Frame Lock Synchronization for Multiprojection Immersive Environments based on PC Graphics Clusters

MARCELO P. GUMARÃES¹, PAULO A. BRESSAN², MARCELO K. ZUFFO
Av. Prof. Luciano Gualberto, 158 – Trav. 3 – Butantã
CEP: 05508-900 – São Paulo – SP – Brazil
Laboratory of Integrated Systems – Polytechnic School – USP
Tel: (+55 11) 3091-5254 – Fax: (+55 11) 3091-5374
{paiva, pbressan, mkzuffo}@lsi.usp.br

Abstract: The availability of PC clusters is an attractive alternative to effectively implement the computational infrastructure of Multiprojection Immersive Environments (MIME), such as CAVE Environments, Panoramas and Power Walls. These virtual reality systems require a good frame synchronization performance (frame locking) in order to offer coherent visual matching among screens rendered by different PC nodes on the cluster. This synchronization is usually based on hardware methods, commercially available on some graphics adapters. In this paper we propose and implement software oriented approaches to support frame lock synchronization based on Common Object Request Broker Architecture (CORBA). The frame lock synchronization software is running on the Pleiades, a six-node PC cluster, which is part of CAVERNA Digital, a five-sided CAVE.

Keywords: multiprojection environment, frame lock, graphic cluster.

1. Introduction

Advances in research on network communications and innovations in personal computer performance make clusters of low-cost commodity components appealing. It has made practical the use of PC cluster to solve many problems historically addressed by massively parallel systems [1,2]. These applications make use of combined computational power of several resources using a low-cost solution. However, to solve graphic problems, a new category of PC cluster is emerging, the PC graphics cluster.

PC graphics cluster differs from PC cluster in their intent. A PC cluster takes a large problem and breaks it up into smaller components, distributes the problems to nodes on the cluster which solves each of their assigned tasks, usually offline and batch-oriented, and synchronizes all the information at the back end. In a PC graphics cluster, the intent is to produce images (frames) of the visual data set. The interactive or real-time rendering in typical graphics applications requires that a PC graphics cluster complete its entire task in a few milliseconds, making latency a challenging issue [2]. The main difficulty of PC graphics cluster consists in provide a coherent, seamless, contiguous display from its distributed visual components [2]. To solve this, the synchronization with tight latency limitations is essential to each node in PC graphic cluster to update its image when the others end.

¹ Supported by Instituto Adventista de Ensino.

² Project supported by Fapesp #99/12693-1 (Brazilian Research Foundation).
Specifically, there are three levels of synchronization:

- **Video signal synchronization** controls the signal that drives the output to a display system on each node;

- **Dynamic data synchronization** makes that each node determine what to draw or give the changing data set information to each node; and,

- **Frame lock synchronization** makes that each node updates its image just when the others finish.

This project follows several works found in literature. In [3] is proposed DICElib, a socket based (TCP/IP) library matching the requirements of low processing time and high speed, also for PC clusters. This library implements synchronization and synchronous data sharing among the nodes and may create new variable types, but it has few functionalities when compared to other communication tools.

In [4] some CORBA implementations (ORBacus and TAO) are compared to others message passing libraries (MPI and PVM), and they show similar or better communication performance to some kinds of benchmarks. In [5] an encapsulated MPI code with CORBA object is used in virtual reality applications, which ones implement a light simulation and visualization tool using VRML and Java.

The main objective of this paper is to evaluate the frame lock synchronization software based on Common Object Request Broker Architecture (CORBA) approach. The research was carried out using a six-node PC graphics clusters, called Pleiades. The software solution was proposed because nowadays the frame lock synchronization is offered just by hardware and is commercially available on few graphics adapters. The proposed solution can be used in virtual reality systems to synchronize points of view. For example, the avatar parameters can be broadcasted for all nodes in PC cluster with the frame lock software.

In section 2 is presented the hardware frame lock synchronization used in recently graphic cards, like 3DLabs WildCat. The section 3 presents the utilization of CORBA in MIME environments, its advantages and features. Then section 4 presents the proposed solution for frame lock synchronization by software, to resolve the “tearing” effects on the screens. After, section 5 shows the performance evaluation realized on Pleiades cluster. Finally, the conclusions, acknowledgments and references in section 6, 7 and 8, respectively.

2. **Hardware Frame Lock Synchronization**

A PC graphics cluster to offer many frames in the same time can use the hardware frame lock synchronization solution. That is possible using the external synchronization with graphic cards, like the architecture presented in Figure 1.

Besides, the technique called “double buffering” is being used, which consists of the frame lock synchronization solutions displays the contents of one buffer while it draws the next frame into a second buffer (non-visible). When it finishes drawing this new frame the buffers can be swapped, either immediately or on receipt of some trigger signal. So the content of the new buffer is displayed while the next frame into the original buffer is drawn [2]. In MIME, it is necessary for the buffer to swap on all
displays of the systems simultaneously, otherwise “tearing” effects will appear across screen boundaries.

To synchronize, there are extra cables with two signals named “Done” (or “Ready”) and “Release” that interconnect the nodes between the graphics cards. The functionality is very simple. Following the Figure 1, when one slave node finishes its image and receives the signal “Done” of its right node, it sends the “Done” signal to the left node. If it is the last node on right side, it just sends “Done” signal when finishes. When the master node, the most left node, receives the “Done” signal, it responds with “Release” signal and performs a buffer swap. Each slave node receives the “Release” signal of the left side, replaces to right side and performs a buffer swap.

3. CORBA for MIME

There has been earlier attempts to standardize the cluster related technologies like message passing (MPI), shared memory (OpenMP) and communication protocol (VIA) to achieve the integration, performance and portability among the applications. But the development of MIME requires more features like object management, communication, support for graphics library, and others.

In this way, we are using CORBA to software development in PC cluster graphics. CORBA is the open standard defined by the Object Management Group (OMG), based on object-oriented model and supporting software development in distributed and heterogeneous environments. The CORBA simplifies the application development process by eliminating the task of writing tedious and error-prone communication code [6]. For this, it defines services and supports necessary to applications that will be carried in a distributed environment. The communication platform, or ORB (Object Request Broker), is the middleware that establishes the client server relationships between objects [7].

In spite of MIME applications seem obvious, the problem is about its communication performance. CORBA is always criticized regarding its intrinsic heavy protocols. Certainly, the early implementations of CORBA were not powerful since quality of service, and efficiency was not integrated in the standard specification. However, recent implementations of CORBA are more robust and real-time features are been addressed [8, 9].

The Figure 2 presents a frame lock synchronization example using the software solution with rendering algorithm based on OpenGL. This example was implemented using the proposed solution and illustrates two tiled images. Therefore others kinds of virtual reality systems are possible with simple changes of the points of view.
4. **Frame Lock Synchronization by Software based on CORBA**

The frame lock synchronization software implemented in this project uses the TAO, one ORB with real-time features. The TAO handles the transfer of messages from a client program to an object located on a remote network node. The TAO hides the underlying complexity of network communications from the programmer. In the CORBA model, programmers create standard software objects whose member methods can be invoked by client programs located anywhere in the network. A program that contains instances of CORBA objects is typically known as a server [10].

When a client invokes a member function on a CORBA object, the ORB intercepts the function call. Also, the ORB redirects the function call across the network to the target object. The ORB then collects results from the function call and returns these to the client.

The Figure 3 presents the frame lock synchronization architecture based on CORBA middleware. The slave nodes (Slave 1 and Slave 2) are servers, where the CORBA objects are implemented. The master (left node) is a client, which invokes the methods on the servers.

The frame lock synchronization by software solution is similar to the hardware solution, the difference is that the software solution does not use extra cables and the synchronization is done using messages on network. However, there is the command “Draw/Show” (draws and presents the frame) and “Done” (the frame was stored in
buffer) that is exchanged between the master and slaves. For each slave, one thread is created on the master.

The Figure 4 presents the steps followed by the frame lock synchronization application. In this example, the steps are:

1. In master node, one thread is started to each slave node. Each thread sends a message “Draw/Show” to its related slave node to starts the process.
2. When the slave node receives the message “Draw/Show” it renders the frame, storages in graphic buffer and sends the message “Done” to master node.
3. The master node stays locked until all the message “Done” are received, and then it restarts the procedure with step 1.

![Figure 4 - Software frame lock synchronization process.](image)

5. Performance Evaluation

The benchmark was realized on Pleiades graphics cluster, where was considered all six nodes. Each cluster node consists of a Dual Pentium III 933 MHz, 1 GB RAM and graphic card. The network is connected by an optical Gigabit-Ethernet (1 Gbit/second) switch. During the test, the computers were only running involved programs and the network had not others traffic beyond synchronization messages. All nodes of PC cluster were running Windows 2000 operating system.

The benchmark was the following: given a vector of chars, broadcast the vector and synchronize the applications one thousand times. We ran this test for vector sizes in range of 4 Kbytes and 4 Mbytes, and the results are showed in the Figure 5, 6 and 7. The synchronization time was considered the total time between the messages “Draw/Show” go to slaves and the messages “Done” go back to master.
The tests were produced considering one computer like master and others like slaves. The master computer represents the machine responsible by the managing cluster in virtual reality applications, such as input/output equipments and distribution tasks.

Two thresholds are important to MIME: 15 Hz and 60 Hz. The first one (15 Hz) is considered an acceptable measure to users interactive with applications. To realize 15 Hz the synchronization time should be on the maximum on 66 milliseconds. The second (60 Hz) is acceptable to stereo applications. To realize on 60 Hz the synchronization time should be on the maximum on 16 milliseconds.

![Figure 5 - Synchronization time.](image)

<table>
<thead>
<tr>
<th>Kbytes Quantity</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 slave</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>11</td>
<td>24</td>
<td>47</td>
</tr>
<tr>
<td>2 slaves</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>9</td>
<td>18</td>
<td>39</td>
<td>69</td>
<td>132</td>
</tr>
<tr>
<td>3 slaves</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td>11</td>
<td>21</td>
<td>47</td>
<td>85</td>
<td>163</td>
</tr>
<tr>
<td>4 slaves</td>
<td>6</td>
<td>7</td>
<td>9</td>
<td>12</td>
<td>19</td>
<td>38</td>
<td>71</td>
<td>151</td>
<td>289</td>
</tr>
<tr>
<td>5 slaves</td>
<td>9</td>
<td>11</td>
<td>14</td>
<td>19</td>
<td>32</td>
<td>62</td>
<td>114</td>
<td>240</td>
<td>438</td>
</tr>
</tbody>
</table>

![Figure 6 - Synchronization time to small messages.](image)
6. Conclusions and Future Work

In this paper, we described the frame lock solution software based on Common Object Request Broker Architecture (CORBA) to realize the coherent visual matching among screens rendered by different PC nodes on the cluster.

Depending on the virtual reality application 15 Hz or 60 Hz is required to update the screens. The tests showed that when the frame lock solution based on CORBA sends just a signal to all machine, like the hardware solution, we can get the synchronization to a Multiprojection Immersive Environments. However, if we use the same software to frame lock and data distribution it must respect the threshold of data and quantity of machines involved.

For the future, we propose to investigate other CORBA communication solutions to improve the scalability of the software, like asynchronous messages and communication services. Tests on others operational systems, heterogeneous network and the increase of the number of nodes are also considered a future project.

7. Acknowledgments

This project is in partly funded by Fundação de Amparo à Pesquisa do Estado de São Paulo, grant # 99/12693-1, with additional support from Intel Foundation and FINEP (Financiadora de Estudos e Projetos).

8. References


