ADVANCED IMAGING

CAVELib Support For PC Visualization Clusters

Multiprojector systems, like the one shown here, are supported by PC-based visualization systems, which have been commonly used as the image generator for interactive 3D displays.

Usability with Open Inventor allows for creating interactive 3D applications.

By Matt Szymanski
Chief Technology Officer
VRCO, Inc.

Visualization systems for driving immersive interactive 3D (i3D) displays are generally characterized by a combination of three hardware components. The first is computational power, which includes the CPU, main memory and storage systems. The second is graphics power, which, at the core, is the GPU (Graphics Processing Unit). The GPU is frequently referred to as a graphics card, graphics pipe or simply pipe, depending upon which hardware vendor one is conversing with. Most modern graphics cards are usually able to output to one or more channels where each channel is then connected to a projector that is displaying to an i3D system. The third component is the communications, which is the glue that holds the first two components together. Communications may be external to the system such as local networks or internal to the system, like system buses.

Many modern-day supercomputers are capable of having multiple CPUs and GPUs, as well as an internal architecture that interconnects them with high-speed buses. These systems easily support multiprojector systems and have been commonly used as the image generator for i3D displays. However, there is a growing trend to use commodity off-the-shelf hardware and PC workstations to drive multiwall visualization display systems.

The difficulty with using workstations is that they are generally a single GPU system. In order to display to a multiwall system, a cluster of PCs need to be interconnected through hardware and have software that allows multiple synchronized images to be rendered. In response to this need, the CAVELib was enhanced back in 1999 to support PC-based visualization clusters, a.k.a. viz-clusters.

OVERVIEW

The CAVELib is an API designed for creating immersive, interactive 3D applications. The CAVELib's power lies in its independence of any specific display system, despite being named "CAVE" Lib. The software is completely configurable at run-time and currently supports up to thirty-two arbitrarily

placed projection screens. Because of this flexibility, it has been shown to work on all commercially available display systems to date, and all tested custom displays as well. The CAVELib itself is not a high-level graphics toolkit, but a framework to create OpenGL-based applications that will display properly on a variety of systems without the need to recompile.

There is a basic philosophy with the CAVELib's support for viz-clusters and that philosophy is let the machines do as much of the work locally as possible. PC clusters, both for data computation as well as for visualization, are quickly replacing the supercomputer because of the low cost of commodity PC hardware. The reason for supercomputers' high cost — even ones that use commodity components for microprocessors and memory — is due, in part, to the cost of designing a proprietary system architecture of buses, motherboards and shared memory.

In order to lower the cost of doing large computations and managing visualization displays, more organizations are looking at PC clusters as an alternative. However, when a cluster of PCs are used to do the same type of work that was traditionally done by a supercomputer, there are going to be differences in how an application needs to be written in order for it to work. No longer is the application written to execute on a single system. Instead, it is written for a collection of systems that need to communicate with each other about the work they are doing, which includes synchronizing their data and information.

Most software developers are quite familiar with making an application work within the local memory of a single computer or supercomputer system, such as an SMP system, but fewer have the experience and knowledge of developing applications that need to do inter-PC communications. With the CAVELib, in order to aid the application developers, it attempts to abstract away as much of the communications' complexity as possible in creating interactive 3D applications for viz-clusters.

PC clusters are being seen as cost-effective alternatives for large computations and managing visualization displays.

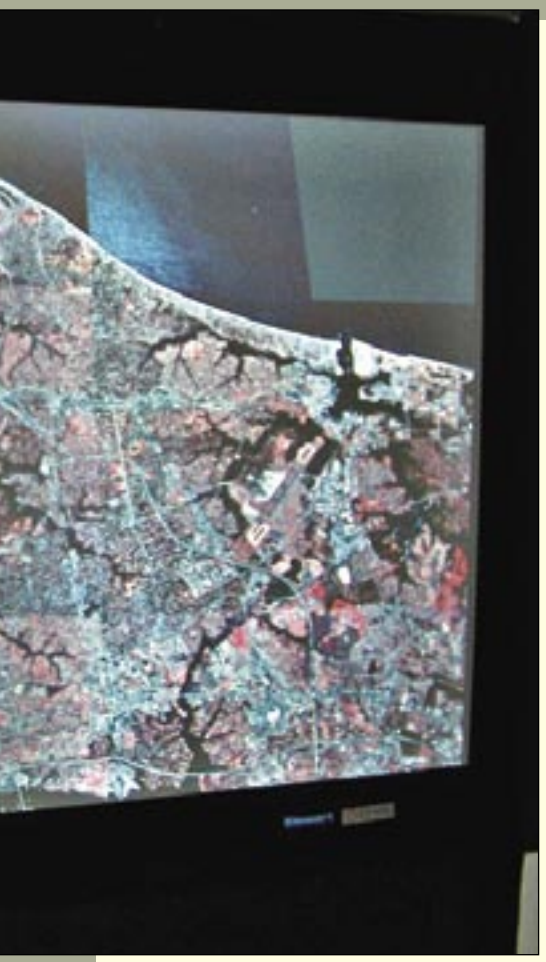
DESCRIPTION

The primary purpose of using PC viz-clusters is to be able to drive multi-projector display systems with a lower cost alternative to supercomputers. The display systems may range from CAVE-like displays with one projector per screen, to tiled-wall displays where a single screen is comprised of an array of projectors in order to create a single high-resolution projected image, or be something in between.

When a CAVELib application is run on a PC viz-cluster, an exact copy of the executable runs on each PC within the cluster, also called a node. This means each node must have access, via network or local disk, to any and all of the data files that the application will be loading and using during run-time.

Each node of the viz-cluster is responsible for rendering graphics to some portion of the display. One of the PC nodes is always called the master node and the rest are all called slave nodes. With the CAVELib, the master node is also required to be one of the graphics nodes. For example, in a 2x2 tiled wall display, composed of four PCs and four projectors, there would be one master node and three slave nodes. Each PC would then be responsible for rendering one-fourth of the overall image. In addition to rendering graphics and managing the synchronization between the PCs, the master node in a CAVELib cluster application is also responsible for obtaining input data, such as trackers and input devices, and then sharing this data with the slave nodes each and every frame.

When using the CAVELib, each node is configured to know only which portion of the screen it is responsible for rendering. The CAVELib guarantees that, for each frame, all internal state data that



may affect the rendering (such as tracking data, time, etc.) are exactly the same for each and every node in the cluster. By using a series of barriers or locks, the CAVELib guarantees that no PC is allowed to execute its rendering action until it has the latest state data from the master node. The primary mechanism for making a cluster work is in the fact that each node is running an exact copy of the CAVELib application and using the same data values for each frame, this will result in the rendered output being deterministic on a per-frame basis. This is the main contributor to how the CAVELib keeps the rendered images synchronized.

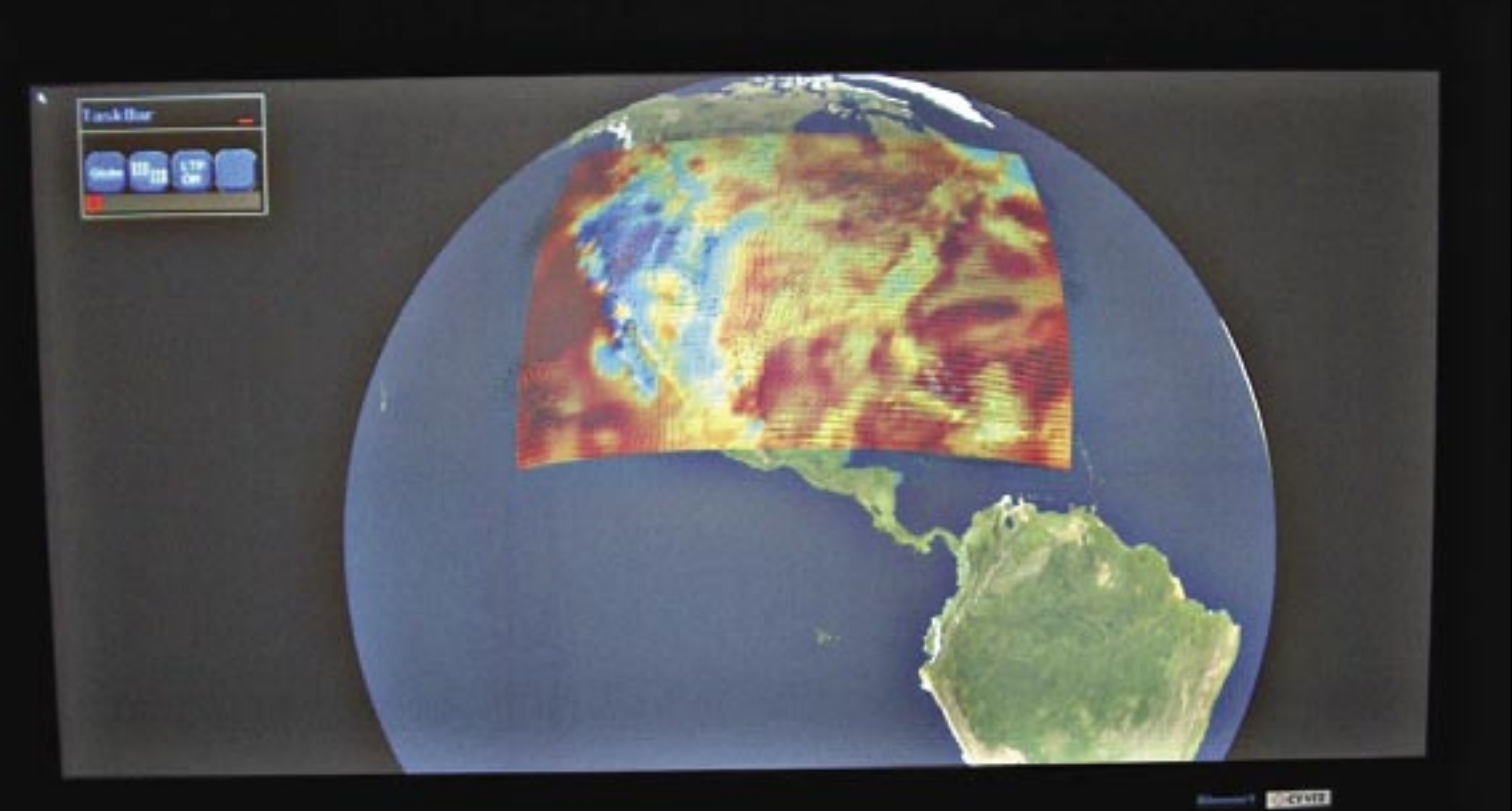
Additionally, the CAVELib has to guarantee that the rendered images are made available to the video projectors at the same time for each node, each frame. The process of doing this is called framelock. Framelock means that all of the PCs do their graphics buffer swaps at the same time. The buffer swap is when the image the application just created is moved from the image buffer into which it was rendered, to the buffer the graphics card accesses to output the image as a video signal. The CAVELib, as part of its support for clusters, provides framelock for the application automatically at run-time.

When discussing viz-clusters, there is another type of synchronization that is often mentioned and that is video genlock. Genlock is the action of each video card sending its video signal out at exactly the same time. It is used for many types of displays and is absolutely required for an active stereo display. In active stereo, a graphics card sends alternating images from the graphics buffers. For each display frame, the graphics card alternates between sending a left-eye and a right-eye image. When using multiple PCs with shutter glasses for active stereo, the glasses and the video image for each node must be synchronized. What this means is that each video card in the viz-cluster must output the left-eye image at the same time that the glasses are synchronized to view the left-eye image, and vice-versa for the right-eye image. The genlock synchronization is something that is done specifically by graphics hardware, usually with a special interconnect or cable that connects the cards in each PC. So far, the CAVELib has been shown to work in active stereo on any viz-cluster that has a genlock mechanism.

The CAVELib support for viz-clusters uses three main technologies: data synchronization, framelock and genlock. The CAVELib's paradigm for cluster support lets each node of the viz-cluster do all of its rendering and responses to per frame data (such as tracker and controller inputs) locally. The CAVELib's architecture is set up to maximize the usage of each individual node by doing computations on the local CPUs, utilizing the local memory as much as possible, and minimizing the use of the network. The CAVELib's cluster support is based on the premise that the data and geometry transfer done inside a PC is more efficient than transferring data between PCs. The added benefit of this approach is that it simplifies the migration of legacy applications into cluster environments as the legacy code does not need to be re-architected to a distributed computing paradigm.

In addition, the CAVELib cluster design fully supports applications that do utilize a distributed scenario in which one node computes geometry information and shares it with the other nodes that perform the rendering. The CAVELib API provides a function call to guarantee the synchronization of the shared render data between all nodes. In this way, a master node may take advantage of a local resource but still be able to share that information to each node, as opposed to having access to the data duplicated on each node.

Typically, with a cluster CAVELib application, each node operates nearly autonomously in a cluster setup. Each node receives the per-frame data from a single master node each frame and then does all of its work on the local CPUs and memory. In a CAVELib application for an SMP system, information about the tracker and user input is stored in shared memory, and each display process or thread has access to it. In a cluster CAVELib application, the CAVELib treats each node akin to being a process and automatically passes this per frame data to each node's local memory without the application developer needing to worry about it. All of this is able to occur without each node having to know what the other nodes in the viz-cluster are responsible for. The approach of hiding this passing of per-frame data is what makes migration of a CAVELib application from an SMP system to a viz-cluster very easy for many existing customers.



A CAVE-like display assigns one projector to each screen, as opposed to a tiled-wall display, in which a single screen consists of numerous projectors to create a single high-resolution image.

COMBINING THE CAVELIB AND OPEN INVENTOR

The CAVELib and Open Inventor from Mercury Computer Systems have been successfully used together for many years. Applications that use these two packages must follow a CAVELib programming paradigm for the creation of the application. In order to support clusters, the CAVELib must maintain its responsibility for all of the window, viewport and graphics contexts creation as well as spawn the display threads or processes when these packages are used together.

The CAVELib is the software responsible for receiving any tracker and input data and makes it available to the application through its API. Within an Open Inventor CAVELib application, Open Inventor provides the scene graph and rendering portions of the application. The CAVELib itself is not a high-level toolkit for creating graphical applications. Instead, the CAVELib's strength is in providing a framework from which applications are developed for supporting multiwall, immersive, interactive 3D display systems. Open Inventor has proven to work very well with CAVELib because its rendering engine can be executed easily from within a CAVELib display process. In fact, the Open Inventor rendering action executes without depending upon any CAVELib functionality, this allows most standard nodes and custom nodes to be rendered within an Open Inventor CAVELib application. Additionally, since an exact copy of an Open Inventor CAVELib application is running on each node, an application's custom nodes and callback nodes will work with the CAVELib, which may not be true for some other cluster solutions.

Developers making use of the CAVELib and Open Inventor will find its structure a little bit of a departure from a traditional desktop Open Inventor application. However, all of Open Inventor's geometry, materials and lights work with CAVELib with no new effort needed by the application developer. Because the CAVELib is designed to primarily support immersive devices, including six DOF trackers and input controllers as opposed to desktop interfaces, support for some of Open Inventor's interactive nodes will differ slightly from how they are used in a standalone Open Inventor application. Finally, Open Inventor's desktop GUIs are typically unsupported with the CAVELib.

SUMMARY

The CAVELib's cluster support is a different approach than many of the other cluster paradigms available due to its minimal bandwidth requirements. Putting the workload inside each PC will allow for a more efficiently scalable solution than one that has a tighter coupling to network performance. Except for instances where active stereo is required, the CAVELib's cluster support has been shown to be hardware-independent.

Since 2000, the CAVELib's cluster support has been used to drive CAVEs, Reality Centers and tiled displays for a variety of organizations. Additionally, the CAVELib and Open Inventor have been used together successfully for displaying in i3D displays driven by both supercomputers and PC clusters. Both toolkits have cross-platform support and have been used individually as well as together for many years.

(Photos Courtesy of VRCO, Inc.)

Reprinted with Permission from Advance Imaging • September 2004