PC Clusters for Virtual Reality

Bruno Raffin*and Luciano Soares †

ID-IMAG, CNRS/INPG/INRIA/UJF
Grenoble - France

Abstract

In the late 90's the emergence of high performance 3D commodity graphics cards opened the way to use PC clusters for high performance Virtual Reality (VR) applications. Today PC clusters are broadly used to drive multi projector immersive environments.

In this paper, we survey the different approaches that have been developed to use PC clusters for VR applications. We review the most common software tools that enable to take advantage of the power of clusters. We also discuss some new trends.

CR Categories: I.3.2 [Graphics Systems]: Distributed/network graphics; I.3.7 [Three-Dimensional Graphics and Realism]: Animation—Virtual reality; I.6.8 [Types of Simulation]: Animation—Distributed; D.1.3 [Concurrent Programming];

Keywords: Virtual Reality; PC Clusters

1 Introduction

PC clusters became popular in the mid 90's for batch processing of hight performance applications. The rendering farms are a possible example of batch processing. Today, PC clusters are very common in the top 500 supercomputer ranking [1]. Their success is related to:

- Their low cost, because they are mainly built of commodity components produced for a mass market ;
- Their modularity that enables to built a cluster adapted to the user's need regarding components, size or performance ;
- Their compliance with standards, that favors software and hardware interoperability ;
- The availability of a large range of open source software solutions that enables to customize, if required, a given software layer.

In the late 90's the emergence of high performance 3D commodity graphics cards opened the way to use PC clusters for high performance Virtual Reality (VR) applications. It was first motivated by the need to have multiple video outputs to drive multi projector immersive environments. One PC was not able to support multiple video outputs, via one or even several graphics cards, for high performance 3D graphics. The clustering approach was not new, as it was used in the early 90's to drive the first Cave with a cluster of SGI workstations [26]. Since these early uses, PC cluster technology and software approaches have deeply evolved, giving rise to different solutions with different advantages and drawbacks regarding performances, portability, ease of use, etc.

*e-mail:Bruno.Raffin@imag.fr
†e-mail:Luciano.Soares@imag.fr

In the following, we survey the different approaches that have been developed to use PC clusters for virtual reality applications. We first focus on parallel rendering, i.e. how to take advantage of the graphics power of a PC cluster to drive a multi projector environment. The solutions range from a duplication of the application on each node with the appropriate synchronizations and data communications to keep the different copies coherent, to graphics oriented parallelizations based on sort-first, sort-middle or sort-last paradigms [44]. We also discuss low level synchronization issues, i.e. swap-lock and gen-lock, in conjunction with stereo rendering constraints.

We put in perspective these different approaches with the evolution of the PC and network components. We also study the trends in video projector technologies as it influences the tasks PCs have to perform, like image blending. But PC cluster benefits go beyond parallel rendering. Since having several computer nodes enables to distribute tasks not directly related to rendering, taking advantage of extra computing or I/O capabilities. One issue is then to coordinate and distribute the different tasks on the cluster. We survey existing approaches from parallel scene graphs to sequential and parallel task coupling. An overview of the most advanced applications regarding their ability to take advantage of the potential of large PC clusters is also presented.

PC clusters basics are reviewed in section 2. Section 3 presents the image lock constraints that a cluster should comply with when driving a multi projector environment. Section 4 discuss the different levels of parallelism that may be present into a VR application. Section 5 reviews some of the most common software tools supporting clusters, while section 6 presents some hardware compositing devices. Display technology issues are discussed in section 7 before to conclude.

2 Basics

2.1 Hardware

A PC cluster is a set of interconnected PCs usually gathered in a single room and dedicated to process high performance parallel applications. Its architecture can range from low-end single CPU PCs connected through a Ethernet or Giga-Ethernet network to high-end multi CPU PCs connected through some high performance network like Myrinet or Infiniband.

In the recent years, the PCI-Express bus is probably the most important hardware evolution for PC clusters. It enables to significantly increase the data transfer bandwidth for two important components:

- The bandwidth of high performance networks like Myrinet was first limited by the PCI and next PCI-X busses rather than by their own technology. The availability of PCI-Express opened the way to very high bandwidth. For instance the Myri-10G cards from Myrinet reach a point-to-point bandwidth of 1.2 GBytes/s.
- The bandwidth for transferring memory to the graphics card also takes advantage of the PCI-Express bus. It has a higher

performance than the AGP bus in both directions. AGP 1X has a speed of 264 Mb/s, the last AGP version, AGP 8X has a speed of 2 Gb/s. PCI-Express speed depends on the number of lanes. Usually for graphics cards a x16 slot having a speed of 4Gb/s is used.

The 64 bits processor architectures (AMD opteron, Intel Itanium or 64 bits Xeon) enable to address beyond 4 GBytes of memory, the limit imposed by the 32 bit architectures. This is useful for memory intensive applications. Also, the 64 bits architectures are usually showing higher performance than their 32 bit counterparts. However experiencing a real application performance increase may not be straightforward. Part of the code may need to be reworked (especially with the Intel Itanium processor).

Graphics cards become more versatile as it is possible to program part of the graphics pipe-line through shaders to achieve high-quality and high performance final rendering results. As PCs usually provide several PCI-Express slots, it is now possible to install several graphics cards, while only one AGP slot was available. Also notice that most commodity graphics cards are now providing 2 digital video outputs.

2.2 Software

One difficulty with clusters is to install and maintain a coherent system on all computer nodes. A classical approach is to rely on a disk cloning mechanism. Then from time to time, when incoherences appear between machines, because one user has changed a configuration on a set of machines and forgot to propagate the change on all machines for instance, the node's disks are cloned from a reference computer.

Linux is the main operating system for clusters. Several Linux cluster distributions, like Rocks [2], provides tools to ease the installation and maintenance of clusters. They usually provide fast cloning mechanisms and some packages for classical cluster tools like the message passing library MPI or the cluster performance monitoring tool Ganglia [42].

Though these tools ease cluster management, the user still has to see a cluster as a set of PCs, each one running its on OS. On-going efforts are focused on developing single image operating systems for clusters, like Kerrighed [3]. The goal is to offer the view of a unique SMP machine on top of a cluster of standard PCs.

2.3 Clusters versus Dedicated Supercomputers

The difference between a PC cluster and a dedicated supercomputer is getting thinner (at least for small to medium size configurations). They are using the same CPUs and GPUs, high performance networks that show about the same performance. They are running the same operating system (Linux). The main difference comes from the vendor ability to deliver turn-key solutions with a strong hardware/software integration, validated configurations and a quality user support.

This was not always the case. For instance, Silicon Graphics (SGI) computers were very common to drive virtual reality facilities in the 90's. At that time these computers had graphics cards [45, 46] powerful enough to display high quality 3D content in real-time. They achieved parallel rendering through special busses (Triangle bus and after Vertex bus), not available on commodity graphic cards. The Onyx computers had a distributed memory with specific hardware to implement a cache coherency protocol on top of which was built a virtual shared memory [7]. The SGI's operating system, IRIX, is a single image OS.

Another important project in the field of parallel rendering and clustering is the PixelFlow [27]. It was developed to support parallel rendering in a complete dedicated hardware. It has an array of renders that compute a full image with a fraction of global primitives needed. At the end a high-performance image-compositing network combines the images from the array nodes in real-time. PixelFlow could be coupled to a parallel supercomputer for an immediate-mode rendering. It also worked in a retained-mode, the geometries being stored inside the PixelFlow hardware. For image compositing PixelFlow used binary trees. One important aspect about PixelFlow is its support for deferred shading in the compositing, enabling high-quality graphics without increasing image-composition bandwidth or redundancy.

3 IMAGE-LOCK

PC clusters are first used in VR to drive multi-projector environments. The first issue is then to ensure that the image streams displayed by the different projectors are coherent, even though they have been computed on different nodes: the images displayed with the different projectors should appear as a single high resolution image. This *image-lock* constraint can be decomposed into three synchronization levels, *gen-lock*, *frame-lock* and *data-lock*, presented in the following.

3.1 Gen-lock

Gen-lock ensures all video signals generated by the cluster are synchronized. Synchronization can occur at a pixel, line or frame level.

A frame level gen-lock is mandatory for active stereo (see section 7.2). If gen-lock is not properly ensured, the user may see from the same eye both, a right and a left eye image displayed by two different projectors. The quality of stereo is then affected.

Systems that do not use active stereo usually do not require gen-lock to obtain a good quality image.

3.2 Frame-lock

Frame-lock, also called swap-lock, ensures the images computed on each PC node are released at the same time, i.e. the buffer swaps are synchronized. Failing to ensure a proper frame-lock results in discrepancies, where images that are displayed at the same time correspond to different rendering steps.

3.3 Data-lock

The *data-lock* goal is to guarantee that the data used to compute the images for the different projectors are coherent. For instance, all images related to the same time frame must be computed from 3D objects that are at the same position or have the same color.

Data-lock is a complex issue that can be tackled at different levels of the application. This is discussed in section 4.

3.4 Implementations

3.4.1 Hardware

Commodity graphics cards usually do not support hardware gen-lock and frame-lock. Some high-end models do provide such features. The NVIDIA Quadro FX 3000G [8] and 3DLabs [9] implement these synchronizations relying on an additional daughter boards through a RJ-45 or DB9 chain. 3DLabs also enables to maintain a minimum swap period between the frames, allowing the application as a whole to maintain an acceptable frame-rate.

Springer et. al. [59] propose to modify commodity graphics cards to control the gen-lock. It is achieved by getting the gen-lock signal from a pin on the master node board, disabling each graphics card slave node gen-lock and feeding them with the gen-lock signal from the master node.

3.4.2 Software

A synchronization barrier, executed on a classical Fast-Ethernet network just before the buffer swap, is sufficient to ensure a proper frame-lock [14, 23, 56, 29].

Gen-lock, as it concerns video signal generation, is close to the hardware and then much more difficult to ensure with a software only approach. However some software approaches that do not require any hardware modification of the graphics card have been developed to implement a frame level gen-lock and enable active stereo on PC clusters.

Rather than to gain total control on signal generation, which would make the software deeply dependent on the graphics card specification, SoftGenLock [12] applies continuous small modifications to converge and maintain gen-locked video signals. The gen-lock signal is propagated along the different machines using the parallel port, a low latency device present on all PCs. It results in a software that only requires access to few specific registers on a graphics card: it can be ported with minimal effort on potentially any graphics card.

An other implementation, based on SoftGenLock [12] core code, has been developed at HLRS [64]. It supports the Linux Kernel 2.6.

4 Parallelism for VR: a Taxonomy

We can distinguish 4 main components more or less intricate in a VR application:

- *Inputs*: Inputs include static data stored into files as well as dynamics data captured by various sensors, like a tracker, a wall-clock time or random number generators.
- *Simulation*: A VR application can involve several simulations ranging from solid object collision detection algorithms to fluid simulations.
- *Animation*: The properties of the objects present into the scene are updated according to inputs and simulation results.
- *Rendering*: For users feedback, an output is computed and rendered on the appropriate devices. The most common outputs are images, sounds and forces.

The goal when using a cluster is to take advantage of the extra resources available to alleviate performance bottlenecks. To achieve this goal it is necessary to split processing amongst the different cluster nodes. There are basically two different approaches:

- *Data parallelism* where several instances of the same task are executed concurrently but on different datasets.
- *Task parallelism* where different tasks are executed concurrently.

Parallelism implies data communications and synchronizations to ensure a proper task coordination. In particular, data redistribution (or sorting) steps are required to make available to the target tasks data computed at source tasks. Depending on the nature and amount of data to be redistributed the cost can vary significantly. For instance, input events retrieved by a position tracker are limited to a few bytes, while graphics primitives can be significantly larger.

We discuss in the following the main approaches used for the different components of a VR application and the required redistribution steps.

4.1 Inputs

Task parallelism is the most common approach. Because input devices are usually independent, they can be distributed on different nodes [6]. For instance a tracker is executed on one node, while a wand is executed on a different one. This also enables to better use the connectors that are available on the different PCs.

Data parallelism is used in some cases for computation and data intensive input devices like multi-camera environments [66, 11].

4.2 Simulations

Task parallelism can be used to execute several simulations concurrently. Large simulations may be internally parallelized. For instance data parallelism is a classical approach for parallelizing fluid simulations. The space where the fluid can evolve is split into regular cells that are then cyclically distributed by blocks [15].

4.3 Animation

The data are distributed onto different nodes (data-parallelism), each node taking care of updating its data [54]. As some data may be replicated on several nodes, to save on communications, care must be taken to manage coherency between the different copies when a data is modified.

4.4 Rendering

Data parallelism is the main approach for rendering. Let us focus on graphics rendering. The standard taxonomy for parallel rendering distinguishes three broad classes, depending where parallelism occurs in the graphics pipeline [44]:

- *Sort-first*: Each task is assigned a sub-section (tile) of the entire image to render. Then each task processes independently the graphics primitives that project into its tile up to obtain the final image.

  This approach is very classical on clusters. Data distribution can occur at the input layer only, all subsequent tasks working locally for its tile. Because this scheme only requires to carry lightweight data over the network (input events), it is widely used for VR applications. However an important part of the data and computations (mainly simulations and animations) are repeated on each task, limiting the benefits of using a cluster to multi-projector rendering and input event capture parallelization. In the following we call this approach the *replication approach*. The other classical scheme is to have the application executed on a single node up to the generation of graphics primitives. Then primitives are distributed to different nodes, each one in charge of computing a tile. Thought this enables to avoid most of the replications of the previous approach, the performance are impaired by the cost of distributing the graphics primitives, which amount is proportional to the complexity of the scene.

- *Sort-middle*: Each task is assigned a set of graphics primitives that it processes up to the rasterization. Then, data are sorted according to the tile they belong to before rasterization is performed. Because commodity graphics cards integrate both geometry processing and rasterization without giving users the ability to retrieve data before rasterization, this approach is seldom used on clusters. A sort middle approach can be used on clusters when geometric transformations are performed on CPU [63].
- *Sort-last*: The geometry dataset is split and sent to different tasks. In a final step, the images computed by each task are redistributed for compositing. Image compositing can be performed by dedicated hardware reading the images from the video output [60], or through software solutions reading back the images in the frame buffer of graphics cards and using the cluster network to move the data [24]. The complexity is then proportional to the image resolution rather than to the scene. It is also easier to achieve load-balancing as there is no locality constraints on graphics primitives when they are initially distributed to the different tasks. This solution has been used mainly for scientific visualization where datasets tend to be very large [37].

5 Software Tools for Clusters

In this section we review some software tools for VR supporting PC clusters. Most of them are open source. This list is not exhaustive, but its covers the most common and advanced uses of parallelism into VR applications. Each tool is positioned according to the parallelism it enables. A complementary classification can be found in [61].

5.1 CaveLib

The CAVELib [49] was developed at the Electronic Visualization Lab to drive the first Cave [26]. Initial versions ran on a a cluster of SGI machines, using a replication approach. Some fixed data, such as a navigation matrix and input device values, were shared through the cluster. Further data sharing could be implemented by transferring blocks of memory between nodes. Nowadays it supports PC clusters following a similar replication approach.

5.2 VR Juggler

VR Juggler [22] is a software framework for developing portable VR applications. PC cluster support was first provided by Net Juggler [14], a library based on MPI for message exchange. VR Juggler II relies on a client/server paradigm where inputs are executed on servers while clients take care of parallel rendering [21]. Both approaches uses a replication approach.

5.3 Syzygy

Syzygy is a software library dedicated to VR applications running on PC clusters [57]. Syzygy supports networked input devices and sound rendering. Syzygy includes two application frameworks both based on a master/slave paradigm. The first one relies on a classical duplication paradigm, while the second proposes to distribute the data from the master at the scene graph level (or animation level following our classification). Syzygy provides a special protocol to transport the scene graph primitives. The replication approach is usually used when using a scene graph is not appropriate, like for volume rendering.

The entire cluster configuration information is stored in a networked accessible database managed though a distributed operational system called Phleet. It enables a dynamics application reconfiguration, to recover from a slave crash for instance.

5.4 DIVERSE

DIVERSE [35] is a modular collection of complementary software packages designed to facilitate the creation of device independent virtual environments. DgiPf is the DIVERSE graphics interface to OpenGL Performer. A program using DgiPf can run on platforms ranging from fully immersive systems such as CAVEs to generic desktop workstations without modification. On clusters DIVERSE relies on a replication paradigm.

5.5 Jinx

Jinx [58] is a fully distributed virtual environment browser, which has a special support for commodity PC clusters and immersive visualization devices. It is used to develop virtual reality applications based on the X3D format. It uses a replication approach to provide cluster support.

5.6 OpenSG

OpenSG [52, 54] is a portable scene graph system. It allows multiple asynchronous threads to independently manipulate the scene graph without interfering each other. As scene graph data can get very big, a distinction of structural and content data is introduced, and a method to replicate the latter only if necessary. OpenSG also runs on PC Clusters and it is implemented as an extension of the multi-threaded model. Changes in the environment are propagated when they are applied to another node. OpenSG has a Multi Display Window mode, used to render one virtual window on a number of cluster servers, making it possible the use of OpenSG in a CAVE configuration. OpenSG can also be used using a replication approach when used with other tools like VR Juggler.

5.7 Chromiun

Chromium [33], the successor of WireGL [32], proposes a stream processing framework for OpenGL graphics primitives. A network of Stream Processing Units (SPU) enables to apply different transformations to the primitive stream. Chromium is mainly used for sort-first and sort-last parallel rendering. SPUs implement various optimization to reduce the amount of data to send over the network.

Chromium enables to executes an unmodified OpenGL application by intercepting the graphics primitives and broadcasting then to rendering SPUs, each one in charge of its own image tile.

Commercial solutions based on a similar approach are available today, from TechViz or IBM for instance.

5.8 OpenGL Multipipe SDK

OpenGL Multipipe SDK [20] was developed by SGI and provides an uniform API to manage scalable graphics applications access across several graphics subsystems, also providing a customizable image compositing interface, which facilitates the development and deployment of parallel OpenGL based applications. It is able to choose and adapt the decomposition strategy for a given problem domain and graphics environment at run time.

5.9 Basho

Hinkenjann et. al. [30] implemented a solution running over AVANGO [62] and OpenGL/Performer [53] called Basho [40]. It pursues a retained-mode approach for distribution of geometry data with a minimal network load. It consists of a front-end for scene graph distribution and load balancing, and a back-end that performs rendering and compositing. The image combiner works in cascade. For load balance, it uses statistical information from the rendering nodes, adapting the balance as necessary. It also supports different kinds of renders from the 3D engine of graphics cards to ray tracing algorithms.

5.10 Covise

Covise [19] relies on a data-flow model for coupling components that can be distributed on the different nodes of a cluster. It also supports collaborative work and immersive environments using a replication approach. It enables to built advanced VR applications like collaborative volume rendering [65].

5.11 OpenMask

OpenMask [41, 5] is a multi-threaded and distributed middleware library for animation and simulation. It enables to define tasks that are distributed on a cluster and communicate through point-to-point messages and signals. Task update frequency is fixed and controlled by the OpenMask scheduler. Parallel rendering is provided by external tools like OpenSG.

5.12 FlowVR and FlowVR Render

FlowVR [4, 13] is a middleware library dedicated to parallel VR applications. FlowVR relies on an extended data-flow model. An application is composed of *modules* exchanging data through a *FlowVR network*. From the FlowVR point of view, modules are not aware of the existence of other modules. A module only exchanges data with the FlowVR daemon that runs on the same node. The set of daemons running on the PC cluster are in charge of implementing the FlowVR network that connects modules. The daemons take care of moving data between modules using the most efficient method. Daemons can also load plug-gins for implementing advanced message handling operations, like filtering, sampling, frustum culling, broadcasts, etc. This approach enables to develop a pool of modules that can next be combined in different applications, without having to recompile the modules. Modules can be multi-threaded or parallel applications relying on other parallelization tools like MPI [28], FlowVR providing the necessary features to implement a parallel code coupling.

FlowVR Render [17] is a library built on top of FlowVR . It implements a sort-first retained-mode parallel rendering. Instead of relying on OpenGL commands, it defines a shader based protocol using independent batches of polygons as primitives. In opposite to OpenGL based sort-first approaches like Chromium, FlowVR Render does not support unmodified OpenGL applications, but provides a more flexible and efficient framework for parallel rendering.

FlowVR and FlowVR Render enable to built complex distributed applications combining different parallelizations at the input, simulation, animation and rendering levels [18, 16].

6 Hardware Compositing Devices

If rendering is the performance bottleneck and that software solutions for parallel rendering do not provide the expected results, it is still possible to take advantage of hardware compositing devices. These devices are not commodity components but most of them can be installed on commodity PC clusters.

6.1 Cluster Level Compositing

A common approach consists in designing additional boards that take as input signal the video signal of standard graphics cards [43, 51, 60, 48]. The data are then combined in real-time to provide as output one or several video signals. A daughter board can be installed on each PC box to retrieve the video signals computed on each box. Data are exchanged between boards through a dedicated network. Boards can be chained or use a switch like the Sepia boards that rely on an Infiniband network. An other approach consists in having a external compositing device.

These devices usually propose different compositing modes like sort-last compositing, tile compositing, eye compositing to built an active stereo signal from left and right eye images computed concurrently.

6.2 PC Level Compositing: SLI and CrossFire

Back in 1996, Voodoo proposed to have two graphic cards installed in the same PC to share graphical processing for one display. Each card rendered half of the image scan lines that where interleaved when generating the video signal.

Nowadays, NVidia [47] and ATI [25] developed similar solutions, but with better load balancing algorithms and different operational modes. Both solutions support Split Frame Rendering (SFR), Alternative Frame Rendering (AFR) and Anti-aliasing. In SFR mode one card computes the upper half of the image and the other the lower half. In AFR mode one card computes all odd frames, the other even ones. The Anti-aliasing mode splits the anti-aliasing workload between the two graphics cards. ATI also supports a Supertiling mode where rendering tiles are defined and distributed dynamically to the GPUs to ensure a better load balancing. This mode is only available for DirectX applications for the moment.

7 Display Technology

The first immersive environments were using large CRT projectors. While PC clusters were emerging as a relevant alternative to dedicated supercomputers, it also appeared that commodity projectors based on LCD or DLP technologies could be used too [36].

7.1 Projectors

Different projection technologies are available today. Usually they fit in four main categories:

- CRT projectors utilize three tubes, one for each color (RGB). The three colors combine and converge into a light beam that draws the image sequentially pixel per pixel. CRTs do not have a fixed number of pixels. As the number of pixels to draw increases, the refresh rate decreases. One problem regarding CRT projectors is that they require periodic calibrations. Today CRT technology tends to be abandoned in favor of other ones.
- LCD projectors contain three separate LCD glass panels, one for each color (RGB). As the light passes through the LCD panels, their opacity is controlled to modulate the amount of color pixels receive. LCD is one of the main technology used for commodity projectors. They require little maintenance.
- DLP projectors are based on an optical semiconductor called a DMD chip, which was invented in 1987 by Texas Instruments. The DMD chip is an array of micro-mirrors (one per pixel) that tilt to control the pixel brightness. Color is provided either by associating 3 DMD chips, one per color (RGB), or using one DMD and a color wheel. Commodity DLP projectors use one DMD, while high end ones uses 3 DMDs. DLP projectors require little maintenance.
- LCOS projectors rely on a new technology that uses LCD filters with a mirror under it. This technology is today used for high-end projectors, like the SRX Sony projector [10] that reaches a resolution of 4096 x 2160 pixels.

7.2 Stereo

Stereo is a fundamental feature for the sense of immersion. Usually stereo technologies can be classified into in three categories:

- Active stereo. Left and right eye images are displayed alternatively. Swap occurs during the vertical blanking, i.e. when no pixel is generated before the next video retrace starts. The user wears glasses with shutters opening alternatively to let him see the right eye images with the right eye only and vice-versa. Gen-lock is mandatory to ensure all projectors display the left or right eye image synchronously. The video refresh rate must be high enough (usually about 120 Hz) to ensure the user does not perceive flickering. Active stereo is available with specific CRT and 3 chip DLP projectors. Single chip DLP projectors supporting active stereo may be soon available [31]. In all cases, quality active stereo is only supported by projectors that have been adapted for that use.
- Passive stereo. Left and right eye images are displayed concurrently by two different projectors. Passive stereo can be obtained with almost any projector. Frame-lock is usually sufficient. No specific hardware is required on the cluster. We can distinguish two different approaches to separate both images:
  - Linear and circular polarization. The left and right eye images are polarized differently, the user wearing glasses with polarizing filters to block the image of the opposite eye. Light can be linearly or circularly polarized. Filters for linear polarization are cheaper than the ones for circular polarization, but image separation is lost when the user tilts his head. Both techniques require special screens that maintain the polarization. Such screens are usually high gain screens, while low gain screens are preferable to smooth hot spot effects that are particularly noticeable when tilling multiple projectors.
  - Infitec. The spectral colors are split into 6 ranges, two for each primary color. Each eye is associated to one range for each color. Specific filters on the projectors and the glasses are used to separate the right and left eye images. Infitec does not impose any constrain on screen materials. However a color shift occurs that requires to be corrected using hardware or software techniques.
- Auto-stereo technologies are becoming more common. They do not require users to wear glasses. Today no projection technology supports auto-stereo. Though some attempts have been made to build display walls for auto-stereo displays [55], this approach is not yet used for immersive environments.

7.3 Image Calibration

When using a large number of projectors, the challenge is to ensure a high quality image calibration, i.e. a correct geometric projector alignment as well as a photometric uniformity.

Promising approaches propose automatic calibration techniques. These approaches use a camera to take pictures of predefined patterns. From these pictures are extracted information on corrections to apply on images displayed by each projector [50, 38, 39, 34]. Corrections are them applied online by graphics cards at each projector. Though quality results are obtained for geometric calibration, photometric uniformity is more difficult to obtain, partly because camera captors are not very precise in measuring color discrepancies. An alternative consists in computing color calibration from values obtained by a spectroradiometer.

8 Conclusion

PC clusters are today the main solution to provide the computing and I/O power required by advanced VR applications and facilities. PC clusters can range from single CPU PCs connected through a standard network, using video game graphics cards to high-end multi CPU PCs connected through a dedicated high performance network and equipped with professional graphics cards.

Software tools have to be adapted and developed to these architectures. The difficulty is to develop software solutions that enable to take advantage of the performance offered by clusters, while keeping the complexity of application development, deployment and execution as low as possible. Today such solutions are available for distributed rendering, and others are emerging to provide extra computing capabilities for processing input data from sensor networks, or for multi modal applications involving 3D graphics, spacialized sound, haptics systems, multiple simulations, etc.

Still, the size of clusters used for VR is moderate (a few tens of nodes), while much larger machines exists. We may see in the future more applications using such larger machines. It will have to face some issues regarding algorithm parallelization, algorithm coupling and software engineering to handle the complexity of the applications. Advantage can be taken from the various developments that have been performed for high performance or distributed computing. But it is important to stress that VR applications tend to be more heterogeneous than classical high performance applications (coupling different codes for sound, graphics, haptics, physical based simulations, etc.) and with a strong constraint on performance (latencies and update frequencies).

Beside clusters, parallelism will also become more present as new processor architectures emerge. It will be common to have a single box machine with multiple CPUs, GPUs and FPGAs. For instance, today it is possible to have a PC with 2 GPUs and 8 dual-core CPUs. Soon the Cell processor that includes 8 cores will be available. The extra computing power available could greatly influence the development of the next generation of VR applications.

References

[1] Top 500 Supercomputers, www.top500.org.

[2] Rocks Cluster Distribution, www.rocksclusters.org.

[3] Kerrighed, www.kerrighed.org.

[4] FlowVR. `http://flowvr.sf.net`.

[5] OpenMASK. http://www.irisa.fr/siames/OpenMASK.

[6] VR Juggler. http://www.vrjuggler.org.

[7] Origin Servers. Technical report, Silicon Graphics, April 1997.

[8] Nvidia quadro fx 3000g solutions for advanced visualization. Technical brief, NVIDIA Corporation, 2003.

[9] New wildcat realizm graphics technology. White paper, 3DLabs, 2004.

[10] *SR Projector, Operating Instructions*, 2005.

[11] J. Allard, E. Boyer, J.-S. Franco, C. Ménier, and B. Raffin. Markerless Real Time 3D Modeling for Virtual Reality. In *Proceedings of the Immersive Projection Technology Workshop*, Ames, Iowa, May 2004.

[12] J. Allard, V. Gouranton, G. Lamarque, E. Melin, and B. Raffin. Softgenlock: Active Stereo and Genlock for PC Cluster. In *Proceedings of the Joint IPT/EGVE'03 Workshop*, Zurich, Switzerland, May 2003.

[13] J. Allard, V. Gouranton, L. Lecointre, S. Limet, E. Melin, B. Raffin, and S. Robert. FlowVR: a Middleware for Large Scale Virtual Reality Applications. In *Proceedings of Euro-par 2004*, Pisa, Italia, August 2004.

[14] J. Allard, V. Gouranton, L. Lecointre, E. Melin, and B. Raffin. Net Juggler: Running VR Juggler with Multiple Displays on a Commodity Component Cluster. In *IEEE VR*, pages 275–276, Orlando, USA, March 2002.

[15] J. Allard, V. Gouranton, E. Melin, and B. Raffin. Parallelizing Prerendering Computations on a Net Juggler PC Cluster. In *Immersive Projection Technology Symposium*, Orlando, USA, March 2002.

[16] J. Allard, C. Ménier, E. Boyer, and B. Raffin. Running large vr applications on a pc cluster: the flowvr experience. In *Proceedings of EGVE/IPT 05*, Denmark, October 2005.

[17] J. Allard and B. Raffin. A shader-based parallel rendering framework. In *IEEE Visualization Conference*, Minneapolis, USA, October 2005.

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

[19] A.Wierse, U.Lang, and R. Rühle. Architectures of Distributed Visualization Systems and their Enhancements. In *Eurographics Workshop on Visualization in Scientific Computing*, Abingdon, 1993.

[20] P. Bhaniramka, P. C. D. Robert, and S. Eilemann. Opengl multipipe sdk: A toolkit for scalable parallel rendering. In *Proceedings of IEEE Visualization 2005*. IEEE, October 2005. Proceedings of IEEE Visualization 2005.

[21] A. Bierbaum, A. Bierbaum, and C. Cruz-Neira. Clusterjuggler: A modular architecture for immersive clustering. In *VR-Cluster'03-Workshop on Commodity Clusters for Virtual Reality, IEEE Virtual Reality Conference 2003*, Los Angeles, March 2003.

[22] A. Bierbaum, C. Just, P. Hartling, K. Meinert, A. Baker, and C. Cruz-Neira. VR Juggler: A Virtual Platform for Virtual Reality Application Development. In *IEEE VR 2001*, Yokohama, Japan, March 2001.

[23] M. Bues, R. Blach, S. Stegmaier, U. Häfner, H. Hoffmann, and F. Haselberger. Towards a Scalable High Performance Application Platform for Immersive Virtual Environements. In *Immersive Projection Technology and Virtual Environements 2001*, pages 165–174, Stuttgart, Germany, May 2001. Springer.

[24] X. Cavin, C. Mion, and A. Filbois. Cots cluster-based sort-last rendering: Performance evaluation and pipelined implementation. In *IEEE Visualization Conference*, Minneapolis, USA, October 2005.

[25] A. Crossfire. http://www.ati.com/technology/crossfire/.

[26] C. Cruz-Neira, D. J. Sandin, T. A. DeFanti, R. V. Kenyon, and J. C. Hart. The Cave Audio VIsual Experience Automatic Virtual Environement. *Communication of the ACM*, 35(6):64–72, 1992.

[27] J. Eyles, S. Molnar, and J. Poulton. PixelFlow: High-Speed Rendering Using Image Composition. In *Proceedings of ACM SIGGRAPH 92*, pages 231–240, Chicago, USA, July 1992.

[28] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: Portable Parallel Programming with the Message-Passing Interface*. Scientific and Engeneering Computation Series. The MIT Press, 1994.

[29] M. P. Guimarães, P. A. Bressan, and M. K. Zuffo. Frame lock synchronization for multiprojection immersive environments based on pc graphics clusters. In *Proocedings of the 5th SBC Symposium on Virtual Reality*, 2002.

[30] A. Hinkenjann, M. Bues, S. Schupp, and T. Olry. Mixed-mode parallel real-time rendering on commodity hardware. In *Proceedings of 5th Symposium on Virtual Reality*, Fortaleza, Brazil, 2002.

[31] A. Hopp. Using a single spatial light modulator for stereo images of high color quality and resolution . In *Proceedings of EGVE/IPT 05*, Denmark, October 2005.

[32] G. Humphreys, M. Eldridge, I. Buck, G. Stoll, M. Everett, and P. Hanrahan. WireGL: A Scalable Graphics System for Clusters. In *Proceedings of ACM SIGGRAPH 2001*, 2001.

[33] G. Humphreys, M. Houston, R. Ng, S. Ahern, R. Frank, P. Kirchner, and J. T. Klosowski. Chromium: A Stream Processing Framework for Interactive Graphics on Clusters of Workstations. In *Proceedings of ACM SIGGRAPH 02*, pages 693–702, 2002.

[34] C. Jaynes, B. Seales, K. Calvert, Z. Fei, and J. Griffioen. The Metaverse - A Collection of Inexpensive, Self-configuring, Immersive Environments. In *7th International Workshop on Immersive Projection Technology/Eurographics Workshop on virtual Environments*, Zurich, Switzerland, May 2003.

[35] J. Kelso, L. E. Arsenault, S. G. Satterfield, and R. D. Kriz. Diverse: A framework for building extensible and reconfigurable device independent virtual environments. In *Proceedings of IEEE Virtual Reality 2002 Conference*, 2002.

[36] K. Li, H. Chen, Y. Chen, D. W. Clark, P. Cook, S. Damianakis, G. Essl, A. Finkelstein, T. Funkhouser, A. Klein, Z. Liu, E. Praun, R. Samanta, B. Shedd, J. P. Singh, G. Tzanetakis, and J. Zheng. Early Experiences and Challenges in Building and Using A Scalable Display Wall System. *IEEE Computer Graphics and Application*, 20(4):671–680, 2000.

[37] K. Liang, P. Monger, and H. Couchman. Interactive Parallel Visualization of Large Particle Datasets. In *Proceedings of the Eurographics Parallel Graphics and Visualization Symposium*, pages 23–30, Grenoble, France, June 2004.

[38] A. Majumder, Z. He, H. Towles, and G. Welch. Achieving color uniformity across multi-projector displays. In *Proceedings of the 11th IEEE Visualization*, page 17, October 2000.

[39] A. Majumder and R. Stevens. Color Non-Uniformity in Projection Based Displays : Analysis and Solutions. *Trasactions of Visualization and Computer Graphics*, 2003.

[40] F. Mannuß and A. Hinkenjann. Towards better quality in virtual environments. In *Proceedings of Eurographics Workshop on Virtual Environments*, 2005.

[41] D. Margery, B. Arnaldi, A. Chauffaut, S. Donikian, and T. Duval. Openmask: {Multi-Threaded | Modular} animation and simulation {Kernel | Kit }: a general introduction. In Simon Richir, Paul Richard, and Bernard Taravel, editors, *VRIC 2002 Proceedings*, pages 101–

110. ISTIA Innovation, June 2002.

[42] M. L. Massie, B. N. Chun, and D. E. Culler. The ganglia distributed monitoring system: Design, implementation, and experienc. In *Parallel Computing*, volume 30, July 2004.

[43] L. Moll, M. Shand, and A. Heirich. Sepia: Scalable 3d compositing using pci pamette. In *Proceeding of the Seventh Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, page 146, April 1999.

[44] S. Molnar, M. Cox, D. Ellsworth, and H. Fuchs. A Sorting Classification of Parallel Rendering. *IEEE Computer Graphics and Applications*, 14(4):23–32, July 1994.

[45] J. S. Montrym, D. R. Baum, D. L. Dignam, and C. J. Migdal. Realityengine graphics. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 109–116, 1993.

[46] J. S. Montrym, D. R. Baum, D. L. Dignam, and C. J. Migdal. Infinitereality: a real-time graphics system. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 293–302, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[47] S. Nvidia. http://www.slizone.com/.

[48] G. Otrage. Orad's dvg: Solutions for scalable graphics clusters, Aug 2004. Graphics Hardware Conference.

[49] D. Pape, C. Cruz-Neira, and M. Czernuszenko. *CAVE User's Guide*. Electronic Visualization Laboratory, University of Illinois at Chicago, 1997.

[50] R. Raskar and J. van Baar. Low cost projector mosaic. In *Proceedings of the Asian Conference on Computer Vision 2002*, Jan 2002.

[51] F. Razo. *SGI InfinitePerformance: Scalable Graphics Compositor Owner's Guide*, 2002.

[52] D. Reiners. Opensg: A scene graph system for flexible and eficient reltime rendering for virtual and augmented reality applications, 2002.

[53] J. Rohlf and J. Helman. Iris performer: a high performance multiprocessing toolkit for real-time 3d graphics. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 381–394, 1994.

[54] M. Roth, G. Voss, and D. Reiners. Multi-threading and clustering for scene graph systems. *Computers & Graphics*, 28(1):63–66, 2004.

[55] D. J. Sandin, T. Margolis, J. Ge, J. Girado, T. Peterka, and T. A. DeFanti. The varriertm autostereoscopic virtual reality display. *ACM Trans. Graph.*, 24(3):894–903, 2005.

[56] B. Schaeffer. Networking Management Frameworks for Cluster-Based Graphics. http://www.isl.uiuc.edu/ClusteredVR/ClusteredVR.htm, 2002.

[57] B. Schaeffer and C. Goudeseune. Syzygy: Native PC Cluster VR. In *IEEE VR Conference*, 2003.

[58] L. P. Soares and M. K. Zuffo. Jinx: an x3d browser for vr immersive simulation based on clusters of commodity computers. In *Proceedings of the ninth international conference on 3D Web technology*, Monterey, California, USA, April 2004.

[59] J. Springer, A. Simon, and J. Plate. Supporting commodity clusters using avango's generic scene graph distribution. In *VR-Cluster'03-Workshop on Commodity Clusters for Virtual Reality, IEEE Virtual Reality Conference 2003*, Los Angeles, March 2003.

[60] G. Stoll, M. Eldridge, D. Patterson, A. Webb, S. Berman, R. Levy, C. Caywood, M. Taveira, S. Hunt, and P. Hanrahan. Lightning-2: A High-Performance Display Subsystem for PC Clusters. In *Proceedings of ACM SIGGRAPH 01*, 2001.

[61] A. Streit, R. Christie, and A. Boud. Understanding next-generation vr: classifying commodity clusters for immersive virtual reality. In *GRAPHITE '04: Proceedings of the 2nd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pages 222–229, New York, NY, USA, 2004. ACM Press.

[62] H. Tramberend. Avocado: A distributed virtual reality framework. In *Proceedings of the IEEE Virtual Reality*, page 14, 1999.

[63] J. L. Williams and R. E. Hiromoto. Sort-middle multi-projector immediate-mode rendering in chromium. In *IEEE Visualization Conference*, Minneapolis, USA, October 2005.

[64] U. Wössner and M. Aumüller. Software-based genlock for active stereo nvidia cards.

[65] U. Wössner, J. Schulze, S. Walz, and U. Lang. Evaluation of a Collaborative Volume Rendering Application in a Distributed Virtual Environment. In *Eighth Eurographics Workshop on Virtual Environments*, 2002.

[66] S. Würmlin, E. Lamboray, O. Staadt, and M. Gross. 3D Video Recorder: A System for Recording and Playing Free-Viewpoint Video. *Computer Graphics Forum*, 22(2):181–193, 2003.