HIGH END VISUALIZATION WITH SCALABLE DISPLAY SYSTEM

Dinesh M. Sarode*, Bose S.K.*, Dhekne P.S.*, Venkata P.P.K.*,
Computer Division, Bhabha Atomic Research Centre, Mumbai, India

Abstract

Today we can have huge datasets resulting from computer simulations (CFD, physics, chemistry etc) and sensor measurements (medical, seismic and satellite). There is exponential growth in computational requirements in scientific research. Modern parallel computers and Grid are providing the required computational power for the simulation runs. The rich visualization is essential in interpreting the large, dynamic data generated from these simulation runs. The visualization process maps these datasets onto graphical representations and then generates the pixel representation. The large number of pixels shows the picture in greater details and interaction with it enables the greater insight on the part of user in understanding the data more quickly, picking out small anomalies that could turn out to be critical and make better decisions. However, the memory constraints, lack of the rendering power and the display resolution offered by even the most powerful graphics workstation makes the visualization of this magnitude difficult or impossible. The initiative to develop high end visual environment at Computer Division, BARC explores how to build and use a scalable display system for visual intensive applications by tiling multiple LCD displays driven by the Linux based PC graphics-rendering cluster. We are using the commodity off-the-shelf components such as PCs, PC graphics accelerators, network components and LCD displays. This paper focuses on building an environment which render and drive over 20 millions of pixels, using the open source software framework. We describe the software packages developed for such a system and its use to visualize data generated by computational simulations and applications requiring higher intrinsic display resolution and more display space.

INTRODUCTION

The visualization is essential in interpreting the large, dynamic data generated from computer simulations and sensor measurements. The visualization processes maps the datasets onto graphical representations and then generate the pixel representation. When the graphical representation containing billions of polygons is mapped to pixels provided by the conventional display device then many polygons fall on the same pixel and are contained within a pixel. This may lead to aliasing or even missing small data features contained in these tiny polygons that could turn out to be critical. If we have a high resolution display, then the large number of pixels shows the picture in greater details and interaction with it enables the greater insight in understanding the data. However, the memory constraints, lack of the rendering power and the display resolution offered by even the most powerful graphics workstation makes the visualization of this magnitude difficult or impossible.

While the cost-performance ratio for the component based on semiconductor technologies doubling in every 18 months or beyond that for graphics accelerator cards, the display resolution is lagging far behind. The resolutions of the displays have been increasing at an annual rate of 5% for the last two decades. The ability to scale the components: graphics accelerator and display by combining them is the most cost-effective way to meet the ever-increasing demands for high resolution. This paper focuses on building the scalable display system which render and drive over 20 millions of pixels. We discuss the system architecture, hardware and software environment for the system. We also discuss the software packages developed for the system and its use to visualize data.

SYSTEM ARCHITECTURE

Unlike the most scalable display systems today, which use high-end graphics machines and high-end projectors, our system is built with low cost commodity components: a cluster of PCs, PC graphics accelerators and LCD displays. In our architecture, a display is constructed by tiling LCD displays. To reduce the seams between the tiles, we removed the outer casing of the LCD displays and tiled them in a custom designed frame. A single workstation drives a single display; system software controls rendering the portion of graphics apportioned to that display. A separate workstation of the cluster is used to control the entire tiled display. (Figure 1)

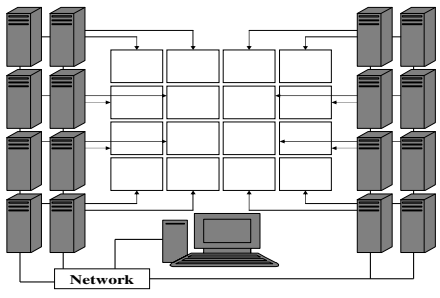

Figure 1: A schematic representation of the system

*{dinesh, bose, dhekne, panikv}@barc.ernet.in

The commodity projectors can also be used for seamless tiled display. The projectors are costly as compared to LCD monitors and they are hard to align**.** The brightness, color coordination, balance and edge blending also needs to be addressed. While tiling LCDs, the borders of the LCDs (seams) provide a grid-like pattern etched on the display, in which we can't display anything. The seams introduce issues while visualization of the data. Our software environment treats the borders as if represented by hidden pixels thereby creating the "panned window" effect.

HARDWARE

We designed the first 2x2 tiled LCD display system that provides a display resolution of 2048 x 1536 (3.1 million) pixels. Each tile is 15″ ViewSonic VE510+ display with maximum resolution of 1024x768. A PC Linux cluster of five workstations running Red Hat 7.2 is used to drive the display system. The workstations are Intel P-IV 1.7 GHz processor, 256 MB RDRAM and 3Dlabs GVX1 pro AGP graphics accelerator card. Communication between them is handled over the Fast Ethernet.

We then scaled the display to 4x4 by tiling 16 17″ ViewSonic VE 700 LCD displays driven by cluster of 17 (16+1) workstations running Red Hat 7.2. Fig. 2 shows the schematic representation of high-resolution display system. The resolution of our new system is 5120x4096 (20 million) pixels. In our scaling efforts, we used two commodity interconnects: Gigabit Ethernet for graphics related communication and Fast Ethernet for file sharing etc.

SOFTWARE ENVIRONMENT

Moving the visualization applications to scalable display system makes software environment much more complex than the shared memory multi processor multi pipe systems. In the PC rendering cluster there is no single place to share the data, the programmer must know the cluster configuration and provide ways to effectively share data through interconnecting network.

The tiled display system must also fulfil three functional requirements: *genlock*, *swaplock* and *datalock*. *Genlock* synchronizes the video frames from each node of the cluster so that final image presented is coherent. It is critical for active stereo on tiled display. Each graphics card in a cluster renders different number of polygons resulting in different rendering times for each frame. *Swap lock* ensures the frame buffers swaps are synchronized. *Data lock* further ensures the synchronization of the views to maintain consistency across the tiles. We have not used any explicit *genlock* for tiled display whereas *swaplock* and *datalock* has been implemented in the system software.

System Software

The software architecture for the system is based on open source Chromium and DMX. Chromium provides the mechanism that allows many existing OpenGL based graphics applications to use the rendering capabilities of the cluster and the resolution offered by tiled display without any modification [4]. The other X11 applications requiring more display space and resolution use DMX (Distributed Multihead X) infrastructure to run on the tiled display system [6]. We developed the graphical control panel software to manage the entire system and to automate time consuming tasks and functionalities often required by users while working with the tiled display system.

Chromium

Chromium is a flexible framework for distributed-rendering on cluster of computers, initiated at Stanford University. The rendering pipeline consists of three conceptual stages: geometry database (graphics primitives), geometry processing (transformation, clipping, lighting etc) and rasterization (scan-conversion, shading and visibility). The graphics accelerator cards of the cluster provide multiple rendering pipelines. The decision to split up and recombine the rendering workload can happen before or after either of the geometry processing or rasterization stage. Thus we can have *sort-first*, *sort-middle*, and *sort-last* architectures for distributed rendering [5].

The *sort-first* architecture distributes the graphics primitives early in the rendering pipeline (geometry processing) to the rendering node that can do the remaining work. This is achieved by dividing the display into tiles and making rendering nodes responsible for all rendering calculations that affect their respective tiles. The screen-space bounding box of the primitive determines the tile into which it falls. The primitives are then distributed over an interconnect network to the appropriate rendering nodes. In *sort-middle*, primitives are distributed arbitrarily to geometry processing units and then screen space primitives are redistributed in the middle of the rendering pipeline to the appropriate rasterizer responsible for a portion of the display screen. The *sort-last* architecture assigns arbitrary subsets of the primitives to the rendering nodes which rasterize into pixels values no matter where they fall in the screen. The rendering nodes then transmit these pixels over an interconnect network to compositing nodes which resolve the visibility of the pixels according to the alpha or Z-buffer entries of the pixels.

The main advantages of *sort-first* are relatively less communication requirements, can deal well with both large number of primitives and large number of pixels. The *sort-last* approach is scalable but requires an image composition network with very high bandwidth. The *sort-middle* approach is very difficult in a cluster-of-PCs system; we can't break the rendering pipeline as it is implemented in hardware.

Chromium implements sort-first, sort-last and hybrid algorithms. It works by replacing the native OpenGL library with its own. It directly operates on the stream of the OpenGL graphics commands issued by the

application. It provides Stream Processing Units (SPU). Each SPU has its input - streams of graphics commands, perform some operation on these commands and passes them on. SPUs can be chained to perform combined operations. The SPUs for tiled rendering are *tilesort, render* and *pack*. *Tilesort* SPU implements the sort-first algorithm. The *tilesort* SPU sorts the OpenGL commands into tiles so that they are packed and sent over a network to the nodes handling the tiles. *Render* SPU passes the stream to the node's local OpenGL implementation. *Pack* SPU packs the stream into a buffer for transmission to cluster servers.

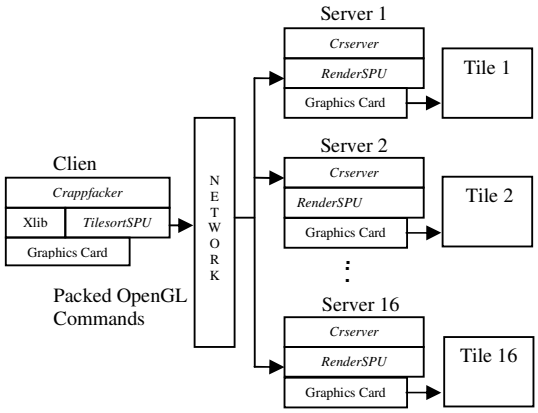

Figure 2: Sort-First configuration

A python script called *mothership* controls running programs through Chromium. It contains the tile layout, SPU chains and nodes used for running the application. The *mothership* runs an event loop waiting for communication from the *crappfakers* and *crservers*. The *crappfaker* will run the OpenGL program and relink it against the defined SPU libraries. The *crserver* is used for tiled rendering and any SPU that requires network access for data transmission. The Figure 2 shows the *sort-first* configuration used by our system.

DMX (Distributed Multihead X)

DMX is an open source framework, distributing X Window sessions across the nodes of a tiled display such that the X display/desktop can span many individual tiles. It is used to run any program that runs under X Windows on the tiled display. The program can use the entire display space and resolution offered by the tiled display [6].

It works by providing a front-end X server acting as a proxy to a number of back-end X servers running on the nodes of the cluster driving tiled display. X clients will connect to front-end X server running on the control node of the tiled display. Rendering and input requests will be accepted by the front-end server and then will be sent appropriately to back-end servers.

Graphical Control Panel

The graphical control panel provides functionality to start (login) and stop (logout) X windows on all rendering tiles, reboot / shutdown the entire cluster or individual nodes, changing display power management, viewing the system information of the cluster nodes such as process tree, memory, cpu, network utilization. It also provides the interface for running scripts that are quite useful for cluster control such as parallel copy when changes are needed across the cluster and parallel execution for performing simple tasks on a cluster. The control panel also continuously monitors the status of the cluster nodes. The software is implemented in two parts. One part handles the entire GUI and runs on the Control PC, which controls the tiled display. Second part implements the functionalities provided by the software which are invoked remotely on the nodes of the cluster.

APPLICATIONS

The tiled display-aware software packages available for visualization, display of images and animations currently includes:

AnuVi

CFD visualizations with high-resolution grid enhance the quality of the visual information. Interacting with large CFD data sets in real time and providing rich and deep visual feedback is achieved only with rendering cluster and tiled display.

To achieve this we have developed AnuVi, the Post-Processor and Scientific Visualization Framework that run on our scalable system through chromium and DMX as well as on desktop. It is an open source software system for 3D computer graphics and visualization supports a wide variety of visualization algorithms including scalar, vector, tensor, texture and volumetric methods. It is capable of delivering visualizations for other disciplines also like Structural, Thermal and Medical imaging. It is based on the open source libraries of VTK and WxWidgets.

The Reactor Safety Division of BARC has initiated computational simulation of tsunami generation, its propagation and finally run up evaluation for protection of public life, property and various industrial infrastructures for the coastal regions of India. Our scalable display system and AnuVi helped in visualizing this huge simulation data. AnuVi was used to generate about 200 high resolution images from the simulation data. We then generated a movie in streaming media format (SM). The movie was played on the system through DMX by using the Blockbuster – an open source movie player [9]. The movie is revealing and enables the user to get greater insight as it displays huge simulation data which includes multiple time steps at the native resolution (20 million pixels) of tiled display. (see Fig. 3)

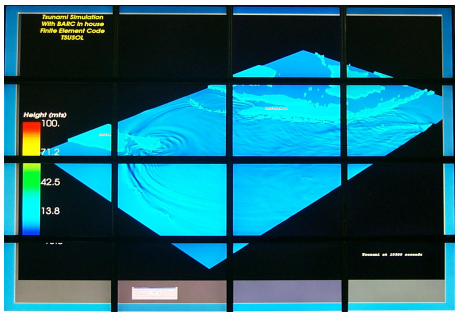

Figure 3: Visualization of tsunami simulation data

CollabCAD

CollabCAD is a java based distributed 2D&3D CAD/CAM and CAE software system being developed by the Computer Aided Design Group of National Informatics Centre, New Delhi [7].

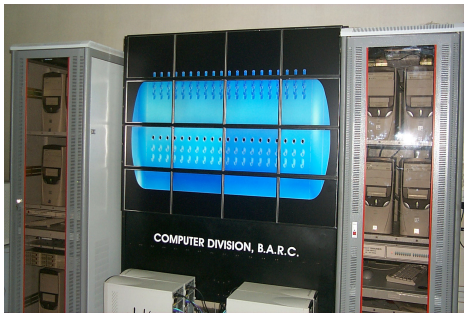

Figure 4: CollabCAD on tiled display

We have augmented the display capabilities of CollabCAD with our scalable display system which is extremely useful for visualizing huge CAD models. It provides detailed view of the individual components of the model and as well as overall interrelationships often required for design reviews. The CollabCAD is also configured to run through chromium. Figure 4 shows one such model.

Tiled MPEG/AVI movie player

The precomputed MPEG/AVI movie is a powerful way of visualizing complex structures or simulations. We developed tiled movie player that plays the MPEG/AVI movie at the native resolution of the tiled display. It is an MPI (Message Passing Interface) application where the master reads the mpeg/AVI movie, decodes it and divides it into several sub-streams. It then redistributes the sub-streams to the slaves. The slaves convert the sub-stream OpenGL textures, which are mapped to the respective tiles managed by the slaves, thus forming the entire frame at the native resolution of the tiled display.

Tiled Image Viewer

It is useful for displaying large-format images, particularly those containing many millions of pixels. The typical display does not have sufficient resolution, requiring scrolling or image shrinking which results in loss of context and lack of details.

CONCLUSION

Our high-end visual environment with commodity components and open source software is a reasonable alternative to multiprocessor, multipipe systems. The advantages are low cost and technology tracking. The large field of view (FOV) coupled with 20 times more pixels than the standard display provides the deep and rich visual experience. We found that the approach is scalable, works well with various computing and display configurations. The chromium and DMX support the scalable display system and hide its complexity from OpenGL as well as X11 applications. Many visual-intensive applications that display large-scale datasets or many simultaneous datasets benefit from it.

ACKNOWLEDGEMENTS

We would like to thank all the people who have contributed for building the scalable display system, its software base and encouraging us in developing the visualization applications.

REFERENCES

[1] Visualization Research with Large Displays,Bin Wei, Claudio Silva, Eleftherios Koutsofios, Shankar Krishnan, and Stephen North,IEEE Computer Graphics and Applications, July/August 2000

[2] Software Environments for Cluster-based Display Systems,Yuqun Chen, Han Chen, Douglas W. Clark, Zhiyan Liu, Grant Wallace, and Kai Li, Proceedings of the 1st International Symposium on Cluster Computing and the Grid, 2001

[3] Deep View: High-Resolution Reality, J. T. Klosowski, P. D. Kirchner, J. Valuyeva, G. Abram, C. J. Morris, R. H. Wolfe, and T. Jackman, IEEE Computer Graphics and Applications, May/June 2002.

[4] Chromium: A Stream Processing Framework for Interactive Rendering on Clusters of Workstations , Greg Humphreys, Mike Houston, Ren Ng, Randall Frank, Sean Ahern, Peter Kirchner, and James T. Klosowski, Computer Graphics SIGGRAPH 2002 Proceedings

[5] A Sorting Classification of Parallel Rendering, S Molnar, Michael Cox, David Ellsworth, Henry Funchs, IEEE Computer Graphics and Applications, 1994.

[6] http://dmx.sourceforge.net/

[7] http://www.collabcad.com/

[8] http://public.kitware.com/VTK/

[9] http://blockbuster.sourceforge.net/