# Hybrid Hardware-Accelerated Image Composition for Sort-Last Parallel Rendering on Graphics Clusters with Commodity Image Compositor

Jorji Nonaka* Kyoto University
Nobuyuki Kukimoto† Tohwa University
Naohisa Sakamoto‡ Kyoto University
Hiroshi Hazama§ Kyoto Technical Institute
Yasuhiro Watashiba¶ Kyoto University
Xuezhen Liu‖ Mitsubishi Precision
Masato Ogata** Mitsubishi Precision
Masanori Kanazawa†† Kyoto University
Koji Koyamada‡‡ Kyoto University

**ABSTRACT**

Hardware-accelerated image composition for sort-last parallel rendering has received increasing attention as an effective solution to increased performance demands brought about by the recent advances in commodity graphics accelerators. So far, several different hardware solutions for alpha and depth compositing have been proposed and a few of them have become commercially available. They share impressive compositing speed and high scalability. However, the cost makes it prohibitively expensive to build a large visualization system. In this paper, we used a hardware image compositor marketed by Mitsubishi Precision Co., Ltd. (MPC) which is now available as an independent device enabling the building of our own visualization cluster. This device is based on binary compositing tree architecture, and the scalable cascade interconnection makes it possible to build a large visualization system. However, we focused on a minimal configuration PC Cluster using only one compositing device while taking cost into consideration. In order to emulate this cascade interconnection of MPC compositors, we propose and evaluate the hybrid hardware-assisted image composition method which uses the OpenGL alpha blending capability of the graphics boards for assisting the hardware image composition process. Preliminary experiments show that the use of graphics boards diminished the performance degradation when using an emulation based on image feedback through available interconnection network. We found that this proposed method becomes an important alternative for providing high performance image composition at a reasonable cost.

**CR Categories:** I.3.1 [Computer Graphics]: Hardware-Architecture—Parallel Processing; I.3.2 [Computer Graphics]: Graphics Systems—Distributed/network graphics;

**Keywords:** Image Compositing, Sort-Last, Parallel Rendering, Hardware-Assisted, Cluster Computing

## 1 INTRODUCTION

Commodity off-the-Shelf (COTS) PC clusters have proven to be a cost-effective alternative to high-end High Performance Computing (HPC) systems. Recent low-cost, high performance commodity graphics accelerators have become an attractive solution to enhancing these PC clusters for building scalable visualization systems. These enhanced PC clusters are also known as *"visualization clusters"* or *"graphics clusters"*, and have received increasing attention as a cost-effective solution to render large data sets at interactive frame rates which was traditionally feasible only by using high-end visualization systems. This popularity is due particularly to the impressive evolution of consumer graphics hardware, which has been providing excellent rendering performance at a reasonable cost. The ability to combine with low-cost software generation locking (genlock) for image synchronization is making these systems more and more attractive not only for large-scale data visualization but also for building cost-effective immersive virtual environments.

Nowadays, state-of-the-art numerical simulations of real-world phenomena can generate a huge amount of data in single or multiple time steps with each step containing many gigabytes of data. As the volume data size and complexity increases at a comparable rate of computational power and resolution of data acquisition devices, the necessity for visualizing these data sets in high-resolution also increases. For this purpose, there already exists commercially available super high definition LCD displays with multi-million pixels such as the 22.2 inch LCD display developed by IBM. In addition, both monitor and projector-based high-resolution displays, such as PC Cluster based scalable display walls, are becoming popular because they are easy to build using readily available commodity components and open source software. Despite requiring more processing power it is sometimes advisable to see the rendering results in stereoscopic fashion in order to enhance the visualization capability. For this purpose, low-cost passive stereo projection and stereo viewing systems built with commodity off-the-shelf components and open source application software making them more affordable than previous commercial solutions, have become available.

Volume rendering is considered an efficient method for extracting useful information from volumetric data sets by generating an image on a two-dimensional view plane. Despite the simplicity of the rendering process, it is computationally intensive and, as a result, several acceleration techniques including parallelization have been proposed. Parallel volume rendering is considered an effective solution for interactive rendering of large volume data sets and for displaying them on aforementioned high resolution display systems. Parallel rendering algorithms for parallel architecture systems such as cluster-based fall into three categories first proposed by Molnar [13]: *sort-first*, *sort-middle*, and *sort-last*. This classification is based on how the geometric primitives are sorted from object space to screen space. *Sort-last* approach has been widely utilized due to the simplicity of the data distribution strategy. After the initial data repartition and distribution there is no need for data transferring between processors until finishing the rendering process. However, it requires the final image composition in order to generate the final result. As a result, the performance becomes highly dependent on the required resolution for displaying the rendered image affecting the amount of data to be transmitted.

*e-mail: jorji@ais.sys.i.kyoto-u.ac.jp
†e-mail: kukimoto@tohwa-u.ac.jp
‡e-mail: naohisas@mbox.kudpc.kyoto-u.ac.jp
§e-mail: h-hazama69@mail.pref.kyoto.jp
¶e-mail: siba@ais.sys.i.kyoto-u.ac.jp
‖e-mail: liu@mpcnet.co.jp
**e-mail: ogata@mpcnet.co.jp
††e-mail: bwv147@mbox.kudpc.kyoto-u.ac.jp
‡‡e-mail: koyamada@kudpc.kyoto-u.ac.jp

IEEE Symposium on Volume Visualization and Graphics 2004
October 11-12, Austin, Texas, USA
0-7803-8781-3/04/$20.00 ©2004 IEEE

Several efficient software parallel compositing solutions have been proposed, however they are not sufficient to handle the increasing frame rate that hardware graphics accelerators are capable of. In addition, the efficiency of most acceleration techniques proposed for software composition is highly dependent on the quantity of the background pixels. As a result, the performance degradation is most pronounced during immersive visualization such as virtual voyages inside the volumetric data.

Several hardware solutions have been proposed in order to overcome this final image composition bottleneck. Most of them still remain in their prototyping stage and others are usually offered as a non-separable part of the whole visualization system solution such as HP sv7, MPC VGCluster, SGI Onyx4 and Orad DVG. Among these, the HP and MPC hardware are geared towards COTS PC Clusters and, in addition, although MPC markets the whole visualization system they also sell the image compositing hardware separately. We therefore acquired this MPC hardware compositor, which is based on framebuffer readback and binary composition tree architecture, to increase the potential of our COTS visualization cluster. Although this compositor possesses high scalability for building large visualization systems, it becomes prohibitively expensive to build such a large system. A simple solution is to use a complete graphics cluster in its minimum hardware configuration with only one hardware compositor. However, it is not possible to handle very large data sets in which the number of sub-volumes exceeds the number of available rendering nodes. To overcome this problem we propose and evaluate the hybrid hardware-assisted image composition technique in order to emulate the hierarchical cascade interconnection of this compositing device for supporting multiple rendering nodes. In this case, the composition task is done on both hardware compositor and graphics board. Some studies for supporting more precise emulation of the cascade interconnection and for utilizing high-resolution or stereoscopic displaying systems will also be discussed.

In Section 2, we briefly review some related works. In Section 3, we briefly describe our utilized graphics cluster with the image compositor. In Section 4, we describe the proposed hybrid hardware-accelerated image compositing method and we present some results. Finally, we conclude and present future directions in Sections 5 and 6.

## 2 Related Work

Many software-based parallel image compositing algorithms have been proposed for sort-last parallel volume rendering on both shared-memory [18] and distributed memory parallel architecture systems [11, 1, 26, 23, 19]. Yang [26] divided these algorithms into two main groups: *buffered* and *sequenced*. In the case of buffered algorithms each processor is responsible for handling a fixed portion of the image and allocates a buffer and receives pixels in the same fixed portion of the image from other processors in order to generate the final image of the portion it handles. In the case of sequenced algorithms, such as *parallel pipeline* and *binary-swap* approach, each processor receives a sub-image from another processor and composites received pixels immediately in each compositing stage. Stompel [22] selected the *direct send*, *binary swap* and *parallel pipeline* methods as the representatives for parallel image compositing algorithms. Among these methods, *binary-swap* is perhaps the most utilized and researched and, as a result, several optimizations such as bounding box, load balancing and compression have been proposed for achieving further acceleration [26, 23, 19]. They exploit the sparsity of the image, eliminating the background pixels as much as possible, diminishing the required data to be communicated during the compositing process. As a result, the performance is highly dependent on the quantity of background pixels. Stompel [22] proposed Scheduled Linear Image Compositing (SLIC); a fully optimized direct send method which is reported that outperforms the direct send and binary swap method even when optimized. Compared to the binary-swap method, the performance increase becomes accentuated during high-resolution image composition. However, it remains dependent on the amount of pixels required to be communicated and especially on the performance of the utilized interconnection network.

Although pure software image compositing solutions have proven adequately efficient for building low-cost graphics clusters without the need for expensive interconnection network, the recent advances in commodity graphics accelerators for supporting real-time volume rendering have influenced the performance demands of the image compositing step [10]. Even higher demands is expected when using some software acceleration techniques proposed for texture-based volume rendering such as early-ray termination, empty-space skipping and occlusion culling [8, 7]. To fulfill these requirements several hardware devices for image compositing have been proposed [15, 27, 9, 21, 6, 20]. These proposed hardware solutions vary considerably and there is still no standard hardware architecture nor common software API for these devices.

The Scalable Graphics Engine (SGE) [6] works as a network-attached hardware frame buffer which supports high-resolution display systems in single or multiple display units. Although it has the ability to perform image tiling and masking for arbitrary-shaped regions, it does not allow depth or alpha composition. The Metabuffer [27] , Lightning-2 [21], Sepia-2 [9]and SGI Compositor [20] are capable of receiving images directly from the DVI digital output of graphics accelerators and require neither frame buffer read backs nor specialized interface cards for the image I/O process. The latter two have become commercially available as a non-separable part of visualization systems such as SGI Onyx4 (with an SGI Compositor) and HP sv7 (with a Sepia-2 system). The SGI Compositor allows image tiling and the hardware genlock enables active stereoscopic visualization on projection-based high-resolution display systems. However, it does not allow cascade interconnection and will therefore support a maximum of only four rendering pipes. The HP Sepia-2, which first appeared as Sepia (ServerNet Enhanced Parallel Image Accelerator) [12], uses depth information for the image composition process. It has been using high-speed interconnection networks such as *ServerNet* (similar to *Myrinet*) and *Infiniband*, for transmitting the images to be composed. The MPC Compositor [15, 16] requires frame buffer read back and utilizes binary compositing tree architecture, simplifying the compositing order change and reducing the latency to the $O(log(n))$ compared with $O(n)$, or $O(n.log(n))$ when the number of layers is taken into account, in the *recursive Clos structure* used in the Sepia-2 system.

## 3 Commodity Graphics Cluster with MPC Compositor

The MPC compositor has eight inputs and one output channel which communicate to the PCs through the PCI32 interface cards on the PC side. This data transmission is done through the LVDS cables which form a customized interconnection network. To use this hardware compositor, only one PCI slot is required at each PC. Our current device driver will not support a second PCI interface card thus the complete PC cluster configuration for a MPC Compositor will be composed of nine nodes. A complete graphics cluster with MPC Compositor is shown in Figure 1. We built a graphics cluster consisting of nine PCs (Fujitsu W600) interconnected by two different interconnection networks (*Fast Ethernet* and *Gigabit Ethernet*). Eight of the nodes are designed to act as hardware-assisted rendering nodes and one node for display and user interaction. The main hardware components of each node are shown in Table 1. Each rendering node is equipped with the

NVIDIA GeForce FX 5950 ULTRA GPU based graphics board and a specialized PCI interface card for sending the rendering results for the compositor. Each node runs the *RedHat Linux 7.3, kernel 2.4.18-3 with SCore patch*, which makes it possible to utilize the lightweight *PM/Ethernet* communication protocol. The *SCore 5.6.1* was used as the cluster management software, and we utilized C++ and MPI library for building parallel graphics applications and used *MPICH/PM* for running these applications.

Figure 1: Graphics Cluster with MPC Compositor

| Component | Type |
|---|---|
| CPU | Intel Pentium 4 - 2.4 GHz |
| Memory | 1024MB DDR SDRAM PC2100 |
| Chipset | Intel 845E |
| AGP | AGP 4x |
| GPU (Rendering) | NVIDIA GeForce FX5950 Ultra (256MB Video RAM) |
| Network card | Intel EtherExpressPro 100 (on-board)<br>Intel Pro 1000/XT |
| Switching Hub | Fujitsu 1516<br>NetGear GS104 and GS108 |

Table 1: Graphics Cluster's components

### 3.1 MPC Compositor

The MPC compositing hardware is composed of the compositor itself and the PCI interface boards (*IFB In* and *IFB Out*) . As we can see in Figure 2, it works by receiving the 32 bit RGBA images and their priority numbers from the *IFB Out* cards. These priority numbers determine the compositing order. The image composition is performed using alpha blending, and the composited image is sent to the *IFB In* card on the display node for subsequent visualization on a display. The compositing operation based on over operation [17] implemented on the programmable FPGA is compatible with the OpenGL *glBlendFunc( GL_SRC_ALPHA, GL_ONE_MINUS_ SRC_ALPHA)* function. The MPC Compositor supports three different channel selections for input and output: one of eight inputs and one output (8:1), two of four inputs and one output (4:1) and four of two inputs and one output (2:1). The latter two enable it to be used in a multi-display system composed of two or four displays, or in a stereoscopic display system composed of one or two displays. Low Voltage Differential Signaling (LVDS) is used for transmitting the compositing information and RGBA pixel values through the PCI interface cards. Thus, the network requirement is drastically reduced. Because we use the sort-last parallel rendering approach, the interprocessor communication is required only for the initial volume data distribution and for the software synchronization signal distribution when using *MPI_Barrier*. As a result, there is no need for specialized and expensive high-speed networks such as *Myrinet* or *Infiniband* as used in HP Sepia-2.

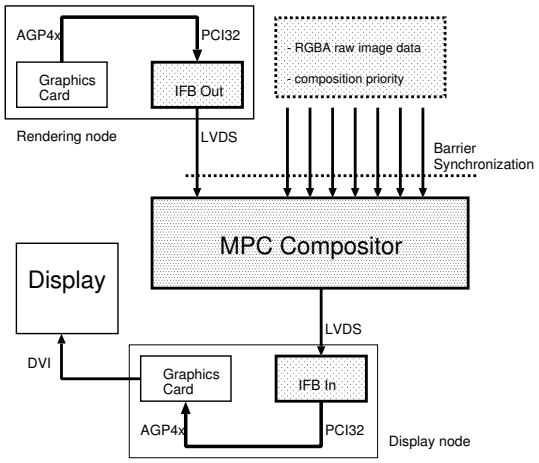


Figure 2: Image Composition Data Flow

The detailed characteristics of this compositor can be verified in [15, 16]. However, some important points should be understood in order to appreciate the possibilities and limitations. This image compositor works as a brute-force RGBA image compositor which can composite a maximum size of 2048x2048 RGBA image at each step. For each input image the compositor requires the priority information for correct alpha blending order (Figure 3). This priority information is just a value which corresponds to the virtual distance from the viewpoint. The sub-images generated at the rendering nodes need to be synchronized before being sent to the compositor, when using *MPI_Barrier* (software synchronization), and the compositing performance will directly depend on the performance of the interconnection network. The expected synchronization overhead for each compositing step is shown in Table 2. The maximum buffer size which we can allocate on the PCI interface cards make it viable to send a RGBA image data with a maximum size of 2048x2048 which is sufficient for most commercially available LCD displays, but is insufficient for compositing the full image in the 9.2 million-pixel IBM 22.2 inch LCD display. Because of the utilization of PCI32 interface, the I/O performance will depend directly on the hardware characteristics of the PC's motherboard especially the chipset.

Table 2: Synchronization overhead for each *MPI_Barrier*

| Network | Latency |
|---|---|
| Fast Ethernet | 1.29 - 5.5 ms. |
| Gigabit Ethernet | 0.49 - 0.61 ms. |

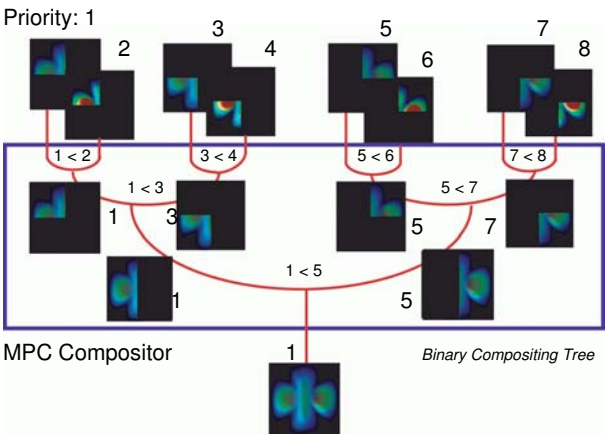


Figure 3: Image Composition using alpha blending and priority information

### 3.2 Hardware-Assisted Barrier Synchronization

This graphics cluster does not require a high-end interconnection network for the image compositing process because it uses LVDS cables to communicate the necessary data for compositing. However, the MPI software synchronization (*MPI_Barrier*) used in this experiment is highly affected by on the data traffic and performance of the interconnection network. As we can see in Table 2, this effect is not so prominent when using *Gigabit Ethernet* but it was easy to verify on *Fast Ethernet* even when isolated from the external network. As a result, inconsistent compositing performance is expected when using this kind of interconnection network. In order to eliminate this undesirable effect when using a low power network we have implemented a low-cost parallel port based barrier synchronization (Figure 4). This device is based on *TTL_PAPERS* [4], and as *SoftGenLock* [2] we have used a reprogrammable logic device. We used a simple, low-cost 32 macro-cell ALTERA PM7032 FPGA. We have implemented a hardware synchronization library which uses LPT poling instead of catching LPT hardware interruption (such as IRQ 7). By using this synchronization hardware and this library, it is possible to obtain the software synchronization performance of *Gigabit Ethernet* even using *Fast Ethernet*. In addition, it may be modified to support software *genlock* in order to use active stereo projection when using the MPC Compositor in the dual four inputs and one output mode.

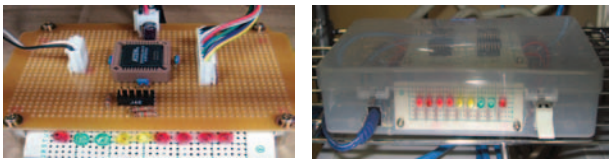

Figure 4: Hardware-assisted barrier synchronization devices

### 3.3 Performance Evaluation

In order to simplify the performance evaluation of the image composition process we will split the volume rendering process from the image compositing process from now on. Considering the utilization of the hardware-assisted volume rendering, the image compositing process will start by obtaining the image data generated from the graphics boards (GB), of the rendering nodes, through the OpenGL *glReadPixel* command. These sub-images are then sent through the *IFB Out* PCI interface cards to the compositor for alpha blending and will be received by the display node through the *IFB input* PCI interface card, then sent to the frame buffer through the *glDrawPixel* command for subsequent visualization (Figure 5).

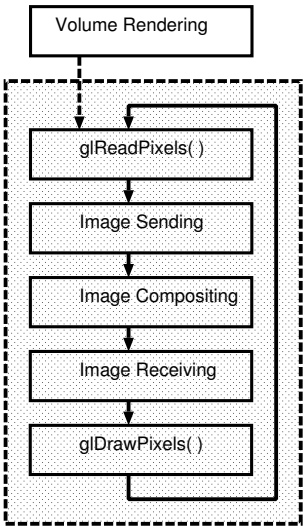


Figure 5: Evaluated Image Compositing Stages

The achieved average framebuffer readback performance for different sizes of synthetic 32 bit RGBA images (Figure 6) is shown in Table 3. We can observe that it is only one fifth of the theoretically available AGP4x bus bandwidth of 1GB/s. This image data is sent to the *IFB Out card* through the PCI32 bus with theoretically available bandwidth of 133 MB/s. In addition, the shared nature of the PCI bus has the potential to decrease even further this data transmission performance.

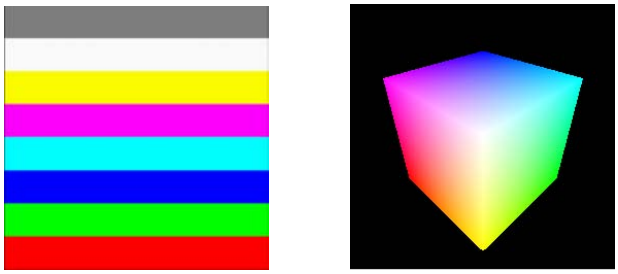

Figure 6: Composited synthetic images

Table 3: Framebuffer readback performance

| Size | Time | Rate |
|---|---|---|
| 256 x 256 | 1.5 ms | 168 MB/s |
| 512 x 512 | 5.9 ms | 169 MB/s |
| 1024 x 1024 | 22.5 ms | 178 MB/s |
| 1600 x 1200 | 40.9 ms | 183 MB/s |

We measured the performance of MPC Compositor with image sizes usually used for performance evaluation of volume rendering algorithms: 256x256, 512x512 and 1024x1024. In addition, we used some of the more common video resolutions of current monitors and LCD projectors: XGA(1024x768), SXGA(1280x1024)

and UXGA(1600x1200). For each measurement, these images were loaded into the *IFB Out* interface cards of the rendering nodes and sent to the compositing unit one after the other without delay. The achieved sustained maximum frame rate for image compositing without considering the image load and display (*glReadPixels* and *glDrawPixels* stages), for the two different interconnection networks present in the graphics cluster is shown in Table 4. The performance variation between the *Gigabit* and *Fast Ethernet* is due to the synchronization overhead while using software based barrier synchronization (*MPI_Barrier*). From this result, we can verify that real-time image compositing (over 30 FPS) is only possible for image sizes smaller than 512 x 512. It is important to remember that this achieved framerate does not include the time required for rendering. Considering that the volume rendering and the framebuffer readback cannot overlap, the total time $T$ for processing one frame will be $T = T_{render} + T_{compositing}$ . Ogata et al. [16] reported performance of up to 25 FPS when using VolumePro 500, a specialized volume rendering board, to render $512^3$ volume data to 512 x 512. We obtained almost constant frame rate while using *Gigabit Ethernet*. However, the variation in the achieved compositing speed became high when using *Fast Ethernet* even when isolated from external network. This variation was especially pronounced during the composition of images with sizes smaller than 512x512. However, we can easily eliminate this undesirable effect by using a low-cost hardware synchronization device, such as that presented in section 3.2.

Table 4: Maximum Image Compositing Performance

| Network | $256^2$ | $512^2$ | $1024^2$ |
|---|---|---|---|
| Gigabit Ethernet | 155.0 FPS | 50.0 FPS | 13.8 FPS |
| Fast Ethernet | 107.0 FPS | 32.0 FPS | 11.5 FPS |
| Fast Ethernet + Hardware Barrier | 157.3 FPS | 51.2 FPS | 13.8 FPS |

| Network | XGA | SXGA | UXGA |
|---|---|---|---|
| Gigabit Ethernet | 18.2 FPS | 11.1 FPS | 7.6 FPS |
| Fast Ethernet | 15.5 FPS | 9.5 FPS | 6.9 FPS |
| Fast Ethernet + Hardware Barrier | 18.2 FPS | 11.1 FPS | 7.6 FPS |

## 4 Hybrid Hardware-Accelerated Image Compositing

Although the MPC Compositor has high scalability, allowing us to build a massively parallel visualization cluster just by connecting these devices hierarchically in cascade as seen in Figure 7, the cost becomes prohibitively expensive for building such large visualization systems. On the other hand, a minimal system consisting of one image compositor will limit the volume subdivision to a maximum of eight because this hardware does not support bi-directional communication between the IFB PCI cards. As a result, it can be insufficient for handling large volumetric data sets where the quantity of sub-volumes surpasses the available rendering nodes. Perhaps the simplest way for emulating this cascade interconnection is to feed back the partially composited image to the rendering nodes. However, the partially composited images should be transmitted through the available interconnection network because the MPC Compositor does not allow bi-directional communication. In the next subsection we will discuss the use of image feedback for emulating this cascade interconnection by using only one MPC Compositor.

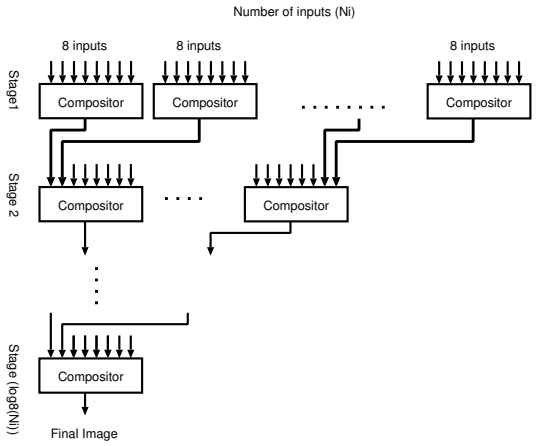


Figure 7: Cascade interconnection of the MPC Compositors

### 4.1 Feedback Image Compositing

A simple approach for emulating the cascade interconnection of MPC Compositors (Figure 7) by using only one MPC Compositor is shown in Figure 8. From the second stage, the composited image obtained from the MPC Compositor is sent back to a rendering node, chosen specifically for feedback, through the available interconnection network on the PC cluster. By using this rendering node for "image feedback" purposes, we will lose one node for rendering. By separating the rendering nodes from this feedback node, it is possible to overlap the image receiving process with the rendering process on the remaining seven rendering nodes. However, the overhead for sending back the partially composited image data through the available interconnection network makes this method less attractive. The image compositing performance when using image feedback, by using *MPI_Send* and *MPI_Recv* functions, is shown in Table 5. We can observe that the performance gain from the hardware compositor will be canceled out by the communication overhead. Instead of sending back the composited images we focused on the use of standard OpenGL functions available on graphics boards to assist the remaining blending process (Figure 9). An increase in performance can be expected when using more lightweight network protocol such as QUANTA [5] or using direct connection through the *IFB PCI cards.* We achieved a transfer time of 358 milliseconds by using *MPI_Send* and *MPI_Recv* for sending a 1024 x 1024 RGBA image through a gigabit ethernet network. We could reduce this time to 30 milliseconds by using QUANTA library for sending the same 1024x1024 RGBA image through this gigabit ethernet network. We also achieved the transfer time of 63 milliseconds for sending a 1280x1024 RGB image between two IFB PCI cards. In addition, acceleration methods for binary-swap image composition such as bounding rectangle and compression can also be used for diminishing the data size for transmission.

### 4.2 Hybrid Image Compositing

Taking into consideration that the OpenGL alpha blending function, *glBlendFunc( GL_SRC_ALPHA , GL_ONE_MINUS_ SRC_ALPHA )*, is implemented on the MPC Compositor's FPGA, we focused on executing the same function, by software or hardware, on the display node side in order to avoid image feedback. The OpenGL alpha blending function *glBlendFunc( GLenum sfactor , GLenum dfactor )* controls how color values in the fragment being processed (the source) are combined with those already stored in the framebuffer (the destination). The argument *sfactor* in-

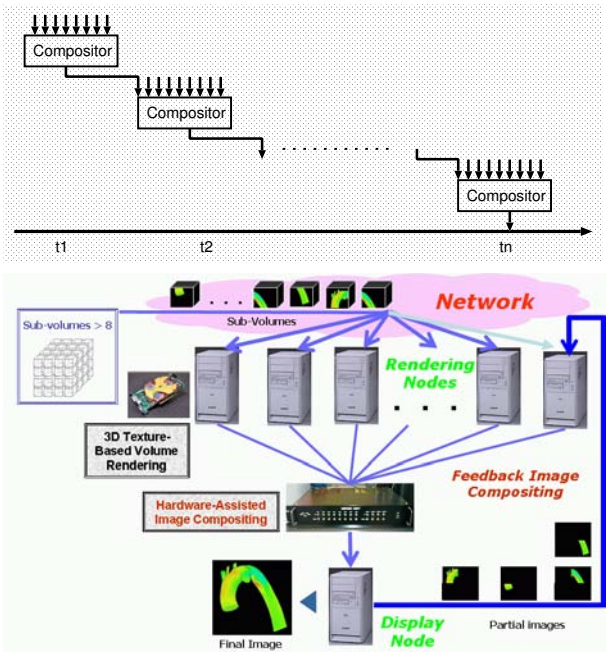


Figure 8: Feedback image compositing scheme

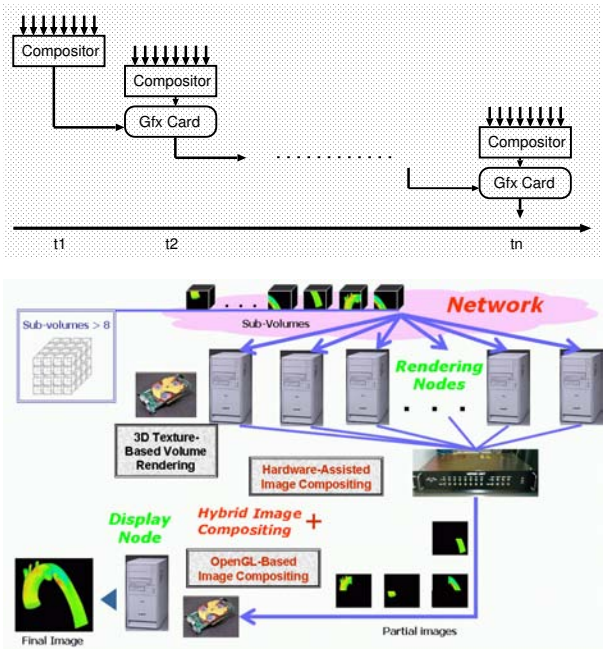


Figure 9: Hybrid Image compositing scheme

Table 5: Feedback Image Compositing Performance

| Network | $256^2$ | $512^2$ | $1024^2$ |
|---|---|---|---|
| Gigabit Ethernet | 96.6 FPS | 28.2 FPS | 7.3 FPS |
| Fast Ethernet | 31.0 FPS | 8.3 FPS | 2.1 FPS |
| Fast Ethernet + Hardware Barrier | 32.9 FPS | 8.8 FPS | 2.1 FPS |

| Network | XGA | SXGA | UXGA |
|---|---|---|---|
| Gigabit Ethernet | 9.7 FPS | 5.9 FPS | 4.1 FPS |
| Fast Ethernet | 2.7 FPS | 1.7 FPS | 1.1 FPS |
| Fast Ethernet + Hardware Barrier | 2.8 FPS | 1.7 FPS | 1.1 FPS |

dicates how to compute a source blending factor; *dfactor* indicates how to compute a destination blending factor [25]. The alpha blending function implemented on the MPC Compositor is, like other OpenGL blending operators, an associative function and can be computed sequentially. However, it is not commutative and, as a result, the compositing order cannot be changed. Considering that the blending factors of source and destination RGBA images are $(Sr,Sg,Sb,Sa)$ and $(Dr,Dg,Db,Da)$, respectively, the resulting RGBA image obtained from source RGBA image $(Rs,Gs,Bs,As)$ and destination RGBA image $(Rd,Gd,Bd,Ad)$ will be $(RsSr+RdDr,GsSg+GdDg,BsSb+BdDb,AsSa+AdDa)$. Taking this into consideration, the blending factor *GL_SRC_ALPHA* on the implemented alpha blending function will be $(As,As,As,As)$ and the blending factor *GL_ONE_MINUS_SRC_ALPHA* will be $[(1,1,1,1)-(As,As,As,As)]$. The diagram of this hybrid hardware-assisted image compositing is shown alongside Figure 9.

The total delay time $T_N$ when using traditional cascade interconnection is represented as :

$$T_N = T_d\lceil log_8 N_i\rceil$$

Where $T_d$ corresponds to the delay time generated from the compositing hardware and $N_i$ is the number of inputs to the compositor [16]. Substituting the compositing costs of stages two onwards, the total delay time $T_N$ for the hybrid compositing approach can be represented as :

$$T_N = T_d\lceil log_8 N_i\rceil + T_O(\lceil log_8 N_i\rceil - 1)$$

Where $T_O$ corresponds to the cost of the OpenGL alpha blending function executed on the graphics board.

The obtained compositing performance using NVIDIA GeForce FX5950 ULTRA GPU based graphics board is shown in Table 6. From this result, we can verify that real-time image compositing is only possible for image sizes smaller than 512x512. However, it is still possible to composite an image size of 1024x1024 at an interactive frame rate.

Table 6: Hybrid Image Compositing Performance

| Network | $256^2$ | $512^2$ | $1024^2$ |
|---|---|---|---|
| Gigabit Ethernet | 142.1 FPS | 45.2 FPS | 12.2 FPS |
| Fast Ethernet | 115.4 FPS | 35.0 FPS | 11.3 FPS |
| Fast Ethernet + Hardware Barrier | 144.6 FPS | 45.6 FPS | 12.2 FPS |

| Network | XGA | SXGA | UXGA |
|---|---|---|---|
| Gigabit Ethernet | 16.1 FPS | 10.0 FPS | 7.1 FPS |
| Fast Ethernet | 14.2 FPS | 9.1 FPS | 6.7 FPS |
| Fast Ethernet + Hardware Barrier | 16.1 FPS | 10.0 FPS | 7.1 FPS |

In order to verify the OpenGL alpha blending performance difference between graphics boards, we also used an ATI Radeon 9800 PRO GPU based graphics board. The obtained results are shown in Table 7 and we verified almost no difference in performance between them.

Table 7: Performance Comparison.

| Screen size | GeForce FX5950 | Radeon 9800 |
|---|---|---|
| 256 x 256 | 142.1 FPS | 141.9 FPS |
| 512 x 512 | 45.2 FPS | 45.9 FPS |
| 1024 x 1024 | 12.2 FPS | 12.3 FPS |

### 4.3 High-Resolution Display Systems

The ever increasing size of volume data has increased its necessity to be visualized in high-resolution and sometimes in immersive manner. Recently, some high-resolution LCD displays with WUXGA (1920x1200) resolution have become available, and cost-effective DLP projectors have become available for building low-cost stereoscopic projection systems and scalable projection-based display systems. Some hardware and software solutions for handling such mega-pixel displays have been proposed. Klosowski et al. [6] proposed the SGE which works as a large virtual frame-buffer for supporting the resolution of 3840 x 2400 on the IBM T221 LCD display. Moreland et al. [14] proposed an efficient software solution for handling mega-pixel displays such as scalable projection-based display systems. We have focused on a more simple solution taking into consideration the MPC Compositor's characteristics. Although the MPC Compositor supports a maximum image composition of 2048x2048 RGBA images in one single step, it is still insufficient for compositing the full image in the 9.2 million-pixel IBM 22.2 inch LCD display (Figure 10), and therefore requires more than one step for compositing by using a simple approach such as shown in Figure 11. Although it is possible to composite such a high resolution image the performance degradation is considerable. The quantity of steps will be directly proportional to the available buffer memory for compositing. This compositor allows multiple outputs which can support tiled displays or stereoscopic projection based displays. In the first case, it is theoretically possible for the four outputs to display UXGA (1600x1200) image size each which makes it possible to generate a 3200x2400 image. However, the rendering performance will be drastically affected because there will remain only two rendering nodes for each output. Using four rendering nodes for each of two outputs, it is possible to generate each of the dual UXGA image separately. These images also correspond to a single stereoscopic UXGA image. In this case, even when compositing two images simultaneously the performance penalty will not be so high because the compositing process is done in one single step.

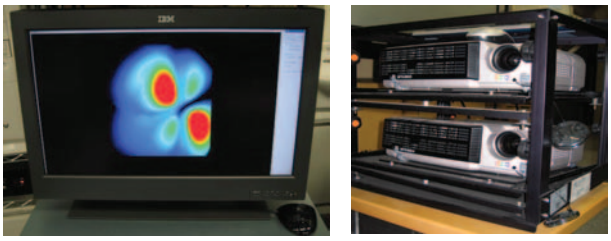

Figure 10: High resolution / Stereoscopic display systems

## 5 Future Works

In the future we are planning to integrate this method with the data streaming based volume rendering system proposed in [24]. This remote visualization system works in a client-server manner, by breaking large volume data into sub-volumes with consideration

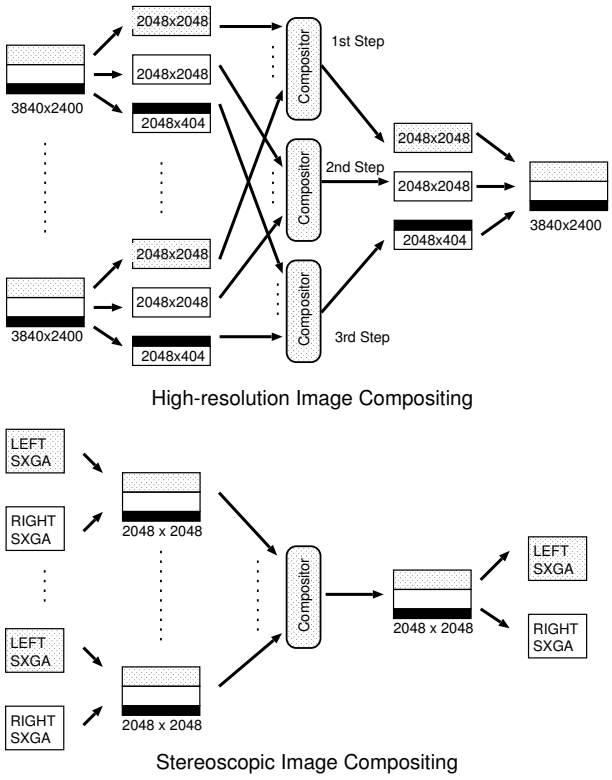


Figure 11: High resolution / Stereoscopic Image Composition

to the available resource (texture memory) on the client side. We think that this hybrid image compositing scheme can be an interesting solution for assisting this parallel volume rendering for handling large data sets where the quantity of sub-volumes can easily surpass the available rendering nodes (eight). By using this visualization framework, we expect to obtain more realistic results such as those in real world situations including both high-resolution and stereoscopic visualization. Some preliminary tests have been done using a shared-memory multiprocessor machine (Fujitsu PRIMEPOWER HPC 2500) as a server, however at the time of writing this paper we only tested datasets smaller than $512^3$.

## 6 Conclusion

In this paper, we have presented the hybrid hardware-accelerated image compositing method for sort-last parallel rendering on a visualization cluster with a MPC compositor. Our approach was focused on using a minimal hardware configuration taking cost into consideration while retaining the ability to handle large data sets which surpass the number of available rendering nodes. The utilized visualization cluster was built using commercially available image compositor and commodity graphics cards for texture-based volume rendering. In addition, the simple yet efficient multi-step image compositing method was also proposed for supporting high-resolution and immersive visualization where the degradation of software image compositing solutions is accentuated. This image composition method has the potential to meet the ever increasing image compositing performance demands brought about by the advances in both hardware and software solutions for hardware-assisted parallel volume rendering. Better results are expected when the next-generation of this compositing device using faster PCI technology such as PCI-X and PCI-Express becomes commercially available.

## ACKNOWLEDGEMENTS

We greatly appreciated the valuable advice and comments from Kwan-Liu Ma (University of California, Davis) who assisted in clarifying the directions of this work. We would like to thank Shin-ichiro Mori (Kyoto University) for his valuable comments and advice on hardware rendering and compositing. Thanks also to Shiguenobu Nonaka (Tohki Electronics) for his helpful assistance and advice about FPGA and PC hardware. We wish to thank Takuji Nakajima and Atsuji Ogasa (Fujitsu Limited), Yoshiyuki Kobayashi and Katsuhiko Hirota (Fujitsu Nagano Systems Engineering) for their helpful assistance in the development of graphics application. Thanks also to the staff of the Academic Center for Computing and Media Studies (ACCMS) and the Center for the Promotion of Excellence in Higher Education (CPEHE) of Kyoto University for their valuable assistance. This work has been supported in part by the Japanese Ministry of Education, Science, Sports and Culture Grant-in-Aid for IT Program, 2004.

## REFERENCES

[1] James Ahrens and James Painter. Efficient sort-last rendering using compression-based image compositing. In *Second Eurographics Workshop on Parallel Graphics and Visualization*, pages 33–40, 1998.

[2] Jeremie Allard, Valerie Gouranton, Guy Lamarque, Emmanuel Melin, and Bruno Raffin. Softgenlock: Active stereo and genlock for pc cluster. In *Proceedings of the Joint IPT/EGVE'03 Workshop*, Zurich, Switzerland, May 2003.

[3] Karl Czajkowski, Alper K. Demir, Carl Kesselman, and Marcus Thiebaux. Practical resource management for grid-based visual exploration. In *Proceedings of The 10th IEEE Symposium on High Performance Distributed Computing*, pages 416–424, 2001.

[4] H. G. Dietz, R. Hoare, and T. Mattox. A fine-grain parallel architecture based on barrier synchronization. In *Proceedings of the International Conference on Parallel Processing*, pages 247–250, 1996.

[5] Eric He, Javid Alimohideen, Josh Eliason, Naveen K. Krishnaprasad, Jason Leigh, Oliver Yu, and Thomas A. Defanti. Quanta: A toolkit for high performance data delivery over photonic networks. *Future Generation Computer Systems*, 19(6):919–933, 2003.

[6] James T. Klosowski, Peter D. Kirchner, Julia Valuyeva, Greg Abram, Christopher J. Morris, Robert H. Wolfe, and Thomas Jackman. Deep view: High-resolution reality. *Computer Graphics and Applications*, 22(3):12–15, 2002.

[7] J. Kruger and R. Westermann. Acceleration techniques for gpu-based volume rendering. In *Proceedings IEEE Visualization 2003*, pages 287–292. IEEE Computer Society Press, 2003.

[8] Wei Li, Klaus Mueller, and Arie Kaufman. Empty space skipping and occlusion clipping for texture-based volume rendering. In *Proceedings IEEE Visualization 2003*, pages 317–324. IEEE Computer Society Press, 2003.

[9] Santiago Lombeyda, Laurent Moll, Mark Shand, David Breen, and Alan Heirich. Scalable interactive volume rendering using off-the-shelf components. In *Proceedings of the IEEE 2001 symposium on parallel and large-data visualization and graphics*, pages 115–121. IEEE Press, 2001.

[10] Kwan-Liu Ma, Eric B. Lum, and Shigueru Muraki. Recent advances in hardware-accelerated volume rendering. *Computer & Graphics*, 27:725–734, 2003.

[11] Kwan-Liu Ma, James S. Painter, Charles D. Hansen, and Michael F. Krogh. Parallel volume rendering using binary-swap image composition. *Computer Graphics and Application*, 14(4):59–68, 1994.

[12] Laurent Moll, Mark Shand, and Alan Heirich. Sepia: Scalable 3d compositing using pci pamette. In *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines*, pages 146–155. IEEE Press, 1999.

[13] Steven Molnar, Michael Cox, David Ellsworth, and Henry Fuchs. A sorting classification of parallel rendering. *IEEE Computer Graphics and Applications*, 14(4):23–32, 1994.

[14] Kenneth Moreland, Brian Wylie, and Constantine Pavlakos. Sort-last parallel rendering for viewing extremely large data sets on tile displays. In *Proceedings of the IEEE 2001 symposium on parallel and large-data visualization and graphics*, pages 85–92. IEEE Press, 2001.

[15] Shigeru Muraki, Masato Ogata, Kwan-Liu Ma, Kenji Koshizuka, Kagenori Kajihara, Xuezhen Liu, Yasutada Nagano, and Kazuro Shimokawa. Next-generation visual supercomputing using pc clusters with volume graphics hardware devices. In *Proceedings of the 2001 ACM/IEEE conference on Supercomputing (CDROM)*, pages 51–51. ACM Press, 2001.

[16] Masato Ogata, Shigeru Muraki, Xuezhen Liu, and Kwan-Liu Ma. The design and evaluation of a pipelined image compositing device for massively parallel volume rendering. In *Proceedings of the 2003 Eurographics/IEEE TVCG Workshop on Volume graphics*, pages 61–68. ACM Press, 2003.

[17] Thomas Porter and Tom Duff. Compositing digital images. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 253–259. ACM Press, 1984.

[18] Erik Reinhard and Chuck Hansen. A comparison of parallel compositing techniques on shared memory architectures. In *Proceedings of the Third Eurographics Workshop on Parallel Graphics and Visualisation*, pages 115–123, 2000.

[19] Kentaro Sano, Yusuke Kobayashi, and Tadao Nakamura. Differential coding scheme for efficient parallel image composition on a pc cluster system. *Parallel Computing*, 30(2):285–299, 2004.

[20] SGI. *SGI InfinitePerformance: Scalable Graphics Compositor Owner's Guide (Hardware)*, February 2002.

[21] Gordon Stoll, Matthew Eldridge, Dan Patterson, Art Webb, Steven Berman, Richard Levy, Chris Caywood, Milton Taveira, Stephen Hunt, and Pat Hanrahan. Lightning-2: a high-performance display subsystem for pc clusters. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 141–148. ACM Press, 2001.

[22] Aleksander Stompel, Kwan-Liu Ma, Eric B. Lum, James Ahrens, and John Patchett. Slic: Scheduled linear image compositing for parallel volume rendering. In *Proceedings of the IEEE Symposium on Parallel and Large-Data Visualization and Graphics*, pages 33–40, 2003.

[23] Akira Takeuchi, Fumihiko Ino, and Kenichi Hagihara. An improved binary-swap compositing for sort-last parallel rendering on distributed memory multiprocessors. *Parallel Computing*, 29(11-12):1745–1762, 2003.

[24] Yasuhiro Watashiba, Jorji Nonaka, Naohisa Sakamoto, Yasuo Ebara, Koji Koyamada, and Masanori Kanazawa. A streaming-based technique for volume rendering of large datasetes. In *Proceedings of The 6th IASTED International Conference on Computers, Graphics and Imaging*, pages –, 2003.

[25] Mason Woo, Jackie Neider, and Tom Davis. *OpenGL Programming Guide: The Official Guide to Learning OpenGL*. Pearson Education, 1997.

[26] Don-Lin Yang, Jen-Chih Yu, and Yeh-Ching Chung. Efficient compositing methods for the sort-last-sparse parallel volume rendering systems on distributed memory multicomputers. *The Journal of Supercomputing*, 18(2):201–220, 2001.

[27] Xiaoyu Zhang, Chandrajit Bajaj, and William Blanke. Scalable isosurface visualization of massive datasets on cots clusters. In *Proceedings of the IEEE 2001 symposium on parallel and large-data visualization and graphics*, pages 51–58. IEEE Press, 2001.