計算機アーキテクチャ　68－2
(1987・11・12)

空間分割型並列処理による光線追跡法の高速化に関する一検討

A Study of Object Space Parallel Processing for Fast Ray Tracing

窪田　英幸　小林　広明　中村　維男　重井　芳治
Hideyuki KUBOTA　Hiroaki KOBAYASHI　Tadao NAKAMURA　Yoshiharu SHIGEI

東北大学　工学部
Faculty of Engineering , Tohoku University

あらまし　本論文では、光線追跡法の高速化を目的としたオブジェクト空間分割型並列処理システムの構成、画像生成の並列処理機構、負荷分散法について述べ、そのシミュレーションによる性能評価の結果を報告する。小規模なシステムでは、オブジェクト空間のマッピングによる静的負荷分散法により並列処理の台数効果が得られる。しかし、効果的な負荷分散が得られるプロセッサ数には上限がある。そこで、高性能な大規模システムの構築のために階層化システムについて検討する。これにより、メモリ要求量を低く抑えつつ、多数のプロセッサを高効率で稼働できることを示す。

Abstract---This paper presents a multiprocessor system for fast ray tracing based on object space parallel processing. A parallel processing scheme for image synthesis and load balancing methods in the system are discussed. Firstly, as static load balancing, a mapping strategy of a regularly subdivided object space into the processors are evaluated by simulation. Moreover, we study a hierarchical multiprocessor system to overcome the limitation of the static load balancing in a large scale multiprocessor system. By using this architecture, nearly "ideal" load balancing can be achieved without noticeable increase in memory requirement for object description.

1．はじめに

近年、コンピュータグラフィックスは、良質のマンマシンインターフェースを提供するとしてＣＡＤ／ＣＡＭといった工業デザインからアニメーションにいたるまで幅広く用いられている。さらに、ディスプレイ技術及びコンピュータ技術の飛躍的な向上に伴い、非常にリアリティの高い画像の生成が可能になっている。しかしながら、画像の品質向上に伴いその処理時間も増大し、汎用計算機などを用いても実用的な処理速度は得られず、高速処理の要求が高まっている。

現在最も高品質な画像を生成できるアルゴリズムとして光線追跡法がある[1]。光線追跡法は比較的単純なアルゴリズムにもかかわらず光の反射、屈折などが扱えるため非常にリアルな画像が生成できる反面、処理速度が非常に遅いという欠点を持つ。このため、現在この処理速度を改善しようという研究が、ソフトウェアとハードウェアの両面から盛んに行われている。特に後者は最近のＶＬＳＩ技術の進歩に伴い、多数のマイクロプロセッサを用いた並列処理システムが、実際にいくつか試作されている[2][3][4]。これらは画素単位にプロセッサを割当てることによって、かなりの高効率の並列処理を行っている。

このような背景のもとで、光線追跡法のための並列処理システムのもう１つの形態として、オブジェクト空間分割型並列処理システムが提案されている[5]。これはオブジェクト空間を部分空間に分割し各々にプロセッサを割当てるものである。

本稿では、光線追跡法とその高速化手法としてオブジェクト空間分割型並列処理システムについて述べる。さらに、静的負荷分散がなされるようなオブジェクト空間の割付法とシステムの階層化について述べ、シミュレーションによる性能評価を行なう。

2．光線追跡法とオブジェクト空間分割による高速化

光線追跡法は[1]、オブジェクト空間（対象とする物体が定義されている空間）からスクリーン上の各画素を通り視点に入射する光線を、逆方向に光学現象に従って追跡し、スクリーン上の各画素の輝度を求めて画像を生成する手法であ

る。この光線追跡法は比較的簡単なアルゴリズムであるにもかかわらず、鏡面や透明体の反射光や透過光も扱え、非常にリアルな画像が得られる。しかしその反面、処理時間が膨大になるという欠点をもつ。この膨大な処理時間は画素すべてに対し光線追跡を行うことと、光線追跡に伴う物体と光線との交差判定を多量に行うことなどが原因である。

光線追跡法のアルゴリズム的な高速化へのアプローチとして空間分割法[6][7]がある。これは光線と交差する可能性のない物体を交差判定から除く手法である。オブジェクト空間は互いに隣接するいくつかの部分空間に分割され、光線追跡は、光線の貫く部分空間を順次決定しながら、各部分空間に含まれる物体と交差判定して行く。本研究で採用した分割法は、オブジェクト空間を３次元等分割し、均一な部分空間（セル）に等分割する方法である。この手法は、光線が進むべき次部分空間の決定が３次元ＤＤＡ[8]により高速に実行できるなど利点が多い。

３．オブジェクト空間分割型並列処理システム

光線追跡法では各画素は独立に計算可能であるため並列処理による高速化が可能である。また、空間分割によって交差判定処理を低減できる。光線追跡法のための並列処理システムの多くは画面分割を行ない各分割画面をプロセッサに割当てる方法を取っている（以降、このようなシステムを画素分割型と呼ぶ。）が、光線追跡法ではオブジェクト空間内で隠面除去と輝度計算を行なうため各プロセッサは空間内の全オブジェクト情報のコピーを持つ必要がある。さらに空間分割法を用いた場合、各プロセッサが持たなければならないデータ量はさらに増大する。

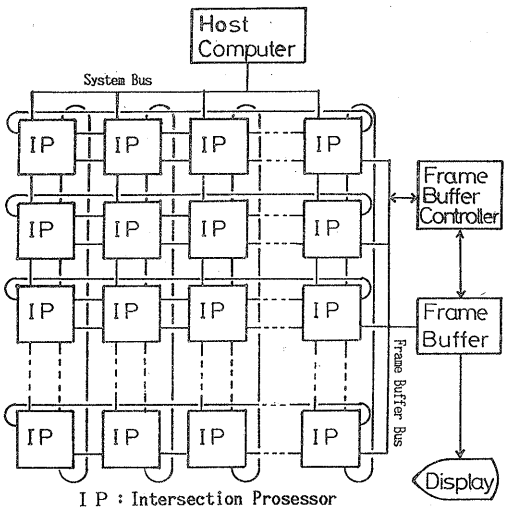


図１　オブジェクト空間分割型並列処理システムの構成
（トーラス型結合形態の場合）

これに対し、本システム（図１）はオブジェクト情報の分散配置及び効率よい交差物体の検出を目的としている。本システムでは分割したオブジェクト空間をプロセッサ(ＩＰ)に割当てるため、各プロセッサは部分空間内のローカルなデータのみを持てば良い。ＩＰはｎ次元の最隣接結合ネットワークで結合されており、これらＩＰ間で光線情報が入ったパケット（光線パケット）を通信することでオブジェクト空間内の光線追跡を行なう。視点から出される第一世代の光線パケットはホストコンピュータから視線が当たるＩＰへ送られ、その後あたかも光線が進むように（実際の光線の進む方向とは逆であるが）ＩＰ空間の中を進行する。もし光線が途中で物体に当たれば、そこで輝度計算がなされ、必要ならば次世代の光線パケットが生成される。光線がオブジェクト空間の境界に達したときはそのパケットは消滅する。このようにしてすべての光線パケットが消滅したとき計算完了となる。

３．１　通信パケットと輝度計算処理

３．１．１　パケット情報

ＩＰ間で通信されるパケットには２種類ある。１つは空間内で伝搬する光線に相当する光線パケットで、もう１つは、交差面が影になるかどうか判定するため光源方向に送られる付影パケットである。これらパケットの内容は以下のものを含み、そのサイズはおよそ２８バイトである。

- 光線パケットと付影パケットを区別するためのフラグ
- 画素の座標（ｘ，ｙ）
- 光線が進む次部分空間を決定する３次元ＤＤＡに必要なパラメータ
- 次に進むセルの指標（ｘ，ｙ，ｚ）
- 光線パケットの場合は光線の強度k_i（初期値は１）、付影パケットの場合は拡散反射成分輝度
- Binary shade tree[1] の深さ

３．１．２　光線パケット処理

ＩＰで光線に交差する物体が検出されたなら、

１）環境光成分をフレームバッファに加算する。

２）各光源ごとに拡散反射成分を計算し、その結果にk_iを乗算し、その値を持つ付影パケットを光源方向に送る。

３）必要なら、次世代パケットの生成を行なう。このとき前世代パケットのk_iと物体の鏡面反射係数または透過係数との積を、次世代パケットのk_iとする。

３．１．３　付影パケット処理

付影パケットがオブジェクト空間の範囲を越えたとき、そのパケットの持っていた輝度を、環境光と同様フレームバッファに加算する。付影パケットが物体にぶつかった場合は、そのパケットを消滅させる。

３．２　部分空間割付法とプロセッサ結合形態

オブジェクト空間を分割し各部分空間をプロセッサに割当てる場合、光線パケットが自律的に伝搬でき、プロセッサ間の通信距離を最小にする必要がある。この条件を満たす方法としてオブジェクト空間を３次元等分割し、その部分空間にプロセッサを割り付けることが考えられる。この場合、プロセッサの結合形態は３次元立方格子となり、１個のプロセッサは隣接する６個のプロセッサとのみ光線パケットの通信を行なうことになる。しかし、この方法は以下の問題点を持つ。

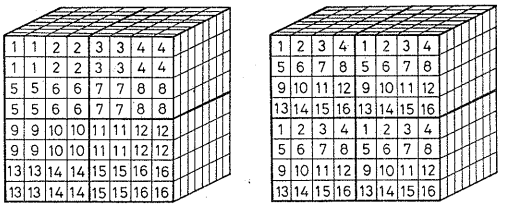
○　最適な分割数（画像によっては最適な分割数は数万に

も及ぶ）だけプロセッサを用意することは困難である。

○　物体が存在しない部分空間にプロセッサが割付られたり、透明体・鏡面がある部分空間を割当てられたプロセッサに処理が集中するなど負荷が偏る。

前者については、一つのＩＰに複数のセルを割付けることによって解決する。最も単純な割付法は、図２(a)のように隣接するセルを一定区間ごとに区切ってＩＰに割付ける方法である。このような割付法をブロック型と呼ぶ。

これに対して、図２(b)のようにオブジェクト空間中の"飛び飛び"の場所に位置するセルを１つのＩＰに割当てることで、オブジェクト空間内の特定の領域に偏って存在する負荷をできる限り多くのＩＰへ分散させる方法が考えられる。この場合、空間の連続性をプロセッサ上で保つためには、システムの各次元方向に沿ってプロセッサをリング又はトーラス状にサイクリックに結合する必要がある。このような割付法を負荷分散型と呼ぶ。



（a）ブロック型　　（b）負荷分散型

図２　部分空間割付法

３．３　前処理の並列処理

空間分割法による光線追跡法では分割された各セルにどの物体が存在するかを求め、そのリストを生成するといった前処理が必要である。この処理は光線追跡に比べ処理量は少ない。しかしながら、１台のプロセッサだけにこの処理を任せるには荷が重いため、この処理をＩＰに行なわせる。１個の物体に対する処理は１台のプロセッサで行なう方が効率のよい処理ができるため、ある物体の処理を１台のＩＰだけに割当てる。また、同一の形状の物体の場合、どの物体の処理の負荷が重いのか予測できないため、各ＩＰが処理する物体数を均等になるよう割当てる。

今回、ある物体を含むセルを見出すのに用いた方法を以下に述べる。まず、座標軸当りの分割数を因数分解し、その因数でオブジェクト空間を分割する。その分割された部分空間に物体が含まれないなら、その部分空間についての処理を終える。この処理をｘ，ｙ，ｚ軸それぞれに対して行なう。分割された部分空間に物体が含まれるなら、部分空間を次の因数で分割する。以上の処理を分割した部分空間が単一のセルになるまで繰返す（図３）。

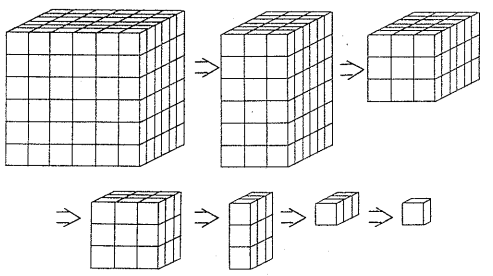
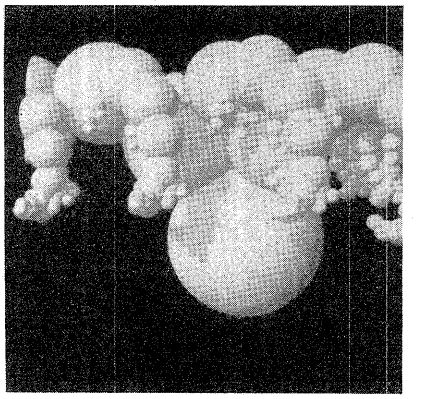


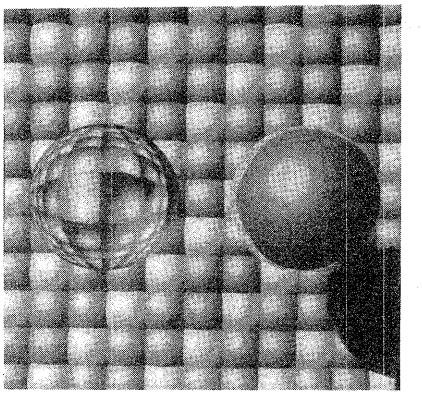
図３　オブジェクト空間の分割過程

４．シミュレーションによる性能評価

２つのサンプル画像を用いて、ブロック型と負荷分散型のマッピング方法についてシミュレーションにより評価した。オブジェクト空間の分割数は各次元方向当り２４個（セル総数は24^3個）とした。画像１(図４(a))は再帰的に定義された乱反射体の大小の球３６９個の画像である。画像２(図４(b))は前面に大きな透明体と反射体の球が１つずつ、後面に１２１個の乱反射体の小球が平面状に並ぶ画像である。



（a）画像１



（b）画像２

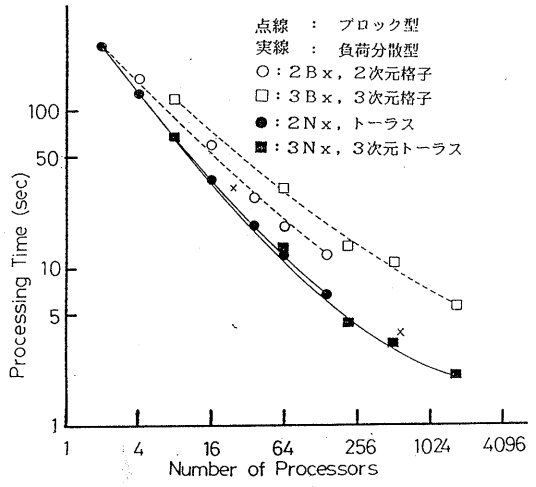
図４　サンプル画像

4．1　シミュレーションの仮定

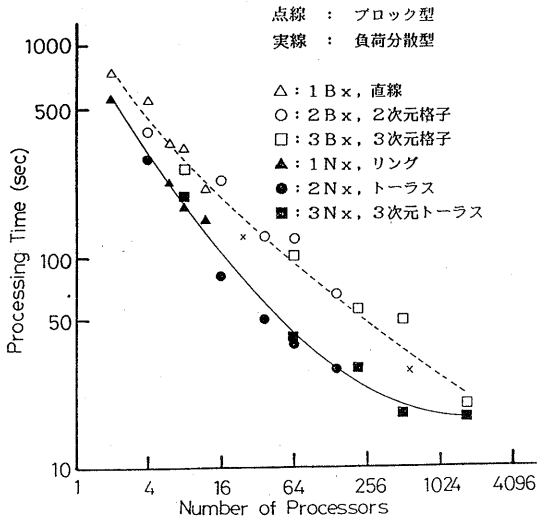
本シミュレータでは、各ＩＰがマイクロプロセッサＶ３０（クロック８ＭＨｚ）、および浮動小数点コプロセッサ８０８７各１個より構成されていることを想定して処理時間を設定した。その主なものを下に示す。この時間は、同様のプロセッサ構成を持つパーソナルコンピュータにおいて、Ｃ言語によって各処理を記述し、その実行時間をもとに決定した値である。

交差判定時間	１３００（不成功時）
	１７５０（成功時）
次部分空間決定時間	３４０
輝度計算時間	４６００
反射ベクトル計算時間	６８０
透過ベクトル計算時間	１１００
パケット解析時間	５８０
パケット作成時間	１６００　単位：μｓ

ＩＰ間のパケット通信に要する時間は、８ビットパラレルインターフェースを想定して、１パケット当り50μｓとした。

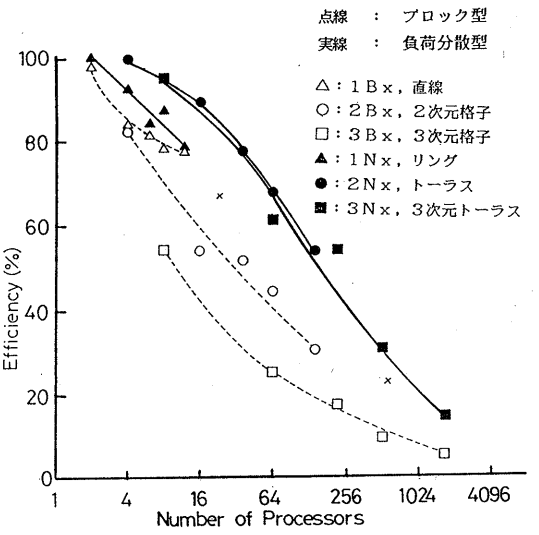


（ａ）画像１

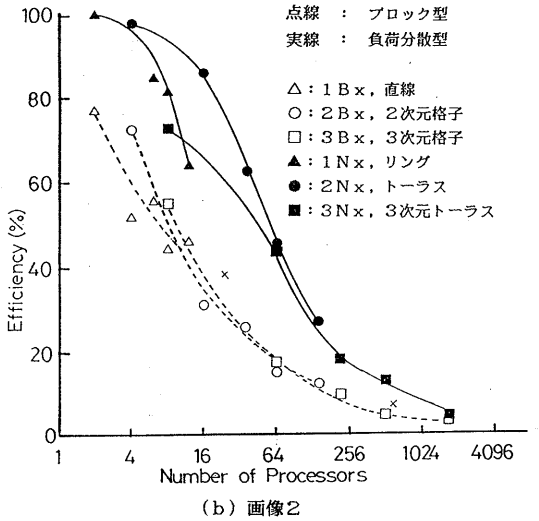


（ｂ）画像２

図５　プロセッサ数と画像生成時間の関係



（ａ）画像１



（ｂ）画像２

図６　プロセッサ数とシステム効率の関係

4．2　シミュレーション結果

4．2．1　画像生成時間及びシステムの効率

図５にＩＰ数と画像生成時間、図６にＩＰ数とシステムの効率の関係を示す。いずれも画素数は１４４×１４４で実験

を行なった。ここで、システムの効率は

（理想的な処理時間）／（実際の処理時間）　×　100　(%)

と定義する。

負荷分散型は効率で最大２倍以上ブロック型を上回り、負荷均等化による画像生成の高速化の効果が得られることがわかる。

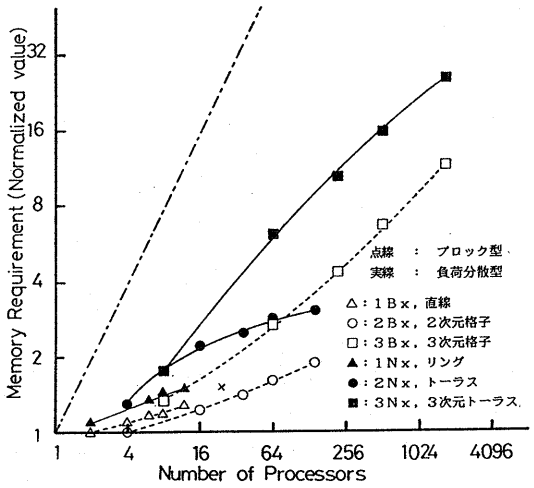
４．２．２　データメモリ要求量

理想的には、ＩＰの台数ｎのシステムにおける各ＩＰのデータ格納に必要なメモリ量は１／ｎになり、総データメモリ量はｎに無関係である。しかし、実際は１つの物体のデータが複数のＩＰに重複して格納されることがあるため１／ｎより多くなる。この点に関してはブロック型より負荷分散型の方が不利である。図７にＩＰの台数ｎとデータメモリ要求量の関係を示す。ここでデータメモリ要求量を

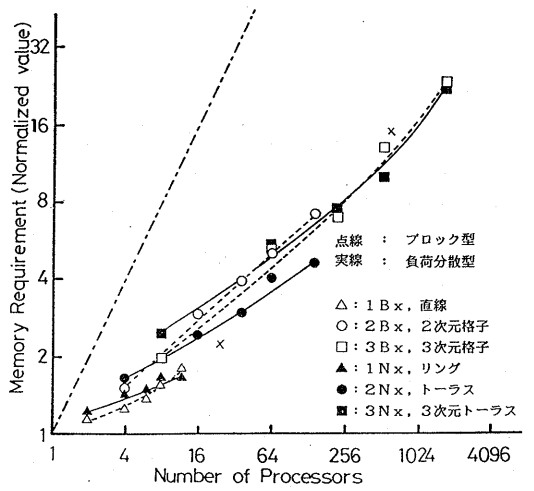
（ＩＰにおけるデータ格納に必要なメモリ量の最大値）

×　（ＩＰの台数ｎ）

と定義する。グラフからわかるように、ブロック型に比べ負荷分散型のデータメモリ要求量は若干多くなる。しかしながら、ブロック型、負荷分散型のいずれの場合においても、全プロセッサに同じデータを持たせた画素分割型システムでのメモリ量（図中、一点鎖線）に比べ下回っており、空間分割型のメモリ要求量での有効性がわかる。



（ａ）画像１



（ｂ）画像２

図７　プロセッサ数とデータメモリ要求量の関係

４．２．３　前処理時間

画像１についてＩＰ数と前処理時間の関係を図８に示す。ここでの単位時間は１平面によって分割される半空間に物体が存在するか判定する時間である。ＩＰ数が増加すると前処理時間は減少する。１台で９４３８単位時間要した処理が１６台で７３８単位時間、３２台で４３２単位時間とＩＰ数が多くなると台数効果は得られないが、並列処理によって処理時間は短縮されることがわかる。

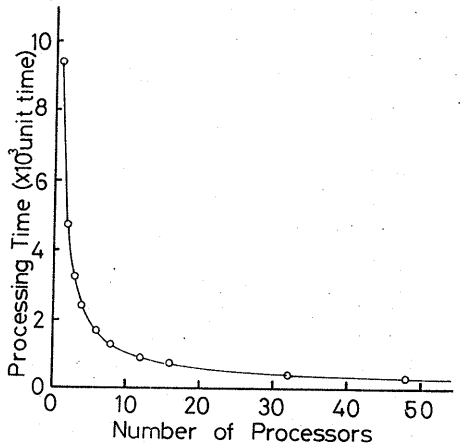


図８　プロセッサ数と前処理時間の関係

５．階層化空間分割型並列処理システム

５．１　システムアーキテクチャ

前章のシミュレーション結果において、部分空間のプロセッサマッピングによる負荷分散型割付法を用いるとＩＰ台数が１６以下の場合、効率が８５％を越える。一方、データ量はＩＰが１６台の場合、全プロセッサに同じデータを持たせた画素分割型システムに比べ６～７分の１に減っている。しかしながら、さらにプロセッサ数を増やすと著しくシステムの効率を低下させる結果になる。

高い効率を維持したままプロセッサ数を増やす方法として、空間分割型における各ＩＰを複数のプロセッサから成るクラスタとして構成し、クラスタレベルで前章で述べた負荷分散型空間割付を行なう階層化空間分割型システムを考える。クラスタ内のプロセッサが単一バス共有メモリ方式によって結合されるとすると１クラスタは図９のような構成になる。各プロセッサは２ポートメモリを持ち、共通バスを通して他のプロセッサからアクセスできるものとし、バスの制御、仲裁は専用のハードウェアが行なうものとする。通信ユ

ニットは隣接するクラスタから送られてくるパケットを各プロセッサがもつキューに書き込みむ。また、各プロセッサから送られてくるパケットを隣接するクラスタへ送るといった機能を果たす。

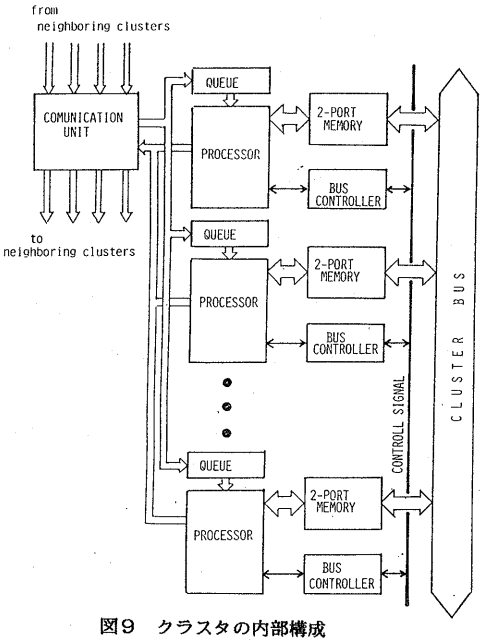


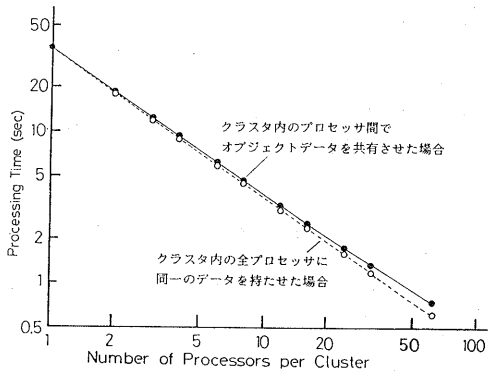
図９　クラスタの内部構成

５．２　シミュレーションによる性能評価

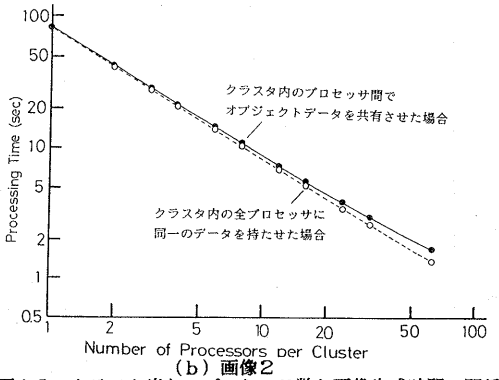
４章の結果から効率、データメモリ要求量の点から最も有効であると考えられるシステム（１６個のクラスタをトーラス状に結合させたもの）について、クラスタ内のプロセッサ数を変化させシミュレーションを行なった。また、条件として以下の２つの場合を考えた。１つは、オブジェクトデータを各プロセッサが持つ２ポートメモリに分散配置させた場合である。もう１つはデータメモリが十分に大きく、クラスタを構成するプロセッサすべてに同じデータを持たせることができ、バスを通してのメモリアクセスが行なわれない場合である。このシステムにおいて、各クラスタ内のｍ個のプロセッサすべてに同一のデータを持たせた場合、総データメモリ要求量はｍに比例して増えるが、システム内の全プロセッサに同一のデータを持たせた画素分割型に比べ、６～７分の１になるという関係は保持される。

５．２．１　画像生成時間及びシステム効率

図１０にＩＰクラスタ当りのプロセッサ数と画像生成時間の関係、図１１に１個のクラスタ当りのプロセッサ数と効率の関係を示す。クラスタを構成するプロセッサすべてに同じデータを与えたシステムでは、１クラスタ６４プロセッサで画像１を0.62秒、画像２を1.34秒（共に144×144画素）で生成できる。効率も８０%以上をを得ており、クラスタ当りのプロセッサ数に比例した台数効果が得られることがわかる。それに対してオブジェクトデータを共有させた場合は、１クラスタ６４プロセッサで画像１を生成するために0.75秒、画像２を生成するために1.67秒かかる。１クラスタ１２プロセッサ以下までのシステムでは８０%以上の効率が保たれることから、単一バス結合によるデータ共有でも１クラスタ１２プロセッサまでならクラスタ当りのプロセッサ数に比例した台数効果が得られる。



（ａ）画像１



（ｂ）画像２

図１０　クラスタ当りのプロセッサ数と画像生成時間の関係

５．２．２　データメモリ要求量

図１２にクラスタ当りのプロセッサ数とデータメモリ要求量の関係を示す。ここで、各プロセッサ間に共有させているデータはオブジェクトデータだけであり、セルにどの物体が含まれるかといった情報はクラスタ内のすべてのプロセッサが持っているものとしたため、１クラスタ１２プロセッサの場合で２倍強の差しか現れていない。しかしながら、テクスチャマッピング（サンプル画像では行なっていない）を行なう画像については共有するデータが増えるため、さらに両者の差が広がるものと考えられる。

６．まとめ

本稿ではオブジェクト空間分割型並列処理システムの概略とＩＰへのオブジェクト空間割付法、システムの階層化について述べ、シミュレーションによる性能評価を行なった。負荷分散型の空間割付を行なうことで、静的な負荷分散が可能であり、ＩＰが１６台の場合、全プロセッサに同じデータを持たせた画素分割型に比べデータ量が６～７分の１になり要求されるデータメモリ量の点で有利になることがわかった。

しかしながら、部分空間の割付による負荷分散では、それ以上ＩＰ数を増大させると負荷分散の効果は減少する。そこで高効率な並列処理システムを構築するために、システムの階層化を検討した。各ＩＰを複数のプロセッサで構成しクラスタ化を行なった場合、クラスタ内の全プロセッサに同じデータを持たせることによってクラスタ当りのプロセッサ数に比例した台数効果が得られることがわかった。また、クラスタ内の各プロセッサを単一バスで結合させ、オブジェクトデータを共有させた場合でも、１クラスタ１２プロセッサにおいて効率８０％以上を維持できる。この場合、クラスタ内の全プロセッサに同じデータを持たせたものに比べ、データ量が２分の１以下になり、システム内の全プロセッサに同一のデータを持たせた画素分割型システムと比較すると１２分の１以下になりデータメモリ量の点で有利であることが示された。

今後の課題として、より大規模なシステムの負荷分散及び動画生成時における負荷分散などが残されている。

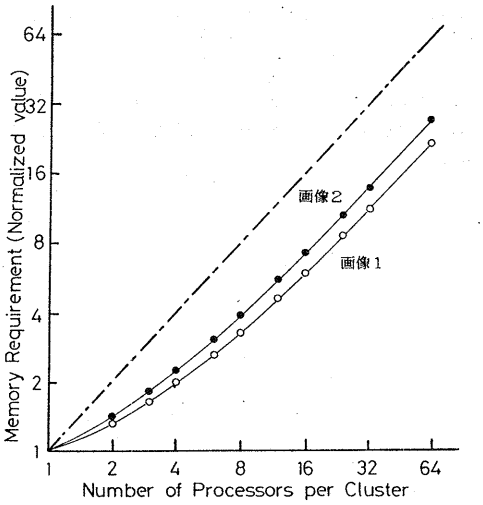
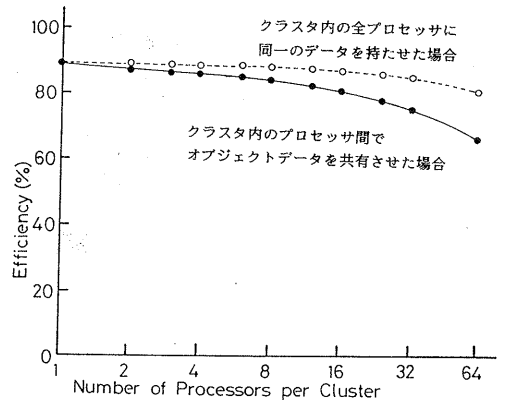
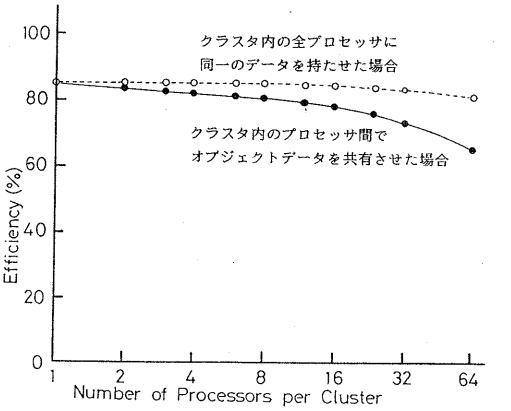


図１２　クラスタ当りのプロセッサ数とデータメモリ要求量の関係



（ａ）画像１



（ｂ）画像２

図１１　クラスタ当りのプロセッサ数とシステム効率の関係

［参考文献］

[1] T.Whitted , "An Improved Illumination Model for Shaded Display", Comm.ACM, Vol.23,No.6, June 1980, pp. 343-349

[2] 西村 他 , "コンピュータグラフィックスシステムＬＩＮＫＳ－１における並列処理の性能評価", 電子通信学会論文誌, Vol.J68-D, No.4, April 1985, pp.733-740

[3] H.Sato et al : "Fast Image Generation of Constructive Solid Geometry Using A Cellular Array Processor ", SIGGRAPH'85 Conference Proceedings, July 1985, pp. 95-102

[4] 吉田 他, "グラフィックス計算機ＳＩＧＨＴの基本構成", 情報処理学会研究報告、Vol.85-CA-60, No.60-5, Dec. 1985

[5] H.Kobayashi, T.Nakamura, and Y.Shigei , "Parallel Processing of an Object Space for Image Syntheses Using Ray Tracing", The Visual Computer,vol3,No.1, 1987, pp.13-22

[6] M.Dippe, and J.Swensen , "An Adaptive Subdivision Algorithm and Parallel Architecture for Realisitic Image Synthesis", SIGGRAPH'84 Conference Proceedings, July 1984, pp.149-158

[7] A.S.Glassner , "Space Subdivision for Fast Ray trcacing ", IEEE CG&A, Vol.4, No.10, Oct. 1984, pp.15-22

[8] A.Fujimoto, T.Tanaka, and K.Iwata , "ARTS: Accelerated Ray-Tracing System", IEEE CG&A, April 1986, pp. 16-26