UC Davis
IDAV Publications

Title

The Design and Evaluation of a Pipelined Image Compositing Device for Massively Parallel Volume Rendering

Permalink

https://escholarship.org/uc/item/0ns753nz

Authors

Ogata, Masato
Murakin, Shigeru
Liu, C. C.
et al.

Publication Date

2003

Peer reviewed

eScholarship.org Powered by the California Digital Library
University of California

Volume Graphics (2003)
I. Fujishiro, K. Mueller, A. Kaufman (Editors)

The Design and Evaluation of a Pipelined Image Compositing Device for Massively Parallel Volume Rendering

Masato Ogata[1*], Shigeru Muraki[2†], Xuezhen Liu[1*] and Kwan-Liu Ma[3‡]

[1] Research and Development Department, Mitsubishi Precision Co., Ltd., Kamakura, Kanagawa, Japan
[2] The Collaborative Research Team of Volume Graphics, National Institute of Advanced Industrial Science and Technology, Tokyo, Japan
[3] Department of Computer Science, University of California, Davis, California, USA

Abstract

An experimental study of software image compositing that we have carried out on a 512-node PC cluster shows the necessity of hardware compositing support to make possible real-time volume visualization scalable with large PC clusters. This paper describes the design and performance evaluation of such a hardware image compositing device. A PC cluster using such devices along with commodity graphics cards can enable simultaneous simulation and volume visualization.

Categories and Subject Descriptors (according to ACM CCS): I.3.1 [Computer Graphics]: Hardware Architecture – Parallel Processing; I.3.2 [Computer Graphics]: Graphics Systems – Distributed/Network Graphics

1. Introduction

Large-scale scientific simulations are increasingly being run on low-cost PC clusters instead of expensive vector supercomputers. There is also a growing interest in conducting simultaneous simulation and visualization on the same PC cluster. To offer this capability, the 3D graphics power of the PC cluster system must be enhanced. This paper presents the design and performance of a hardware image compositing device that we have developed to support scalable, real-time parallel volume rendering.

For the design of the hardware image compositing device, a comprehensive experimental study of software image compositing was performed on a 512-node PC cluster. The test results not only guided our design but also justified the use of hardware image compositing for scalable real-time rendering. The resulting compositing device has an 8-input-1-output pipelined design. We have built a volume graphics enhanced PC cluster system using this hardware compositing device and NVIDIA GeForce 4 cards [1].

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 presents experimental results, which show the bottleneck in the software compositing process, on a 512-node PC cluster. Section 4 describes our design principles in response to the experimental results. The architecture of the compositing hardware is given in Section 5. Section 6 presents the performance of the hardware device as well as the PC cluster system using this device.

[*]345 Kamimachiya, Kamakura, Kanagawa, Japan 247-8505;
e-mail: {ogata, liu}@mpcnet.co.jp
[†]2-41-6 Aomi, Koto-ku, Tokyo, Japan 135-0064;
e-mail: s-muraki@aist.go.jp
[‡]One Shields Avenue, Davis, CA95616-8562, USA;
e-mail: ma@cs.ucdavis.edu

2. Related Work

In object-space parallel rendering, the image compositing often dominates the overall rendering performance, since it requires inter-processor communication. Parallel image compositing has thus been an active area of research and both software and hardware approaches have been proposed. For polygon parallel rendering, a hybrid sort-first and sort-last approach has been investigated [2]. For volume rendering, a sort-last configuration is generally used [3]. The binary-swap compositing (BSC) algorithm introduced by Ma et al. [4] has been adopted by many parallel-rendering systems. Lee et al. have developed a pipelined algorithm that is particularly efficient for mesh networks [5]. Lightning-2 [6] and Sepia-2 [7] are two hardware image compositing designs. Lightning-2 provides a DVI-to-DVI interface and employs scan-line based pixel mapping. Sepia-2 is a custom PCI card using pipelined associated blending operations.

It is worth noting that the PC cluster built with Sepia-2 uses ServerNet-2, a high-speed network based on a hierarchical

© The Eurographics Association 2003.

switched topology, with one Sepia-2 board for every graphics accelerator. For a large PC cluster, ServerNet-2, a non-blocking Clos switch network [8, 9], can scale efficiently; however, the multi-stage communication inherent in the compositing process is like to incur a significant communications overhead and the extra wiring needed could be costly. As shown below, our compositing hardware design results in an overall low-cost and simple system configuration.

3. The Compositing Bottleneck Using Software

The BSC algorithm has been shown to demonstrate scalable performance on large tree networks [4]. When BSC is used, an object-space partitioning scheme needs to be employed, and we must be able to determine the image compositing in a straightforward manner. The compositing process, which pairs up processors in order of compositing, starts as soon as all processors complete rendering of their local sub-volumes. If there is a total of N_P processors, then the compositing process takes $\lceil \log_2(N_P) \rceil$ stages, as in simple binary compositing. However, BSC is more efficient than simple binary compositing as a result of keeping all the processors involved in the full course of the compositing and by reducing the image data that must be transferred and computed while moving up the compositing tree. This is done, at each compositing stage by having the two processors involved in a composite operation that splits the image plane into two pieces, with each processor taking responsibility for one of the two pieces. Figure 1 illustrates BSC when eight processors are used.

PC clusters that commonly employ switch networks are unable to attain comparable scalable rendering performance, especially when hardware-accelerated rendering is used. Furthermore, BSC is less optimal when the number of processors utilized is not a power of two. To assist our design of compositing hardware, we evaluated an implementation of BSC for volume rendering on the SCore Cluster III [10].

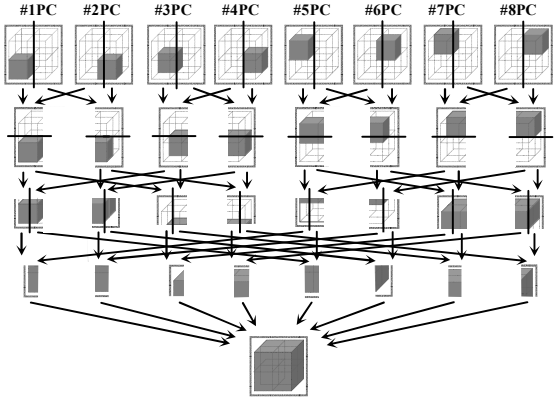


Figure 1: *The principle of parallel composition of Binary-Swap-Compositing*

3.1. Experiment on Software Image Compositing

The purpose of our performance study is to gain an understanding of the communication requirements of BSC on a typical high-performance PC cluster like the SCore III. We investigated whether it is possible to maintain high frame rates (e.g., 30 frames per second) as more processors are added to cope with larger problems. The SCore Cluster III is a PC cluster developed by the Real World Computing Partnership (RWCP). The system specification of SCore III is shown in Table 1. In the system, the Cluster System Software, SCore, is used. Since the high performance communication library PMv2 and a Myrinet network are used, the communication overhead is only 7μs with MPI (Message Passing Interface).

Table 1: *The specification of RWCP SCore Cluster III*

CPU	Pentium III 933MHz (512PC x 2CPU)
Memory	512MB
Network	Myrinet 2GBits/sec
OS	SCore4.2.1 (Real World Computing Partnership)
	Linux Kernel 2.2.16

The parameters of the volume data used in the experiment are shown in Table 2, and the generated images are shown in Figure 2 (1) and (2) respectively. Total processing time T can be divided into rendering time T_r and compositing time T_c. Thus, the total processing time can be expressed as:

$$T = T_r + T_c . \quad (1)$$

T_r, T_c, and T, which are proportional to screen size, are shown in Figure 3. The comparison between T and T_c is shown In Figure 3 (a) and (b), and that of T and T_r is shown in Figure 3 (c) and (d).

The maximum and minimum values of the compositing time measured in the experiment are shown in Table 3.

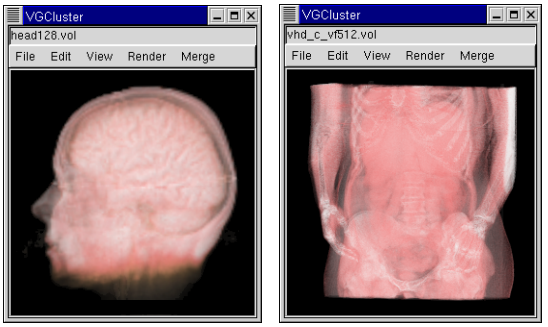
Table 2: *The conditions of the time measurement.*

Test No.	Volume data (voxels)	Screen size (pixels)
(1)	Human head (128 x 128 x 128)	256^2
		512^2
		1024^2
(2)	Human body (512 x 512 x 512)	256^2
		512^2
		1024^2

Table 3: *The minimum and maximum processing time for composition using BSC. (ms)*

Dataset	Screen size (pixels)					
	256^2		512^2		1024^2	
	Min	Max	Min	Max	Min	Max
Human head	7.6	31.3	39.2	80.7	161	333
Human body	6.8	31.9	37.7	81.1	162	334

© The Eurographics Association 2003.



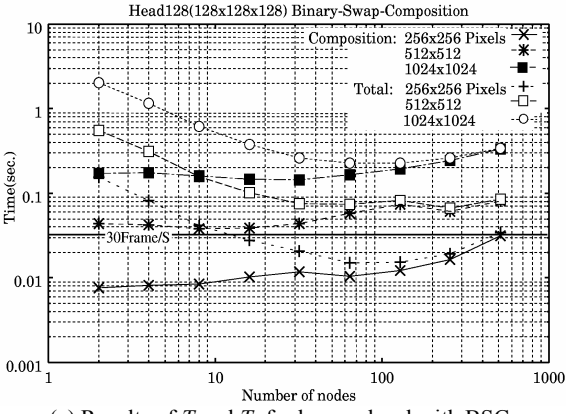
(1) Human head 128^3 (2) Human body 512^3

Figure 2: *The images of the volumes used in the experiment.*

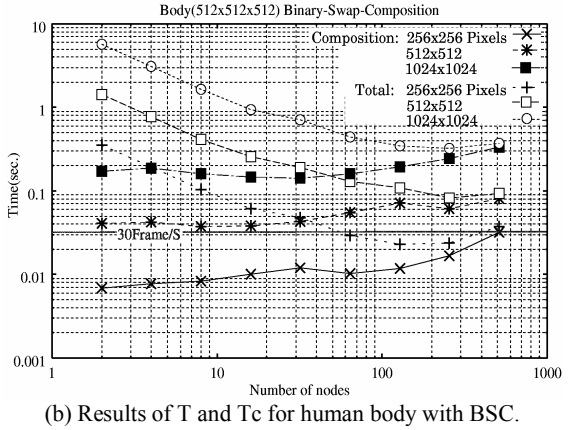
3.2. Image generation bottleneck at video rate

To speed up the rendering process, it is necessary to increase the number of processors. However, the compositing overhead also increases with the increase of the number N_P of node PCs (See Figure 3 (a) and (b)).

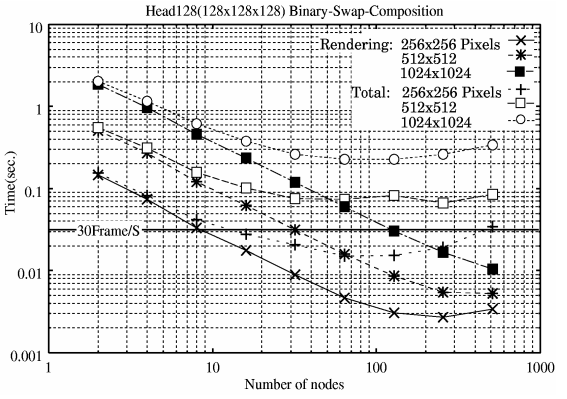
In the software implementation of BSC method, not only T_r but also T_c is parallelized. The rendering time decreases by T_r/N_P corresponding to the number of N_P, as shown in Figure 3 (c) and (d). Conversely, the compositing time increases after a slightly decreasing.



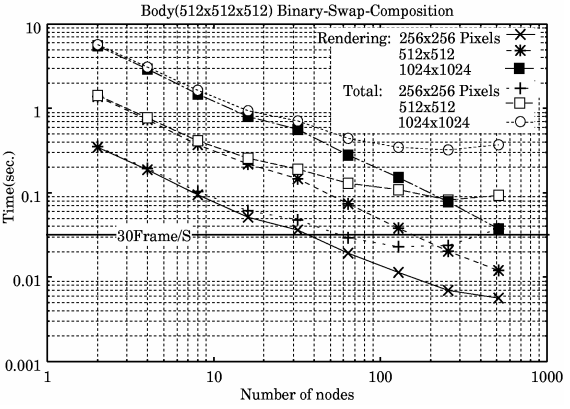
(a) Results of T and T_c for human head with BSC.



(b) Results of T and Tc for human body with BSC.



(c) Results of T and T_r for human head with BSC.



(d) Results of T and T_r for human body with BSC.

Figure 3: *Measured BSC time (seconds).*

Below is our understanding of the compositing time increase based on our experimental results. It is logical to assume that the compositing process includes both processes that can be parallelized and those that cannot. Provided the compositing time which can be parallelized is T_{cp}, and the compositing time which can not be parallelized is T_{cf}, T_c can be expressed as:

$$T_c = T_{cp} / N_P + T_{cf} \quad . \tag{2}$$

It is clear that the compositing time increases when the number of processors is increased in Figure 3 (a) and (b), since the second term T_{cf} in Equation (2) increases. This is called the communications overhead. The compositing time for a size of 512^2 pixels is about 40 ms. That is to say, in a massively parallel environment, there is a bottleneck preventing the realization of image generation at video rate by software implementation even though the BSC method is used.

4. Proposed Hardware

For the above reasons, it is necessary to use a special type of hardware for image generation at video rate. The necessary

© The Eurographics Association 2003.

functions and performances for implementations are as follows.

4.1. Reduction of scale of circuitry

- Reduction of number of communication times for each node
  The Sepia-2 system, acquiring the compositing data in the compositing phases via the special network switch the minimum number of communication times negotiated by the software, is $\lceil \log_2 N_P \rceil$ per node PC.
  In our proposed method, the compositing unit completes the entire process after the composting data is transferred from each node PC. For this reason, the minimum number of software negotiations is only once per node PC, no matter what the number of parallel node N_P is. Therefore, it can sustain the processing efficiency in a massively parallel environment.

- Adoption of the binary tree method
  The basic principle of the Binary Space Partitioning (BSP) [11] is illustrated in Figure 4(a) and (b). In BSP, the number of node PCs contributing to the compositing process is reduced by half along the binary tree structure. Therefore, the wiring, which is a primary factor in determining the scale of the hardware circuitry for communication of data exchange, can be simplified. Thus, it is suitable for making pipelined high-speed hardware from the viewpoint of reducing circuit scale.

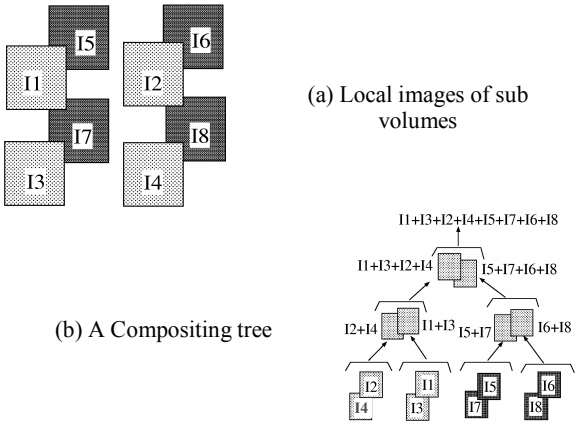


Figure 4: *The image composition of local images using a compositing tree.*

A naïve example of an 8-input-1-output compositing unit is shown in Figure 5 (a). In this example, the unit consists of an 8 x 8 crossbar switch in the first stage, a 4 x 4 crossbar switch in the second stage, and a 2 x 2 crossbar switch in the last stage. However, when using a binary tree, we need only to consider the adjacent connection of the two nodes, which have sub-images along the binary tree structure. It is not necessary to consider the arbitrary connections of the node PCs. Therefore, the compositing unit needs only a type of 2x2 crossbar switch, radically simplifying the circuit. A diagram is shown in Figure 5 (b).

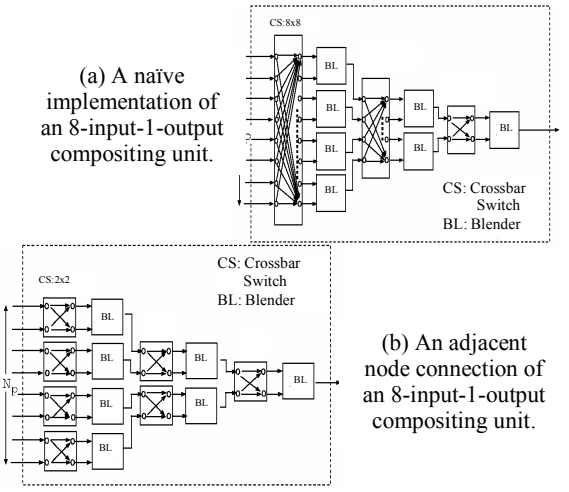


Figure 5: *Reduction of the scale of the circuitry.*

4.2. Compensation for synchronization delay

Generally, both the image data items to be composited have to be synchronized. This is carried out by the MPI synchronization command or its equivalent. This is not a complete solution, because the software causes the different delays among the node PCs as shown in Figure 6. The delay is about 20μs when the Myrinet network is used, whereas it is about 200μs for an Ethernet network is used. Thus, the compositing unit needs a mechanism to compensate for these delays.

4.3. Processing scalability

A massively parallel system has to be achieved by simply connecting the 8-input-1-output compositing unit without effecting any structural changes. The diagram is shown in Figure 7. Since the binary tree method is used, a hierarchical connection can be easily realized for a massively parallel system. The total delay time T_N can be calculated from

$$T_N = T_d \lceil \log_8 N_P \rceil. \quad (3)$$

Where T_d is the delay time generated from the 8-input-1-output compositing unit. N_P is the number of inputs to the compositing unit.

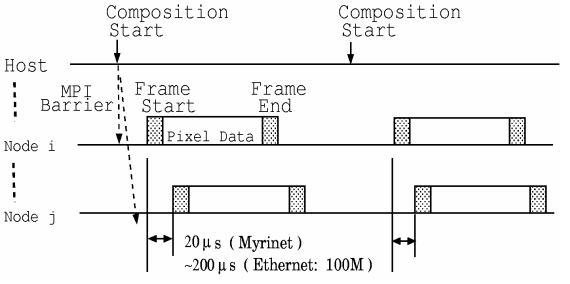


Figure 6: *The delay compensation mechanism for the data input to the image-composition device.*

© The Eurographics Association 2003.

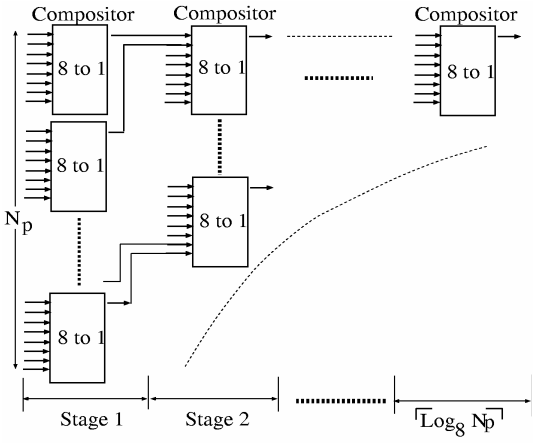


Figure 7: *The implementation of a massively parallel system by hierarchical connection.*

5. Design of the Prototype Pipeline Compositing Unit

5.1. Appearance of the prototype hardware

A photo of the prototype hardware is shown in Figure 8. The compositing unit talks to each PC through a PCI interface board

Figure 8: *The appearance of the prototype pipeline compositor.*

5.2. Pipeline compositing circuit

A block diagram of the designed pipeline compositing circuit is shown in Figure 9. There are 8 input channels and one output channel. The priority and sub-image consisting of color information (R (red), G (green), B (blue), A (alpha)) are transferred through the input channel. As the priority is unchanged per frame, it is transferred once at the beginning of the frame data. The priority is used for selector controlling to determine which sub-image, transferred through A or B channel, is front to the viewpoint. The compositing of the two images is carried out separately for R, G, and B in the BLENDER shown in Figure 9. The compositing circuit is composed of pipelines with a bit-width of 36 (8bits each for R, G, B, A respectively, and a 4-bits control bit) and 21 pipeline stages. The operating frequency is 33MHz, the speed of PCI32 bus. The FIFO in Figure 9 is for the compensation of the delay time among the images input to the compositing circuit. It can compensate for up to a maximum of 1 ms delay time. For the parallel system over 8 node PCs, the compositing can be implemented by connecting the compositing unit hierarchically.

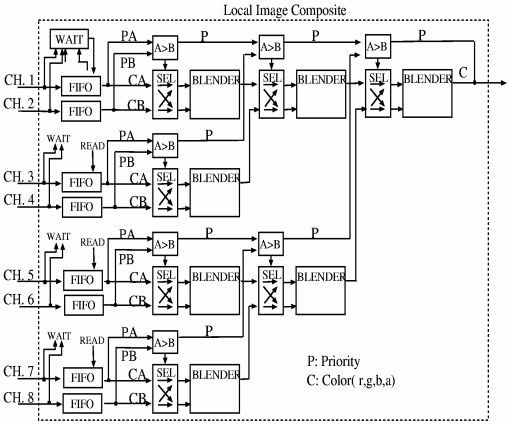


Figure 9: *A block diagram of the pipeline compositor.*

5.3. Compositing commands

The list of compositing commands between the interface board and the compositing unit is shown in Figure 10. The commands are categorized as follows.

- Hardware initialization
- Frame data transfer
- Priority setting
- Status informing

Every command is composed of 36 bits. The upper 4 bits are for command identification, and the lower 32 bits are for parameters corresponding to the commands. The bit assignment is shown in Table 4.

In order to make it follow the hierarchical connection scheme of the compositing unit, the series of commands passes through the compositing unit, i.e., the commands are output from each unit without any change in format except for the priority number and pixel data. The pixel data in the frame data feeds back the compositing result through the compositing unit. The priority number is substituted for the highest priority number in the compositing unit and output to the following compositing unit.

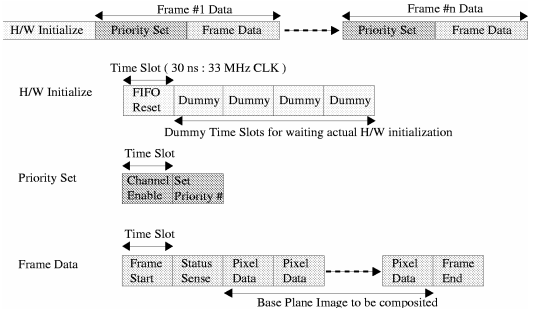


Figure 10: *Command list of the compositor.*

© The Eurographics Association 2003.

Table 4: *Assignment of 36 bits Data.*

Bit	35-32	31-24	23-16	15-8	7-0
Dummy Command	0	-	-	-	-
Pixel Data	1	Red	Green	Blue	Alpha
Channel Enable	2	1	-	-	-
Channel Disable	2	2	-	-	-
Set Priority	2	4	-	Priority (9bits)	
FIFO Reset	2	8	-	-	-
Frame Start	2	16	-	-	-
Frame End	2	32	-	-	-
Channel not Ready	4	1	-	-	-
Pixel Count Mismatch	4	2	-	-	-
Illegal Command	4	4	-	-	-

5.4. Electric interfaces

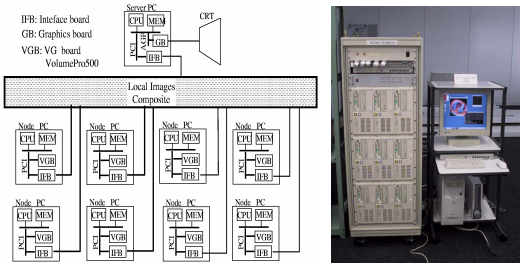
The transfer between the interface board and the compositing unit is carried out using the LVDS (low voltage differential signaling). This process converts the parallel data into serial data to prevent skewing of signal wave caused by high-speed transfer. Converting from the CMOS/TTL signal to the LVDS format is carried out using the VLSI chip (DS90CR483/484, National Semiconductor Inc.). The interface boards and compositing unit are connected via LVDS cables, which can transfer 48-bit-wide parallel data at 33MHz. In the prototype, the cabling is composed of 8 wires, each assigned 6 bits of parallel data plus 1 wire for clock signal. The basic clock frequency 33MHz of the PCI bus (PCI32) is multiplied up to 198 MHz ($33\ \text{MHz} \times 6$) using a PLL (Phase Locked Loop). Thus, the parallel data is serialized at a clock speed of 198 MHz. The compositing unit, receives the data in a frequency of with 198MHz clock speed, and converts the serial data into parallel data using the DS90CR484. The transfer speed is also 33×6 Mbits/s for each line. The effective bandwidth of the 8 lines is 1.58 Gbits/s. This speed is higher than the PCI bus speed of 1.064 Gbits/s (PCI32). This means that the unit has efficient processing margin. Most of the logic circuits use Altera's FPGA and most logic descriptions use VHDL in the design of the interface board and the compositing unit.

6. Performance Evaluation

For evaluating the prototype compositing unit, a sort-last parallel volume rendering system - VG Cluster 1 was built. The processing speed is measured using the system. In this system, VolumePro500 [12], high-speed volume rendering engines were used to render the sub-image and the 8-input-1-output prototype compositing unit was used for the compositing sub-images.

6.1. The parallel volume rendering system

The configuration of the parallel volume rendering system using the prototype compositing unit is shown in Figure 11 (a), and a photo of the system is shown in Figure 11 (b). In this system, rendered sub-images on node PCs are transferred to the compositing unit via the PCI interface board (IFB) inserted into the PCI bus. The output image from the compositing unit is composited taking into account the occlusion of the sub-images, and is then transferred to the frame memory of the graphics board in the server PC through the PCI interface board (IFB), and finally generating the total image. The sub-image may be generated using the high-speed volume engine (VGB) or by software. However, the massively parallelization is necessary for high-speed rendering when using software rendering (e.g. ray-casting).



(a) Configuration (b) Appearance

Figure 11: *The parallel volume rendering system using the prototype compositing unit.*

6.2. Measurement of the processing time of the prototype compositing unit

The evaluation of the performance of the compositing unit is carried out using the above system. The measured results for processing time of the compositing unit are shown in Table 5. The measured data includes the sub-image data-transfer-time through the interface board (IFB output), compositing time, and receiving time of the final image through the interface board (IFB input).

Table 5: *The performance of the pipeline compositor (ms).*

Screen size (pixels)	IFB output $T_{pcio}(P^2)$	Compositing	IFB input $T_{pcii}(P^2)$
256^2	2.47	0.69	2.48
512^2	9.39	0.69	8.75
1024^2	37.07	0.69	33.84

Actual measurement is possible within 8 parallel PCs when using the prototype compositing unit. The compositing time of the system over 8 parallel PCs is estimated using Equation (4) based on the measurement results. The compositing unit of the system over 8 parallel PCs is connected using hierarchical connection, as shown in Figure 7. The delay time of the compositing unit can be calculated from Equation (3). Since there are 21 pipeline stages in the compositing unit, and the operating frequency is 33MHz, the calculated delay time for the 8-input-1-output unit 636ns. If the number of processors is N_P and image size is P^2 pixels, the compositing time T_H can be calculated from Equation (4).

$$T_H = T_d \lceil \log_8 N_P \rceil + T_{pcio}(P^2) \ . \quad (4)$$

© The Eurographics Association 2003.

Where $T_{pcio}(P^2)$ is the transfer time of the 2D sub-image from the interface board to the compositing unit, which is determined by the screen size of P^2. Furthermore, $T_{pcio}(P^2)$ can be expressed as:

$$T_{pcio}(P^2) = R_{pci} \cdot \frac{1}{R_{effect}} \cdot P^2 . \quad (5)$$

Where $R_{pci} = \frac{1}{33 \times 10^6}$ seconds, is derived from the PCI standard. R_{effect} is the coefficient of the effective efficiency of the PCI bus, which in turn depends on the chipset to control the PCI32 on the main board of the PC; it is calculated as about 0.86 from Table 5. Thus, we obtain as $R_{pci} \cdot \frac{1}{R_{effect}}$ to be 3.5×10^{-8} .

Table 6: *Time measured on VG Cluster: Human head 256^3. (sec)*

Mode	Node PC	Screen size (pixels)			
		256^2		512^2	
		Rendering	Compositing	Rendering	Compositing
HH	2	0.0139	0.0032	0.0261	0.0107
	4	0.0134	0.0032	0.0255	0.0108
	8	0.0132	0.0032	0.0253	0.0108
SH	2	0.1350	0.0032	0.5198	0.0108
	4	0.0680	0.0032	0.2685	0.0109
	8	0.0352	0.0032	0.1396	0.0108
SS	2	0.1341	0.0106	0.5198	0.0428
	4	0.0674	0.0194	0.2685	0.0787
	8	0.0345	0.0291	0.1396	0.1182

Table 7: *Time measured on the VG Cluster: Human body 512^3. (sec)*

Mode	Node PC	Screen size (pixels)			
		256^2		512^2	
		Rendering	Compositing	Rendering	Compositing
HH	2	-	-	-	-
	4	0.1201	0.0032	0.1314	0.0108
	8	0.0820	0.0032	0.0940	0.0108
SH	2	4.9084	0.0032	18.6686	0.0108
	4	2.6981	0.0032	9.6096	0.0109
	8	1.3876	0.0032	4.9882	0.0108
SS	2	4.9084	0.0118	18.6686	0.0486
	4	2.6981	0.0222	9.6096	0.0916
	8	1.3876	0.0324	4.9882	0.1313

Table 8: *Time measured on SCore Cluster III at SS mode of the rendering software implemented in the VG Cluster: The human head 256^3. (sec)*

Mode	Node PC	Screen size (pixels)			
		256^2		512^2	
		Rendering	Compositing	Rendering	Compositing
SS	2	0.1341	0.0106	0.5198	0.0428
	4	0.0674	0.0194	0.2685	0.0787
	8	0.0345	0.0291	0.1396	0.1182
	16	0.0176	0.0379	0.0690	0.1525
	32	0.0095	0.0473	0.0353	0.2024
	64	0.0048	0.0562	0.0188	0.2293
	128	0.0032	0.0657	0.0104	0.2663
	256	0.0026	0.0738	0.0071	0.3009
	512	0.0033	0.0828	0.0065	0.3351

Table 9: *Time measured on SCore Cluster III at the SS mode of the rendering software implemented in the VG Cluster: The human body 512^3. (sec)*

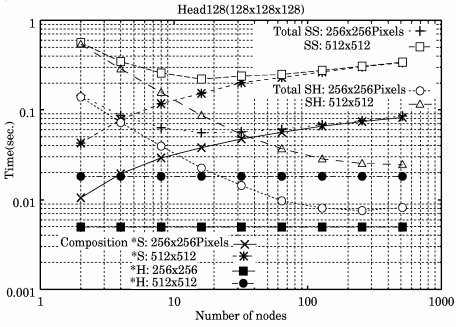
Mode	Node PC	Screen size (pixels)			
		256^2		512^2	
		Rendering	Compositing	Rendering	Compositing
SS	2	4.9084	0.0118	18.6686	0.0486
	4	2.6981	0.0222	9.6096	0.0916
	8	1.3876	0.0324	4.9882	0.1313
	16	0.7303	0.0417	2.6941	0.1383
	32	0.3830	0.0509	1.4247	0.2089
	64	0.2027	0.0600	0.7662	0.2479
	128	0.1145	0.0694	0.4071	0.2853
	256	0.0646	0.0780	0.2214	0.3219
	512	0.0356	0.0864	0.1335	0.3899

6.3 Relaxation of the saturation of the processing performance in a massively parallel system

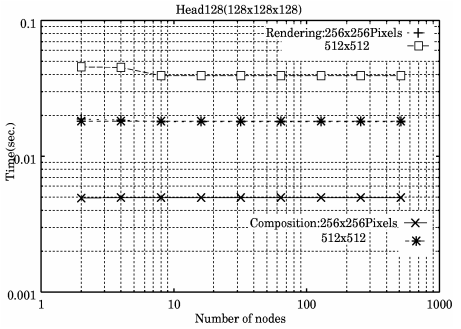
For the evaluation, a parallel rendering program was implemented in the VG Cluster. With this program, the rendering and compositing modes can be selected as hardware (H) or software (S). For example, HS in the Tables means hardware rendering / software compositing. Because measuring of time is impossible over 8 parallel PCs due to the limitations of the VG cluster, in case of SS, the evaluation is carried out using the time measured from SCore Cluster III with the above program implemented. The results are shown in Table 6-9. In case of the SH and HH mode, the results obtained from SCore Cluster III were used for S, and the results estimated from Equation (4) were used for H for over 8 PCs. We regard this assumption as reasonable because, 1) rendering and compositing are clearly separated; 2) the access to compositing unit from each PC is only once at the starting of the composition regardless of number of PCs. The estimated processing time over 8 parallel PCs is shown in Figure 12. It is clear that the

© The Eurographics Association 2003.

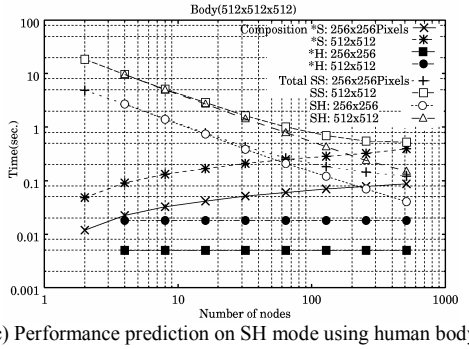
saturation of the processing performance is improved from this Figure.



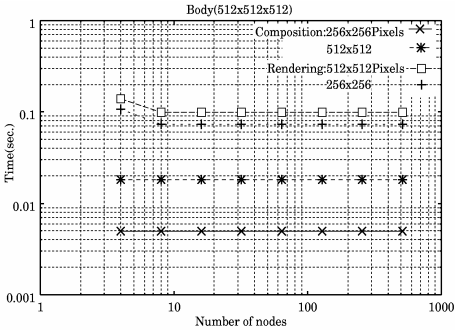
(a) Performance prediction on SH mode for human head



(b) Performance prediction on HH mode for human head



(c) Performance prediction on SH mode using human body



(d) Performance prediction on HH mode for human body

Figure 12: *Performance prediction of the volume rendering using the composition unit under massively parallel conditions. (sec)*

7. Conclusion

A key finding of our study is that for PC cluster system like this to support real-time volume visualization and to be scalable to large-scale problems, hardware support for image compositing is needed. Our design for a hardware compositing unit closely follows the lessons we have learned through the experimental study conducted on the 512-node PC cluster.

The 8-node prototype system we have built using this hardware compositing unit and commodity graphics cards can deliver a 25Hz update rate for rendering 512^3 voxels to 512^2 pixels. This performance will scale with system size and will profit from the increasing performance of newer commodity graphics cards. Volume graphics enhanced systems such as this are thus ideal for conducting simulation-time volume visualization as a means of tracking simulations.

Acknowledgements

We would like to thank Associate Prof. Yutaka Ishikawa of the Department of Information Science at The University of Tokyo for his kind support in the evaluation experiment. Thanks are due to Mr. K. Kajihara of Graduate School of Information Science & Engineering of Tokyo Institute of Technology for his assistance in preparing this manuscript. We are also grateful to Prof. Yasunori Dohi of Yokohama National University's Electrical Engineering and Computer Science Division for his advice on this paper.

References

1. S. Muraki, M. Ogata, K.-L. Ma, K. Koshizuka, K. Kajihara, X. Liu, Y. Nagano and K. Shimokawa, "Next-Generation Visual Supercomputing using PC Clusters with Volume Graphics Hardware Devices," Proceedings of IEEE/ACM Supercomputer Conference, 2001.
2. R. Samanta, T. Funkhouser, K. Li, and J. P. Singh, "Hybrid sort-first and sort-last parallel rendering with a cluster of PCs," Proceedings of SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware, pp. .97-108, 2000.
3. S. Molnar, M. Cox, D. Ellsworth and H. Fuchs, "A Sorting Classification of Parallel Rendering," IEEE CG & A, Vol. 14, No. 4, pp. 23-32, 1994.
4. K.-L. Ma, J. Painter, C. Hansen, and M. Krogh, "Parallel Volume Rendering Using Binary-Swap Compositing," IEEE Computer Graphics and Applications, pp. 59-68, July 1994.
5. T. Lee, C. Raghavendara, J. Nicholas, "Image Composition Schemes for Sort-Last Polygon Rendering on 2D Mesh Multicomputers," IEEE Transactions on Visualization and Computer Graphics, Vol. 2, No. 3, pp. 202-217, 1996.
6. G. Stoll: "Lightning-2, A High-Performance Display Subsystem for PC Clusters," Proceedings of SIGGRAPH 2001, pp. 141-148, 2001.
7. S. Lombeyda, L. Moll, M. Shand, D. Breen and A. Heirich, "Scalable Interactive Volume Rendering Using Off-the-Shelf Components," Proceedings of IEEE Symposium on Parallel and Large-Data Visualization and Graphics, pp. 115-158, 2001.
8. C. Clos, "A Study of Non-Blocking Switching Networks," Bell System Technical Journal, Vol. 32, pp. 406-424, 1953.
9. Joseph Y. Hui, "Switching and Traffic Theory for Integrated Broadband Networks," Kluwer Academic Publishers, Springer-Verlag, 1990.
10. Y. Ishikawa, "SCore Cluster System Software", Lecture material, RWC 2001 Final Exhibition & Symposia, Tokyo, Japan, 2001.
11. H. Fuchs, Z. Kedem, and B. Naylor, "On Visible Surface Generation by A Priori Tree Structures," Proceedings of the ACM SIGGRAPH '80 Conference, pp. 124-133, 1980.
12. H. Pfister, J. H. Hardenburg, H. Lauer, and S. Sailor, "The VolumePro Real-Time Ray-Casting System," Proceedings of the ACM SIGGRAPH '99 Conference, pp. 131-138, 1999.

© The Eurographics Association 2003.