基礎論文

# 動的負荷分散レンダリングを用いたCAVEシステム

小木哲朗[*1]，内野孝哉[*1]

**Dynamic Load-Balanced Rendering for a CAVE System**

Tetsurou OGI and Takaya UCHINO

**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 user's viewpoint. This paper describes the implementation of this dynamic load-balanced rendering method and presents the results of an evaluation experiment.**

**Key Words: Load-balancing, Immersive projection display, PC cluster**

## 1. はじめに

近年、PCの高性能化に伴いCAVE型の没入仮想環境の構築にPCクラスタが用いられるようになってきた[1][2]。従来の大型のグラフィックスワークステーションを用いたシステムでは使用する計算機によってディスプレイの性能が固定的に決まってしまうが[3][4][5]、PCクラスタを用いる場合はシステムの構成によって性能を柔軟に変更することができる。例えば1つの映像の描画に複数台のPCを用いることでレンダリング負荷を分散させ、全体のパフォーマンスを向上させることができる。この際、アプリケーションレベルで個別に並列化処理を行うASE (Application-level Synchronized Execution)形式や並列化に対応したグラフィックスAPIの置き換えを行うGLR (GL-DLL Replacement)形式などが提案されている[6][7][8][9][10]。しかしながら、多面スクリーンで構成されるCAVEシステムでこれらの分散レンダリングを用いる場合、従来の方法では幾つかの問題がある。

PCを用いたCAVEの基本構成では1面スクリーンに対して1台(あるいは偏光立体視の場合は2台)ずつのPCが割り当てられていることが多いが、この状態から分散レンダリングを行うためには、従来の方法ではPCをスクリーン枚数に対応する台数ごとに追加していかなくてはならない[11]。例えば3面スクリーン構成のCAVEで左右の視点映像を1台ずつのPCでレンダリングしている場合、PCの台数は6台、12台、18台と6台ずつ追加していくことになり、これではコスト的に拡張は容易ではない。

一方、利用者をスクリーンで囲い込む多面ディスプレイではすべてのスクリーンでレンダリング負荷が均等ではないという特徴がある。例えば立体視表示された仮想オブジェクトを利用者が正面から見ている場合、左右のスクリーンに対するレンダリング負荷は小さい。また利用者の視点位置が変化するに従い、このレンダリング負荷の高いスクリーンも切り替わる。視点位置が変化していないときでも、オブジェクトが別のスクリーンに移動する場合、レンダリング負荷が高いスクリーンが切り替わる。

また視点が動いていないときでも、仮想世界でオブジェクトが運動や移動を行う場合は、レンダリング負荷が高いスクリーンが切り替わる。このようなレンダリング負荷が不均等な状況では、常に低負荷のPCが存在し、計算機資源が有効に利用されていると言えない。そのため、レンダリング負荷の変化に応じて高負荷のスクリーンに割り当てるPCのノード数を集中的に増やすことができれば、効率的なレンダリングを行うことが可能となる。例えば図1のように、利用者が3次元物体を正面から見ている場合は、正面スクリーンに大部分のPCを割り当て、横から見る場合には側面スクリーンにPCを割り当て直すという方法である。またPCとスクリーンの

筑波大学大学院　システム情報工学研究科

*1Graduate School of Systems and Information Engineering, University of Tsukuba

対応を分離することで、スクリーンの枚数とは独立に任意にクラスタの PC 台数を変更することが可能になり、システムとしてのパフォーマンスを効率的に向上できることが期待される。

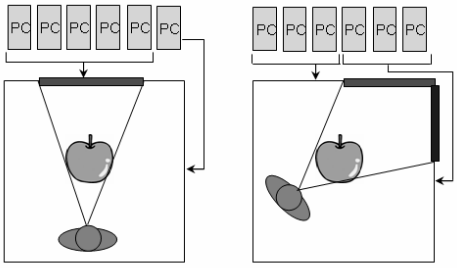



**図 1　視点位置による使用スクリーン領域の変化**
Fig.1 Change of the assignment of the computer resources

本研究では、CAVE システムにおけるレンダリング負荷の変化に合わせて、計算機資源の割り付けを動的に変更することが可能な動的負荷分散レンダリング機能を備えた CAVE システム「CS Gallery (Cyber Space Gallery) 」の開発を行った。以下動的負荷分散レンダリングの基本原理、実装方法及び評価実験の結果等について論じる。

## 2. システム構成

### 2.1　システム概要

本研究で開発した動的負荷分散 CAVE システム「CS Gallery」は、正面、側面、床面の 3 面のスクリーンで構成されている(図 2)。立体映像の提示には円偏光方式を用い、各スクリーンに対して DLP プロジェクタ NEC LT245J を 2 台ずつ用いている。また利用者の視点位置、視線方向を計測するため、磁気センサ Ascension Flock of Birds を使用している。

**図 2　CS Gallery Display**
Fig.2 CS Gallery Display

映像を生成するための計算機には、7 台の Linux PC(HP XW6200, Intel Xeon 3.4GHz, NVIDIA FX3400)を用いたクラスタシステムを構成している。1 台の PC はコントロール用計算機として用い、残りの 6 台の PC はレンダリング用計算機として使用する。

コントロール PC は、センサデータの取り込み、各ノードに対する画像合成モードの設定や後述する分散レンダリングの動的な制御に使用する。6 台のレンダリング用 PC は 3 台を右目用の映像生成に、残りの 3 台を左目用の映像生成に使用している。各レンダリング PC 上では同じ描画プログラムが実行され、コントロール PC から送られる制御パラメータに従い、対応するスクリーン映像の描画を行っている。

また PC クラスタを用いたシステムでは全体の映像空間を複数台の PC でレンダリングするため、各 PC が同期を取って動作することが必要である。本システムでは、コントロール PC がセンサから受け取った視点位置やフレームカウンタ等の同期情報を各ノード PC に送り、共有メモリを介してレンダリングプロセスへ渡すことで全体の同期を取っている。

### 2.2　コンポジタボード

分散レンダリングを行うためには、各 PC で描画された映像を重畳するためのコンポジタが必要であるが、これにはカスケード接続型のシステムとチェーン接続型のシステムが存在する(図 3)。

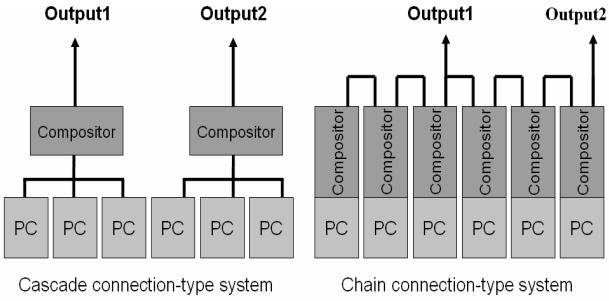



**図 3　カスケード接続型とチェーン接続型**
Fig.3 Cascade connection-type and chain connection-type compositors

カスケード接続型のシステムでは、各 PC で描かれた映像をコンポジタボードに集めて 1 つの映像を出力する[12]。この方法を用いて多面スクリーンに対するレンダリング PC の割り当てやクラスタの台数を変えるためにはハードウェア的な接続関係を変更する必要があり、これを動的に制御することは困難である。

一方、チェーン接続型のシステムでは、各 PC にそれぞれコンポジタボードが搭載され、各 PC は前の PC から受け取った映像に自分の描いた映像を重畳して次の PC に渡す。この操作を順に繰り返していくことで、最後の PC から最終的な合成映像が出力される[13]。この方法ではスクリーンとレンダリング PC との対応や PC の台数が変わってもハードウェアの接続は同一であるため、ソフトウェアの処理によって動的に接続関係を変更することが可能である。

そのため本システムでは、6 台のレンダリング用

PC にチェーン接続型のコンポジタボードである ORAD DVG を導入し、動的負荷分散 CAVE システムのための PC クラスタのシステムを構築した。本研究で使用したコンポジタボードでは専用のビデオバスを用い、画像合成は 60Hz でオンボード処理されるため、画像合成における遅延は発生しない。また、この画像処理はノード PC の CPU に対するオーバーヘッドはないため、ノード PC の台数が増えても各 PC のパフォーマンスには影響がない。

### 2.3 ソフトウェア

動的負荷分散機能を備えたアプリケーションプログラムの開発には、CABIN Lib.[1]を拡張した、OpenGL、glut ベースの汎用の CAVE 型ディスプレイ用ライブラリである glCC lib.に、DVG API を用いて負荷分散機能を組み込んだライブラリ「CSG Lib」を開発し使用できるようにしているこうすることによって開発者が動的負荷分散の機能の実装について意識せずに、glut の形式に近い形のプログラミングによって動的負荷分散 CAVE 用アプリケーションを開発することができる。

## 3. 動的負荷分散レンダリング

### 3.1 分散レンダリング

PC クラスタを用いた基本的な構成の CAVE では、各 PC が特定のスクリーン映像を描画するが、本システムでは PC とスクリーンの対応を動的に変更するため映像空間全体を仮想的なウィンドウとして用いる方法を取った。この方法では、正面・側面・床面の映像を合わせた仮想ウィンドウを開き、その中を縦と横に 4 分割した領域にそれぞれ正面、側面、床面の映像を描く(図 4)。

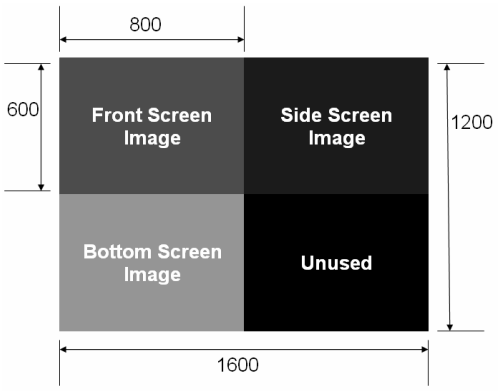



図 4　全体の映像空間のウィンドウ

Fig.4 Graphics window of the entire image space

レンダリング用の各ノード PC は映像空間内の指定された領域の映像を描き、コンポジタボードを通して映像を重畳することで全体の映像空間を生成する。この全体映像からスキャンコンバータによって各スクリーンに対応する映像領域を切り出すことで、スクリーン映像が各プロジェクタに送出される。本システムでは仮想ウィンドウとして 1600x1200 の解像度のウィンドウを設定し、これを 4 つに切り分けた 800x600 の解像度の領域が各スクリーンに提示される。仮想ウィンドウの右下の 1/4 の領域は未使用とした。また図 5 はシステム全体の構成と分散レンダリングの処理の流れを示したものである。

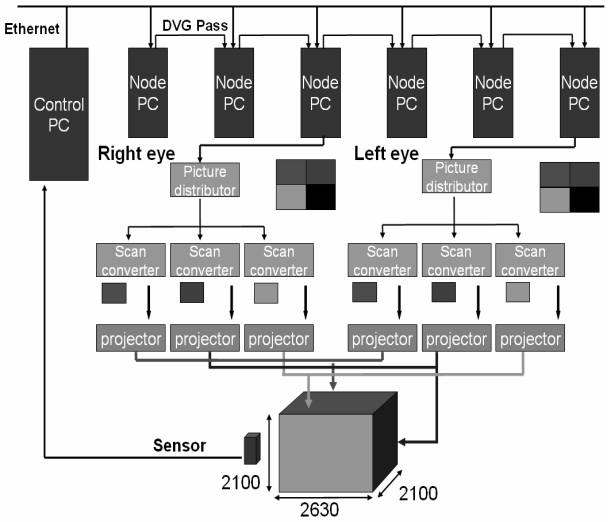



図 5　CS Gallery システムの構成

Fig.5 System configuration of CS Gallery

立体視を行うために、左目用と右目用の映像空間の生成にはそれぞれ 3 台ずつの PC を割り当て、ORAD DVG の AID(Added Image Division)の機能を用いて分散レンダリングを行う[14]。この方法では各ノード PC がレンダリングするビューポートサイズを処理中に変更することができるため、指定するレンダリング領域を変更することで動的に負荷のバランスをとることが可能となる。またこの際、ビューフラスタムカリング処理を併用することで分散レンダリングの効果を向上させることができる。

本システムでは、片眼の映像にそれぞれ 3 台ずつの PC を使っているため、正面・側面・床面のスクリーン領域の分割とは別に、全体の映像空間を三分割したレンダリング領域を各 PC に割り当てている。従って各 PC は一定の視体積に対する映像をレンダリングするのではなく、スクリーンの一部分あるいは正面・側面・床面の複数のスクリーンをまたがる視体積を設定して映像をレンダリングする。動的負荷分散レンダリングの制御方法は映像空間を三分割する分割境界線の位置を左右に移動することで各 PC のレンダリング領域を変更する方法を行った。

図 6 に示すように切り分けたウィンドウのうち正面スクリーン側の映像(①)を 1 番目のノード PC で、側面スクリーン側の映像(②)を 2 番目のノード PC で、床面スクリーン側の映像(③)を 3 番のノード PC でレンダリングし、3 つの映像を重畳することで仮想ウ

ィンドウ内の映像空間を生成する。この分割境界線の位置はコントロール PC で決められ、UDP の通信によって各ノード PC に送られる。各ノード PC では送られた分割境界線の位置に従って同期してフラスタムの値を設定し直す。

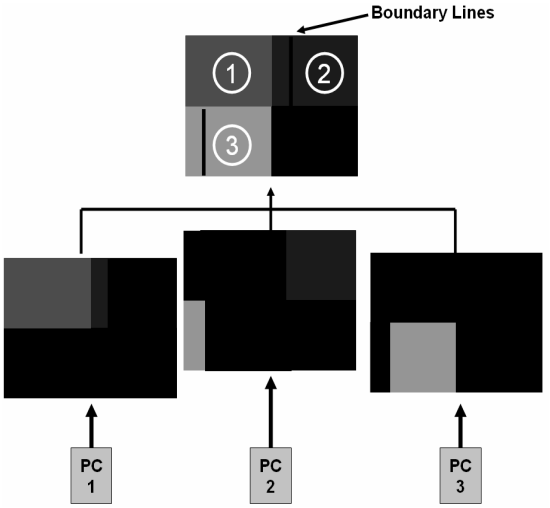



**図 6　負荷分散レンダリング**
Fig.6 Load-Balanced Rendering

### 3.2　動的負荷分散制御

前述したように CAVE システムでは仮想世界の体験中に、オブジェクトの位置や利用者の視点位置の変化に従って、各スクリーンのレンダリング負荷は変動する。このような負荷の変動に対して動的な負荷分散を実現するためには、画面を分割する各領域の幅を各 PC のレンダリング負荷の変化に従って変更する必要がある。このシステムでは、各ノード PC はレンダリング速度を常時計測しコントロール PC にデータを送る。コントロール PC は、レンダリンタイムが遅かったノード PC に対して、担当するレンダリング領域の幅を小さくするように分割境界線の位置を移動させる。

各分割境界線の位置の制御は、それぞれ隣接する PC 間だけで別々に判定する方法を取っている。すなわちノード 1 とノード 2 の間でレンダリングタイムが大きい方に最初の分割境界線を移動させ、ノード 2 とノード 3 の間でレンダリングタイムが大きい方に次の分割境界線を移動させる。全ノードのレンダリングタイムを同時に考えて最適化するように分割境界線を決める方法も考えられるが、クラスタの数が増えて分割数が多くなった場合、分割境界線の位置を決める計算負荷が逆に大きくなってしまうことを避けるため、ここでは個々に判定させる方法を取っている。また、コントロール PC は各ノード PC から送られてきたレンダリングタイム情報を用い、隣接する PC 間のレンダリングタイムの比率に従って各境界線の移動量を決めている。具体的にはレンダリングタイムの比の値について小数点以下の切り捨てた整数値が n であれば、n ピクセルの移動量とした。こうすることにより、1 つの PC だけに負荷が集中する状態が発生した場合には、移動幅を大きく取ることができるため、急激に負荷が偏ったときの対策となる。

一般にバーチャルリアリティのアプリケーションでは、利用者の視点移動や仮想物体の運動によって負荷の変動が生じるが、この場合負荷は徐々に変化するため、急激に偏ることは稀である。そのため本研究で用いた負荷分散制御の方法は簡易な方法でありながら十分に制御可能と考えられる。本システムでは 1 回の境界線の移動量は最大 60 ピクセルとし、境界線の制御はコンポジタによる映像の合成に合わせて 60Hz で行うようにした。そのため、急な負荷の変動がある場合でも 14 回の更新で境界線は画面の端から端まで（800 ピクセル）を移動することができる。また急な変化に対して境界線を速く追従させる必要がある場合は、一回の制御における移動量を大きくすることで対応できる。

また分割境界線を変更するかどうかの判定は、遅い方のレンダリングタイムが速い方の 1.2 倍以上なら移動させ、1.2 倍未満なら移動させないという判定基準にした。これは、レンダリングタイムの少しの差で分割境界線を変えていると、返って全体のパフォーマンスを落とすことになるからである

### 3.3　クラスタの拡張

クラスタを構成する PC の台数を増やすことによって全体のパフォーマンスを向上させる場合、従来のクラスタのシステムではスクリーンの枚数ごとに PC を増やしていかなくてはならなかったが、本システムでは 1 台ずつ PC を増やすことが可能である。例えば、3 面スクリーンの CAVE の映像生成に 3 台の PC を使用している状態から PC の台数を 4 台に増やすとする。その場合、映像空間の分割数を 1 つ増やして分割境界線の初期位置を調整することで対応することができる(図 7)。

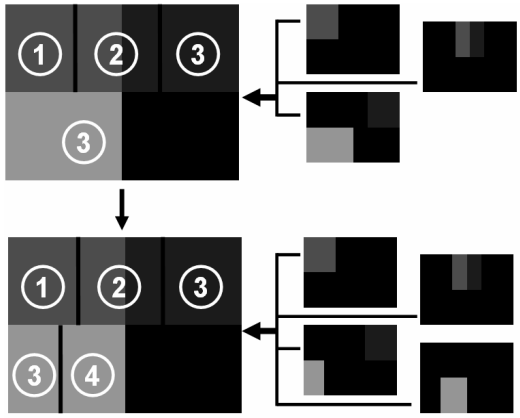



**図 7　PC3 台から 4 台にクラスタを拡張**
Fig.7 Method of changing the number of PCs

また、3 台の PC の内 1 台が故障して 2 台に減ってしまった場合でも、この方法では分割数を 1 つ減らすことで対応することができる。このように本手法ではスクリーン構成とクラスタ PC の対応が分離されているため、レンダリング領域の分割数と分割境界線の初期位置を調整することで、容易にクラスタの拡張や縮小を行うことが可能となる。

### 3.4 ビューフラスタムカリング

一般に映像の透視投影変換におけるビューフラスタムは、近面(Near)・遠面(Far)・左側面(left)・右側面(right)・上面(top)・下面(bottom)の 6 つのクリップ平面で構成されている。ビューフラスタム内に収まらない外側の部分はクリッピング処理されて描画されないが、ライティングや座標変換の処理は行われるため、無駄な計算負荷が生じる。さらに、動的負荷分散レンダリングでは 1 つのノード PC が複数のスクリーンにまたがり複数のビューフラスタムで映像を描くため、クリッピングされた部分の幾何演算負荷がパフォーマンスに大きく影響を与えてしまう。そのため本システムではビューフラスタムカリングを併用することで、動的負荷分散レンダリングの効果をより有効にしている。

ここではカリングの計算負荷が大きくならないように、各オブジェクトに代表点を設定しておき、代表点がフラスタムの中に入っているか入っていないかを調べる方法を採用した(図 8)[15]。この方法では各オブジェクトの代表点がビューフラスタムを構成する 6 つのクリップ平面のうちどれか 1 つに対してでも外側であればカリングを行い、そのほかの場合はカリングを行わない。

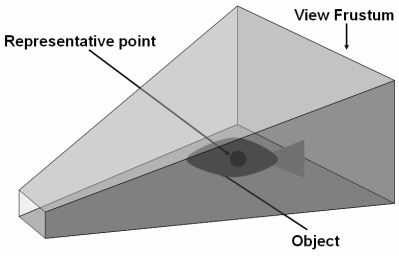



**図 8　ビューフラスタムカリング**

Fig.8 View Frustum Culling

この方法を CAVE のような多面スクリーンの映像に用いる場合、オブジェクトがスクリーンの境界付近にあると、オブジェクトの一部があるスクリーンには映っているのに他のスクリーンには映っていない等の問題を生じてしまう(図 9 )。この問題を解決するために、オブジェクトを球のバウンディングボックスと考えて、カリングの判定をするためのクリップ平面をバウンディングボックスの半径分だけ外側に広げた平面でカリングの判定を行う方法を用いた。この方法によりオブジェクトが部分的に欠けることなく全てのスクリーン間でカリングの判定結果が同一になるようにした。しかし、この方法ではオブジェクトサイズが大きい場合は、バウンディングボックスの半径だけ描画領域が広がるためカリングの効果が弱まってしまう。そのため代表点を持つ 1 つのオブジェクトが適当な大きさになるようにオブジェクトを階層的に分割して保持するなどの処理が必要となるが、この問題に関しては今後の課題である。

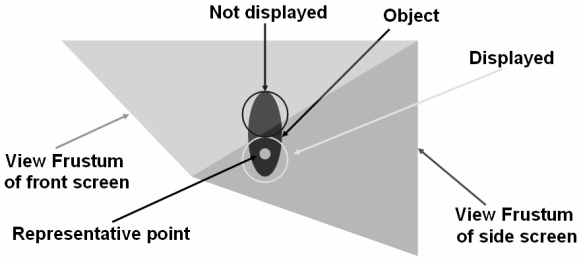



**図 9　多面スクリーンでのカリングの問題**

Fig.9 Problem of View Frustum Culling in multi-screen displays

## 4. 評価実験

### 4.1 オブジェクト移動実験

本研究で開発した動的負荷分散レンダリングの効果を調べるため、オブジェクトが移動する場合と、視点が移動する場合について評価実験を行った。まずオブジェクトが移動する際の評価実験としては、魚群の動きをシミュレーションした Boid の映像[16]を取り上げ、本手法を適用した(図 10)。このシミュレーションでは魚群の動きは、次の三つの規則から成る。

1. 各魚は近くの魚と衝突しないように自分から距離をとる。
2. 各魚は近くの魚と同じ向き、同じ速度で動いて、動きをあわせる。
3. 各魚は近くの魚の群れの中心に向かう。

この三つの動きを各魚にさせることで秩序を持った魚群の動きが生まれる。魚群は乱数に従った不規則な動きをするためレンダリング負荷は不均一にスクリーン間で絶えず変化し、全体のパフォーマンスは遅いノードに引きずられて低下する。このシミュレーション映像を CS Gallery に提示し、動的負荷分散レンダリングのアルゴリズムを適用した。図 10 は描画した映像空全体の映像、図 11 は CS Gallery に提示している様子を示したものである。

魚の中心にカリングのための代表点を作り、分割境界線の初期位置は均等な分割位置として始めた。本システムでは 6 台の PC を使って左右の視点映像を描画しているが、左眼の映像用のノード PC と右眼の映像用のノード PC は同じ領域の映像を描くため、実験のデータとしては片眼に対する PC のデータを取った。また全体のパフォーマンスは遅いノー

ドに引きずられるため、各 PC のフレームレートの最小値を取って、全体のフレームレートとした。

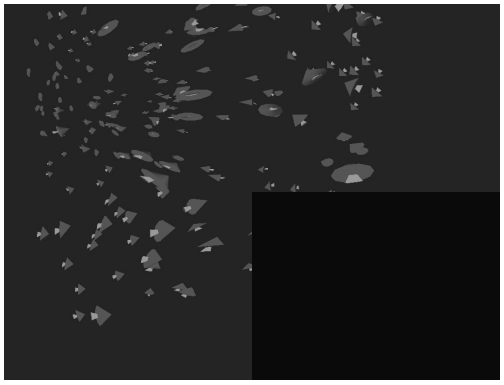


**図 10　シミュレーション映像**

Fig.10 Entire Image of Boid Simulation

図 11　　CS Gallery へ提示

Fig.11 Boid Simulation displayed in the CS Gallery

図 12 は、動的負荷分散レンダリングを行った場合と行わなかった場合のパフォーマンスの違いを時間軸で比較したものである。ここでは魚の数を 250 とし、シュミレーションループごとにレンダリングの更新周波数(Hz)の変化を示している。この結果から、従来の負荷分散を行わないクラスタ構成ではパフォーマンスが大きく変動していることが分かる。これは魚群の映像が動くとスクリーンごとの負荷が変化し、負荷が大きいところと小さいところができてしまう。負荷が均等にかかっているときは動的負荷分散レンダリングを行っている場合との差は少ないが、1 つの PC に負荷が偏ってしまうと同期を取っているため、遅いノード PC に他のノード PC が引きずられ、全体のパフォーマンスが落ちてしまう。一方、動的負荷分散レンダリングを行っている場合は、大きな変化後の負荷バランスが取れるまでの間は多少パフォーマンスが低下しているが、全体として高いパフォーマンスを示している。 t 検定を行った結果、有意水準 1%で有意であった。

図 13 は、レンダリング負荷の変化によって負荷分散の有効性がどう影響するか調べるために、描画する魚の数を変えながら、動的負荷分散処理を行った場合と行わない場合、またカリング処理を行った場合と行わなかった場合の比較を行った結果である。この結果から負荷の変化によらず、動的負荷分散レンダリングの効果は認められた。またこの効果はカリング処理によってより効果的であることが確認された。

図 14 は、クラスタのノード PC の台数を 1 台から 6 台まで変えて 3 面スクリーンに提示したときのパフォーマンスを示したものである。この結果から任意の台数で動的負荷分散レンダリングを実装することができ、ノード PC の台数の増加によってレンダリングパフォーマンスが向上されていることが示された。

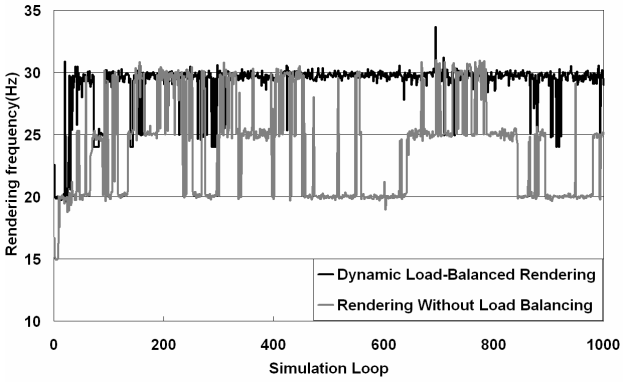



**図 12　動的負荷分散処理の結果の時系列比較**

Fig.12 Time variations in rendering performance

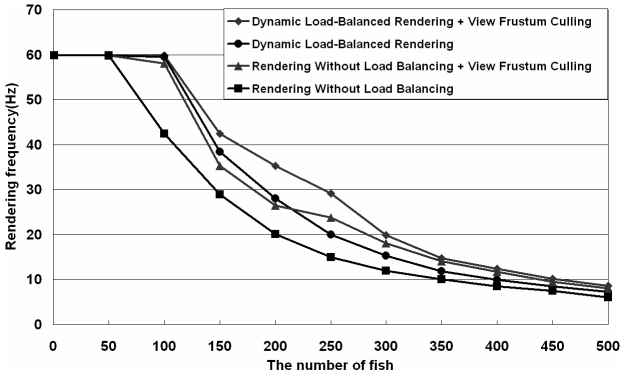



**図 13　レンダリング負荷によるパフォーマンスの変化**

Fig.13 Rendering performance for Changing the number of fish

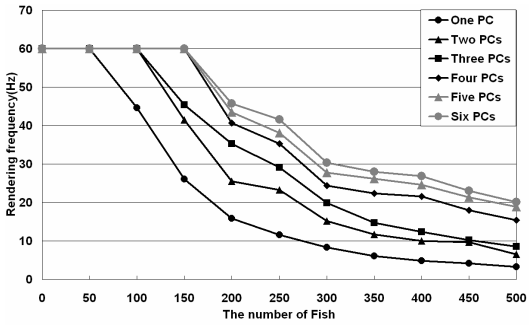



**図 14　クラスタの拡張の結果**

Fig.14 Rendering Performance for various numbers of node PCs

### 4.2 視点移動実験

次に多面ディスプレイにおける視点移動に対する動的負荷分散処理の効果を計測した。

描画する映像は、ディスプレイ空間の(0.4,0.5,0.6)（床面中央が原点、単位 m）から各軸の正の方向へ 0.02mごとに合計 1000 個(=10*10*10)の球(半径 0.02、経度方向の分割数 50、緯度方向の分割数 50)を表示し、このオブジェクトを動かさずに視点を①(0.3,-0.3,0.2) → ② → ③ (-1.0,0.3,0.2) → ④ → ⑤(0.2,0.0,1.7)→⑥→①の順に 120 秒かけて少しずつ移動して一周させて、それぞれの位置からオブジェクトを観測させた(図 15)。

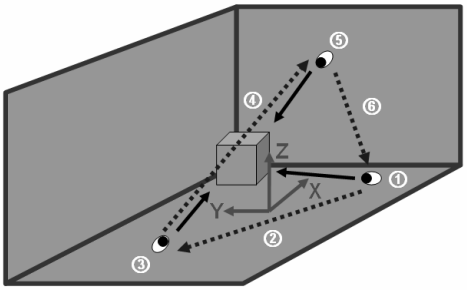



**図 15　視点位置の移動**

Fig.15 Movement of the viewpoint

図 16 は、視点移動したときの各スクリーンのパフォーマンスの推移を示している。実験データを見てわかるように、動的負荷分散処理をしないレンダリング方法では、1 つのノードに負荷が偏ってしまって全体のパフォーマンスは一番パフォーマンスが悪いノードに引きずられる。一方、動的負荷分散処理を適用すると、各ノードのパフォーマンスが近くなり全体的なパフォーマンスが上がっている。つまり、視点移動に伴ってレンダリング負荷の高いスクリーンが切り替わると、高負荷のスクリーンに計算資源を割り当てることで全体のパフォーマンスが上がり、本提案手法が有効に機能していることが示された。

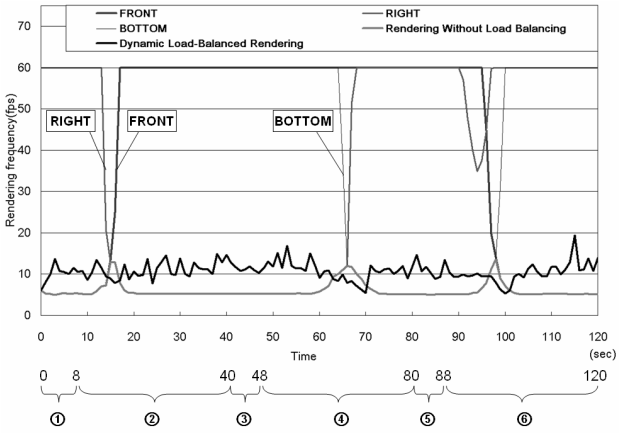



**図 16　視点移動実験のレンダリングパフォーマンス**

Fig.16 Rendering performance in the viewpoint movement experiment

## 5. まとめ

本研究では、チェーン接続型のコンポジタボードを使用することで、レンダリング負荷に対する動的負荷分散処理を行うことが可能な PC クラスタ型の CAVE システムの開発を行った。動的負荷分散レンダリングによる映像の重畳を行うことでパフォーマンスが向上し、計算機のレンダリング能力をより効率的に利用できることが示された。

この動的負荷分散レンダリングは、映像提示を行うプロジェクタ台数と独立に任意台数のクラスタを構成することができるため、種々の CAVE システムに柔軟に対応することができ、ディスプレイの構成に依存しない、スケーラビリティのある共通のハイパフォーマンスレンダリング手法として利用することが期待される。

## 謝辞

本研究は、情報通信研究機構の民間基盤技術研究促進制度に係る研究開発の一部として行った。

## 参考文献

[1] 小木哲朗：PC-CABIN による共有型没入仮想環境の構築，第 32 回可視化情報シンポジウム講演論文集，可視化情報 Vol.24，Suppl. No.1，pp.305-308，2004.

[2] Isakovic, K., Dudziak, T., Köchy, K.: “X-Rooms: A PC-based immersive visualization environment”, Web 3D Conference, Tempe, Arizona, pp.173-177, 2002.

[3] 山田俊郎,棚橋英樹,小木哲朗,廣瀬通孝:完全没入型 6 面ディスプレイ COSMOS の開発と空間ナビゲーションにおける効果，日本バーチャル学会論文誌「プロジェクション型没入ディスプレイ」特集号,vol4, No3,p531-538,1999

[4] Carolina Cruz-Neira, Daniel J. Sandin, Thomas A. DeFanti ,Electronic Visualization Laboratory (EVL), The University of Illinois at Chicago, ,Surround-Screen Projection-Based Virtual Reality：The Design and Implementation of the CAVE，Proceedings of SIGGRAPH’93，pp.135-142，1993.

[5] 廣瀬通孝，小木哲朗，石綿昌平，山田俊朗，“多面型全天周ディスプレイ（CABIN）の開発とその特性評価”，電子情報通信学会論文誌，J81-D-II-5，pp.888-896，1998.

[6] 林幸子,宮地英生,小野謙二:画像重畳装置とクラスタ装置を使った高速可視化システムの開発,.第32回可視化情報シンポジウム講演論文集,可視化情報 Vol.24,Suppl. No.1,pp.299-302,2004.

[7] G. Humphreys, M. Houston, R. Ng, R. Frank,S. Ahern, P. D. Kirchner, and J. T. Klosowski:“Chromium: A Stream Processing Framework forInteractive Rendering on Clusters”, Proceedings ofSIGGRAPH 2002, 2002.

[8] 橋本直己,長谷川誠,佐藤誠：“マルチプロジェクションディスプレイD-visionの開発”,映像情報メディア学会誌, 58,

3, pp. 409-417, 2004.
[9] 中嶋正之,高橋裕樹:”PCクラスタを用いた高精細大型立体ディスプレイシステムの開発”, 放送文化基金「研究報告」平成 12 年度助成・援助分
[10] Y. Chen, H. Chen, D. Clark, Z. Liu, G. Wallace, and K. Li: “Software Environments for Cluster-Based Display Systems”, IEEE International Symposium on Cluster Computing and the Grid, 202-210, May 2001.
[11] Jacobson, J., Rendard, M. L., Lugrin, J. L., Cavazza, M.: The CaveUT System: Immersive Entertainment Based on a Game Engine, ACE 2005, 2005.
[12] 村木 茂, 鈴木靖子, 藤代一成, ボリュームグラフィックス(VG)クラスタによる3D LICレンダリングの並列化, 情報処理学会研究報告 CAD 108-12, August 2002
[13] Ulf Assarsson and Tomas Möller, ``Optimized View Frustum Algorithms for Bounding Boxes'', paper , journal of graphics tools, vol. 5, no. 1, pp. 9-22, 2000.
[14] http://www.orad.tv/index.asp
[15] James H. Clark, “Hierarchical Geometric Models for Visible Surface Algorithm”, Communications of the ACM, vol. 19, no. 10, pp. 547–554, October 1976.
[16] Reynolds, C. W. "Flocks, Herds, and Schools: A Distributed Behavioral Model, in Computer Graphics," 21(4) (SIGGRAPH '87 Conference Proceedings) , pp.25/34.,1987

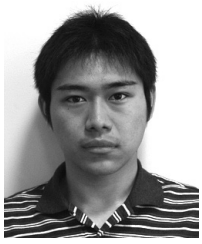
**（2006 年4 月4 日受付）**

［著者紹介］

**小木哲朗　（正会員）**

1986年東京大学大学院工学系研究科修士課程修了．同年（株）三菱総合研究所入社．1994年東京大学大学院工学系研究科博士課程修了．1996年東京大学IML助教授．1999年通信放送機構研究員．2004年より筑波大学大学院システム情報工学研究科助教授．没入型ディスプレイ技術，臨場感通信等の研究に従事．博士（工学）．

**内野孝哉　（学生会員）**



2006 年筑波大学第三学群情報学類卒業. 同年筑波大学大学院システム情報工学研究科博士課程入学, 現在に至る. 没入型ディスプレイを用いたクラスタシステムに関する研究に従事.