Cluster-based solution for virtual and augmented reality applications

Antoine Tarault, Thomas Convard, Patrick Bourdot, Jean-Marc Vézien

Cluster based solution for virtual and augmented reality applications

Antoine Tarault
LIMSI-CNRS
BP 133
F-91403 ORSAY CEDEX, FRANCE
+33 (0)1-69-85-81-70
tarault@limsi.fr

Thomas Convard
LIMSI-CNRS
BP 133
F-91403 ORSAY CEDEX, FRANCE
+33 (0)1-69-85-80-32
convard@limsi.fr

Patrick Bourdot
LIMSI-CNRS
BP 133
F-91403 ORSAY CEDEX, FRANCE
+33 (0)1-69-85-81-73
bourdot@limsi.fr

Jean-Marc Vézien
LIMSI-CNRS
BP 133
F-91403 ORSAY CEDEX, FRANCE
+33 (0)1-69-85-81-67
vezien@limsi.fr

ABSTRACT

This paper presents a cluster based architecture designed for an augmented reality interface. It can manage and synchronize video flows as well as deformable objects coming from several computers, on top of OpenSceneGraph. It was also designed to be easy of use and lies on commercial software called RTMaps®, which allows a graphical programming of the cluster based application. We present this system applied to an augmented reality interface for remote driving called SACARI.

Keywords

Augmented Reality, graphical Cluster, Scene graphs

1. INTRODUCTION

In recent years, PC platform have become the most used architecture to develop Virtual and Augmented Reality (V&AR). However, the hardware of a PC is not powerful enough to support high-end application such as a display in a full immersive environment. Indeed, this kind of displays, such as CAVE [1] or Workbench [2] needs several stereoscopic projections. One solution would be to integrate multiple graphic boards in a single PC. This approach, exploited in NVidia® SLITM and Alienware® Video Array™, is unfortunately not yet validated for VR applications. The only remaining solution is to use a PC cluster to display a VR application in several screens. Compared to a shared memory solution, this approach is far less expensive and more performing, but needs more software developments.

The latest software have allowed to use such PC clusters, but they are uneasy of use and not always tuned for specific applications.

Our system allows the distribution of a V&AR application over the nodes of a cluster, each part of the application being a "component" that can be placed in a graphical interface.

The different constraints of this work were:

1. Easy of use: the different components of the system should be easily integrated
2. Modularity: each part of the system should be tested separately
3. Genericity: the system should support Virtual Reality applications as well as Augmented Reality applications

2. RELATED WORK

There are three main approaches in the literature for cluster graphics, each corresponding to a level of distribution: the application level, the scene graph level and the graphical commands level.

The approach of Chromium [5] (formerly WireGL [6]), is to broadcast OpenGL commands emitted by the master to the slaves. This approach was developed in order to perform tiled rendering, but is also viable for multi wall VR rendering systems. The main advantage of this method is that desktop applications can be ported to PC clusters without any modification of their codes, by tuning Chromium configuration files. But in this kind of architecture, the slave graphical nodes are the replica of the master, except for the point of view of the scene and the projection. The slave nodes cannot integrate data coming from other cluster nodes to their scene. The scene cannot be different from the master node in any way. That means that most of the load of the application is on the master node. Another backdraw is that for the moment, such architecture doesn't support adaptative stereo.

Another option is to replicate the VR application on each node of the cluster (Figure 1). This is the approach chosen in Net-Juggler [3], a cluster extension for VR-Juggler [4]. In this case, applications must share input events and must be synchronized at each frame buffer swap. This is particularly efficient when an application was already developed with VR-Juggler. However, this scheme is the opposite of a true distributed system: for a complex application, resource consuming tasks will be replicated "as is" on each node of the cluster.

The last approach, the closest to ours, is to replicate only the scene descriptor (also known as scene graph) over the PC cluster. This architecture, developed in Syzygy [7] and Avango [8] enables a single master application to manage and share synchronized updates of a scene with slave nodes using a distributed scene graph.

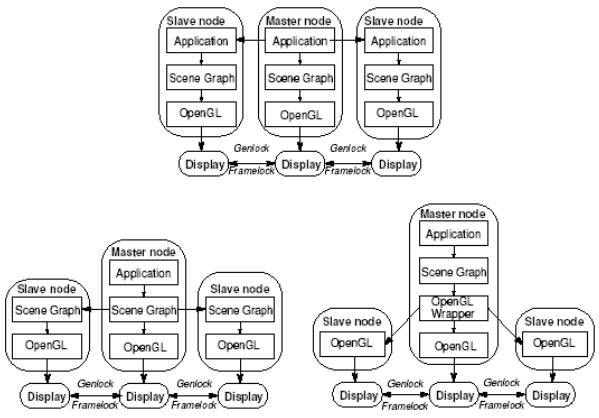

Figure 1. (top) Application replication approach. (left) scene graph distribution approach. (right) OpenGL command broadcasting approach.

Our system can be defined as being between these two last approaches. You can tune it to a totally applicative level, or to a level closest to scenegraph level, depending on the need of your application and features of the cluster.

3. ARCHITECTURE

Our cluster based system is composed of two main products: RTMaps® and OpenSceneGraph. Our immersive device is biplane, but the system is extensible to any number of faces. We have developed a *package* that uses OpenSceneGraph for RTMaps®, and we have extended this package for it to support a *CAVE* multi-displays environment.

We chose to use a master-slave architecture for our system. All the information is collected on the master node: 6 DOF trackers for the wand and the head, and data related to the scene graph. Each slave node is a graphical node connected to a face of the CAVE (see fig. 2). The master node communicates with the slave node via a gigabit network.

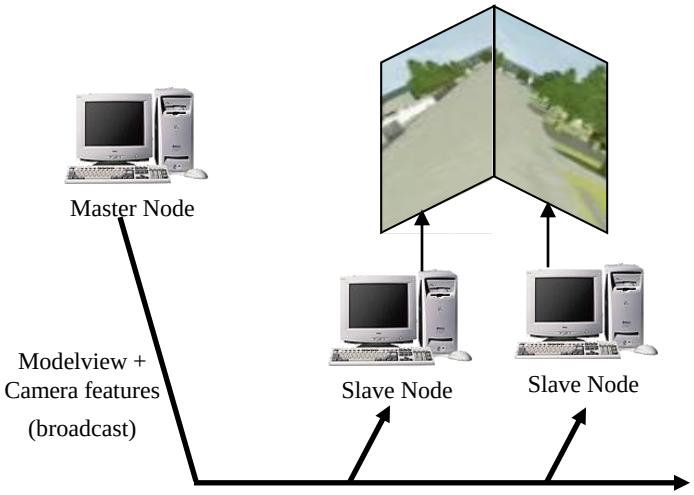

Figure 2. Our 2 face CAVE architecture

3.1 Graphical Data

There are two main kinds of graphical data that are exchanged between the master node and the slave nodes.

First, the *Modelview* matrix and the virtual camera features are transmitted. The *Modelview* matrix corresponds to the position and orientation of the virtual camera in the scene. This matrix is transmitted to each slave node, for them to know the position of the virtual camera. Then a transformation proper to each slave node is applied depending on the position of the screen towards the main camera (see fig. 3).

Other features of the camera are transmitted:

- Stereo features (stereo mode (anaglyph, quad buffer), screen distance, fusion distance, eye separation)
- Frustum

Then, stereo features and frustum can be adjusted in real time.

The *Modelview* matrix and other camera feature are synchronized in the slave nodes with the master Node, together with the update of the scene, for the refresh of the screens to be synchronous.

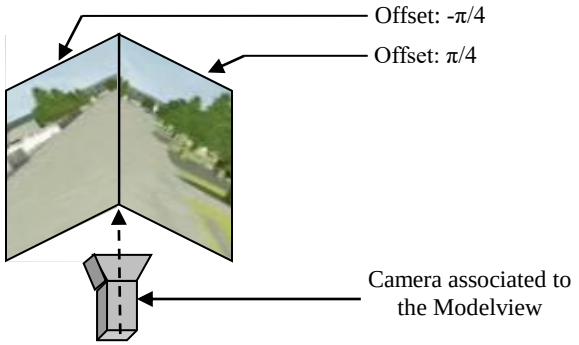

Figure 3. Distribution of the Modelview to a 2 faces CAVE

3.2 Distribution of the peripheral data

To collect the peripheral data, we use a software developed by the LIMSI-CNRS: VeServer [9]. It is a real time client/server architecture that can drive synchronously the peripheral he is in charge with. It can collect data from numerous devices (ARTTrack tracker, MotionStar tracker, ViaVoice voice recognition software, …), coming from different computers. We use that software to take back peripheral data to the master node. The mouse is managed separately, its position on the OpenSceneGraph window being directly related to this last software. Clicks, movements and wheel rolling of the mouse are transformed to RTMaps® events. We multicast all the peripheral data that are useful to the slave nodes, in a FIFO buffer with the RTMaps® software (see fig.4). By useful, we call data that are relevant for the slave node such as a mouse click that changes the state of a node of the graph. Not useful data are those that only affect the camera position, for example.

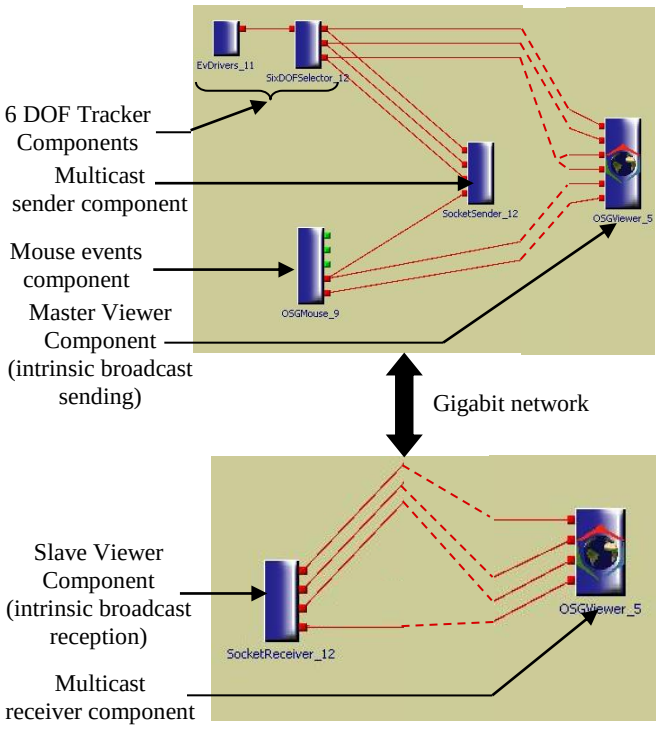

Figure 4. Distribution of the different date to the slave Node

3.3 Distribution of the Video Data

Another challenge in designing a cluster architecture for an augmented reality application is the distribution of the video stream. In such a system, we have two separate video streams, one corresponding to the left eye, and one corresponding to the right eye. To resynchronize left and right eye, each image is attached to a Timestamp, corresponding to the capture time of the two images. The images can be sent from the remote site to the slave nodes with a classical JPEG compression.

4. APPLICATIONS

We have tested this architecture on two kinds of applications. First, a car driving simulator, then, a remote stereo video visualization.

4.1 The car driving simulation

This application is a simulation of the driving of a semi-autonomous vehicle. Several data must be transmitted from the master node to the slave nodes (see fig. 5):

- The path of the vehicle, chosen by the remote driver
- The last position of the next trajectory
- The orientation of the vehicle at the end of the next trajectory

This data is only coming from a 6DOF tracker. The mouse events for starting/stopping the simulation are only used by the master node, because those data only affect the position of the camera.

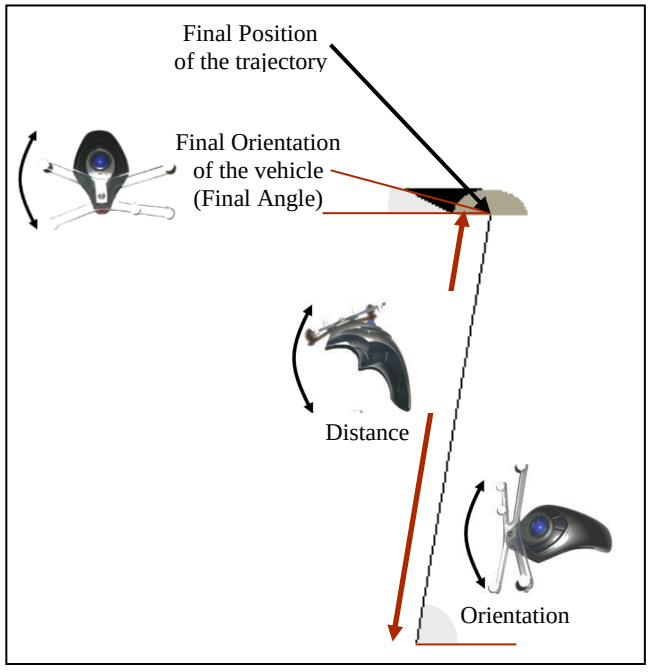

Figure 5. Data associated to the supervision of the vehicle

This example allow discovering the first design rule in such a cluster architecture: graphical data must be totally decorrelated with other data. In this case, the path of the vehicle must only be dependant of the data directly concerning it: distance, orientation, final angle, and calculated path. The distributed diagram is then the one presented figure 6.

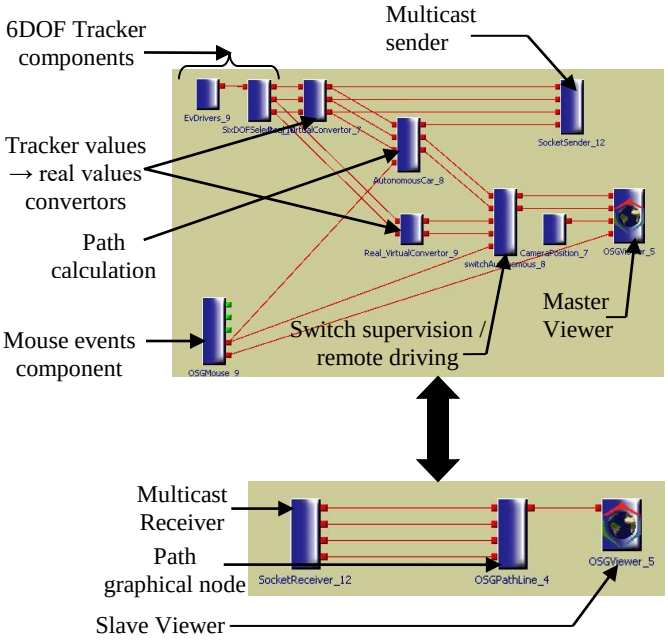

Figure 6. RTMaps® diagram for the driving application

We can see that the graphical data specific to the slaves are only duplicated on the slaves node. That's why this system is fully distributed: each part of the cluster has specific computing. Master node is in charge with path calculation, collision of the vehicle with the ground, peripheral data grabbing, and camera position calculation. Slave nodes are only in charge with graphical nodes, which are not necessary on the master node but for possible monitoring.

Tests for this applications where made with Performer® town (see fig. 7).

Figure 7. Remote driving / supervision application

System benches were measured on a cluster system composed of one master node and two slave nodes equipped with NVidia Quadro NVS 400 graphic cards for genlocking and framelocking

capabilities (see table 1). The network theoretical bandwidth is 1 Gbits/s (700 Mbits/s in practice). Cluster nodes run on Microsoft Windows operating system (see table 1).

Table 1. System benches

Configuration	Performances
Single	
1 Master + 2 Slaves	

4.2 Remote Stereo Video Application

The purpose of this application is to transmit a stereo video stream to a remote user. The stereo camera is mounted on a turret for the remote user to see most of the remote environment.

The main advantage of this system is that is specifically designed for video capture and transmission through a network. Video is captured in the remote site via a RTMaps® component that we developed. A timestamp is associated to each pair of frame. This video is transmitted to each slave node, because as we said before, data that is only relevant for graphics is directly transmitted and computed by the slave nodes. In this application not only video is necessary for the slave nodes, but also position of the head, for the texture to be moved on the CAVE walls (see fig. 8). In this application, no camera's position broadcasting is necessary, because the video texture doesn't depend on the virtual camera position, but on the head orientation.

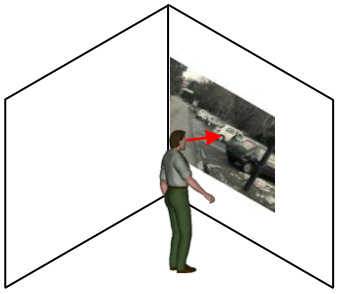

Figure 8. Remote stereo video principle

5. CONCLUSION

Our system has proven to be really easy of use, modular, and generic, supporting VR applications as well as AR applications. The main backdraw of the system is that each application has a specific distribution on the master and the slaves. But it is also its force: we have a truly distributed system, which can be tuned depending of the performances of each node of the cluster.

Further developments of this system could be to manage adaptative stereo, which is a changing stereo depending on the position of the user. For the moment, we only consider the user being at one specific point. Such a system should be easily integrable, because we currently transmit all the camera features to each node of the graph.

6. REFERENCES

[1] Cruz-Niera, Sandin, D., DeFanti, T. Surround-Screen Projection-Based Virtual Reality : The Design and Implementation of the CAVE. In *Proceedings of SIGGRAPH '93,* Anaheim, CA, 1993.

[2] Kruger, W., Bohn, C., Frohlich, B., Schuth, H., Strauss, W., Weshe, G. *The Responsive Workbench: A Virtual Work Environment.* IEEE Computer 28, 1995, 42-48.

[3] Allard, J., Gouranton, V., Lecointre, L., Melin, E., Raffin, B. NetJuggler : Running VRJuggler with MultipleDisplays on a Commodity Component Cluster. In *IEEE Virtual Reality Conference '02*, Orlando, Florida, 2002, 273

[4] Bierhaum, A., Just, C., Hartling, P., Meinert, K., Baker, A., Cruz-Neira, C. VRJuggler, A Virtual Platform for Virtual Reality Application Developpment. In *IEEE Virtual Reality Conference '01,* Yokohama, Japan, 2001

[5] Humphreys, G., Houston, M., Ng, R., Frank, R., Ahern, S., Kirshner, P. D., Klosowski, J.T. Chromium: A Stream Processing Framework for Interactive Rendering on Clusters. ACM Transactions on Graphics. In *Proceedings of SIGGRAPH '02,* San Antonio, Texas, 2002, 693-702

[6] Humphreys, G., Eldridge, M., Buck, I., Stoll, G., Everett, M., Hanrahan, P.: WireGL: A Scalable Graphics System for Clusters. In *Proceedings of SIGGRAPH '01*, Los Angeles, California, 2001, 129-140

[7] Schaeffer, B., Goudeseune, C. Syzygy: Native PC Cluster VR. In *Proceedings of IEEE Virtual Reality Conference '03*, 2003, 15-22

[8] Tramberend, H. Avango: A Ditributed Virtual Reality Framework. In *Proceedings of IEEE Virtual Reality '99,* JW Mariott Hotel, Houston, Texas, USA, 1999

[9] Touraine, D., Bourdot, P., Bellick, Y., Bolot, L. A framework to manage multimodal fusion of events for advanced interactions within Virtual Environments, *8th Eurographics Workshop on Virtual Environment (EGVE '02),* Barcelona, 2002