社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2003－CG－111 (5)
2003／5／12

高並列ボリューメトリックシミュレーションのための空間分割方法に関する一考察

村木 茂
(独) 産業技術総合研究所
ボリュームグラフィックス連携研究体

あらまし
スーパーコンピュータによるシミュレーション計算と，グラフィックスワークステーションによる可視化処理の間には，データ転送のボトルネックが存在する．構造格子上のシミュレーション計算と可視化処理をPCクラスタの分散メモリ空間上で同時に行えば，このボトルネックを解消し，PCの持つ高い計算性能とグラフィックス性能を活かした，非常にコストパフォーマンスの高いボリューメトリックシミュレーションが実現できると考えられる．しかし，この実現のためには，計算と可視化の双方にとって効率の良い負荷分散方法の開発が必要である．本報告では，開発中の計算可視化システム-VGクラスター-を用いて，いくつかの適応的空間分割法を評価し，この問題について考察する．

A Study on Space Subdivision Methods for Large-scale Parallel Volumetric Simulations

Shigeru Muraki
Collaborative Research Team of Volume Graphics, AIST

Abstract
Traditional large-scale simulation methods had a bottleneck problem for transferring numerical results from the simulation process on a super computer to the visualization process on a graphics workstation. The author believes that the high performance *volumetric simulation* is possible by performing these two processes on a single pc cluster system by distributing a common *structured grid* object space into node PCs of the cluster. However, to realize such the simulation system, it is very important to develop an efficient space subdivision technique, which is suitable for both computation and visualization. In this paper, I study on this matter by evaluating a few object space subdivision methods using a prototype visual-computing system, the VG cluster.

1. はじめに

PC の計算性能の向上と価格低下により，スーパーコンピュータに代わる大規模シミュレーション用計算機として，PC クラスタシステムが注目されている．一方，PC 用グラフィックスエンジンの性能向上も目覚しく，一部の機種では，ハードウェア 3D テクスチャマッピング機能を利用した，高速ボリュームレンダリングも可能になってきている[1]．構造格子（*structured grid*）ボリュームデータの，分割統治型ボリュームレンダリング[2]は，このような PC クラスタに最も適したアプリケーションの一つであるが，実装する際の最大の問題は，並列に生成した部分画像を合成する，画像重畳処理（*image compositing*）の効率である．筆者らは，PC クラスタ用フレーム重畳装置を開発することで，この問題を克服し [3]，現在は，この装置を用いた PC クラスタシステム（VG クラスタ）上で，シミュレーション計算と可視化処理を効率よく実行するための方法を研究している．本稿では， VG クラスタを実際に使用していて気づいた，いくつかの興味深い点について報告する．

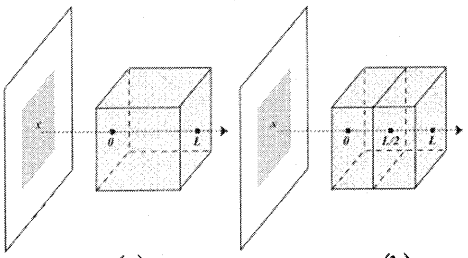
2. 分割統治法

2.1. 分散ボリュームレンダリング

構造格子のボリュームレイキャスティングでは，図 1(a)に示すように，画面上の点 **x** における輝度が，

$$I(\mathbf{x},\mathbf{r}) = \int_0^L C(s)\mu(s)\mathrm{e}^{-\int_0^s \mu(t)dt} ds \qquad (1)$$

のような積分で与えられる[4]．ここで，**r** は視線方向，$C(s)$ は視線上の点が視線方向に向かって放射する光の輝度，$\mu(s)$は光の吸収率である．今，図 1(b)に示すように，ボリュームを前後二つのサブボリュームに分割すると，式(1)は，



(a) (b)

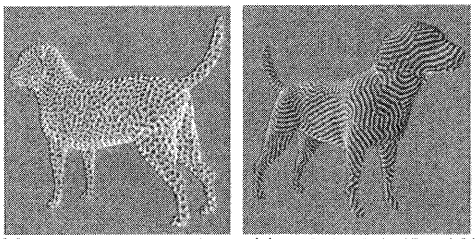
図1　分割統治型ボリュームレンダリング

$$
\begin{aligned}
I(\mathbf{x},\mathbf{r}) &= \int_0^L C(s)\mu(s)\mathrm{e}^{-\int_o^s \mu(t)dt}ds \\
&= \int_0^{L/2} C(s)\mu(s)\mathrm{e}^{-\int_o^s \mu(t)dt}ds \\
&\quad +\mathrm{e}^{-\int_o^{L/2}\mu(t)dt}\int_{L/2}^{L} C(s)\mu(s)\mathrm{e}^{-\int_{L/2}^s \mu(t)dt}ds \\
&\simeq I_f(\mathbf{x},\mathbf{r}) + \{1 - A_f(\mathbf{x},\mathbf{r})\} I_b(\mathbf{x},\mathbf{r})
\end{aligned}
\tag{2}
$$

のように変形できる．ここで，$I_f(\mathbf{x},\mathbf{r})$，$I_b(\mathbf{x},\mathbf{r})$は，それぞれ前後のサブボリュームのボリュームレンダリングの色成分，$A_f(\mathbf{x},\mathbf{r})$は前のサブボリュームのボリュームレンダリングのアルファ値（不透明度）である．つまり，分割したサブボリュームを別々にレンダリングした画像（サブイメージ）を，画面上で重畳（アルファブレンディング）することにより，完成画像を並列に生成することができる[1]．このような並列ボリュームレンダリング法は，サブイメージの前後関係が一意に決定できる凸分割であれば，いくつに分割されていても適用でき，PC クラスタによる大規模ボリュームデータの並列レンダリングに有効である．

分割によって生成されたサブボリュームのボリュームレンダリングは，PC クラスタの各ノード PC のグラフィックスプロセッサ(GPU)をプログラムして，高速に行うことができる[1]．一方，生成されたサブイメージは，通常のネットワークを使って集められ，CPU パワーを使って重畳されるので，グラフィックスエンジンの性能を最大限に利用することが難しい．そのため，筆者らは，PC クラスタでの重畳処理を高速化するための専用ハードウェアを開発している[3]．この装置は，オクトリー状の階層接続が可能で，ネットワークスイッチを利用する同様の目的の装置[5]に比べて大規模な並列ボリュームレンダリングに適しているが，サブボリュームが座標軸に垂直な平面で再帰的に分割され，平衡 2 進木（*balanced binary tree*）や *k-d* 木[6]で表現されていなければならない．

[1]式(2)の最後は，近似になっているが，この近似は$\mu(s)$の積分の指数関数を近似する際に出るもので，分割によって生じる問題ではない．



(a) a=0.1, b=1.1, d=20　(b) a=0.1, b=1.45, d=20

図2　反応拡散系シミュレーション

2.2.　局所近傍演算

本研究の目的は，PC クラスタによるシミュレーション計算と可視化の同時処理にある．ここでは，分割統治型ボリュームレンダリングの特徴を活かすために，シミュレーション計算が局所的に実行できるものだけを考える．

局所計算は，数値流体力学などの辺微分方程式や，3 次元画像処理でのフィルタリング処理などで，非常に多く見られる．図 2 は，反応拡散現象[7]を 3 次元構造格子上でシミュレーションした例である．この反応は，拡散性の活性因子 u と，抑制性因子 v の相互作用として，偏微分方程式，

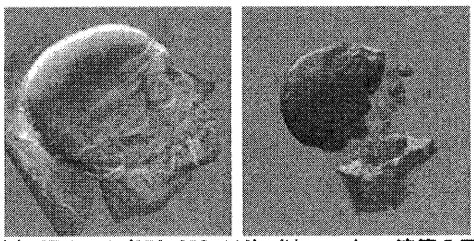
$$
\begin{cases}
\partial u/\partial t = \nabla^2 u + u^2 v + a - u, \\
\partial v/\partial t = d\nabla^2 v + b - u^2 v,
\end{cases}
\tag{3}
$$

で定義される．式(3)に現れる∇^2はラプラシアン（*Laplacian*）と呼ばれる 2 階微分演算子で，離散データに対しては，

$$
\begin{aligned}
\nabla^2 u(i,j,k) &= \frac{\partial^2 u(i,j,k)}{\partial x^2} \\
&\quad + \frac{\partial^2 u(i,j,k)}{\partial y^2} \\
&\quad + \frac{\partial^2 u(i,j,k)}{\partial z^2} \\
&\simeq u(i-1,j,k) + u(i+1,j,k) \\
&\quad + u(i,j-1,k) + u(i,j+1,k) \\
&\quad + u(i,j,k-1) + u(i,j,k+1) \\
&\quad - 6u(i,j,k)
\end{aligned}
$$

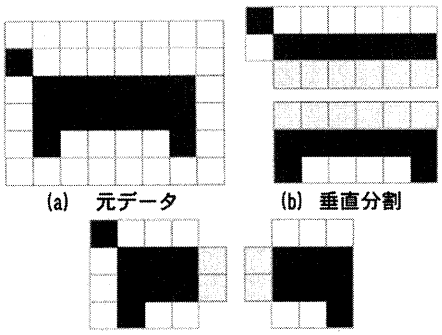
のような局所近傍演算で近似できる．式(3)を解くには，各ボクセルにおいて式(3)を計算し，u, v が収束するまで，適当な初期値 u_0, v_0 から，

$$
\begin{cases}
u_{T+1} = u_T + \left(\partial u/\partial t\right)_T \Delta T \\
v_{T+1} = v_T + \left(\partial v/\partial t\right)_T \Delta T
\end{cases}
\tag{4}
$$



(a) CT_head (256×256×113) (b) min/max 演算 5 回

図 3 モルフォロジ演算



(a) 元データ (b) 垂直分割

(c) 水平分割

図 4 空間分割と袖領域

のような更新を繰り返せばよい．パラメタ *a,b,d* の僅かな違いにより，図 2(a)，(b)のように，全く異なる模様（チューリングパターン）が出現するため，動物の表皮模様の出現や，生物の形態形成のメカニズムとの関連が研究されている．

局所演算の別の例として，図 3 のようなグレースケールモルフォロジ演算もある．図 3(a)は 256×256×113 の頭部 CT データのボリュームレンダリングであり，図 3 (b)は，各ボクセルに対して，そのボクセルと 6 近傍の濃度値の最大値（*max*）と最小値（*min*）を計算し，その濃度値を *min/max* で置き換える処理を 5 回繰り返した結果のボリュームレンダリングである．単純な演算で，不用な構造を除去することができるため，医用画像処理で盛んに応用されている．

2.3. 空間分割と袖領域

局所近傍演算は並列処理に適しているが，分散メモリ型の PC クラスタでは，袖領域処理（*ghost point exchange*，又は，*computational boundary exchange*）が問題となる．図 4 は，袖領域の問題を 2 次元で示している．

図 4 (a) をシミュレーション計算の対象となるボリュームデータとすると，拡散反応やモルフォロジ演算を行うのは，形状の定義されたボクセル（黒）だけであるから，背景ボクセル（白）は無視してよい．これらの処理を，2 つの PC で並列処理する場合，各 PC の黒ボクセルの数が，できるだけ均等になるように対称空間を分割すれば，計算負荷のバランスが良くなると考えられる．分割統治型ボリュームレンダリングを使用する制限から，座標軸に垂直な平面で分割するとすれば，図 4 (b)，(c) のような 2 通りの分割が考えられる．しかし，局所近傍演算では，計算対象のボクセルの近傍ボクセルの濃度値も必要であるため，図 4 (b)，(c) のように分割されたサブボリュームの境界では，灰色で示したような，袖と呼ばれるボクセルの濃度値情報を，隣接する PC からネットワークを経由して受け取らなければならない．図 4 (b)，(c) を見比べると，演算対象の黒ボクセル数は同じだが，図 4 (b) の方が袖ボクセル数が多く，ネットワークへの負荷が大きいことがわかる．一方，可視化のコストに比例するサブボリュームの体積（図 4 では面積）で比べると，図 4 (c) の左側のサブボリュームは 16 ボクセルで，他より大きい．つまり，袖領域処理では図 4 (c) が優れ，可視化処理では図 4 (b) が優れていることになり，空間分割の問題が単純ではないことを示している．

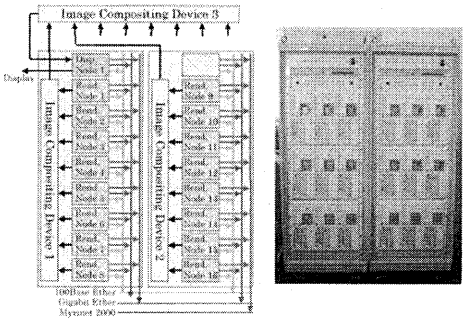


図 5 17 ノード PC クラスタシステム

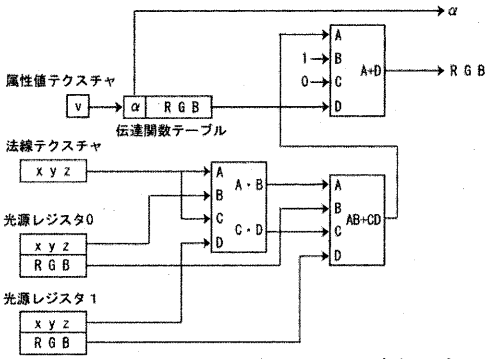
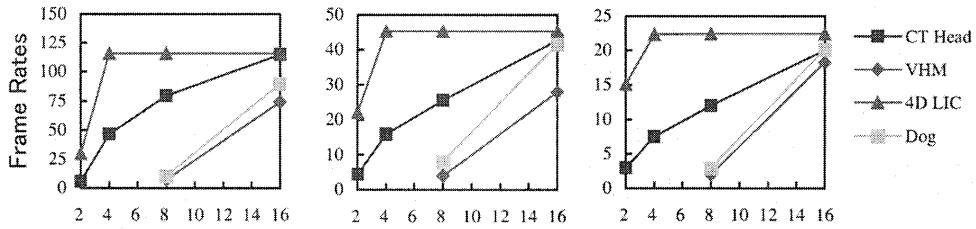


図 6 GeForce 4 レジスタコンビナ設定

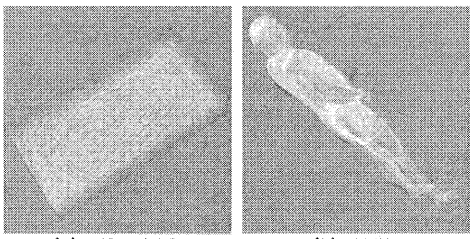
3. 実験

図 5 に示すような，Intel 製 Xeon Dual プロ



(a) 256×256　　(b) 512×512　　(c) 768×768

図 7　パイプラインモードでのフレームレート (Hz) (カーテシアン分割)



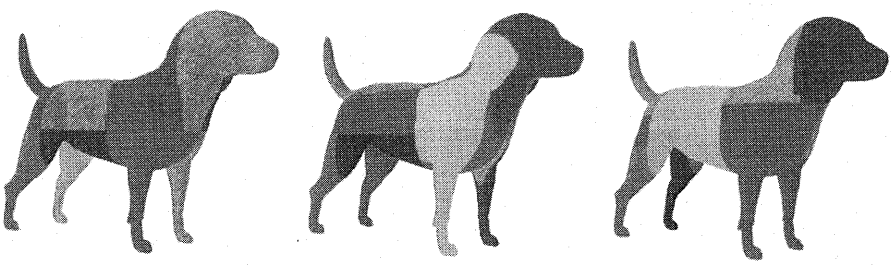
(a) 4D_ LIC　　(b) VHM

図 8　評価用ボリュームデータ．(a) 4D_LIC (240×120×60×16) と (b) VHM (430×240×939)．

セッサ（1.7 GHz）の PC システム 17 台から成る，PC クラスタを製作し，分割統治型シミュレーションの実験を行った．この PC クラスタ[2]は，2 つのサブクラスタで構成され，それぞれが，三菱プレシジョン製フレーム重畳装置を内蔵し，8 台のレンダリングノード PC（Rend_node）の画像出力を一つに合成できる．本実験では，2 つのサブクラスタの画像出力を，もう一つのフレーム重畳装置で合成し，一方のサブクラスタの表示ノード PC（Disp_node）の画面に表示することにより，最大 16 個のレンダリングノードを用いた，分割統治型ボリュームレンダリングが行えるようになっている．各レンダリングノードは，nVIDIA 製 GeForce 4 Ti 4600 グラフィックスエンジンを持ち，図 6 に示すレジスタコンビナー設定による，テクスチャベースのハードウェアボリュームレンダリング[1,8]を行う．この方法では，1 ボクセルあたり 4 バイト（濃度 1 バイト，法線 3 バイト）のグラフィックスメモリが消費され，GeForce 4 Ti 4600 が 128M バイトのメモリを持つことから，各 PC は，最大 256^3 ボクセル程度のボリュームデータを可視化できる．また，PC 間のメッセージパッシング性能の評価のために，100Base-TX，ギガビットイーサー，Myrinex 2000，の 3 種類のネットワークを備えている．

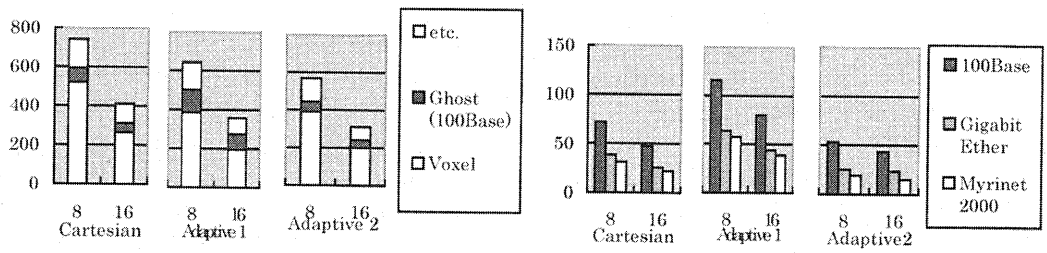
[2] ボリュームグラフィックス（VG）を得意とする PC クラスタということで，VG クラスタと呼んでいる．

まず，4 つのボリュームデータを用いて，本 PC クラスタのボリュームレンダリング性能を調べた（図 7）．データは，図 2，図 3 の生成に用いた，Dog（512×348×143）と CT_Head（256×256×113）の他に，図 8 に示す 4D_LIC（240×120×60×16）と VHD（430×240×939）を用いた．4D_LIC は，竜巻のシミュレーションで作った静 3 次元ベクトル場ボリューム（240×120×60）から，位相シフト LIC 法[9]で，流れを強調する 16 個の動ボリュームデータを生成したものである．4D_LIC は，16 個の時系列ボリュームを連続表示する時の表示速度，他のものは，それぞれの図に示したものと同様のシーンで，対象を回転させた時の表示速度を，3 種類の画面サイズ（256×256，512×512，768×768）で比較している．レンダリングノード数は，2，4，8，16 としたが，Dog と VHD は，グラフィックスメモリの不足のため，4 以下の計測ができなかった．また，レンダリングと重畳処理は，別スレッドで制御し，パイプライン処理を実現している．図 7 では，すべての画面サイズにおいて，4D_LIC 以外のデータで，レンダリングノード数の増加に伴い，フレームレートがほぼ線形に向上することが確認された．特に，レンダリングノード数が少ない場合に，ノード数の増加以上にフレームレートが向上する，スーパーリニア効果（*super-linear effect*）が観察された．この原因としては，空間分割により空ボクセルの一部が除去されることや，GeForce 4 のキャッシュ効果，主メモリの不足など，いくつかが考えられるが，PC クラスタによる大規模ボリュームデータの並列処理の有効性を顕著に示している．また，4D_LIC においては，レンダリングノード数が 4 以上において，フレームレートの向上が見られないが，これは各時刻のボリュームデータのサイズが小さいため（240×120×60），サブイメージのレンダリングスレッドが非常に早く終了し，重畳処理スレッドの終了を待っているためである．このことは，筆者らが開発したフレーム重畳装置の PCI バス転送速度の限界を示すとともに，ソフトウェアによる



(a) カーテシアン分割　　(b) 適応分割 1　　(c) 適応分割 2

図 9　シミュレーション計算と可視化のための空間分割方法



(a) 全シミュレーション時間 (秒)　　(b) 袖ボクセル交換に要した時間 (秒)

図 10　反応拡散シミュレーション (1 万回反復)

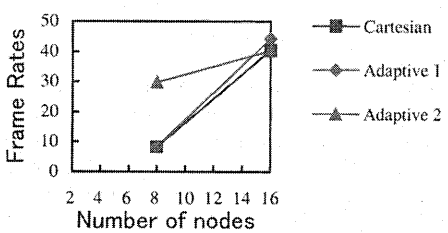


図 11　ボリュームレンダリングのフレームレート

重畳処理[10]では，もはや最新のグラフィックスエンジンの描画速度を活かすことができないことを示している．

次に，図 2 に示した反応拡散シミュレーションを使って，シミュレーション計算と可視化を同時に行う場合の，空間分割の問題を調べた．図 2 の生成に使用したデータは，犬のメタボールモデル（メタコーポレーション(株)製作）から生成した，512×348×143 ボクセルの内部が 1，外部が 0 の 2 値ボリュームデータで，ユーグリッド距離変換によって厚さ 5 ボクセルの体表面ボクセル(合計 1,105,026 ボクセル）を抽出している．このデータを可視化しながら，体表ボクセル上で反応拡散系シミュレーションを行うために，座標軸に垂直な面で 2 分割を繰り返し，8 個，又は，16 個のサブボリュームに分割した（図 9）．前節で述べたように，分割は座標軸の数だけの自由度があり，16 領域に分割する場合，3^{15} (=14,348,907) 通りの分割が可能となるが，それらをすべて調べることは，あまりにも計算コストが高い．そこで，再帰的に 2 分割を行う時点で，ボリュームデータの体積をなるべく均等化する図 9(b)のような分割（*Adaptive 1*）と，袖ボクセルの数をなるべく少なくする図 9(c)のような分割（*Adaptive 2*）の性能を比べることにした．図 9(a)は比較用の，負荷バランスを全く考慮しないカーテシアン（*Cartesian*）分割である．

図 10(a)は，図 2(a)の斑点模様が発生するまで，式(4)の更新を 1 万回繰り返した時間にしめる，計算時間（*Voxel*）と袖ボクセル交換時間（*Ghost*）である．レンダリングノード数が 8 と 16 の場合を，3 種類の空間分割方法について比較している．どの分割法でも，レンダリングノード数が 8 の場合には，16 の場合の約 2 倍の計算時間がかかっていることから，反応拡散系シミュレーションのような局所近傍演算が，PC クラスタによる並列処理に適していることが分かる．また，カーテシアン分割では，負荷バランスが不均衡なため，他の二つよりも計算時間が多くかかっている．適応分割 1 と適応分割 2 では，計算時間がほぼ同じだが，袖領域処理の時間で，適応分割 2 の方が 2 倍程度優れていた．図 10(b)は，袖領域処理の時間を，3 種類のネットワークで比較したものである．袖領域処理は，データの並べ替えに要する時間を含み，負荷バランスの影響も受けるので，ネ

ットワークの通信性能の違いだけでは評価できない．この実験では，ギガビットイーサーのパフォーマンスは，100Base-TXの2倍程度に過ぎず，ギガビットイーサーとMyrinet 2000の差はさらに小さかった．

図11は，512×512画素のボリュームレンダリング時間の，ノード数と空間分割方法による違いを示している．適応分割1は，ボリュームレンダリングの負荷バランスを良くするはずであったが，期待に反して8ノード時は適応分割2の方が高速であった．また，16ノード時の分割方法による差はわずかであった．適応分割は，分割後の状態のみで評価するので，繰り返し分割した後に最適な分割が得られるとは限らない．これは，ゲームプログラミングで，1手先だけを評価しても最適な戦略が決められないことと似ている．

4. おわりに

本稿では，筆者らが開発したフレーム重畳装置と，テクスチャベースのハードウェアボリュームレンダリングを用いることで，ボリュームレンダリングの性能を大幅に強化したPCクラスタシステム（VGクラスタ）の17ノードシステムを製作し，反応拡散系シミュレーションなどにおけるボリュームレンダリングとシミュレーション計算の性能を評価した．フレーム重畳装置は，オクトリー状の多段接続で，最大512PCまでの並列ボリュームレンダリングが可能な設計になっているが，今回の実験で，設計どおりに働くことが確認できた．しかし，インターフェースにPCIバスを用いているため，時系列ボリュームデータのアニメーション表示のように，サブイメージのレンダリングが非常に早く終了する場合に，重畳処理能力が不足する現象が観測された．PC用グラフィックスエンジンのレンダリング性能は，今後益々向上すると予想され，また，画素の情報量が128ビットに達するエンジンも現れはじめていることから，より高速な重畳装置の開発が必要になることは明白である．PCI Expressなどの次世代バスの動向を見ながら，慎重に検討する必要があるだろう．

一方，空間分割方法では，最適な分割をあきらめ，ボリュームサイズの均等性と袖ボクセルの数を評価基準とし，適応的な再帰的2分割を繰り返す方法をテストした．その結果として，袖ボクセル数を最少化する方法が，総合的に良いパフォーマンスを示したが，非常にデータ依存性の高い問題であり，今後，さらに詳細な検討が必要である．特に，異なる種類のネットワークやグラフィックスエンジンの混在するGridのような環境では，ハードウェア構造にあわせた複数の空間分割方法を組み合わせることで，システムの性能を最大限に引き出せるのではないかと期待している．

本研究は，科学技術振興事業団計算科学技術活用型特定研究開発推進事業13-D4「広域ビジュアルコンピューティング技術」の成果である．

参考文献

[1] Rezk-Salama, C., Engel, K., Bauer, M., Greiner, G., Ertl, T., Interactive Volume Rendering on Standard PC Graphics Hardware Using Multi-Textures and Multi-Stage Rasterization, *Proc. ACM SIGGRAPH/Eurographics Workshop on Graphics Hardware 2000*, pp. 109-118, 2000.

[2] Ma, K.-L., Painter, J.S., Hansen, C.D., Krog, M.F., Parallel Volume Rendering Using Binary-Swap Compositing, *IEEE CG&A*, Vol.14, No.4, pp.59-68, 1994.

[3] Muraki, S., Ogata, M., Ma, K-L., Koshizuka, K., Kajihara, K., Liu, X., Nagano, Y., Shimokawa, K.: Next-generation visual supercomputing using PC clusters with volume graphics hardware devices, *CD-ROM Proc. IEEE SC2001*, November 2001.

[4] Krueger, W., The Application of Transport Theory to the Visualization of 3D Scalar Field, *Computers in Physics*, vol. 5, pp. 397-406, 1991.

[5] Lombeyda, S.,Moll, L., Shand, M., Breen, D., Heirich, A., Scalable Interactive Volume Rendering Using Off-the-Shelf Component, *Proc. IEEE Sympo. Parallel and Large-Data Visualization and Graphics*, October 2001.

[6] 浅野哲夫訳，コンピュータ・ジオメトリ　計算幾何学：アルゴリズムと応用，近代科学社，2000.

[7] Turing, A. M., The chemical basis of morphogenesis, *Phil. Trans. Roy. Soc.*, B237, pp. 37-72, 1952.

[8] Cabral, B., Cam, N., Foran, J., Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware, *Proc. ACM Symp. on Volume Visualization*, 1994.

[9] Cabral, B., Leedom, L.: Imaging vector field using line integral convolution, *Computer Graphics (Proc. SIGGRAPH 93)*, pp.263-270, August 1993,.

[10] Humphreys, G., Houston, M., Ng, R., Frank, R., Ahern, S., Kirchner, P. D., Chromium: A Stream-Processing Framework for Interactive Rendering on Clusters, *ACM Trans. Graphics (Proc. SIGGRAPH 2002)*, Vol. 21, No. 3, pp.693-702, July 2002.