, 2003.