Dynamic Load-Balanced Rendering for a CAVE System

Tetsuro Ogi
University of Tsukuba
1-1-1 Tennoudai, Tsukuba,
Ibaraki 305-8577, Japan
+81-029-8535191
tetsu@cc.tsukuba.ac.jp

Takaya Uchino
University of Tsukuba
1-1-1 Tennoudai, Tsukuba,
Ibaraki 305-8577, Japan
+81-029-8536574
uchino@gil.cs.tsukuba.ac.jp

ABSTRACT
Recently, PC clusters have been used to construct CAVE-like immersive projection displays. However, in order to improve the rendering performance of PC cluster-based CAVE systems, the number of node PCs should be increased in accordance with the number of screens that are included. In this research, a mechanism for dynamic load-balanced rendering in a PC cluster system incorporating an arbitrary number of nodes was developed. This system constructs the cluster system by using a chain connection-type compositor board, whereby load-balancing can be controlled dynamically in response to the movement of the virtual objects or of the user's viewpoint. This paper describes the implementation of this dynamic load-balanced rendering method and presents the results of an evaluation experiment.

Categories and Subject Descriptors
I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism – *virtual reality.*

General Terms
Performance

Keywords
Load-balancing, Immersive projection display, PC cluster

1. INTRODUCTION
The performance of PCs has improved significantly recently, and PC clusters are now being used to construct CAVE-like immersive virtual environments [1]. When a large-scale graphics workstation is used in such a system, the performance of the display is fixed according to the specification of the computer [2]. However, when PC clusters are used, the rendering performance can be flexibly-changed according to the system construction. For example, the rendering load can be distributed by using several PCs to render an image, and therefore the total performance of the system can be improved. However, there are still some problems that occur when these distributed-rendering methods are applied to a CAVE-type system consisting of multiple screens. In the basic configuration of a PC-based CAVE system, one or two PCs are assigned to each screen in order to generate a stereo image. Under these conditions, it is necessary to add extra PCs corresponding to the number of screens in order to realize distributed rendering [3].

On the other hand, CAVE-like displays that surround the user with multiple screens operate in such a way that the rendering loads are not evenly distributed among all of the screens. For example, when a user looks at a stereo image of a virtual object from a frontal position, then the rendering load on the side screens is small, whereas when the user moves around in the display-space, the screens that require the highest rendering loads change in response to changes in the user's viewing position. In this situation, the total rendering load is not distributed equally among all screens, and therefore some PCs always experience low loading, i.e. the total computer resources are not used efficiently.

If the number of node PCs that are assigned to a high-load screen could be actively increased according to changes in the rendering load, an efficient means of load-balanced rendering could be achieved. For example, if a user were to look at a three-dimensional object from the front, most of the PCs would be assigned to the front screen, and when he looks at the object from the side, the resources could be reassigned to the side screens. Therefore, by separating the assignments between the PCs and the screens, a cluster system consisting of an arbitrary number of PCs that are independent of the number of screens could be constructed. It is expected that the efficiency of the rendering performance of PC-based cluster systems could be improved by the use of such a technique.

In this study, a CAVE-like multi-screen display named CS Gallery (Cyber Space Gallery) was developed, which includes a dynamic load-balanced rendering function. This system changes the assignment of the computer resources dynamically according to changes in the rendering load. This paper describes the basic principles of the dynamic load-balanced rendering function, the method of implementation, and the results of evaluation experiments.

2. SYSTEM CONFIGURATION

2.1 System General View
The CS Gallery consists of three screens, one at the front, one on the right, and one on the floor. In this system, two *NEC LT245J* DLP projectors are used for each screen to represent stereo images using the circular-polarization method. In addition, the electroma-gnetic sensor known as *Ascension Flock of Birds* is used to measure the user's viewing position and direction.

For the PC cluster-system that is used to generate the stereo images, seven Linux PCs (HP XW6200, Intel Xeon 3.4GHz, NVIDIA

VRST'06, November 1–3, 2006, Limassol, Cyprus.

FX3400) are used. One PC controls the cluster system, while the other six PCs are used to render the images.The control PC receives data from the sensors, sets the parameters for the image composition at each node PC, and dynamically controls the load-balanced rendering. The six rendering PCs are used to generate the left-eye and the right-eye images. The same rendering program runs on each of the rendering PCs and generates the images for the assigned screens according to control parameters sent from the control PC.

It is also a requirement that each PC renders the corresponding images synchronously, because the entire image of the virtual world is rendered via the distributed PCs. In this system, the control PC transmits synchronization data (such as the viewing position or the frame number of the animation data) to the rendering PCs, and therefore synchronization can be achieved among the rendering PCs.

2.2 Compositor

In order to perform distributed rendering, a compositor is required to compose the images rendered by each node PC. Existing compositors can be classified into two types; cascade connection-type systems and chain connection-type systems (as shown in Figure 1).

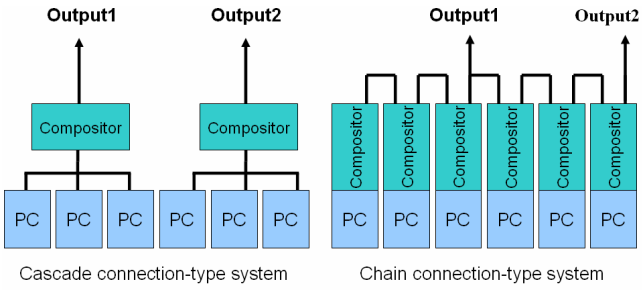

Fig.1 Cascade connection-type and chain connection-type compositors

The compositor for a cascade connection-type system collects the images rendered by each node PC and outputs a composed image [4]. In this method, it is necessary to change the physical connections in order to change the assignment of the PCs to the individual screens or to alter the number of cluster PCs. Therefore, it is difficult to use this method for the dynamic control of load-balanced rendering for the CAVE system.

On the other hand, in the case of chain connection-type systems, compositor boards are installed in each PC and these are then connected in sequence. Each PC receives the output image from the preceding PC, synthesizes the rendering image using the compositor board, and then sends the synthesized image to the next PC. By repeating this operation sequentially, the final synthesized image is output from the last PC [5]. When using this method, even if the assignments between the PCs and the screens or the number of cluster PCs are changed, the physical connections between the hardware are not changed. Therefore, dynamic control of the connections can be performed by changing the functional connections via the software. In this research, the chain connection-type compositor board ORAD DVG was used.

3. DYNAMIC LOAD-BALANCED RENDERING

3.1 Load-Balanced Rendering

In most CAVE systems that use PC clusters, each PC renders an image for a specific screen. However, in this system, a graphics window that represents the entire space of the virtual world is used in order to control the assignment between the PCs and the screens dynamically. In this method, the graphics window is divided into four areas, and the images for the front, right and bottom screens are rendered in each area, as shown in Figure 2. In addition, each rendering PC renders an image in a specified area and the entire image-space can be generated by composing these images through the compositor boards. The image areas corresponding to each screen are segmented from the entire image-space by using scan-converters, and these are then displayed by the projector. In this system, a virtual window of 1600x1200 resolution is defined, and each quarter-area of 800x600 resolution is projected on each screen. The lower-right area of the window is not used. Figure 3 shows the system configuration of the CS Gallery and the processing flow of the distributed rendering.

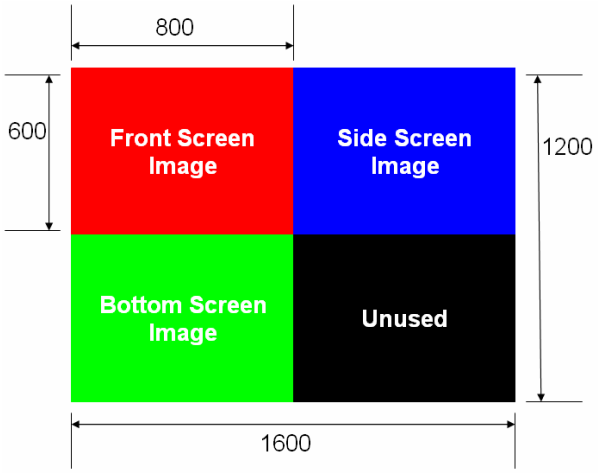

Fig.2 Graphics window of the entire image space

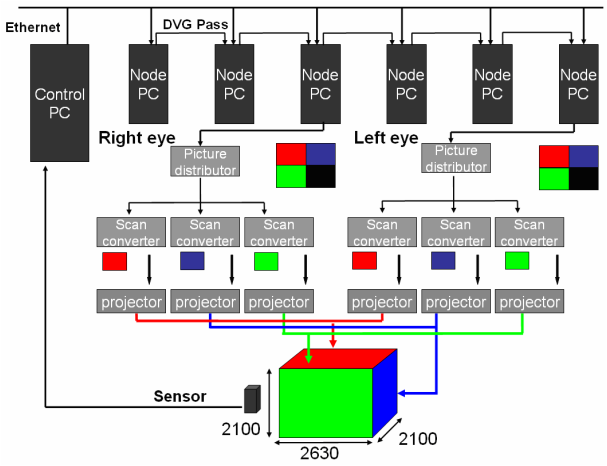

Fig.3 System Configuration of CS Gallery

In order to generate a stereo image, three PCs are assigned to the left-eye and the right-eye images respectively, and load-balanced

rendering is performed using the AID (Added Image Division) function of ORAD DVG. In this method, since the size of the view-port can be changed even when the program is running, dynamic load-balancing can be realized by changing the rendering area for each node PC. In addition, the effect of the distributed rendering can be improved by using this technique in conjunction with the 'view frustum culling' method.

In this system, since three PCs are used to render each eye-image, the entire image-space is divided into three parts besides the division for the screen assignment, and each part is assigned to each PC. Therefore, it is a requirement that each PC is able to render an image for the viewing frustum that is defined for one part of one screen or over several screens. In order to control the dynamic load-balanced rendering, a technique for changing the rendering area for each PC was used. In this method, the rendering is changed by moving the position of the boundary lines, where the image-space is divided into three parts.

The image space is divided by boundary lines, as shown in Figure 4. The front-screen area (1), the right-screen area (2), and the bottom-screen area (3) are rendered by the first-node, the second-node and the third-node PCs, respectively. Then, by composing these three images, the entire image-space can be generated. The positional data for these boundary lines is calculated in the control PC, and this data is sent to each node PC by using the UDP protocol. In each node PC, the viewing frustum is redefined synchronously according to the positional-data for the boundary lines.

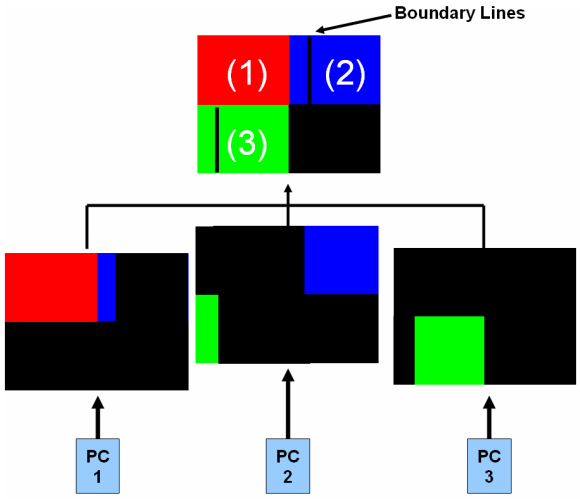

Fig.4 Load-Balanced Rendering

3.2 Dynamic Load Balancing Control

In the CAVE system, the rendering load for each screen changes according to the movement of the user's view-point or of the object while the user is experiencing the virtual world, as mentioned above. By changing the widths of the areas divided by the boundary lines in response to changes in the rendering-load on each PC, dynamic load-balanced rendering can be realized. In this system, each node PC measures the rendering time during every rendering loop and sends it to the control PC. Then, the control PC moves the positions of the boundary lines such that the widths of those divided areas that require increased rendering time can be reduced.

In order to control the position of each of the boundary lines, a method of comparing rendering time between neighboring PCs was used. Namely, the boundary line between the areas covered by node 1 and node 2 is moved by comparing the rendering times between node1 and node 2, and the next boundary line between node 2 and node 3 is moved by comparing node 2 and node 3. A method of controlling the positions of the boundary lines so that the rendering time is optimized in the entire node PCs can also be considered. However, in this system, a method of comparing two neighboring PCs was used in order that the calculation load would not become too large to optimize the rendering time when the number of cluster PCs is increased.

3.3 View Frustum Culling

In general, a view frustum for perspective projection consists of six clipping planes. Although objects that are located outside the viewing volume are clipped and are not rendered, the lighting and the geometric transformations for these objects are still calculated unnecessarily. In the dynamic load-balanced rendering technique, the geometry calculations for the clipped objects have a considerable influence on performance, because one node PC often renders images using two or more view frustums for several screens. Therefore, dynamic load-balanced rendering can be implemented more effectively if it is used in conjunction with view frustum culling.

In the case of the view frustum culling method, judgments about whether representative points for each object are in the view frustum or not are used so that the calculation load needed for culling is not so large [6]. When a representative point for each object is outside one of the clipping planes, the object is culled away. Though culling methods that use strict shape models for objects have been considered [7], the total performance would decrease due to the large calculation load.

4. EVALUATION EXPERIMENT

4.1 Experimental Method

In order to evaluate the effectiveness of the dynamic load-balanced rendering method developed in this study, the method was applied to visualize a boid that simulates the movement of a school of fish [8]. In this simulation, when the school of fish moves irregularly according to a random number, the rendering load between the screens also changes irregularly and the entire performance is influenced by the slowest node. In this study, the algorithm for dynamic load-balanced rendering was applied to visualize the movement of the school of fish in the CS Gallery (as shown in Figure 5).

In this experiment, a representative point for view frustum culling was defined at the center of each fish, and the initial positions of the boundary lines were set at equal intervals. Though this system uses six node PCs to generate a stereo image, data to render single images was measured in the experiment because the rendering loads for the left-eye images and for the right-eye images were almost the same. The frame-rate of each PC was recorded and the performance of this system was defined using the minimum value of the frame rates among all of the PCs.

Fig.5 Boid Simulation displayed in the CS Gallery

4.2 Experimental Result

Figure 6 shows the results of time variations in performance when dynamic load-balanced rendering was used and when it was not used. In this graph, the frame rate for rendering an image of 250 fishes in the CS Gallery was plotted in every simulation loop. From the results, we can see that the rendering performance was changed significantly when the dynamic load-balanced rendering method was not used. In this case, the rendering load for each screen was uneven because the number of fishes rendered on each screen changed in response to the movement of the school of fish. When the rendering load was balanced among all of the screens, there were few differences in rendering performance between using and not using the dynamic load-balanced rendering method. However, when the rendering load was unbalanced, the performance of the faster nodes was affected by the slower nodes through the synchronization process, and the overall performance was reduced. On the other hand, when dynamic load-balanced rendering was used, a high rendering performance was realized, except for immediately after gross changes in the rendering load.

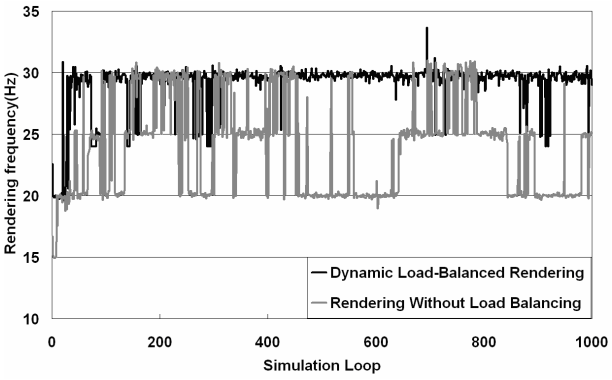

Fig.6 Time variations in rendering performance

Figure 7 shows the results of changes in the rendering performance when the rendering load is changed. In this graph, the frame rates are compared between using and not using the dynamic load-balanced rendering function, and also between using and not using the view frustum culling function, when the number of fishes is changed. From these results, we can see that the dynamic load-balanced rendering method was independently effective for changes in the rendering load, and that the effect was enhanced when it was used in conjunction with the view frustum culling function.

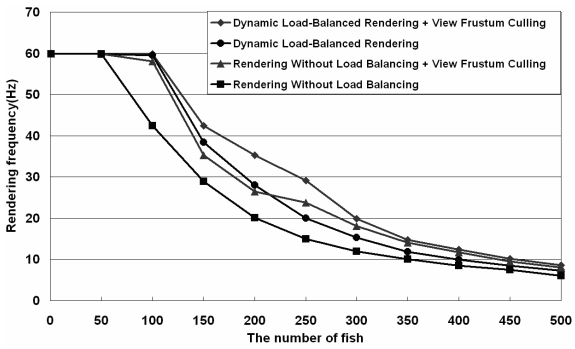

Fig.7 Rendering performance for Changing the number of fish

5. CONCLUSION

In this research, a PC cluster-based CAVE system that implemented a dynamic load-balanced rendering function using a chain connection-type compositor board was developed. In the dynamic load-balanced rendering, the full rendering power of the computer could be used effectively and the total performance was improved by composing the images that were rendered by each PC. This method can be flexibly applied to CAVE-like systems of various constructions, because an arbitrary number of PCs can be used independently of the number of projectors. Therefore, it is expected that the proposed method could be used as a common high-performance rendering technique that is independent of the display construction.

6. REFERENCES

[1] Benjamin Schaeffer, Camille Goudeseune: "Syzygy: Native PC Cluster VR," Proceedings of IEEE Virtual Reality 2003, IEEE, pp. 15-22, March 2003..

[2] C. Cruz-Neira, D.J. Sandin, T.A. DeFanti: "Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE," Proceedings of SIGGRAPH'93, pp. 135-142, 1993.

[3] J. Jacobson, M. L. Rendard, J. L. Lugrin, M. Cavazza: "The CaveUT System: Immersive Entertainment Based on a Game Engine," ACM SIGCHI International Conference on Advances in Computer Entertainment Technology (ACE 2005), Valencia, Spain, 2005.

[4] M. Ogata, S. Muraki, X. Liu, and K.L. Ma: "The design and evaluation of a pipelined image compositing device for massively parallel volume rendering," Proc. 2003 Eurographics/IEEE TVCG Workshop on Volume Graphics, pp. 61-68, 2003.

[5] ORAD, http://www.orad.tv/

[6] James H. Clark: "Hierarchical Geometric Models for Visible Surface Algorithm," Communications of the ACM, Vol. 19, No. 10, pp. 547-554, October 1976.

[7] Ulf Assarsson and Tomas Möller: "Optimized View Frustum Algorithms for Bounding Boxes," Journal of Graphics Tools, Vol. 5, Number. 1, pp. 9-22, 2000.

[8] Reynolds, C. W. Flocks, Herds, and Schools: "A Distributed Behavioral Model, Computer Graphics," SIGGRAPH '87 Conference Proceedings, Vol. 21, Number. 4, pp. 25-34, 1987.