Scalable Vision-based Gesture Interaction for Cluster-driven High Resolution Display Systems

Xun Luo*
Office of the Chief Scientist R&D
Qualcomm Inc.

Robert V. Kenyon†
Electronic Visualization Laboratory
University of Illinois at Chicago

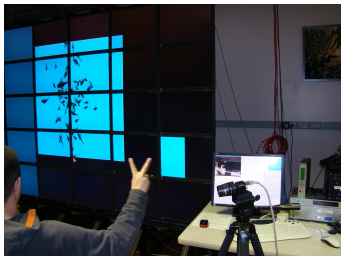

Figure 1: A user interacts with a display wall using gestures, for 2D window resizing and moving tasks. Frame images are captured by a single camera on the tripod but processed by multiple nodes of the cluster driving the display wall.

ABSTRACT

We present a coordinated ensemble of scalable computing techniques to accelerate a number of key tasks needed for vision-based gesture interaction, by using the cluster driving a large display system. A hybrid strategy that partitions the scanning task of a frame image by both region and scale is proposed. Based on this hybrid strategy, a novel data structure called a scanning tree is designed to organize the computing nodes. The level of effectiveness of the proposed solution was tested by incorporating it into a gesture interface controlling a ultra-high-resolution tiled display wall.

Index Terms: H.5.2 [Information Interfaces and Presentation]: User Interfaces—Input Devices and Strategies; D.1.3 [Programming Techniques]: Concurrent Programming—Parallel Programming

1 INTRODUCTION

A successfully implemented solution to obtain multi-million-pixel resolution is to construct display systems that consist of multiple screen tiles driven by a computer cluster. Such display systems include projection-based walls, flat panel-based walls, and flat panel-based tabletops. For many applications that run on these displays, vision-based gesture interaction could be advantageous compared to instrument-based counterparts under certain circumstances. However, vision-based gesture interaction faces the performance challenges for low latency and high throughput of frame image processing. A promising way to address these performance challenge is using scalable computing techniques to accelerate the frame image processing, without loss of interactivity.

In this paper we describe the scalable computing techniques for vision-based gesture interaction that utilize the computing power inherent in cluster-driven displays. The processing of captured video frames from a single camera is parallelized by multiple nodes of the cluster. Consequently, we are able to achieve a lower latency and a higher throughput for frame image processing. Figure 1 shows a prototypical vision-based gesture interface in use by a display wall. Being an application-level solution, this work complements other research in the literature that uses computer cluster to speed up frame image processing at the programming model level, such as the RPV environment [2] and the FlowVR middleware [1].

*e-mail: xun.luo@ieee.org
†e-mail:kenyon@uic.edu

IEEE Virtual Reality 2009
14-18 March, Lafayette, Louisiana, USA
978-1-4244-3943-0/09/$25.00 ©2009 IEEE

2 DESIGN AND IMPLEMENTATION

2.1 The Hybrid Task Partitioning Strategy

To take advantage of the better load balancing inherent in by-region task partitioning, and lower the computation overhead as much as possible by exploiting the characteristics of by-scale task partitioning. A hybrid approach is proposed as follows:

1. The task partitioning is a two-stage process. The first stage breaks down the workload with the by-scale strategy. Large workloads at high scale levels are further partitioned in the second stage which uses the by-region strategy.

2. In the first stage, each cluster node is assigned to process the whole frame image at a single scale level, or within a certain scale level range: a) The workloads at small scale levels which are less computationally demanding are grouped and assigned to a single cluster node. In this way, the light workloads are aggregated to avoid under-utilized cluster nodes. b) A cluster node which has been assigned to scan at a large scale level further partitions its workload using the by-region strategy, and assigns the processing of sub-regions to several additional cluster nodes. In this way task partitioning enters the second stage with the by-region strategy. c) If the workload of scanning at a certain scale level is comparable to those at either aggregated small scale levels or partitioned large scale levels, it is assigned to a single cluster node.

3. The partitioning process is completed when an optimized overall system performance is achieved.

2.2 The Scanning Tree

Figure 2 gives a graphical presentation of the scanning tree data structure which manages the cluster nodes. The scanning tree can have two or three levels. Every node of the tree represents a cluster node and is indexed with a unique key, which is its IP address. Each edge of the scanning tree maps to a duplex network link. A node of the scanning tree has two data attributes: scan_region and scale_range. The scan_region attribute is a quadruple $[X_{left}, X_{right}, Y_{top}, Y_{bottom}]$ that specifies the unscaled sub-region coordinates to be scanned in the frame image. The scale_range attribute is a triple $[S_{start}, S_{step}, S_{stop}]$ that denotes the start, step and stop of scanning scales performed by the node on its scan_region.

3 EVALUATION RESULTS

A gesture interface is implemented using the OpenCV library, with KLT feature-based template comparison. A baseline implementation that does not use the scalable computing techniques is also implemented for evaluation purposes. The gesture interface is then

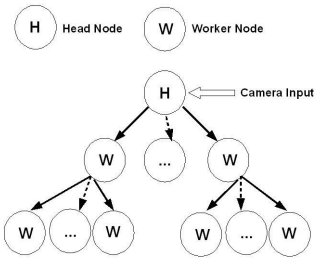

Figure 2: The scanning tree.

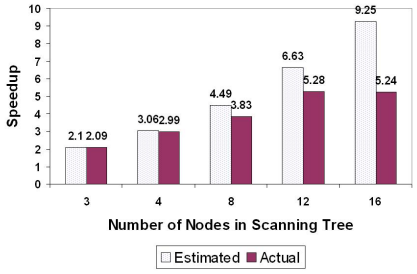

Figure 3: Estimated vs. Actual Speedup Values.

integrated with an ultra-high-resolution tiled display wall – LambdaVision – at the Electronic Visualization Laboratory at University of Illinois at Chicago (EVL-UIC). The wall has 55 LCD screen tiles, and is driven by a 32-node cluster. Each node in the cluster has a 64bit architecture with two 2GHz AMD processors and 4GB RAM. Nodes are interconnected with gigabit network interfaces. A graphics streaming middleware SAGE[1] is used by the display wall for frame rendering and UI management. The head node of the computer cluster is connected to a Dragonfly camera. The Dragonfly captures frame image at 640×480 resolution, 30 Hz rate and 8-bit gray scale. Communications among the scanning tree nodes are handled by the open source library QUANTA. The gesture interface runs as a standalone application and communicate with SAGE through socket connection.

A set of experiments are conducted to measure three groups of metrics: speedup values under different scanning tree configurations, workload balance across nodes in the scanning tree, as well as performance impacts to the graphics streaming.

3.1 Speedup

Five scanning trees are constructed that has 3, 4, 8, 12 and 16 nodes respectively. The estimated speedup values using method described in Section 2.1 are derived without taking account of communication costs. Figure 3 shows the actual speedup values measured for the five scanning tree configurations. The measured values are very close to the estimations for the 3-, 4- and 8-node scanning trees. For 12- and 16-node scanning trees the discrepancies are relatively large due to communication costs.

3.2 Load Balancing

Figure 4 illustrates the workload distribution estimated by the method described in Section 2.1 and the actual measured numbers, with an 8-node scanning tree configuration. Although the numbers in the top and bottom plots are not directly comparable due to different units used, it can be seen that the actual workload distribution

[1]SAGE and the later mentioned QUANTA software are both developed by EVL-UIC.

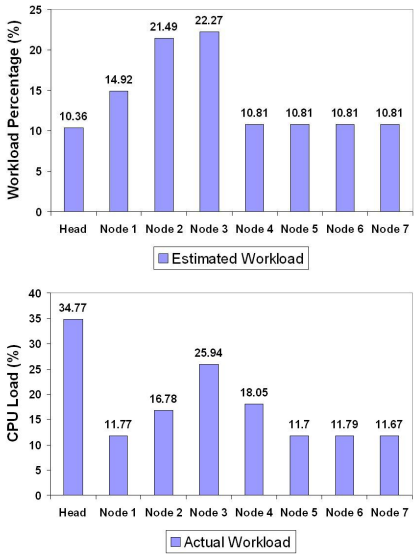

Figure 4: Estimated and actual workload distributions across a 8-node scanning tree. Note that the units used by top and bottom plots are different.

roughly follows the estimated outcome, with the exception at the head node. The larger readings at the head node is because that the head node is also in charge of video capturing, whose CPU usage is not easily separated when doing application-level profiling.

3.3 Performance Impacts

The OpenGL "Atlantis" application is run over SAGE, with the application window covering 20 display tiles. The profiling tools that come with SAGE is used to measure its display bandwidth and display frame rate. Two conditions are tested, i.e. with and without the gesture interface running. For the "with gesture interface running" condition, a 12-node scanning tree is used for parallel processing. Without gesture interface in place, the display bandwidth is $485.54 \pm 59.74 Mbps$. While with gesture interface running, the display bandwidth is $430.77 \pm 49.84 Mbps$. Display bandwidth drops by 11% when the gesture interface is active. Similar results are observed on display frame rates. Without the gesture interface on the frame rate is 69.21 ± 5.26. With gesture interface active, frame rate becomes 61.03 ± 6.44. The frame rate decrease is about 12%. Considering the gain of a five-fold speedup for gesture detection and recognition, the performance impacts to graphics streaming are relatively insignificant.

4 Conclusion and Discussion

To our best knowledge, this paper is the first to address scalable vision-based gesture interaction for large display systems. The solution can be further improved by compressing the frame image distributed across scanning tree nodes. We have already completed this enhancement and will publish the results in a subsequent paper.

References

[1] J. Allard and B. Raffin. Distributed physical based simulations for large vr applications. In *IEEE Virtual Reality Conference*, Alexandria, USA, March 2006.

[2] D. Arita, Y. Hamada, S. Yonemoto, and R. ichiro Taniguchi. Rpv: A programming environment for real-time parallel vision - specification and programming methodology. In *IPDPS '00: Proceedings of the 15 IPDPS 2000 Workshops on Parallel and Distributed Processing*, pages 218–225, London, UK, 2000. Springer-Verlag.