博士学位論文
題目：画像生成用超並列処理に関する研究

平成 9 年 1 月 20 日 (月)

Revised for LaTeX2e 2004-9-29(Wed), no text changed.

情報科学研究科情報基礎科学専攻
山内 斉

目 次

図 目 次

1　緒論

1.1　本論文の背景及び目的

計算機によって駆動される初のディスプレイ装置が 1950 年に MIT において実現された。Whirlwind I と名付けられたこのベクトル走査型のディスプレイ装置は計算機に対話的な表示能力を与えることに成功した。その後、ディスプレイ装置は、表示物体数に上限のあるベクトル型からラスタ型へと発展し、表現力が飛躍的に向上した。ここにコンピュータグラフィクスという分野が誕生する。

初期のコンピュータグラフィクスはまず二次元の物体を表示することに重点がおかれていたが、やがて三次元物体の表現法へと進化していく。三次元物体の表示においては最初はワイヤーフレームという線の表現が主であったが、やがて面を用いて物体を表現するようになった。1980 年に入って Whitted らによりレイトレーシング法が発表されると物体間の光の相互作用を考慮した画像が生成されるようになり、計算機による現実世界の光のシミュレーションを行うことが可能になった。光学現象の解析手法はここに至って、理論的、実験的な解析についで、計算機シミュレーションという第 3 の手法を確立する。端的に言えば建築家は実際に建物を建てずとも建築物の詳細な照明設計を行えるようになったのである。現在、その応用分野は建築物のデザイン、照明のデザイン、工業製品のデザインなど、デザインの分野から、芸術、ゲーム、操縦訓練のためのシミュレータ、自然環境保護のための視環境の事前評価のためのシミュレーションなど広範囲に及んでいる。

上記のような応用分野においては、人と計算機が協力して作業を進めていくために、対話性というものが重要となる。この対話性として、表現力の高さと高速な表示能力が求められる。特に今日では、計算機の作成するシーンに対して現実世界と区別をつけ難いほどの表現力と実時間の画像生成能力を要求される。これまでに文献 [1, 12] に見られるように高速な画像生成のためのハードウェアが提案され、利用されている。これらのハードウェアは、サーフェイスモデルとして定義された三次元物体を局所照明モデルを用いてほぼ実時間で表示することが可能である。

しかしながら局所照明モデルは物体間の光の相互作用を考慮しない照明モデルであり、応用分野によっては表現力が不十分とされる。照明シミュレーションでは、たとえば舞台監督は舞台にどのように照明器具を配置すると効果的な演出が可能であるかや、あるいは建築家であれば建築物に対しどのように照明器具を配置すればより少ないエネルギで十分な照明が得られるか、ということなどに関心がある。局所照明モデルではこのようなことをシミュレートすることは不可能である。これに対し、大域照明モデルは物体間の光の相互作用を扱うことのできる照明モデルであり、上記のような要求に答えることができる。これまでに提案された大域照明モデルに基づく画像生成ア

ルゴリズムのうち、最も写実的な画像を生成可能なアルゴリズムはマルチパスレンダリング法 [4, 25] と呼ばれている。

マルチパスレンダリング法はレイトレーシング法 [24] とラジオシティ法 [5, 7] と呼ばれる 2 つの写実的画像生成法を組み合わせた画像生成法である。この画像生成法は生成される画像の質という点では、他の手法に追随を許さないほど優れているものであるが、膨大な計算時間を要するという欠点が存在する。局所照明モデルであればある 1 つの物体の明るさの計算はその物体と光源のみで決定できるが、大域照明モデルではある 1 つの物体の明るさは周囲にある物体全てから影響を受ける可能性があるため、その計算が膨大になるのである。つまり対話性における表現力については解決するアルゴリズムが存在するが、対話性におけるもう一つの重要な点、高速性をも同時に満たす画像生成法についてはこれまで未解決である。マルチパスレンダリング法はこの低速であるという欠点のため、高い表現力を有しているにもかかわらず、現在までほとんど用いられていない。そのため、表現力と高速性を合わせ持つ画像生成法が多くの分野で渇望されている。本論文はこの要望に対する 1 つの解答を与えるものである。

一方、近年 VLSI 技術の発展は著しく、従来では不可能であった数万個のマイクロプロセッサを実装する並列計算機を実現可能になってきた。マルチパスレンダリング法は、大域照明モデルを実現するため、光の伝播を物理的にとらえ、シミュレートするアルゴリズムという側面も持つ。自然現象としての光の伝播は空間内のあらゆる部分で並列に発生していると考えられる。よって、マルチパスレンダリング法を高速化するために、そこに内在する自然な並列性を並列計算機によって利用するというのは自然な考えであろう。この並列性を扱う計算モデルにはどのようなものが存在するであろうか。

従来の逐次処理を基本とする計算機は、RAM [18] と呼ばれる計算モデルを基本としている。この RAM は強力で単純な計算モデルであり、これを並列化した P-RAM [18] モデルが並列計算においては利用されることが多い。しかし、P-RAM モデルは共有メモリ型の並列計算モデルであり、共有メモリに対するアクセスのコストは RAM と同じく場所によらず一定である。現実の超並列計算機においてはこのような仮定は成立せず、そこには通信コストの問題や、複数の計算要素によるアクセス競合という問題が支配的である。つまり、P-RAM のような計算モデルを用いた超並列計算機の設計は、物理的な計算機とかけはなれたものとなる。このようなモデルを念頭において作成されたアルゴリズムは、実際の並列計算機上で動作させた場合にモデルとは大きく異なる挙動を示すであろう。したがって実際の物理的なモデルを考慮した計算モデルを構築する必要がある。このような計算モデルには、距離によるコストや、必要ならばエネルギの保存などということも考慮に入れた物理的に妥当な仮定が導入されなければならない。本論文ではこのように物理的なモデルを考慮した計算モデルの一つとして、オブジェクト空間分割型

並列計算モデルを導入する。空間的に離れた部分を計算機においても離れた位置に置くことによって実際の物理現象に対応する独立性を活用した並列計算を行うことが可能である。また、ある現象はより低レベルな現象を基本として発生するととらえると、並列計算機の構成はそれに従った階層的な構成となる。

本論文の目的はマルチパスレンダリング法を超並列計算によって高速化することである。そのために写実的画像生成アルゴリズムであるマルチパスレンダリング法を物理的な光の伝播現象のシミュレーションとしてとらえ、このアルゴリズムに内在する自然の並列性に着目する。この並列性を抽出するための並列計算モデルとしてオブジェクト空間分割型並列計算モデルを提案する。これは文献 [28] においてレイトレーシング法の並列化の基礎として提案されているもので、本論文ではこの並列計算モデルをより一般的にとらえ、マルチパスレンダリング法に適用し、マルチパスレンダリング法を並列化する。次に並列化されたマルチパスレンダリング法のアルゴリズムを効率良く実行する計算機アーキテクチャとして階層的な並列処理を行う $(M\pi)^2$ を提案する。また、超並列画像生成におけるフレームバッファへのアクセス競合を避けるためのキャッシュドフレームバッファシステムを提案する。そして、これら提案したマルチパスレンダリング法の並列化手法を $(M\pi)^2$ 上で実行した際の性能評価を行ない、マルチパスレンダリング法の超並列計算による高速化が可能であることを示す。

1.2 本論文の内容

本論文は以下のような構成になっている。

第 2 章は本論文の並列化の対象となっているマルチパスレンダリング法の並列化手法について述べる。まず、これまでの写実的画像生成法について概観し、画像生成の基礎式であるレンダリング方程式とその最良の近似とされているマルチパスレンダリング法について述べる。そしてこのレンダリング方程式を並列化するためのオブジェクト空間分割型並列計算モデルについて言及する。最後にこのオブジェクト空間分割型並列計算モデルに基づいてマルチパスレンダリング法を並列化し、並列化マルチパスレンダリング法のアルゴリズムを提案する。

第 3 章は、第 2 章で述べた並列マルチパスレンダリング法を効率良く実行するためのアーキテクチャ $(M\pi)^2$ を提案する。$(M\pi)^2$ は、オブジェクト空間分割型並列計算モデルに基づいて抽出された並列処理単位が階層的な構成を持つことに着目し、計算機そのものが階層的な構造を持つ。最初にこの階層化された $(M\pi)^2$ のアーキテクチャについて述べ、次に超並列画像生成時に発生するフレームバッファへのアクセス競合を解決するためのキャッシュドフレームバッファシステムを提案する。そしてキャッシュドフレームバッファ

システムの制御法を示し、待ち行列理論を用いて理論的な解析を行う。最後に $(M\pi)^2$ の能力を十分に引き出すための負荷分散法について静的な面と動的な面から言及する。

第 4 章ではこれまでに述べた並列マルチパスレンダリング法の有効性を評価する。はじめにソフトウェアシミュレーションにより並列化されたマルチパスレンダリング法を $(M\pi)^2$ 上で動作させた場合の評価を行なう。次にキャッシュドフレームバッファシステムを含んだ $(M\pi)^2$ の動作についての評価を行なう。

第 5 章は結論である。

2　マルチパスレンダリング法のためのオブジェクト空間分割型並列計算モデル

2.1　緒言

本章では、現在最も写実的な画像生成法として知られており、本論文で注目するマルチパスレンダリング法とその並列化手法について述べる。まず、写実的画像生成法としてのマルチパスレンダリング法を概観し、その内部に存在する並列性について述べる。そしてこれまでに行なわれてきた写実的画像生成法の並列化手法とその限界を述べる。次にこれまでの並列化手法とは異なった並列化のための新しい並列計算モデルであるオブジェクト空間分割型並列計算モデルについて述べる。最後にオブジェクト空間分割型並列計算モデルを用いて並列化したマルチパスレンダリング法のアルゴリズムについて述べる。

2.2　写実的画像生成法

2.2.1　レンダリング方程式と照明モデル

コンピュータグラフィクスが生まれてから写実的な 3 次元画像を生成するために多数の画像生成モデルの研究がなされてきたが、1986 年に J.Kajiya によってレンダリング方程式 [13] が発表され、この問題に結着がついた。レンダリング方程式は物体間の光の相互作用を物理的にモデル化した方程式である。写実性とは何かという問いに対する本論文の解釈は、「光の伝播現象を物理的に忠実にモデル化した画像生成法こそが写実的な画像生成法である」というものである。もちろんこのレンダリング方程式にも限界はあり、たとえば量子力学的な効果などは含まれていない。しかし、レンダリング方程式は通常の画像生成においては十分な解を提供するものであり、現在のコンピュータグラフィクスにおける画像生成の基礎となっている。レンダリング方程式を式 (1) に示す。この式の物理的な意味を図 1 に示す。

$$I(x,x') = g(x,x')\ [\ \epsilon(x,x') + \int_{\Omega} \rho(x,x',x'')\ g(x',x'')\ I(x',x'')\ dx''] \quad (1)$$

ここで、

$I(x,x')$	:	点 x' から x へ伝播する光量
$g(x,x')$	:	点 x' から x に対する幾何的項。x' から x への可視性を意味する。可視の場合には 1 であり、不可視であれば 0 となる。
$\epsilon(x,x')$	:	点 x' から x への放射光量
$\rho(x,x',x'')$	:	点 x'' から x' を経て x へ伝播する光の割合 (双方向反射拡散関数)

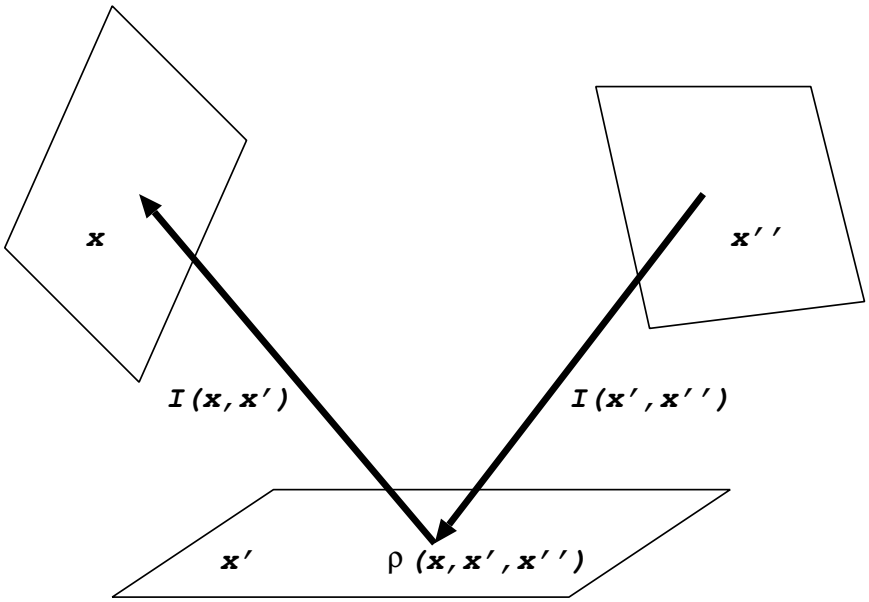

図 1: レンダリング方程式の意味

である。図 1 中の x から x' への光量 $I(x,x')$ を計算するには、まず x' が x から見えるかどうかである $g(x,x')$ を計算する。そして可視の場合には、x' の明るさを求める。この x' の明るさは、x' が自己放射する光 ϵ と周囲の他物体からの光に自己の反射率を乗じたものの和である。この周囲からの光をコンピュータグラフィクスの分野では環境光と呼ぶ。 x' に対する環境光の明るさは、他物体 x'' から x' への光 $I(x',x'')$ に x' 上の反射率 ρ を乗じたものを環境全体に対して積分したものである。物体によっては入射、放射方向が異なると反射率が異なることがあるため、ρ は x,x',x'' の関数となっている。

これまでに提案されている画像生成法は、ある物体の輝度を考える際に周囲の物体間の光の相互作用を考慮するかしないかにより、局所照明モデル に基づくものと大域照明モデルに基づくものに分類されていた。局所照明モデルは、式 (1) 中で x を視点とし、x' の明るさを求める際に積分項を無視するか、あるいは光源のみを対象に積分を行う。また、局所照明モデルでは他物体の影響である環境光を一定値として近似する。したがって、このモデルでは、注目する物体そのものの明るさとその物体が可視かどうかによって画像生成が行なわれる。通常、局所照明モデルにおいては物体そのものの明るさを決定するために光源を参照するが、光源以外の物体を考慮することはない。局所照明モデルに基づく画像生成アルゴリズムの代表的なものとしてスキャンライン法や z-buffer 法 [9] などがある。スキャンライン法も z-buffer 法も視点に対し最も近い物体を表示することで可視性の判定を行う。たとえば、

z-buffer 法では、z-buffer と呼ばれるスクリーンの各画素 (pixel) から物体までの距離を示すバッファを用意する。そして表示させたい全ての物体に対し、その物体と視点までの距離を画素ごとに求め、バッファの内容と比較する。バッファの内容により比較している物体の距離の方が視点に近い場合には注目する物体と視点との距離でバッファを更新し、その物体を表示の候補として保持する。この操作により全ての画素の内容は視点から最も近い物体となり、z-buffer の内容は可視の物体までの距離で埋められることになる。この処理は n 個の物体がある場合には 1 つの画素に対し n 個の物体を一回づつ比較することで達成されるため、画素数が N であれば、$O(Nn)$ の計算量で良い。また可視性の判定には個々の物体と視点さえ決定していれば物体間の関係を知る必要がないためにハードウェア化が容易であり、実際多くの画像生成用計算機で利用されている。さらにスキャンライン法や z-buffer 法は物体の一貫性 (coherence) を利用した増分計算を用いることで、更に高速化が可能である。たとえば、物体が平面で構成されていれば平面上の 2 点からその間を増分計算により線形補間することが可能である。このような性質もハードウェア化に向いており、局所照明モデルに基づく画像生成法は、現在、既に実用に耐える高速性を持つに至っている。

ところが、局所照明モデルでは、式 (1) の積分項を無視したために、物体間の光の相互作用を表現できないという欠点が存在する。ただし、局所照明モデルにおいても、シャドウポリゴンやマッピング手法を用いて擬似的に影や反射などを表現することが可能である。しかしながら、そのような方法は物理的なモデルによる裏付けを持たないうえに、画像生成を行う人間にある種の職人芸的な能力を必要とする。したがって、これらの手法は光学現象のシミュレーションという応用分野、例えば、建築物デザイン、照明デザイン、視環境シミュレーションなどに適用することができない。結論として局所照明モデルは画像生成の高速性のために正確性を犠牲にした照明モデルである。

これに対し、式 (1) の積分項を考慮した照明モデルが大域照明モデルである。この照明モデルでは、他の物体との光の相互作用を考えるため、反射や、屈折、透過、影などの光学的現象を表現することが可能である。大域照明モデルに基づく画像生成アルゴリズムの代表的なものが、レイトレーシング法 [24] とラジオシティ法 [5, 7] である。これらのアルゴリズムは、物体間の光の伝播を考慮するため、n 個の物体に対し、$O(n^2)$ の相互作用の計算を必要とする。レイトレーシング法ではこれを画素ごとに行うために、画素数 N に対し、$O(Nn^2)$ の計算量が必要となる。さらにレイトレーシング法では、物体表面での光の相互作用が反射と屈折を伴うので、実際の計算量は $O(Nn^2)$ にとどまらず指数関数的に増大する。また、ラジオシティ法は、各物体がそれぞれ他の物体にどの程度の影響をおよぼすかを決定するために環境に対してサンプリングを行なう。このため、サンプリング数を M とし、物体数を n とすれば、$O(Mn^2)$ の計算量となる。これらは先の局所照明モデルのアルゴ

リズムが持つ計算量のオーダに比べ、$O(n)$ 倍大きい。さらに、これら大域照明モデルに基づくレンダリングアルゴリズムは、物体上に存在する影などのために物体の一貫性を利用した増分計算による輝度計算の高速化が困難である。

以上のように写実的な画像生成のためにはレンダリング方程式として定式化された大域照明モデルに基づく画像生成法が不可欠である。しかしながら、大域照明モデルに基づく画像生成アルゴリズムは、計算量が大きいという欠点が存在する。次節以降では大域照明モデルに基づく画像生成アルゴリズムについてさらに詳しく検討していき、この計算量が大きいという問題の解決法を考えていくことにする。

2.2.2 マルチパスレンダリング法

大域照明モデルに基づく画像生成アルゴリズムには次のものがある。

- レイトレーシング法 [24]
- ラジオシティ法 [5, 7]
- マルチパスレンダリング法 [25]

このうちマルチパスレンダリング法はレイトレーシング法とラジオシティ法の逐次的な組み合わせである。レイトレーシング法とラジオシティ法の画像生成のアルゴリズムは一見全く異なる方式のように思われるが、それらの基礎式の差はレンダリング方程式の双方向拡散反射関数の近似方法のみである。

式 (1) の $\rho(x, x', x'')$ は、点 x'' から x' を経て x へと伝播する光の割合を示す項であり、物体表面の光学的性質を示す。コンピュータグラフィクスの分野では式 (2) に示すように、この項を主として鏡面反射項と拡散反射項の 2 つの要素に分けて考えてきた。

$$\rho(x, x', x'') = \underbrace{k_s \rho_s(x, x', x'')}_{\text{鏡面反射項}} + \underbrace{k_d \rho_d}_{\text{拡散反射項}} \tag{2}$$

ここで、

ρ_s : 鏡面反射率
ρ_d : 拡散面反射率
k_s : 鏡面の反射強度の割合
k_d : 拡散面の反射強度の割合
ただし、$k_d + k_s = 1$ である

である。

鏡面反射とは、鏡に対する反射のように光線の入射方向に対し、射出方向が一意に決まるものであり、広くは透過や屈折という現象も含まれる。一方、拡散反射とは、射出光線が入射方向にかかわりなくあらゆる方向に散乱する反射である。レイトレーシング法とは鏡面反射項を考慮した画像生成アルゴリズムであり、ラジオシティ法とは、拡散反射項を考慮した画像生成アルゴリズムである。そしてレイトレーシング法とラジオシティ法を組み合わせ、鏡面反射項と拡散反射項を共に扱う画像生成法がマルチパスレンダリング法である。マルチパスレンダリング法が現在最も写実的な画像を生成可能である方法とされるのはこのようにレンダリング方程式を忠実に実現したためである。以下それぞれ詳しく検討していくことにする。

レイトレーシング法　レイトレーシング法 [24] は大域照明モデルに基づく代表的な画像生成アルゴリズムの一つである。 レイトレーシング法の概要を、図 2 に示す。この画像生成法はスクリーン上の各画素ごとに可視の物体を視点から探索し、可視であると判明した物体の光学的特性に従って再帰的に光線を追跡する方法である。レイトレーシング法の扱うことのできる物体の光学的特性としては、反射、屈折、透過などがある。この画像生成アルゴリズムは、レンダリング方程式 (式 (1)) の双方向反射拡散関数をレイの追跡が可能な鏡面反射としてモデル化したものである。このアルゴリズムは、計算の際に必ず視点を必要とするため、視点依存の大域照明アルゴリズムとも呼ばれている。

レイトレーシング法おける各画素の輝度は次式によって求められる。

$$I = G + k_s S + k_t T \tag{3}$$

ここで

G	:	拡散反射輝度
k_s	:	鏡面反射率
S	:	鏡面反射方向から入射する輝度
k_t	:	透過率
T	:	透過方向から入射する輝度

である。ただし、G はある物体表面ごとに局所的にのみ求める。なぜなら、入射光に対して一様な反射をする拡散反射光は光の通過してきた履歴を失なうため、レイトレーシング法では追跡できないためである。このアルゴリズムでは透過光は鏡面反射を行なってきた光として追跡する。

ラジオシティ法　ラジオシティ法 [5] は物体間の光の伝播を熱エネルギの伝播ととらえ、物体間の熱放射の平衡状態を計算することによって物体の明るさを計算する画像生成法である。このアルゴリズムは物体の表面を理想的な

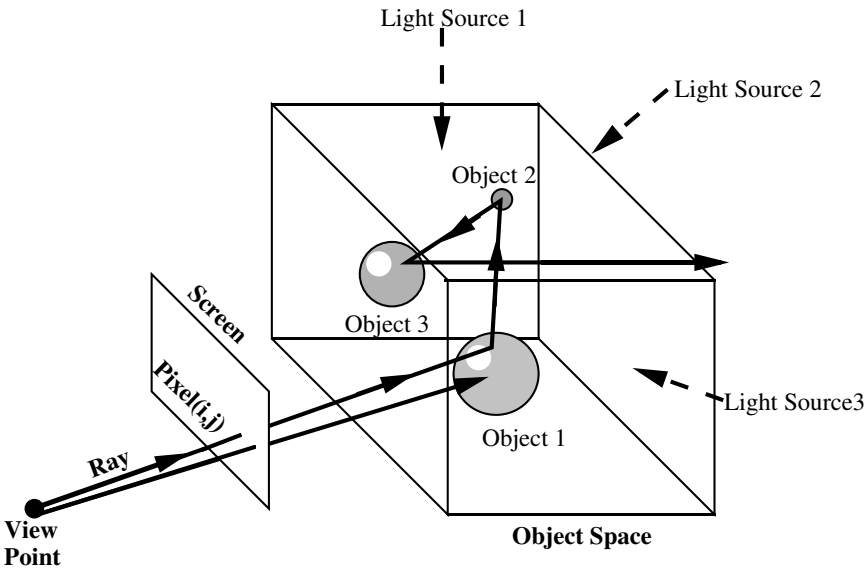

図 2: レイトレーシング法

拡散面であると仮定しており、レンダリング方程式の双方向拡散反射関数の拡散項を扱うことができる。また、ラジオシティ法では物体は小さな面に分割されており、これをパッチと呼ぶ。このアルゴリズムは計算の際には視点を必要としないために視点独立の大域照明アルゴリズムと呼ばれる。図 3 はラジオシティ法の概念図であり、複数のパッチ間でラジオシティが交換される様子を示している。

ラジオシティ法における各パッチの輝度は次式によって求められる。ここで添え字 i, j はそれぞれ i 番目、j 番目のパッチを示している。

$$B_i = E_i + \rho_i \int_{env} B_j F_{ij} \tag{4}$$

B:(ラジオシティ値) パッチから放射されるエネルギ値の総計 (energy/unit time/unit area)

E:(放射値) パッチから自己放射されるエネルギ (energy/unit time/unit area)

ρ: (反射率) 入射する光に対する放射する光の割合

F_{ij}: (フォームファクタ) パッチ i に対してパッチ j が影響する割合

マルチパスレンダリング法　これまでに述べたように、コンピュータグラフィクスの分野では双方向拡散反射関数を鏡面反射項と拡散反射項に分解して考えている。空間中に拡散面が存在するとそこに入射する光は一様な方向に反

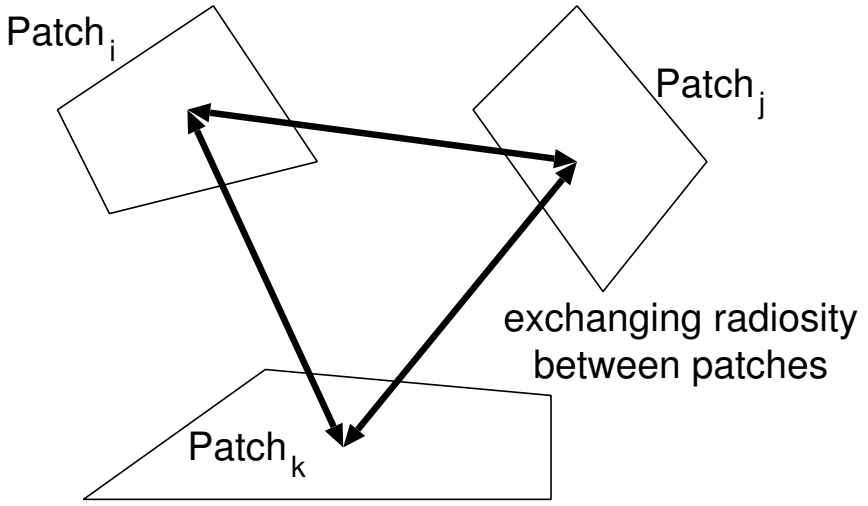

図 3: ラジオシティ法

射され、これまでに伝播してきた履歴を失う。したがって、この履歴を正しく扱うことを考えると、物体と光の相互作用が鏡面反射と拡散反射の 2 種類であっても、それらの組み合わせである 4 種類の光の伝播を考える必要がある。この 4 種類の光の伝播メカニズムを図 4 に示す。レイトレーシング法は図 4 における (a) のような鏡面間を伝播してきた光を扱うことができるが、ひとたびそこに拡散面が介在すると拡散面で反射した光を扱うことができなくなるため、図 4 における (b),(c),(d) のような光の伝播を扱えない。ラジオシティ法は図 4 における (d) のような伝播を扱うことが可能であるが、その他の伝播を扱うことができない。つまり、レイトレーシング法もラジオシティ法もそれぞれ単独ではレンダリング方程式の双方向拡散反射関数の拡散項と鏡面項を正しく扱うことができない。そこでそれぞれの利点を合わせた画像生成法がマルチパスレンダリング法 [4, 25] として提案されている。

マルチパスレンダリング法は、まず双方向拡散反射関数の拡散項をラジオシティ法によって求める。この時、拡散面間に介在している鏡面項を扱うことが可能なように拡張されたラジオシティ法を用いる。そしてその結果を利用してレイトレーシング法により画像を生成する。このような 2 段階の画像生成法により、鏡面反射と拡散反射を経た 4 種類の光の伝播を全て扱うことが可能になる。ところが、レイトレーシング法もラジオシティ法も膨大な計算時間を必要とする写実的画像生成法であり、これを逐次的に組み合わせたマルチパスレンダリング法はさらに膨大な処理時間を必要とする。そのため、局所照明モデルによる画像生成法のように実時間での画像生成はこれまで不可能であった。そこで本論文ではこのマルチパスレンダリング法の持つ膨大な処理時間を並列計算によって短縮することを考えた。次節ではまず、マル

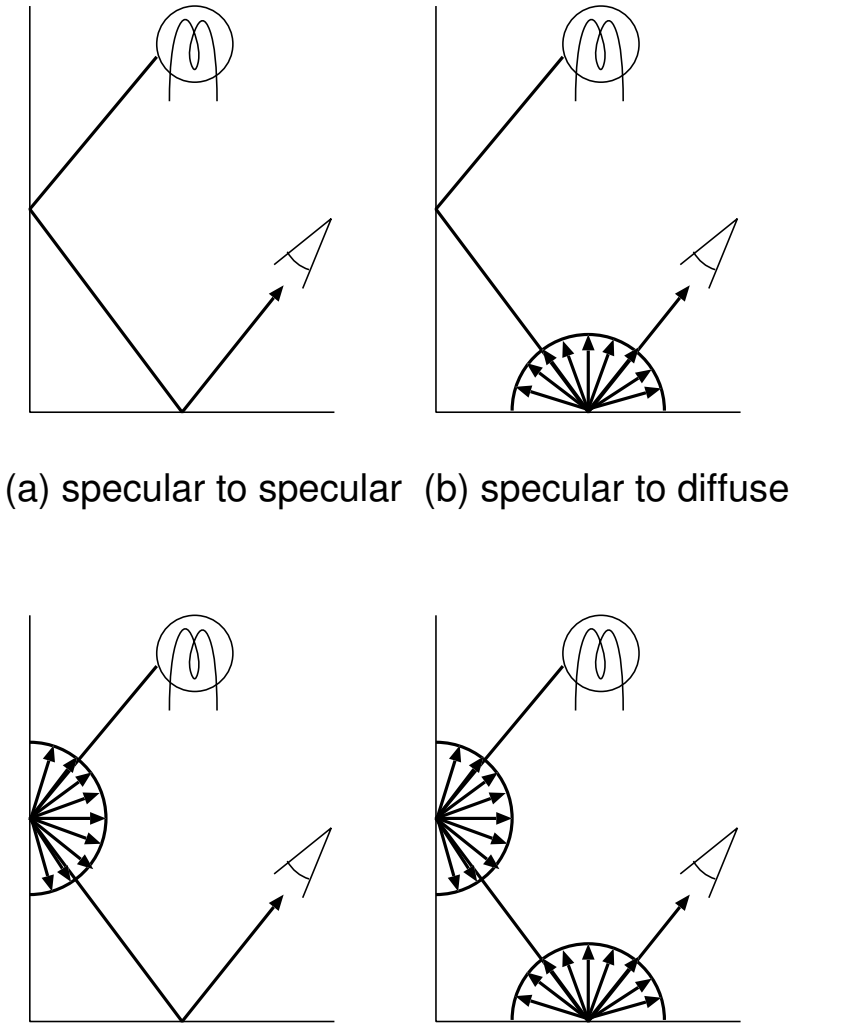

図 4: 4 種類の光の伝播形態

チパスレンダリング法の中に含まれる並列性について検討することにする。

2.2.3 マルチパスレンダリング法の既存の並列化手法

これまでのところ、マルチパスレンダリング法そのものを並列化の対象としてとらえたものはみあたらず、マルチパスレンダリング法の各ステージを個別に並列化したもの [3, 6, 8, 21, 22, 26] が存在している。そこで本節ではマルチパスレンダリング法の各ステージを個別に並列化する手法とその問題点について考えていくことにする。

レイトレーシング法のこれまでの並列化手法 レイトレーシング法におけるほとんどの計算時間はレイと物体の交差判定計算に費やされていることが報告されている [24]。したがって、これまでのレイトレーシング法の高速化はいかにレイと物体の交差判定を少なくするかということと、いかに交差判定を高速に行うかということに重点が置かれていた。

レイと物体の交差判定を減らす最も確実な方法は物体の数を減らすことであるが、これは写実性とは方向を異にする解決方法であり、写実的画像生成法の応用分野である照明シミュレーションなどには利用できない。そこで物体を空間位置によってグループ化し、交差判定が容易な境界立体で囲んでその境界立体との交差判定を行なうことが考えられた。この空間を分割するための方法として Octree 法 [10] や等分割法 [2] が考案されている。

Octree 法による空間分割では、空間を適応分割するために物体の存在に応じた無駄の少ない分割が可能であるが、その反面、分割の処理に時間がかかることやレイが空間を伝播する際にある部分空間からそれに隣接する部分空間を求める際のオーバヘッドが大きいという特徴がある。等分割法では物体の存在に適応した分割はできないが、空間分割の処理が単純で軽いことや、次空間探索のオーバヘッドが少ないという特徴がある。特に次空間探索に 3DDDA 法 [2] という増分計算法を用いることでこのオーバヘッドを軽減できる。

また、レイと物体の交差判定の計算を高速化する方法として交差判定計算の規則性を利用した専用ハードウェアを用いるもの [26] と並列計算がある。レイトレーシング法においては比較的早い時期から画素の輝度計算の独立性が知られており、これを利用した画素並列型の計算機 [3, 8, 22] がいくつか製作されている。

このうち専用のハードウェアを用いるものは 1 つの輝度の計算を細粒度の並列処理によって高速化するものであり、画素の計算の独立性ほどの並列性を利用することはできず、速度向上率も限られたものとなる。

画素の輝度計算の並列性を利用したこれまでの画素並列計算機としては一台のホストから各計算要素へ視線情報を割り当て、物体情報をブロードキャストして並列処理を行うマスタースレーブ型の並列処理方法 [8] や、分散共

有メモリ型の並列計算機を用い、各計算要素に画素ごとの計算を並列に行わせる方法 [3, 22] 等が提案されている。

大域照明モデルに基づく画像生成において、ある画素に寄与するレイは、空間中の物体と反射、屈折を行なうため、定義空間のどの部分へ伝播していくのかをあらかじめ知ることはできない。このために各計算要素は各画素の輝度の計算に際して定義空間のあらゆる場所にアクセスする可能性がある。したがって、定義空間内の物体情報は各計算要素の全てから見えなくてはならない。この時、定義空間内の情報をブロードキャストする方法 [8] ではブロードキャストするホストコンピュータの性能で限界が決まり、計算要素の台数を増加させた時にある所で性能の飽和が起こる。一方、文献 [3, 22] の方法は定義空間内の物体を共有メモリ上に置く方法である。レイが視点から発っせられて一回目の反射を行なう時には、近くの画素を担当するレイは同一の物体と交差する可能性が高い。ディレクトリキャッシュ方式の分散共有型の並列計算機による並列レイトレーシング法はこの参照の局所性を利用して共有メモリへのアクセス競合を緩和している。しかし、基本的に一度物体と反射、屈折を行なったレイは定義空間に対して不規則なアクセスを行うため、これらのレイの参照の局所性は急激に低下し、共有メモリへのアクセス競合が発生する。その結果システム性能が低下する。

以上のようにレイトレーシング法における画素並列処理には物体情報を論理的に共有しなくてはならないことと、物体情報へのアクセスパターンが不規則であるという性質があり、この 2 つの性質が性能の向上をさまたげている。文献 [22] では 48 台までの計算要素でレイトレーシング法を実行した結果、非常に良いスケーラビリティを得ているが、計算要素の台数が比較的少ない並列計算機であることと、各計算要素が比較的大きなキャッシュサイズを持ち、かつ画像生成に用いたデータセットが比較的小さいという条件があった。このような場合には物体情報は全てキャッシュに収まってしまい、結果として全ての計算要素が物体情報をコピーにより保持することと同等になる。そのために非常に良いスケーラビリティが得られていると推察される。よって、並列計算が真に必要となる大規模な画像生成問題を扱った場合には共有メモリへのアクセスの競合やキャッシュのミス率が増大し性能が低下することが予想される。

ラジオシティ法のこれまでの並列化手法　ラジオシティ法の並列化については、これまでいくつかの手法が提案されている [6, 21, 22]。

文献 [22] は階層的ラジオシティ法 [11] をディレクトリキャッシュを備えた分散共有メモリ型の並列計算機を用いて並列処理している。階層的ラジオシティ法は、物体間のラジオシティ値の交換を細かく分割されたパッチレベルで行うのではなく、いくつかのパッチをまとめて一つのパッチとみなしてラジオシティ値の交換を行うために通常のラジオシティ法よりも高速な実行が可能である。ここで並列化された階層的ラジオシティ法では、グループ化さ

れたパッチ間のフォームファクタ計算の独立性を利用している。

文献 [6] の方式はマススレイブ型の計算機を用いた方法である。この並列計算機はマスタープロセッサが独立に計算可能なフォームファクタマトリクスの一部を単位としてスレーブプロセッサに分配し、各スレーブプロセッサがその計算を並列実行する方法である。この方法はスレーブプロセッサの台数が増加するとマスタープロセッサの処理が飽和し、それ以上システムの性能が向上しないという欠点が存在する。

文献 [21] では分散メモリ型の並列計算機を用いてラジオシティ法を並列化している。この方法ではパッチの情報が各計算要素のローカルメモリに分散して蓄えられ、各計算要素はフォームファクタの計算を独立して行う。この時、各計算要素は自己の保持するパッチと他のパッチとのフォームファクタ計算のために他の計算要素の持つ物体情報にアクセスする必要がある。このアクセス速度を向上させるために文献 [21] では AMP と呼ばれる小さな半径を持つ特殊な構成のネットワークを用いて各計算要素を接続している。

以上のように、ラジオシティ法の並列化には基本的にフォームファクタ計算の独立性が利用されている。フォームファクタ F_{ij} とは、パッチ i からパッチ j が幾何的にどれだけ見えるか示す値であり、パッチ i とパッチ j だけで決定するものではない。たとえばある 2 つのパッチ間に一枚のパッチが存在することにより、2 つのパッチ間が遮られ、フォームファクタ値が減少する可能性がある。したがって、フォームファクタ計算のためには環境中にある全てのパッチ情報にアクセス可能でなくてはいけない。このように、フォームファクタ計算の独立性を利用した並列計算の場合もレイトレーシング法における画素並列性を利用する場合と同じデータ共有の問題が生じる。文献 [22] ではスケーラビリティのある結果を示しているが、これは比較的大きなキャッシュを持つ計算要素群に対し比較的小さな画像生成問題を与えたやめに共有情報が全ての計算要素のキャッシュに入ってしまったためと考えられる。そのため大きな画像生成問題であれば、共有メモリへの物体情報のアクセス競合の問題が発生するだろうと推測される。また、文献 [21] では分散した物体情報へのアクセスのために通信量が増大し、計算要素の数が 20 台程度になると性能の向上がみられなくなっている。

2.3 オブジェクト空間分割型並列計算モデル

これまでに見たように画素並列性を利用したレイトレーシング法の並列化や、フォームファクタ計算の独立性を利用したラジオシティ法の並列化には以下の二点の特徴がある。

- 各計算要素は物体情報の全てにアクセス可能でなくてはいけない
- 各計算要素の物体情報へのアクセスパターンは不規則である

これらの特徴のために並列計算機は物体情報を以下のいずれかの方法で持つ必要がある。

- 共有メモリ型の並列計算機の場合
  - 物体情報を共有メモリに保持する
- 分散メモリ型の並列計算機の場合
  - 各計算要素のローカルメモリに物体情報を分散させる
  - 全ての計算要素が物体情報を各ローカルメモリにコピーして持つ
  - 物体情報は 1 つの計算要素が一元して管理し、ブロードキャストによって各計算要素に渡す

共有メモリ型の並列計算機を用いて物体情報を共有メモリに保持した場合には、共有メモリへのアクセス競合が発生し、性能向上のために計算要素の台数を増加させてもある点からは性能の向上がみられなくなる。本論文では、共有メモリ型の並列計算機には共有メモリへのアクセス競合問題が本質的に存在し、大規模な並列計算機を構築して台数にみあった性能を出すことが困難であると考え、これより後では分散メモリ型の超並列計算機に焦点をしぼって検討する。

分散メモリ型の並列計算機を用いて物体情報を各計算要素に分散させた場合には、必要な物体情報を持つ計算要素と通信を行ない物体情報にアクセスしなくてはならない。この場合には必要な物体情報へアクセスするための通信量が増大し、通信の飽和によってシステムの性能が低下する。

分散メモリ型の並列計算機において全ての計算要素に定義空間中の物体情報をコピーして持たせる方法では、計算要素数に比例したメモリ量が必要となる。一般に超並列計算機には計算要素が多数あるため、総メモリ量は比較的大きいが、各計算要素に付属するメモリは小容量であることから、並列処理を適用しないと現実的に解けない大規模な問題に対してコピー方式は有効でない。

これまでに提案されているラジオシティ法、レイトレーシング法に対する並列化手法は問題の持つ並列性を抽出するものではあるが、その問題を実行する並列計算機についてはあまり考慮していない。また、これまでに提案された手法は基本的にデータを共有する解法であり、P-RAM [18] と呼ばれる並列計算モデルに基づいている。単一計算要素の計算機や小規模な並列計算機であれば P-RAM のメモリアクセスコストが距離によらず均一であるという仮定は良い近似となるが、現在の大規模な並列計算機ではこのような仮定はもはや成立しない。つまり、アルゴリズムを並列化する際の計算モデルとそれを実行する並列計算機のアーキテクチャの間にギャップが存在する。このギャップは大規模な並列計算機になればより顕著になり、実行時のオーバヘッドとして現われ、結果的に性能低下をひきおこす。

本論文では画像生成問題とは空間中で発生する物体間の光の相互作用の問題であるととらえ、この考えを基礎とした並列計算モデルであるオブジェクト空間分割型並列計算モデルを提案する [15–17, 28–30]。光は、物体上で反射、屈折、透過、拡散などの現象を繰り返しながら空間を伝播していく。つまり、物体間の光の相互作用は、光があるひとつの物体上で物体と相互作用することと空間を伝播していくということの繰り返しである。この光と物体の相互作用こそが並列処理の単位となるべきである。なぜならこの光と物体の相互作用は定義空間内に存在する物体上で局所的に並列に発生するため、他の物体情報に対する大域的なアクセスを行なう必要がないからである。大規模な並列計算機であれば共有部分へのアクセスや、離れた計算要素へのアクセスにはコストがかかることは明白であり、アクセスする部分を限定できるというこのオブジェクト空間分割型並列計算モデルは現実の並列計算機の実際を反映した並列計算モデルであるといえる。

この並列計算モデルに基づいた並列化により、画素並列計算やフォームファクタの独立性に基づく並列計算の問題である物体情報の共有という問題がなくなる。この光と物体の相互作用を並列計算の要素とすると、各物体につき 1 つの計算要素を割り当て、相互作用を行う物体を担当する計算要素間を接続することが考えられる。そして接続された計算要素間で光の情報をやりとりしながら画像を生成することが考えられる。この方法は物体の共有という問題を解決することはできるが、どの物体間で相互作用が発生するかはやはり計算してみないとわからないため、あらかじめ計算要素間を接続することはできない。また、計算要素間の結合はハードウェア上の制約から無限に自由というものではない。さらに大規模な画像生成問題で扱う物体情報は膨大なものになるため、超並列計算機といえども 1 つの計算要素を 1 つの物体に割当てることは現実的ではない。

そこで、物体そのものを計算要素に割り当てるのではなく、物体の存在する空間の一部を計算要素に割り当てる。1 つの空間中に複数の物体の存在を許容すれば物体数に比較して計算要素数を少なくすることができる。このように画像生成問題は定義空間を分割することで独立性の高い副問題へと分解され、各部分空間が計算要素に割当てられることになる。光は空間中を連続して直進する性質があるため、光の伝播は空間的な一貫性を持つ。そこで、計算要素に部分空間を割当てる時には隣合う部分空間を物理的に隣接した計算要素に割当てる。これによって通信は必ず隣接した計算要素間で局所的に発生し、大域的な通信が不必要になる。そして各計算要素は自己の担当する空間内の物体情報のみを保持すれば良いために物体情報の共有という問題も解決される。これがオブジェクト空間分割型並列計算モデルの基礎となる考えである。つまり、オブジェクト空間分割型並列計算モデルとは画素やフォームファクタの計算の独立性を並列計算の基礎にするのではなく、物体の定義された空間そのものを分割し、分割された部分空間を並列計算の単位とする

並列計算モデルである。

オブジェクト空間分割型並列計算モデルは大域照明画像生成問題そのものを並列化するための並列計算モデルであり、特定の画像生成アルゴリズムの並列化を行うものではない。そのため、同じ大域照明モデルを基礎としたレイトレーシング法とラジオシティ法のどちらをも統一して並列化可能である。これまではレイトレーシング法とラジオシティ法の並列化はそれぞれ別の問題と考えられていたが、この並列計算モデルにより両アルゴリズムを統一化して並列化することが可能になる。両アルゴリズムを統一して並列化可能であるということはこれまで分割して考えられていたそれぞれのアルゴリズムを実行する並列計算機のアーキテクチャをオブジェクト空間分割型並列計算モデルによって統一して扱うことが可能になるということを意味する。

2.4 並列マルチパスレンダリング法

本節ではマルチパスレンダリング法をオブジェクト空間分割型並列計算モデルに基づいて並列化する。マルチパスレンダリング法は、ラジオシティ法とレイトレーシング法を逐次的に接続した画像生成法である。まず、ラジオシティ法により拡散反射要素間の輝度が計算される。その結果をレイトレーシング法で利用しつつ画像を生成する。ラジオシティ法は視点を必要としない視点独立の画像生成法であり、レイトレーシング法は視点を必要とする視点依存の画像生成法である。視点は並列レイトレーシング法のステージではじめて定まるため、視点の変更のみであれば、並列ラジオシティ法のステージを再計算する必要はない。以下各ステージの並列化について述べる。この 2 つのステージを直列に実行すれば並列マルチパスレンダリング法となる。

2.4.1 オブジェクト空間分割型並列計算モデルに基づく並列ラジオシティ法

ラジオシティ法の基礎式は 2.2.2 節の式 (4) に示されている。この式はラジオシティ法におけるギャザリング方程式と呼ばれているものであり、あるパッチのラジオシティ値は自己の放射ラジオシティ値とまわりのパッチのラジオシティ値の積分値に自己の反射率を乗じたものの和であることを示している。したがって、ギャザリング方程式ではある物体の輝度を求めるために環境中の他の物体全ての輝度へアクセスする可能性がある。この方程式はそのままでは物体間の相互作用の計算である。そのため、物体とそれに相互作用する光の計算を並列計算の単位とするオブジェクト空間分割型並列計算モデルに従った並列化が難しい。

漸近的ラジオシティ法 [7] は、これまでのギャザリング方程式を基礎としたラジオシティ法の計算では、比較的輝度の低いパッチからも光を集めるという無駄な計算に時間を費やしている点に着目し、輝度の高いパッチを優先

的に考慮することを考えたラジオシティ法である。その際、輝度の高いパッチを選択してギャザリングを行うのではなく、輝度の高いパッチが他のパッチにどの程度影響を与えるかという考えを導入した部分に重要性がある。輝度の高いパッチが光を放射するために漸近的ラジオシティ法の基礎式はギャザリング方程式に対しショッティング方程式と呼ばれている。

ギャザリング法があるパッチの輝度を決定するために環境全体のパッチにアクセスすることに対し、漸近的ラジオシティ法では輝度の高いパッチが環境中に光を放出し、環境中のパッチに輝度を分配する。本論文ではこの光の放出の際に環境中のパッチを直接選択するのではなく、光を実際に環境に分配することを考えた。このことによって、物体間の光の相互作用の計算が、物体からの光の放出と物体が光を受けるという処理に分解される。この 2 つの処理はそれぞれの物体上で局所的に発生する。つまり、オブジェクト空間分割型並列計算モデルの基礎である物体と光の相互作用へと処理単位を分割可能である。そこで本論文ではラジオシティ法の並列化にあたって、この漸近的ラジオシティ法に着目した。

ギャザリング方程式を漸近的ラジオシティ法のショッティング方程式に変換するためには、式 (5) に示されるフォームファクタの相互関係式を用いる。この式をギャザリング方程式に適用すると式 (6) のようにあるパッチの輝度が他の環境中に与える影響を示す式に変形することができる。この式はラジオシティ値 B_i がパッチ i から環境中に放射した影響によってパッチ j のラジオシティ値 B_j がどの程度影響を受けるかを示している。ここで、A_i はパッチ i の面積を示している。

$$F_{ij}A_i = F_{ji}A_j \tag{5}$$

$$B_j \text{ due to } B_i = \rho_j B_i F_{ij} A_i / A_j \quad \text{for all } j \tag{6}$$

フォームファクタ F_{ij} を求めるために本論文ではヘミキューブ法 [5] を用いた。ヘミキューブ法ではパッチ上から放射されるラジオシティ値はヘミキューブ上の画素の解像度により離散化される。離散化されたラジオシティ値をデルタラジオシティ値と呼ぶ。そしてそのデルタラジオシティ値を運ぶ光を計算機上にモデル化したものをデルタラジオシティレイと呼ぶ。この物体とデルタラジオシティレイの相互作用は、オブジェクト空間分割型並列計算モデルで並列処理の基本単位とした光と物体の局所的な相互作用である。そこで、これを並列計算の単位として漸近的ラジオシティ法を並列化する。ヘミキューブ上の各画素を通過するデルタラジオシティレイの計算が独立であることを利用すると、デルタラジオシティレイをパッチから環境中へと並列に放射させることが可能になる。この様子を図 5 に示す。

並列化した漸近的ラジオシティ法のアルゴリズムは次のようになる。

1. 前処理として以下の処理を行う。

   - 空間を部分空間へ等分割する。物体情報も同じ格子で分割される。

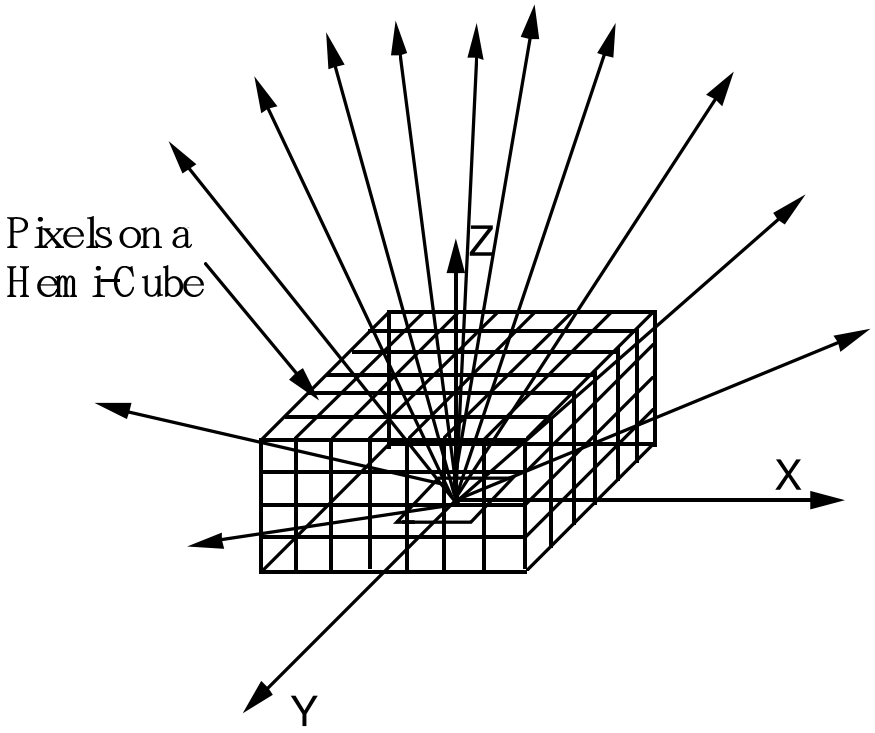

図 5: ヘミキューブ上からのデルタラジオシティレイの並列放射

- 物体情報を計算要素へと分配する。

2. 担当空間内にユーザの定めた放射輝度以上の輝度を持つパッチが存在する場合には最大輝度のパッチを選択し、そのパッチからデルタラジオシティレイを生成する。

3. 処理しなくてはならないレイが部分空間内に存在する場合にはデルタラジオシティレイと空間中の物体との交差判定を行う。交差判定の結果により次のいずれかの処理を行う。

   - 物体とデルタラジオシティレイが交差した場合には交差した物体のラジオシティ値を増加させる。
   - 物体とデルタラジオシティレイが交差しなかった場合には、次空間へとデルタラジオシティレイを伝播させる。次空間の計算には 3DDDA [2] 法を用いる。次空間を担当する計算要素が自己でない場合には通信によりデルタラジオシティレイを次空間を担当する計算要素へ送り、デルタラジオシティレイを次空間へ伝播させる。定義空間を外れた場合にはデルタラジオシティレイは消滅する。

4. システム全体からデルタラジオシティレイがなくなったら計算終了。

この並列漸近的ラジオシティ法では、二次曲面のような滑らかな鏡面要素を扱うために鏡面要素はパッチに分解されないで保持される。デルタラジオシティレイは鏡面要素上では、反射、屈折、透過を行い、空間中を伝播していく。

デルタラジオシティレイの生成と空間内の物体と光の相互作用は各計算要素上で並列に実行される。この様子を図 6 に示す。

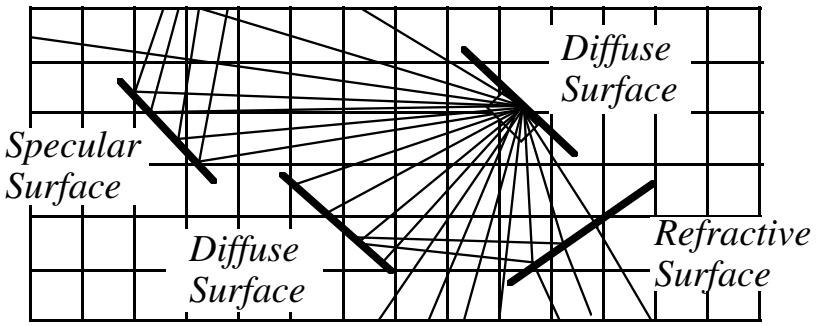

図 6: 並列漸近的ラジオシティ法

2.4.2 オブジェクト空間分割型並列計算モデルに基づく並列レイトレーシング法

並列漸近的ラジオシティ法により拡散要素の計算終了後、視点とスクリーンを決定し、並列レイトレーシング法により画像を生成する。第 2.2.2 節の式 (3) のレイトレーシング法の基礎式は並列レイトレーシング法でもそのまま用いることが可能である。これまでの画素並列レイトレーシング法では画素を単位とし、複数の物体との交差判定、輝度計算を含んだ処理が並列処理の単位であったが、オブジェクト空間分割型並列計算モデルに基づく並列レイトレーシング法では一つのレイが一つの空間内を通過することを並列処理の単位とする。この考えに従って並列化されたレイトレーシング法のアルゴリズムは次のようになる。

1. 前処理として以下の処理を行う。これは並列漸近的ラジオシティ法と共通であるので、並列漸近的ラジオシティ法の直後に並列レイトレーシング法を行うのであれば必要ない。
   - 空間を部分空間へ等分割する。物体情報も同じ格子で分割される。
   - 部分空間と物体情報を計算要素へと分配する。
2. 視点とスクリーンの情報を全ての計算要素に分配する。
3. 視点から発っせられるレイが最初に通過する空間が自己の担当する空間であるかどうか各計算要素が並列に計算し、自己の担当する空間が最初の通過空間であれば初期レイを発生させる。
4. 処理しなくてはならないレイが存在する場合にはレイとそのレイが存在する空間内の物体との交差判定を行う。交差判定の結果により次のいずれかの処理を行う。
   - 物体とレイが交差した場合には、反射、屈折、透過などの輝度計算を行い、反射レイなどの二次レイが発生する場合には二次レイを発生させる。ただし、レイの反射、屈折回数がユーザの定めた回数以上の場合には二次レイは発生させない。
   - 物体とレイが交差しなかった場合には、次空間へとレイを伝播させる。次空間の計算には 3DDDA [2] 法を用いる。次空間を担当する計算要素が自己でない場合には通信を用いてレイを次空間を担当する計算要素へ送り、レイを次空間へ伝播させる。定義空間を外れた場合にはレイは消滅する。
5. システム全体からレイがなくなったら計算終了。

二次レイは反射、屈折の履歴として反射係数を保持する。反射係数はレイ発生時には 1 に初期化されており、反射などを行う度に物体の反射係数が乗

ぜられていく。この反射係数によって複数の反射の影響を表現している。この並列レイトレーシング法が実行される様子を図 7 に示す。スクリーン上の複数のピクセルを通過するレイが物体の定義空間内で並列に追跡されている。

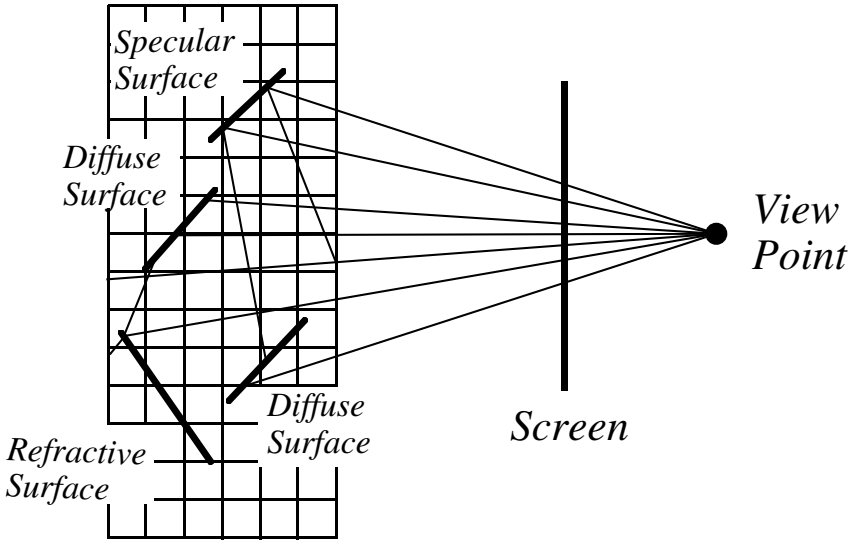

図 7: 並列レイトレーシング法

2.5 結言

本章では、マルチパスレンダリング法を並列化するためのオブジェクト空間分割型並列計算モデルを提案し、並列化したマルチパスレンダリング法のアルゴリズムを示した。

最初に大域照明問題の基礎式であるレンダリング方程式をとりあげ、現在写実的画像生成法としてとりあげられているレイトレーシング法とラジオシティ法はこのレンダリング方程式の近似であることを述べた。次にこれまで提案されているレイトレーシング法とラジオシティ法を概観し、これまでの並列処理手法とその欠点について指摘した。これまでの並列化手法はレイトレーシング法とラジオシティ法の固有の並列性に着目したものであり、空間の一貫性や、両者に共通する並列処理単位を無視していることに問題がある。本論文ではレイトレーシング法とラジオシティ法に共通する並列性としてレンダリング方程式に存在する物体と光の相互作用に着目し、それらの相互作用が空間中で局所的に発生することを指摘した。そしてこれらをふまえた並列計算モデルとしてオブジェクト空間分割型並列計算モデルを提案した。このオブジェクト空間分割型並列計算モデルはレンダリング方程式を並列化可能な並列計算モデルであり、マルチパスレンダリング法の構成要素であるレイトレーシング法とラジオシティ法を統一して並列化可能である。最後にオ

ブジェクト空間分割型並列計算モデルに基づいてマルチパスレンダリング法を並列化し、そのアルゴリズムを示した。

3　超並列画像生成システム $(M\pi)^2$

3.1　緒言

本章では 2 章で説明したオブジェクト空間分割型並列計算モデルに基づく並列マルチパスレンダリング法を効率良く実行するためのアーキテクチャ $(M\pi)^2$ について述べる。$(M\pi)^2$ の名は、マルチパスレンダリング法のための超並列画像生成システム (Massively Parallel Image system for Multi-Pass Image synthesis methods) に由来する。$(M\pi)^2$ は多数の計算要素 (Processing Element : PE) をトーラス状に接続した分散メモリ型の超並列計算機システムである。このシステムは、粒度に応じた並列処理を行なうために階層化された構成になっている。

最初に $(M\pi)^2$ の基本的なアーキテクチャについて述べ、階層的な並列処理がどのように行なわれるかについて述べる。次に $(M\pi)^2$ の並列レイトレーシング法においてフレームバッファアクセスへの競合による性能低下を避けるための階層化フレームバッファシステムを提案する。このシステムの挙動がキャッシュシステムに類似することから本システムをキャッシュドフレームバッファシステムと呼ぶ。最後に $(M\pi)^2$ の性能を十分発揮させるための負荷分散法について述べる。

3.2　オブジェクト空間分割型並列計算モデルに基づく階層型超並列計算機アーキテクチャ $(M\pi)^2$

本節ではオブジェクト空間分割型並列計算モデルに基づく並列マルチパスレンダリング法を効率良く実行するための超並列画像生成アーキテクチャとして $(M\pi)^2$ を提案する。

これまでの並列計算機の多くは、並列処理の粒度に応じた階層化についてほとんど考慮されていなかった。そのために大きな粒度から小さな粒度まで一括して同じ計算要素が処理しており、並列性を効率良く利用できないことが多かった。小さな粒度を処理するための計算要素に大きな粒度のタスクが投入された場合には並列性を十分に活用できず、大きな粒度を処理するための計算要素に小さな粒度のタスクが投入された場合には、処理要素間の通信のオーバヘッドが相対的に高くなり並列処理の効果が期待できない。$(M\pi)^2$ は画像生成における並列処理の粒度に着目し、それぞれの粒度に最適な並列処理形態を適用させているという特徴を有する。

マルチパスレンダリング法はオブジェクト空間分割型並列計算モデルにより、部分空間内の物体と光線の相互作用を処理単位として並列化される。この部分空間内には同時に複数の物体と光の相互作用が存在することが可能である。この複数の物体と光との間の各相互作用は独立した処理であり 1 つの光と 1 つの物体の相互作用を単位としてさらに並列化される。この 1 つの物

体と光の相互作用もさらに物体と光の交差判定、次空間探索処理、反射・屈折光の生成などの処理に分割され、並列処理可能である。このように並列処理自体が粒度の異なる階層的な並列処理 から成っているため、$(M\pi)^2$ もそれに合わせて階層化されているのである。具体的には最も粒度の粗い部分空間ごとの並列性には分散メモリ型の並列処理を適用し、部分空間内に複数存在するレイの処理に共有メモリ型の並列処理を適用し、さらに各レイの処理をいくつかに分けこれにパイプライン処理を適用する。

粒度の関係と処理内容を表 1 にまとめた。この表で階層レベルとは $(M\pi)^2$ におけるハードウェアの構成のレベルである。処理の内容はその階層での並列処理の単位となるものであり、たとえば、PE レベルは複数のレイプロセッシングブロックから構成されており、各レイプロセッシングブロックが 1 つの光と物体の相互作用を並列して実行する。本節では $(M\pi)^2$ におけるシステムレベルの計算要素を PE で表わすことにする。$(M\pi)^2$ とは独立した一般的な計算要素はこれまで通り計算要素と呼ぶ。

表 1: 階層的並列処理

階層レベル	階層の構成要素	処理内容	粒度	処理形態
システムレベル	PE	部分空間内の光と物体の相互作用 (光と物体は複数可)	粗	分散メモリ型のマルチプロセッサシステム
PE レベル	レイプロセッシングブロック	1 つの光と物体の相互作用	中	共有メモリ型のマルチプロセッサシステム
レイプロセッシングブロックレベル	ユニット	交差判定・次空間探索・輝度計算など	細	パイプライン処理

以下、各階層ごとのアーキテクチャについて説明する。

3.2.1 システムレベルの構成

2 次元の $(M\pi)^2$ のシステム構成の概要を図 8 に示す。$(M\pi)^2$ は分散メモリ型の超並列計算機であり、多数の PE がトーラス状に結合されている。PE の結合形態としては図 9 に示すように 1 〜 3 次元状に PE を配置した形態が考えられる。

$(M\pi)^2$ はホストコンピュータ、多数の PE、キャッシュドフレームバッファシステム (CFBS: Cached Frame Buffer System)、グローバルフレームバッファ(GFB: Global Frame Buffer) そしてディスプレイから構成されている。ホストコンピュータと PE はシステムバス (SB: System Bus) を介して物体情報や制御情報をやりとりする。光はレイの情報を持つレイパケットとして PE 間のネットワーク (IPN: Inter-PE Network) を伝播する。このレイパケッ

トは並列レイトレーシング法ではレイの情報を持ち、並列漸近的ラジオシティ法ではデルタラジオシティレイの情報を持っている。光の伝播計算の結果として生ずる輝度情報は輝度パケットとして PE からローカルフレームバッファバス (LFBB: Local Frame Buffer Bus) を介してキャッシュドフレームバッファシステムへ送信される。キャッシュドフレームバッファシステムはグローバルフレームバッファへのアクセスを緩和しつつグローバルフレームバッファへと計算結果をアキュムレートしていく。グローバルフレームバッファに蓄えられた画像が最終的にディスプレイに表示される。キャッシュドフレームバッファシステムについては第 3.4 節で再びとりあげる。

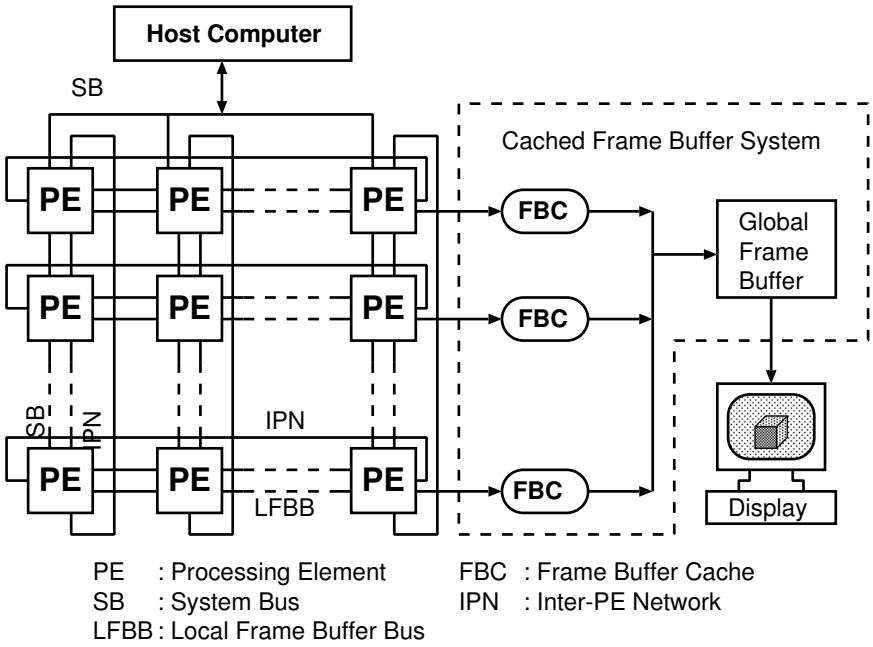

図 8: $(M\pi)^2$ のアーキテクチャ

3.2.2 PE レベルの構成

図 10 に PE レベルの構成を示す。PE はレイプロセッシングブロックを計算要素とする共有メモリ型の並列計算機である。PE の内部には初期レイパケットを発生するための初期レイ発生ユニット (FRGU: First Ray Generation Unit) とレイパケットのレイ情報と部分空間内の物体の相互作用を計算するレイプロセッシングブロック (RP block: Ray Processing Block)、レイパケットをレイプロセッシングブロックに分配するレイディストリビュータ (RD: Ray Distributor)、次空間へとレイパケットを伝播させるためのネットワークインターフェースユニット (NIU: Network Interface Unit)、PE の担当する

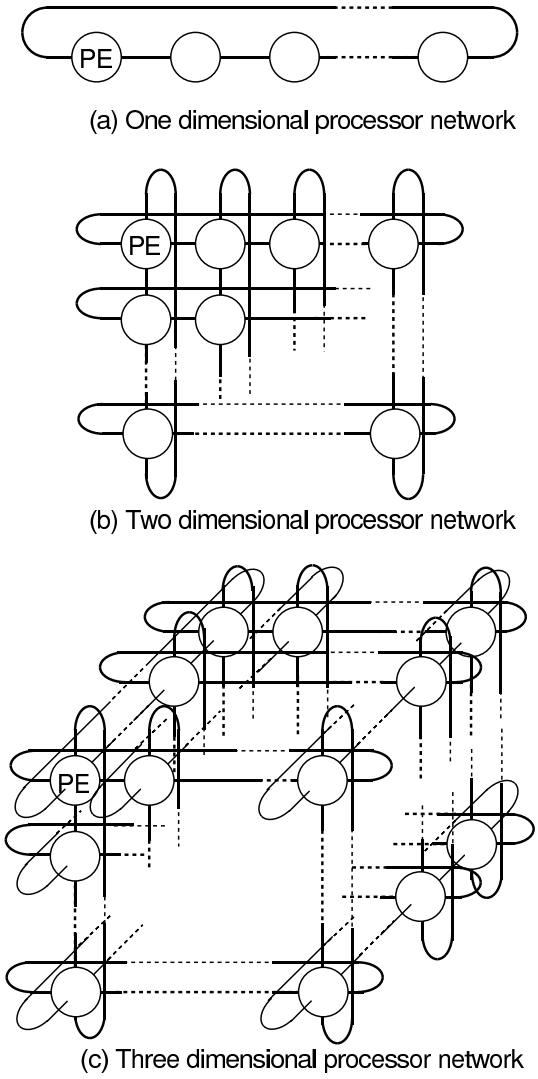

図 9: プロセッシングエレメントの結合形態

空間の物体情報を保持するローカルメモリが存在する。

PE はある空間中の光と物体の相互作用を計算する単位である。この空間の内部には同時に複数のレイパケットと複数の物体が存在する可能性がある。レイプロセッシングブロックは 1 つのレイパケットと空間中の複数の物体との相互作用を計算する単位である。PE 内部ではあたかもスーパスカラプロセッサが演算器に動的に演算を分配するように、レイディストリビュータが低負荷のレイプロセッシングブロックを選択しそのブロックへと動的なレイパケットの分配を行う。このようにして複数のレイプロセッシングブロックが並列動作する。

また、ネットワークインタフェースユニットと初期レイ発生ユニット、レイディストリビュータ、レイプロセッシングブロックはそれぞれ独立した処理単位であり、たとえば初期レイ発生ユニットからレイディストリビュータ、レイプロセッシングブロックをみるとパイプライン構成になっていることがわかる。各パイプラインステージの動作時間が一定でないためにこのパイプラインは段数分の速度向上率を得ることは難しいが、パイプライン処理によってある程度の速度向上が期待される。

3.2.3 レイプロセッシングブロックレベルの構成

図 11 にレイプロセッシングブロックレベルの構成を示す。レイプロセッシングブロックは 4 つのユニットから構成されている。すなわち、交差判定ユニット (ROIU : Ray-Object Intersection Calculation Unit)、次空間探索ユニット (DDU : Direction Decision Unit)、二次レイ発生ユニット (SRGU : Secondary Ray Generation Unit)、輝度計算ユニット (ICU : Intensity Calculation Unit) である。レイプロセッシングブロックは 1 つのレイパケットと空間内の複数の物体との相互作用を計算する計算要素である。この相互作用の計算は、交差判定、次空間探索、二次レイ発生、輝度計算という生産者— 消費者関係にある要素に分割可能であり、各レイパケットについてこれを逐次的に処理する。このような関係はパイプライン処理によって高速化することが可能であり、レイプロセッシングブロック内では実際にパイプライン処理が行なわれる。この各ユニットが $(M\pi)^2$ における基本的な処理単位である。

レイプロセッシングブロックにおけるパイプライン処理は、レイディストリビュータによってレイプロセッシングブロックにレイパケットが入力される所からはじまる。レイプロセッシングブロックに入力されたレイパケットは交差判定ユニットに入力され、交差判定ユニットが共有ローカルメモリ中の物体情報にアクセスしながら入力レイパケットと相互作用する物体を探索する。交差判定の結果、レイパケットの存在する空間内に交差物体が存在しないことが判明した場合にはそのレイパケットは次空間探索ユニットへと送信される。次空間探索ユニットはレイパケットが次にどの空間へと進むかを

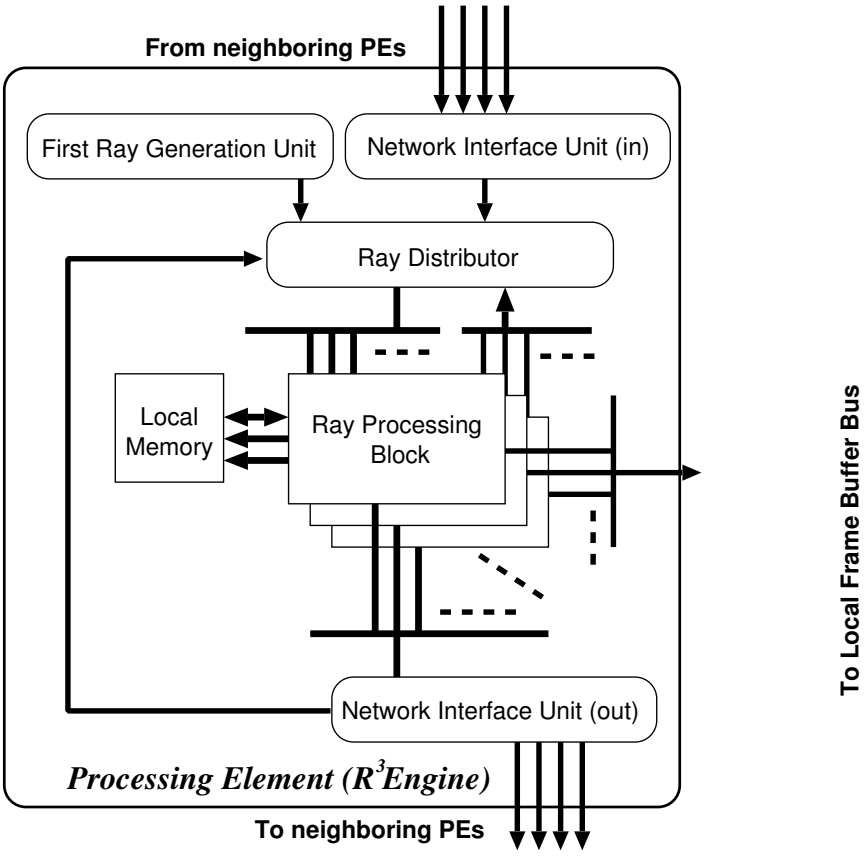

図 10: PE レベルの構成

3DDDA 法 [2] を用いて計算しその結果をレイパケットに記録してネットワークインタフェースユニットへと送信する。もしも空間中に入力されたレイパケットと交差する物体が存在するならば、そのレイパケットは輝度計算ユニットと二次レイ発生ユニットの 2 つのユニットへ同時に送信される。二次レイ発生ユニットは、鏡面反射、屈折、透過の現象によって面から新たに発生するレイパケットを生成し、レイディストリビュータに送信する。輝度計算ユニットはその名の通り物体表面の輝度を計算するユニットであるが、並列レイトレーシング法のステージと並列漸近的ラジオシティ法のステージでの動作が若干異なる。並列レイトレーシング法のステージでは、交差した物体表面の輝度を計算し、輝度の情報を輝度パケットとしてキャッシュドフレームバッファシステムへと送信する。この時、必要ならば同時にシェーディング処理等を行なう。並列漸近的ラジオシティ法のステージにおいてはラジオシティ値を計算し、パッチの未放射ラジオシティ値を増加させる。この時、パッチの未放射ラジオシティ値がユーザの定める放射値よりも大きい場合にはそのパッチの情報を初期レイ発生ユニットへと通達する。

このパイプライン処理は空間内の物体数やレイパケットの持つレイ/デルタラジオシティレイの情報によって次空間探索ユニットかまたは二次レイ発生・輝度計算ユニットのいずれかのみを動作させるためにパイプラインステージ分の高速化は望めないが、ある程度の速度向上が期待できる。

3.3 $(M\pi)^2$ における負荷分散法

3.3.1 $(M\pi)^2$ における負荷不均衡問題

本節では $(M\pi)^2$ における負荷分散法について静的な方法と動的な方法について述べる。

一般に並列計算機では、計算中に負荷の不均衡が発生するとその並列計算機の性能は大幅に減少する。たとえば、100 の計算要素が存在しても負荷の不均衡によって 1 つの計算要素しか働かなければ、その並列計算機の性能は最大値の 100 分の 1 しか発揮されないことになる。そのため、負荷分散法は並列計算機の能力を引き出すための重要な課題である。

本論文では、オブジェクト空間分割型並列計算モデルに基づいてマルチパスレンダリング法を並列化した。オブジェクト空間分割型並列計算モデルでは、各計算要素に部分空間を割り当て、その部分空間ごとの光と物体の相互作用を並列処理の単位としている。したがって、部分空間中に含まれる物体の多寡により、並列処理単位である部分空間の持つ負荷が左右される。一般に物体は定義空間中に一様に存在するとは限らないので、各部分空間の持つ負荷は異なり、これがシステム中の負荷の不均衡となって現われる。たとえ物体数が均一であっても定義空間中に存在する光線数が均一でない場合にはやはりシステム中の負荷に不均衡が発生する。

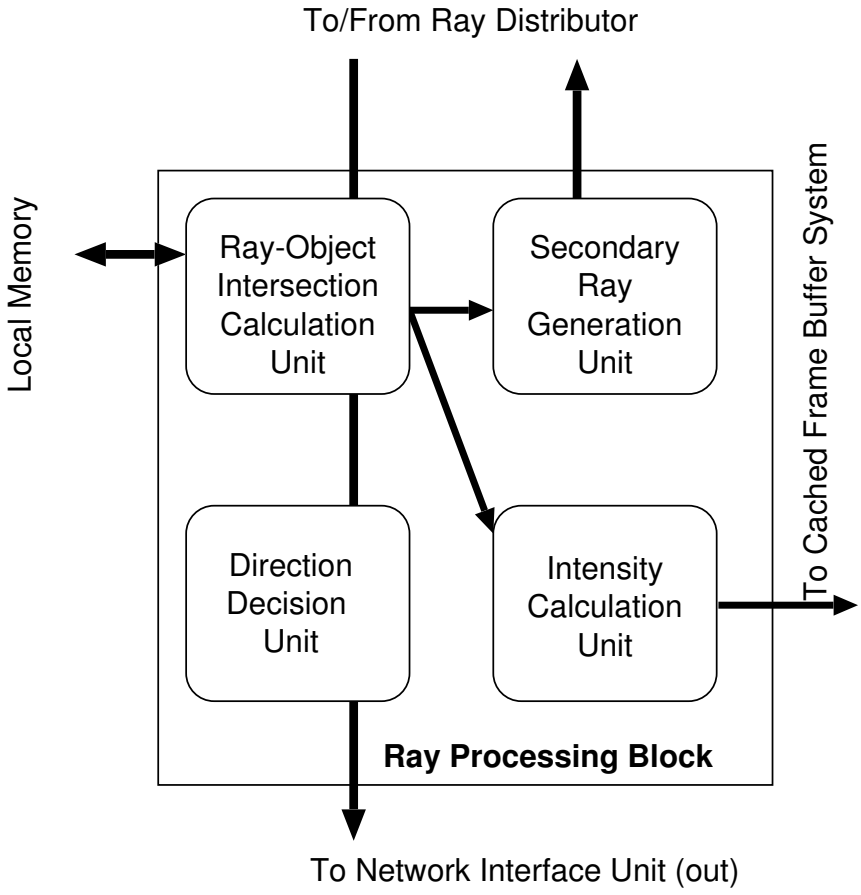

図 11: レイプロセッシングブロックレベルの構成

本論文ではこの負荷の不均衡問題を解決するために部分空間の PE への割り当てを工夫して負荷分散を図る方法と、PE 内部の複数のレイプロセッシングブロックを利用した負荷分散法を提案する。部分空間の PE への割り当てを工夫して負荷分散を図る方法では、計算中に PE 間で物体情報の割り当ての変更やレイの割付けの変更が起きないため、これを静的な負荷分散法と呼ぶ。レイプロセッシングブロックを利用する方法は、計算中にレイプロセッシングブロックの負荷に応じてレイの割付けを決定することで負荷の分散を図るため、これを動的負荷分散法と呼ぶ。以下、これら 2 つの負荷分散法について述べる。

3.3.2 静的負荷分散法

前節で述べたように、負荷分散の不均衡の第一の原因は、部分空間中の物体数が空間全体では均一でないために発生する。したがって、これを均一にすることができれば負荷の不均衡が解消される。

ここで、オブジェクト空間分割型並列計算モデルによる並列化では、部分空間の数は PE の数と必ずしも一致させる必要がないことを利用する。オブジェクト空間分割型並列計算モデルにおいては部分空間はあくまで並列処理の単位であり、1 つの PE が複数の部分空間を担当することが可能である。そこで、部分空間数が PE 数よりも大きくなるように定義空間を分割し、離れた部分空間を同一の PE に割付ける。物体は空間的局所性を持ち、いくつかの部分空間を占めることが多いので各 PE が離れた空間を受け持つことができれば、それぞれの PE には異なる物体が割付けられることが期待される。これを利用して負荷分散を行う。この方法はあらかじめ静的に物体の配置を決定することで負荷分散を図る方法なのでこれを静的負荷分散法と呼ぶ。

最も単純な空間の PE に対する割付法は、図 12 に示すようにある範囲の部分空間を各 PE が担当する方法である。これをブロック型の部分空間割付法と呼ぶ。この場合の部分空間と PE 番号の対応を式 (7) に示す。ここで、x, y, z は PE の x, y, z 方向の指標であり、この方法は空間分割をすすめたにもかかわらず、近くの空間を PE が保持する方法である。そのため、空間分割によって抽出した並列性をシステム全体に分散することにはならない。

$$\text{PE No}(x,y,z) \text{ of Subspace } (X,Y,Z) = (\lfloor \frac{X}{\lceil S_x/N_x \rceil} \rfloor, \lfloor \frac{Y}{\lceil S_y/N_y \rceil} \rfloor, \lfloor \frac{Y}{\lceil S_y/N_y \rceil} \rfloor) \tag{7}$$

ここで、

x, y, z	$\cdots$	PE の x, y, z 方向の指標
X, Y, Z	$\cdots$	部分空間の指標
S_x, S_y, S_z	$\cdots$	空間の x, y, z 方向の分割数
N_x, N_y, N_z	$\cdots$	x, y, z 方向の PE 数

である。よって、システムの総 PE 数は $N_x \times N_y \times N_z$ である。例えば 1 次元の $(M\pi)^2$ では、$N_y = N_z = 1$ であり、2 次元の $(M\pi)^2$ では、$N_z = 1$ となる。

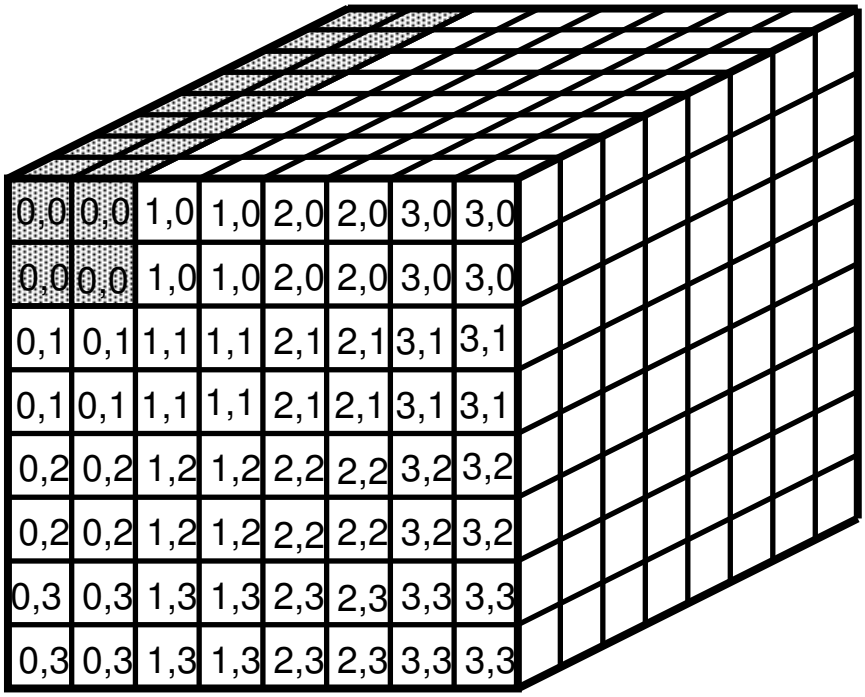

図 12: ブロック型の部分空間割付法 (PE 次元 = 2)

これを分散させて持たせる方法が、図 13 に示した分散型の部分空間割付法である。この場合の部分空間と PE 番号の対応を式 (8) に示す。オブジェクト空間分割型並列計算モデルに基づき、空間的に隣合った部分空間は隣合う PE に割付けなければならない。分散型割付法はこの要請を満たしながら部分空間を PE 全体に分散させることができる方法である。PE 番号がモジュロ演算で示されているように、物理的な PE の接続に端が発生しないよう、$(M\pi)^2$ はトーラス状のトポロジを持った構成となっている。

$$\text{PE No}(x,y,z) \text{ of Subspace } (X,Y,Z) \quad = \quad (X mod N_x, Y mod N_y, Z mod N_z) \tag{8}$$

分散型の割付法は、3 次元の $(M\pi)^2$ に対し、最も有効である。しかし、実際の並列計算機は次数が多いほどコストがかかるため、より低次の構成でも同程度の性能が発揮できると良い。そこで、より低次の $(M\pi)^2$ でも性能が向上するような割付法として、スキュー化分散型割付法 を提案する。図 14 にスキュー化分散型の部分空間割付法を示す。この場合の部分空間と PE 番号

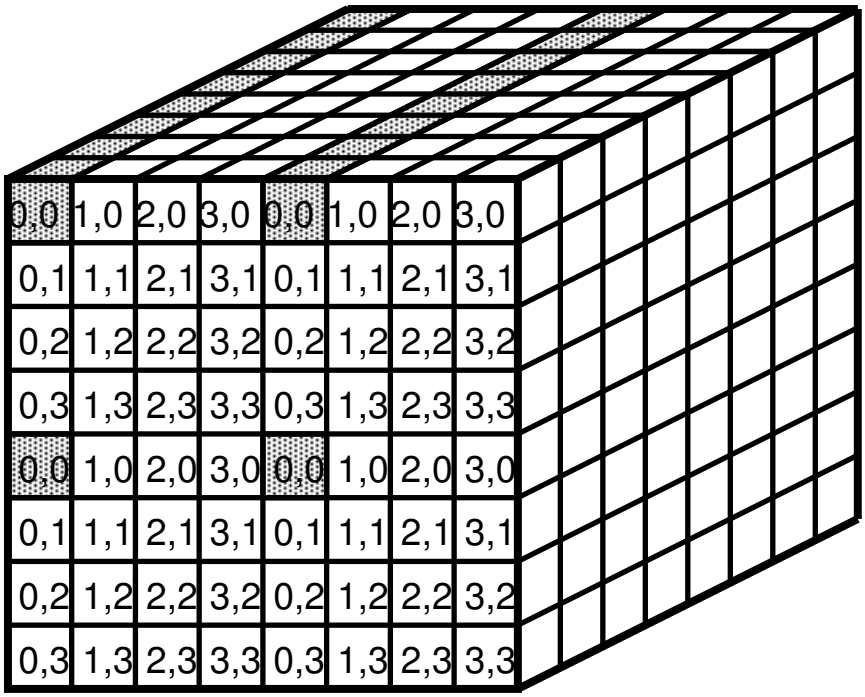

図 13: 分散型の部分空間割付法 (PE 次元 = 2)

の対応を式 (9), (10) に示す。これらの式では PE がそれぞれ x 方向に存在する場合と、x, y 平面に存在することを仮定している。スキュー化分散型の部分空間割付法もオブジェクト空間分割型並列計算モデルの要請に答え、隣合う部分空間を隣合う PE に割付けることが可能である。このスキュー化分散型の割付法は、物体が持つ軸方向のコヒーレンスを低次の $(M\pi)^2$ でも分散することが可能になるよう考案された。多くの現実の物体は重力の存在のために垂直軸方向にコヒーレンスを持っている。たとえば、壁は垂直に立ち、柱も木も然りである。これを分散型の割付法により 3 次元の構成を持つ $(M\pi)^2$ にマッピングした場合には、全ての次元方向のコヒーレンスを分散させることが可能であるが、より低次の $(M\pi)^2$ に分散型の割付法を適用した場合には、物体の持つコヒーレンスを分散できない方向が存在する。これは空間の次元数 (3 次元) を 1,2 次元の構成の $(M\pi)^2$ にマッピングしたために発生する。そのため、定義空間の軸方向にコヒーレンスを持った物体の存在によって同一の PE が同一の物体を含む部分空間を担当する傾向にある。スキュー化分散型割付法はこの PE の存在しない軸方向の担当 PE をずらしていくことで物体の持つコヒーレンスにより負荷の不均衡が発生することを避けることが可能である。

For the one-dimensional $(M\pi)^2$ system:

$$\text{PE No}(x) \text{ of Subspace } (X, Y, Z) \quad = \quad (X + Y + Z) \ mod \ N_x \qquad (9)$$

For the two-dimensional $(M\pi)^2$ system:

$$\text{PE No}(x, y) \text{ of Subspace } (X, Y, Z) \quad = \quad ((X + Z) \ mod \ N_x, Y \ mod \ N_y) \qquad (10)$$

3.3.3 動的負荷分散法

前節で示した静的な負荷分散法は計算前にあらかじめどのように各部分空間を PE にマッピングするかを決定するものであり、実際の計算中にはこれだけでは解決できない負荷の偏りが存在する可能性がある。そこで、計算中に動的に負荷の分散を行うことを考える。

負荷の不均衡は各 PE に割付けられる物体数が不均一であることが第一の原因と考えられるが、これは静的負荷分散法によってある程度緩和される。ここで PE への物体数の不均一が残った場合に、物体を移動することによってこの不均一を解消し、性能向上を図ることは困難である。なぜなら、計算時の物体移動はシステム中のにおける物体の探索、物体情報の通信による移動といった大きなオーバヘッドを持つ可能性があり、このために逆にシステム性能が低下する可能性があるためである。さらにこれは空間中の物体の位置が保たれるというオブジェクト空間分割型並列計算モデルに反した方法で

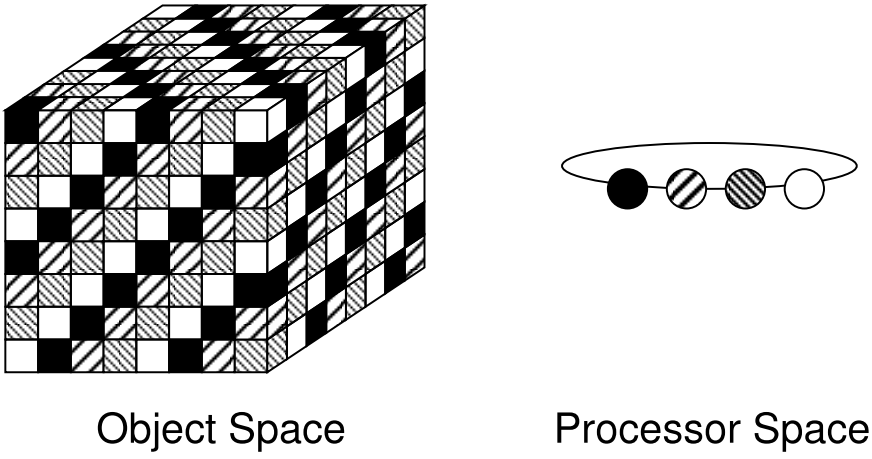

図 14-a: 1 次元の $(M\pi)^2$ のスキュー化分散型割付け法

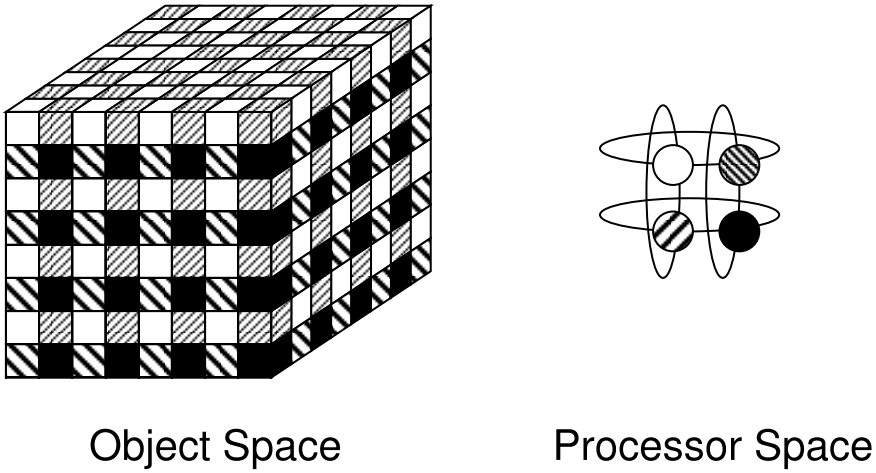

図 14-b: 2 次元の $(M\pi)^2$ のスキュー化分散型割付け法

図 14: スキュー化分散型の部分空間割付法

ある。したがって、負荷を均衡させるための物体情報の移動は静的にも動的にも行うことができない。

ここで、静的負荷分散法によって、各 PE が複数の空間を担当するようになったことを考える。オブジェクト空間分割型並列計算モデルの本来の考えでは一つの物体に一つの計算要素を割付けるというものであったが、扱う物体数が現実的でないほど多数であるために物体でなく空間に計算要素を割付けたのであった。そのため、1 つの部分空間内にそれぞれ複数のレイと物体が存在する可能性があり、PE 内に複数のレイプロセッシングブロックを設け、複数のレイを並列に処理するという考えを 3.2.2 で述べた。

しかし、PE 内部でレイプロセッシングブロックを増加させるのではなく、そのかわりにさらに PE を増加すべきではないかという考えがでてくるであろう。ところが同数のレイプロセッシングブロックを利用するのであれば、PE 数のみを増加させてレイプロセッシングブロック数を抑えるよりもある程度まではレイプロセッシングブロック数を増加させる方が効果的であると予想される。図 15 をみてみよう。この図では PE 数が 2 に対し、空間数が 4 あり、静的負荷分散によって各 PE が 2 つずつ空間を担当している。この図では、静的負荷分散法によって定義空間内に物体の偏りが存在しているにもかかわらず負荷が各 PE へと分散している。このシステムの性能を向上させようと PE 数を増加させると、図 16 のように負荷の不均衡を各 PE 間で分散させることができなくなってしまう。そこで、PE 数を増加させるのではなく、PE 内部のレイプロセッシングブロックを増加させてみたのが図 17 である。図 16 と図 17 は同数のレイプロセッシングブロックを持っている。そのためシステムとしては同程度の潜在的能力を持っていると考えられる。しかし、図 16 の場合には何も物体が存在しない部分空間を担当する PE と、物体の存在する部分空間を担当する PE が存在するために負荷の不均衡が発生しているが、図 17 では各 PE が静的負荷分散法の効果をそのまま利用して担当する物体数を均衡させている。このように、同一のレイプロセッシングブロック数のシステムを考えた場合には、PE 数を抑え、レイプロセッシングブロックを増加させて動的なレイ割り当てを行なうことで負荷分散が行なわれることがわかる。この負荷分散法は静的負荷分散の効果があればより効果的になる。しかしながら各 PE 内部のレイプロセッシングブロック間では各レイプロセッシングブロックが複数の空間内の物体情報にアクセスする必要があるため、レイプロセッシングブロック間で物体情報の共有を行わなければならない。そのため、レイプロセッシングブロック数を増大させるには限界がある。したがって、レイプロセッシングブロック数が一定の時を考えると、PE 数と PE 内部のレイプロセッシングブロック数の間にはトレードオフが存在する。

これがレイプロセッシングブロックを用いた動的負荷分散法である。この動的負荷分散法は物体の移動やレイの移動を PE を越えて行なう必要がない

ために、動的な負荷分散に伴なう PE 間の通信というオーバヘッドが発生しない方法であることも特徴的である。これは並列処理にあたって物理的に意味のある計算モデルを用いたことと、そこから導かれる粒度を考慮した階層型の並列計算機 $(M\pi)^2$ であるからこそ可能であるといえる。さまざまな粒度を同一に扱うこれまでの並列計算機では、細い粒度を並列化しても局所的に扱うことができず、大きなオーバヘッドを強いられると思われる。

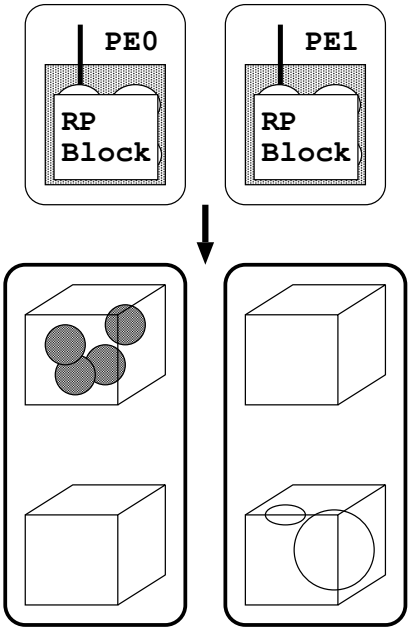

図 15: 動的負荷分散法 (1) 静的負荷分散が成功している例

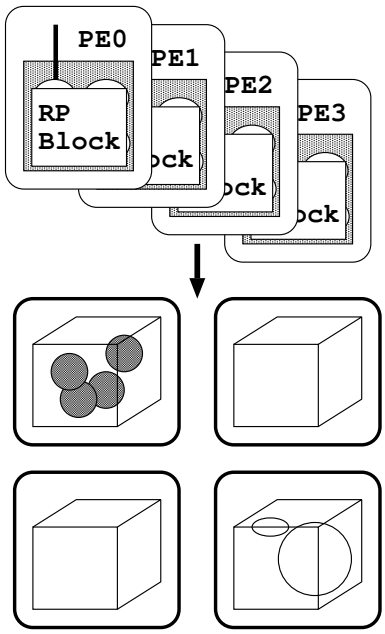

図 16: 動的負荷分散法 (2) PE 数増加のため静的負荷分散に失敗した例

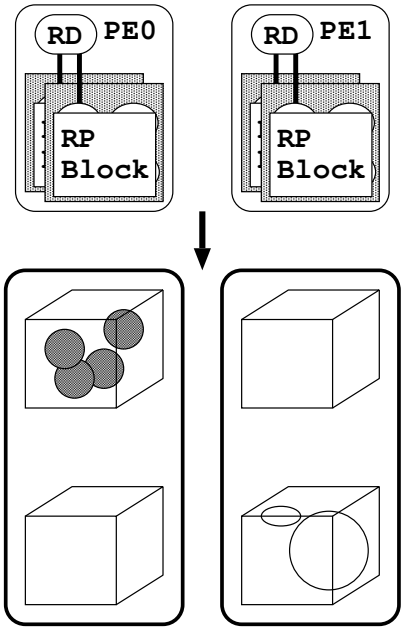

図 17: 動的負荷分散法 (3) RD による動的負荷分散法

3.4 キャッシュドフレームバッファシステム

3.4.1 $(M\pi)^2$ におけるフレームバッファアクセス問題

オブジェクト空間分割型並列計算モデルに従った並列マルチパスレンダリング法では、画像生成のための物体情報を共有する必要がない。よって、物体情報を共有したために発生するアクセス競合や、物体情報のコピーによるメモリ量の増大といった問題は解決され、スケーラビリティのある並列計算機の構築が可能になる。ところが、画像を実際に生成するためには各計算要素で得た輝度情報を統合する処理が必要である。通常、画像情報はフレームバッファと呼ばれるメモリ上に蓄えられる。本システムでは、画像生成に際し、各計算要素で発生した輝度情報を輝度情報パケットとしてフレームバッファへと送信する。輝度情報パケットは光と物体との一回の相互作用によって生じた輝度情報と、その輝度情報が関与するスクリーン上の画素の位置を保持している。一つの画素の輝度は複数の光と物体の相互作用の結果が関与する可能性があるため、最終的に画素毎に輝度情報をアキュムレートしてフレームバッファ上に保持する必要がある。$(M\pi)^2$ のような並列計算機においてはこのフレームバッファは共有資源であり、PE 数、あるいはレイプロセッシングブロック数の増大に伴なってフレームバッファ上でアクセス競合が発生する可能性がある。また、近年プロセッサ速度とメモリ速度の差が増大する傾向があり、メモリであるフレームバッファが PE の速度に十分対応できなくなる可能性があり、この場合のアクセス競合は深酷な問題となる可能性がある。

近年の共有メモリ型の計算機システムにおいては、共有メモリへのアクセス競合を緩和するために、共有メモリ自体を分散させる方法や、キャッシュメモリを用いる方法などがある [27]。フレームバッファを各 PE へ分散させる場合には、フレームバッファをどのように分散させ、統合するかという問題が存在する。負荷の分散のためには、各 PE へフレームバッファを均一に割り当てることが望ましい。しかし、本論文の並列計算方式では視点などのパラメータが変化することにより各 PE の担当するピクセル数や、処理要素の担当するフレームバッファ内のピクセル位置が変化する。そのためフレームバッファの部分集合を各 PE 上に固定したハードウェアとして割り当てることは困難である。一方全ての計算要素が最終的に利用されるフレームバッファと同じ大きさのフレームバッファのコピーを持つ方式も考えられる。この方式であればフレームバッファの分散の問題はないが、コストの面からの不利がある。将来コストの面が解決された場合にもシステム全体に不規則に分散するフレームバッファ上の輝度情報を統合、アキュムレートする際にやはりアクセス集中の問題が発生する。

一方キャッシュメモリを用いる方法はどうであろうか。キャッシュメモリは、プロセッサとメモリシステムの速度差を埋める重要な技術であり、分散

型の並列計算機においては、分散キャッシュメモリが参照の局所性を利用して共有メモリへのアクセス競合を緩和することが可能である [27]。 そこで本論文ではフレームバッファの設計に際し、汎用のマルチプロセッサシステムの階層構造を利用したキャッシュドフレームバッファシステムを提案する。これまでにフレームバッファに対するキャッシュメカニズムがいくつか提案されている [20, 23]。しかしながら、基本的にそれらは局所照明モデルに基づく z-buffer アルゴリズムを用いた高速ポリゴン描写のために設計されている。そのために大域照明モデルに基づくオブジェクト空間分割型並列計算への適用が困難である。マルチパスレンダリング法の第 2 ステージであるレイトレーシング法の計算においては、隣合うピクセルから発っせられるレイは同一の物体に交差しやすい傾向を持っている。これは PE が近くのレイを担当する傾向があることを示しており、フレームバッファへのアクセスに多くの局所性を有していることを意味している。これらをふまえ、我々はスクリーンの画素を規則的なキャッシュブロックに分解し、いくつかの PE に対してフレームバッファキャッシュを用意した。図 18 はキャッシュドフレームバッファシステムの概要を示している。キャッシュドフレームバッファシステムは木構造の構成を持っており、図 18 は 2-ary 木 (binary-tree) の構成のキャッシュドフレームバッファシステムを示している。一般にキャッシュドフレームバッファシステムでは m 分木の構成を考えることができる。図 8 もまた $(M\pi)^2$ におけるキャッシュドフレームバッファシステムを示している。ここでは、いくつかの PE がローカルフレームバッファバスを介してフレームバッファキャッシュ(FBC: Frame Buffer Cache) に接続されており、フレームバッファキャッシュがフレームバッファを共有している。ここではフレームバッファキャッシュとフレームバッファを明確に区別するため、以後、フレームバッファをグローバルフレームバッファと呼ぶ。また、図 8 では計算要素のグループを 1 つのラインとしているが、空間的局所性を利用するために空間的にまとまったキューブ状のグループを用いることも考えられる。輝度情報はグローバルフレームバッファに送信される途中に存在するいくつかのフレームバッファキャッシュによりキャッシュ、アキュムレートされていくことでグローバルフレームバッファへの直接アクセスを緩和する。

3.4.2 キャッシュドフレームバッファシステムの制御法

次にどのように輝度情報をキャッシュするかについて説明する。図 19 に示すようにフレームバッファはいくつかのブロックに分割されている。このブロックは複数の画素から成っている。1 つの計算要素はスクリーンの一部分をブロック状に担当する傾向があるので、キャッシュの管理単位を正方形か長方形のブロックとしている。各計算要素は、輝度を求め、それを輝度情報パケットとしてローカルフレームバッファバスを介してフレームバッファキャッ

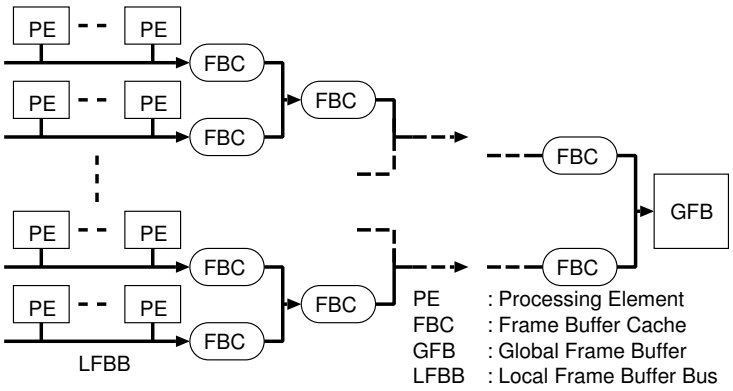

図 18: m 分木のキャッシュドフレームバッファシステム (m = 2)

シュへと送信する。複数の計算要素が同時にパケットを送信した場合には調停が行なわれる。輝度情報パケットには輝度に加えてスクリーン上のどのピクセルの輝度かという情報が含まれている。輝度情報パケットがフレームバッファキャッシュに到達すると、以下のいずれかの操作が行われる。

- Cache miss
  輝度情報パケットの示すピクセルを含むブロックがフレームバッファキャッシュ中に存在しない場合。
  - フレームバッファキャッシュ上に割り当てられていない領域があればそれを割り当て、輝度情報をキャッシュする。
  - フレームバッファキャッシュが全て利用されている場合には、LRU 方式によって 1 つのブロックをグローバルフレームバッファに書き出し、その領域を解放する。のちに空いた領域を新たに割り当て、輝度情報をキャッシュする。
- Cache hit
  輝度情報パケットの示すピクセルを含むブロックがフレームバッファキャッシュ中に存在する場合には、送信されてきた輝度情報をフレームバッファキャッシュ中の輝度情報に加算する。

輝度はグローバルフレームバッファ上に加算されていくだけであるため、各計算要素とフレームバッファキャッシュはグローバルフレームバッファの内容を参照する必要がない。したがって、キャッシュ間のコヒーレンスを保つ機構は必要ない．計算終了時にはキャッシュ上の有効な輝度値を全てグローバルフレームバッファにアキュムレートする。

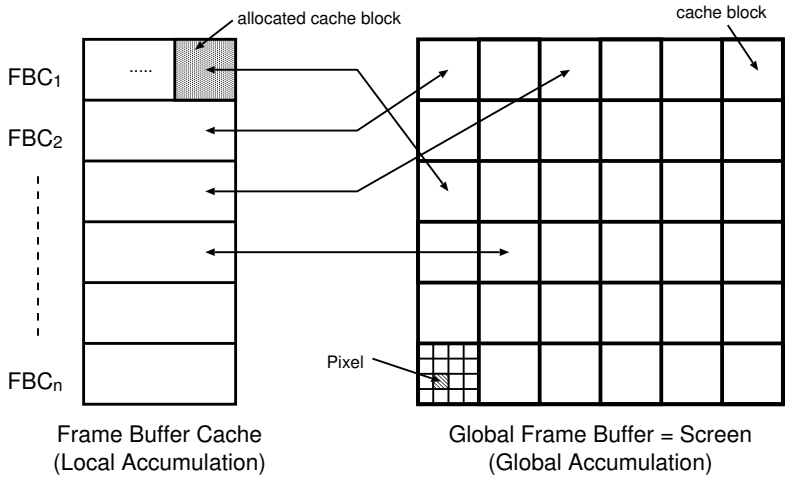

図 19: キャッシュドフレームバッファシステム

3.4.3 キャッシュドフレームバッファシステムの理論解析

本節では、前節のキャッシュドフレームバッファシステムの性能を解析的に評価した。 図 20 は、キャッシュドフレームバッファシステムの性能を待ち行列理論 [14] により解析的に性能するためのモデルを示している。各計算要素 (PE) は通信ユニット (C_1) を介しフレームバッファキャッシュ (FBC_1) に接続されており、各 i 段目の (FBC_i) は $i+1$ 段目の (C_{i+1}) へと接続されている。また、各 i 段目の (C_i) は i 段目の (FBC_i) へと接続されている。各計算要素はグローバルフレームバッファ への輝度情報を平均発生率 λ_{PE} のポアソン分布で発生すると仮定する。通信ユニットの処理状況は待ち行列理論の M/D/1 を仮定し、フレームバッファキャッシュは M/M/1 で処理を行うと仮定する。M/M/1 の待ち行列にポアソン分布間隔で入力があった場合には、Burke の定理 [14] により退去に対してもポアソン分布間隔が保証されているが、M/D/1 の場合の入力間隔がポアソン分布であっても退去間隔はポアソン分布になるとは限らない。しかし、計算要素の輝度発生間隔が短い場合には計算ユニットからの退去間隔をポアソン分布とみなすことが可能であるので、ここでは M/D/1 の退去の間隔をポアソン分布と仮定する。この正当性については次節でシミュレーションにより検討する。フレームバッファキャッシュはミスした場合にのみ次段へ通信するため、ミス率を基本とした。ここで解析に用いた記号は表 2 にまとめた。ここではグローバルフレームバッファの挙動はフレームバッファキャッシュと同等とした。ただし、グローバルフレームバッファ上では必ず輝度情報はヒットする。各 FBC_i と C_i の内部は図 21 のようにモデル化されている。

通信ユニットでは、PE の輝度発生率が初段の入力となる。途中フレーム

.

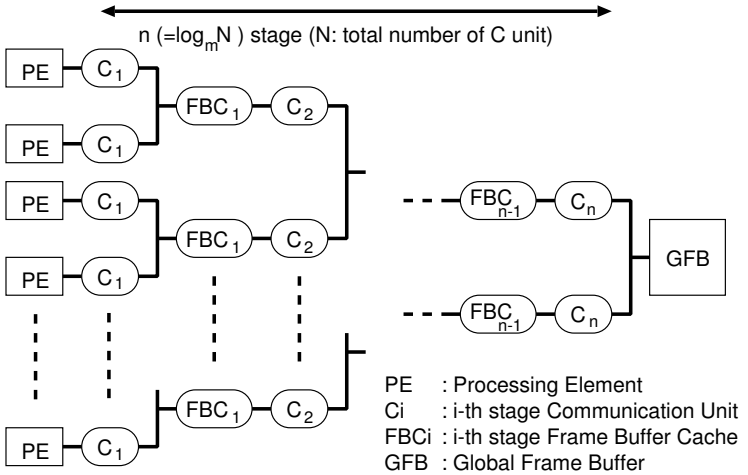

図 20: キャッシュドフレームバッファシステムの理論解析モデル (m-ary (m = 2) $\log_m$n stage)

バッファキャッシュがヒットすることによってこの平均到着率が減少するため、i 段への平均到着率 λ_{C_i} は、

$$\lambda_{C_i} = m^{i-1}P_M^{i-1}\lambda_{PE} \tag{11}$$

となる。また、平均サービス間隔 μ_{C_i}、利用率 ρ_{C_i} は、

$$\mu_{C_i} = \frac{1}{T_{C_i}} \tag{12}$$

$$\rho_{C_i} = m^{i-1}P_M^{i-1}\lambda_{PE}T_{C_i} \tag{13}$$

となる。FBC の i 段への平均到着率 λ_{FBC_i}、平均サービス間隔 μ_{FBC_i}、利用率 ρ_{FBC_i} については、それぞれ次のようになる。

$$\lambda_{FBC_i} = m^{i}P_M^{i-1}\lambda_{PE} \tag{14}$$

$$\mu_{FBC_i} = \frac{1}{T_{mp}} \tag{15}$$

$$\rho_{FBC_i} = m^{i}P_M^{i-1}\lambda_{PE}T_{mp} \tag{16}$$

ただし、T_{mp} はフレームバッファキャッシュの平均処理時間で、

$$T_{mp} = P_M T_M + (1 - P_M)T_H \tag{17}$$

である。

M/M/1 と M/D/1 に対する Pollaczek-Khinchin の平均値公式 [14] より、これらのキューでの平均待ち時間は以下のようになる。

$$W_{C_i} = \frac{\rho_{C_i}}{2(1-\rho_{C_i})}T_{C_i}$$

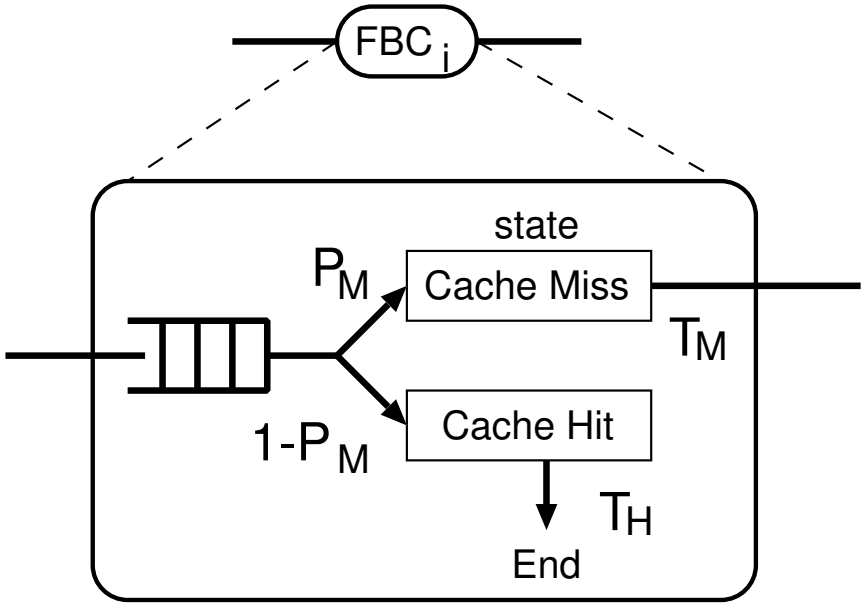

図 21-a: フレームバッファキャッシュユニットの動作

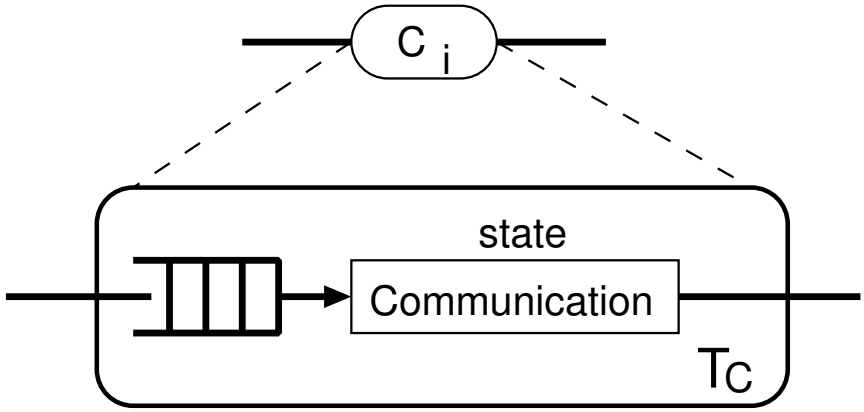

図 21-b: 通信ユニットの動作

図 21: フレームバッファキャッシュ要素と通信要素の解析モデル

$$= \frac{m^{i-1}P_M^{i-1}\lambda_{PE}{T_{C_i}}^2}{2(1-m^{i-1}P_M^{i-1}\lambda_{PE}T_{C_i})} \quad (18)$$

$$W_{FBC_i} = \frac{\rho_{FBC_i}}{(1-\rho_{FBC_i})}T_{mp}$$

$$= \frac{m^iP_M^{i-1}\lambda_{PE}{T_{mp}}^2}{1-m^iP_M^{i-1}\lambda_{PE}T_{mp}} \quad (19)$$

i 段までのレイテンシは、$i-1$ 段までのフレームバッファキャッシュがミスしており、i 段でヒットするものとして求められるので、

$$L_i = \sum_{k=1}^{i}(W_{C_k} + W_{FBC_k}) + iT_{C_i} + (i-1)T_M + T_H \quad (20)$$

となる。各 L_i の発生確率を P_i としてその期待値をフレームバッファキャッシュシステムの木の段数 n 段までとると

$$L_{mean} = \sum_{i=1}^{n} P_iL_i$$

$$= \sum_{i=1}^{n} P_M^{i-1}L_i \quad (21)$$

となる。

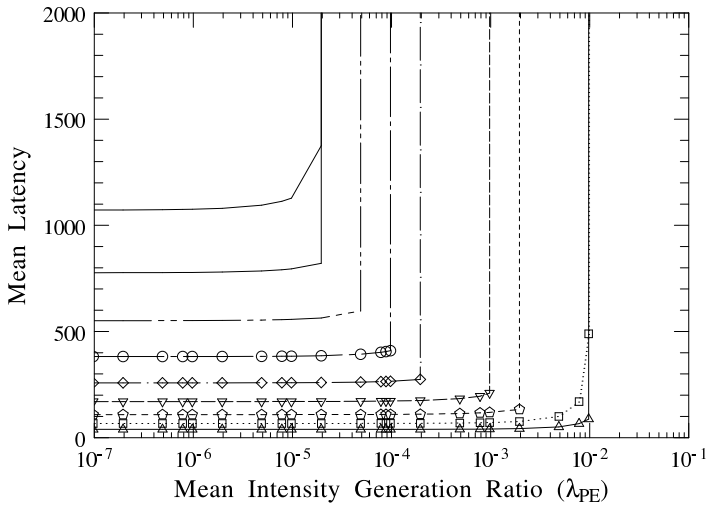

図 22: CFBS におけるキャッシュのミス率と平均レイテンシの関係・理論解析 (4-ary 5-stage)

この輝度がキャッシュドフレームバッファシステムへ入力されてからグローバルフレームバッファへと到達するまでの平均レイテンシ L_{mean} をキャッシュドフレームバッファシステムの性能評価の指標として用いる。ここでは 1024

表 2: CFBS の理論解析における記号の意味

PE	Processing Element
FBC	Frame Buffer Cache
C	Communication unit
$\lambda_{PE}(=\lambda_{C_1})$	mean arrival rate at the first C from PE
λ_{FBC_i}	mean arrival rate at the i-th FBC
$\lambda_{C_i}(\mathrm{i} \neq 1)$	mean arrival rate at the i-th C
μ_{FBC_i}	mean service rate at the i-th FBC
μ_{C_i}	mean service rate at the i-th C
ρ_{FBC_i}	utilization of the i-th FBC
ρ_{C_i}	utilization of the i-th C
P_M	probability of FBC Miss
$P_H(=1-P_M)$	probability of FBC Hit
T_M	processing time of an FBC when FBC Misses
T_H	processing time of an FBC when FBC Hits
T_{mp}	mean processing Time of an FBC
T_{C_i}	processing time of the i-th C
m	m-ary tree of CFBS
N	total number of C units
n	number of stages of the m-ary tree ($=\log_m \mathrm{N}$)
W_{C_i}	queue waiting time of the i-th C
W_{FBC_i}	queue waiting time of the i-th FBC
L_i	total latency of cache access at the i-th stages of the CFBS
L_{mean}	mean latency of the CFBS

表 3: CFBS の理論解析のためのパラメータ

parameter	time (normalized by 1 byte sending time as 1 clock)
T_{C_1}	8
$T_{C_i}(i \neq 1)$	64
T_H	12
T_M	71 (4x4 pixel/cache block)

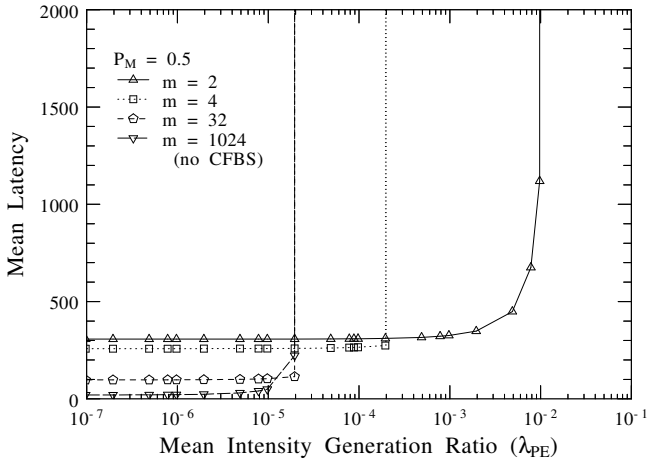

図 23: CFBS の構成と平均レイテンシの関係・理論解析 (ミス率 = 0.5)

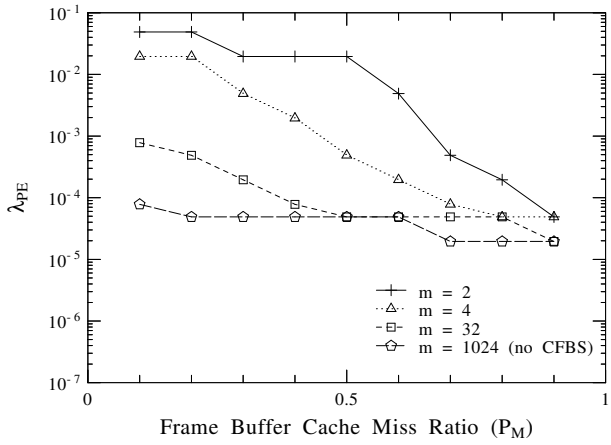

図 24: CFBS の性能飽和点とフレームバッファキャッシュのミス率の関係・理論解析

PE がキャッシュドフレームバッファシステムに接続されていると仮定した。図 22 は 4-ary 5-stage のキャッシュドフレームバッファシステムの理論解析による挙動を示している。ここでは表 3 に示すパラメータを用いた。このパラメータは、1 byte を送信する時間で正規化した時間を基準にしている。1 つの輝度情報パケットは、スクリーン上のピクセル位置 x, y として 4bytes、RGB の値をそれぞれ 1 byte づつ、計 7 bytes の情報を保持すると仮定した。ここでは、キャッシュブロックの大きさを 4×4 pixel に仮定しており、最初の段は輝度が 1 パケットづつ送信され、次段からはキャッシュブロックの置き換え (block size $7 \times 4 \times 4 = 112$ bytes) による通信が発生するとしている。輝度の平均発生率が増大するにつれ、ある地点で急激に平均レイテンシが飽和してしまう様子がわかる。次に、ミス率を一定にし、キャッシュドフレームバッファシステムの構成を変化させた場合を見てみる。図 23 はミス率が 0.5 の時のキャッシュドフレームバッファシステムの平均輝度発生率と平均滞在時間の関係をキャッシュドフレームバッファシステムの構成を変化させて示している。キャッシュドフレームバッファシステムの木の高さが高い、つまり m 分木の m が小さいほど高い平均輝度発生率に対して性能が飽和しにくいことがわかる。しかし、木が高い場合にはキャッシュドフレームバッファシステムのコストが増大し、かつ平均滞在時間が大きくなるというトレードオフが存在する。ここで $m = 1024$ の場合はキャッシュドフレームバッファシステムが存在せず、1024 PE がそのままグローバルフレームバッファに接続されていることを示している。キャッシュドフレームバッファシステムなしの場合 ($m = 1024$) に比べ、2-ary 10-stage のキャッシュドフレームバッファ

システムは数百倍の負荷に耐えることがわかる。

以上からキャッシュドフレームバッファシステムを構成することでグローバルフレームバッファへのアクセスが緩和されることがわかる。また、キャッシュドフレームバッファシステムを構成する場合には m が小さいものを構成すると性能が良いことがわかる。しかし、m が小さい場合にはキャッシュドフレームバッファシステムのためのコストの増大と平均滞在時間の増加があり、このトレードオフを考慮に入れる必要がある

3.5 結言

本章では、オブジェクト空間分割型並列計算モデルに基づく並列マルチパスレンダリング法を高速に実行するための並列計算機アーキテクチャ $(M\pi)^2$ を提案した。

まず一般に並列処理はフラットな構造を持っているのではなく、粒度によって分類されるいくつかの階層を持つことを述べ、この階層を考慮した並列処理が効果的であることを述べた。オブジェクト空間分割型並列計算モデルに基づく並列処理では、その階層は部分空間を単位とする処理、部分空間内の 1 つの光と物体の相互作用の処理、そして光と物体の相互作用計算をパイプライン的に分割した各処理へと階層化される。これら粒度の異なる処理を効率良く扱うために $(M\pi)^2$ は階層的な構成を持った並列計算機となっている。その階層は分散メモリ型の超並列計算機であるシステムレベル、共有メモリ型の並列計算機である PE レベル、そしてパイプライン処理を行なうレイプロセッシングブロックレベルに分けられる。

また、並列処理において避けて通ることのできない負荷分散法について静的な方法と動的な方法について考察した。静的な方法は物体の定義空間を PE 数に対してより細く分割し、分散型の割付法によって各 PE に割付けられる物体数を均一に保つ方法である。この方法は低次の $(M\pi)^2$ においては不十分な面が存在するため、さらに低次のシステムにおいて有効なスキュー化分散割付法を提案した。静的な負荷分散法だけでは解決できない負荷の不均衡を解決するため、静的負荷分散法と組み合わせると効果的な動的な負荷分散法についても提案した。

最後に、$(M\pi)^2$ のような超並列画像生成システムでは、画像生成の際のフレームバッファへのアクセス競合による性能低下が考えられるため、フレームバッファへのアクセス競合を緩和するキャッシュドフレームバッファシステムを提案し、その性能についての理論解析を行なった。

4　性能評価

4.1　緒言

本章では、3 章で提案した $(M\pi)^2$ の性能評価を行なう。性能評価においては実際にシステムを作成することが望ましいが、莫大な開発期間を要し、さらに構築後には設計パラメータの変更が容易でないという問題がある。そこで本論文では、ソフトウェアシミュレーションにより $(M\pi)^2$ の性能評価を行った。

はじめにシミュレーションのモデルを示し、次にシミュレーションの結果を示し、最後に考察を行う。

4.2　$(M\pi)^2$ のシミュレーションによる性能評価

4.2.1　シミュレーションモデル

本論文では、$(M\pi)^2$ の各ユニットは Sparc Chip(40MHz) と同等の性能を持つと仮定した。そして、それぞれのユニットの処理を C++ 言語により記述し、その実行時間を測定することによって各ユニットの処理時間を求めた。実行は、Sparc Chip(40MHz) を CPU に持つ Sun Sparc station IPX (Sun OS 4.1.3) 上で行なった。測定には getrusage システムコールを用い、各処理を 10000 回呼び出すのに要した時間の平均をユニットの動作時間として採用した。C++ コンパイラとしては FSF の g++ Ver. 2.4.5 を使用した。また、各 PE 間の通信路は、40MBytes/sec のバンド幅を持つと仮定した。シミュレーションにあたっては、1 byte のデータをこの通信路で送信する時間 25 nsec を 1 シミュレーションクロックとした。これらに基いて求めた各ユニットのシミュレーションクロック数を表 4 に示す。またシミュレーションに用いたシーンの画像を図 25 に、このシーンのパラメータを表 6 に示す。シミュレーションには実際に並列化マルチパスレンダリング法による画像を生成しつつ $(M\pi)^2$ の動作をシミュレートする離散時間実行駆動型シミュレーション方式を用いた。離散時間実行駆動型シミュレーション方式では、実際に画像を生成しながら、並列動作をシミュレーションする。

4.2.2　シミュレーション結果

オブジェクト空間分割型並列計算モデルに従った並列化マルチパスレンダリング法を $(M\pi)^2$ で動作させた場合の性能評価をシミュレーションにより行なった。最初に並列ラジオシティ法における性能評価について示し、次に並列レイトレーシング法についての性能評価の結果を示す。

表 4: 各ユニットの動作時間

	Simulation clocks
Generation time of an initial ray with Hemi-cube initialization	1053×10^3
Generation time of an initial ray	4379
Ray-object intersection calculation time	$1028 \sim 2446$
Intensity calculation time	349
Next subspace calculation time by 3DDDA with initializing	3151
Next subspace calculation time by 3DDDA	1106
Secondary ray generation time	1818
Ray-packet generation time	164
Transmission time of a packet	164

表 5: キャッシュドフレームバッファシステムのシミュレーションパラメータ

Accessing time to an FBC (Hit)	12 clocks
Accessing time to an FBC (Miss)	7+4×(# of dirty pixels)
Accessing time to the GFB without FBCs	14 clocks
Each FBC size	1/128 of Total GFB size
Cache associativity	2
Cache block size	4×4 pixels

表 6: シーンパラメータ

Number of total patches	18859
Number of subspaces	32768
Screen size	512×512
Number of pixels on a hemi-cube	1200
Maximum number of reflections of each ray	3

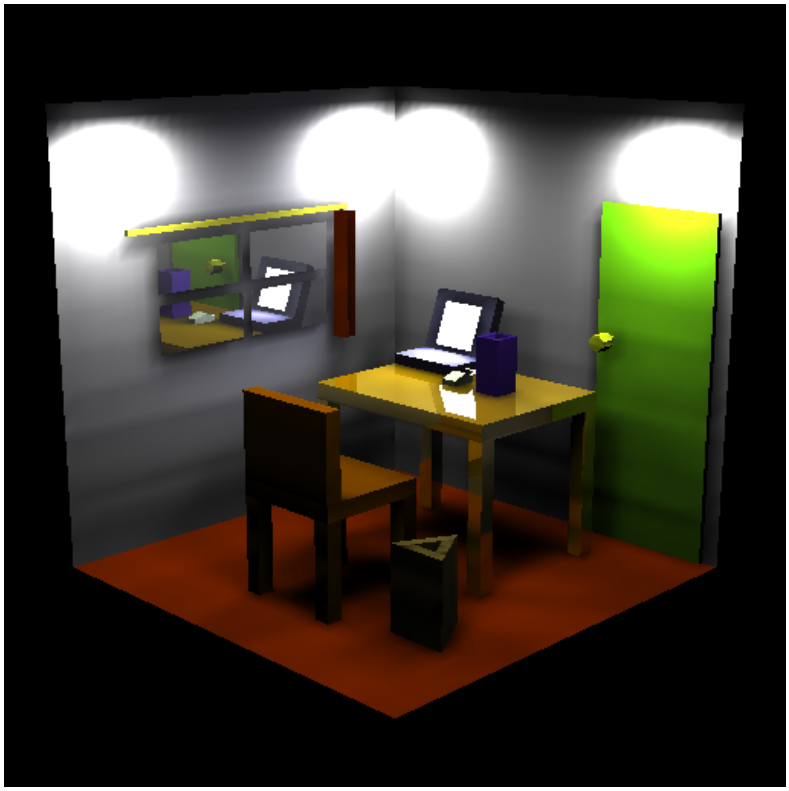

図 25: テストシーン

並列ラジオシティステージ法の性能評価　図 26, 27, 28 にそれぞれ 1,2,3 次元の $(M\pi)^2$ での並列ラジオシティ法における PE 数と速度向上率及び実行効率の関係を示す。ここで、速度向上率とシステムの実効稼働率はそれぞれ式 (22), (23) で定義される。

$$\text{速度向上率} = \frac{\text{単一 PE による画像生成時間}}{\text{システムの画像生成時間}} \tag{22}$$

$$\text{実効稼働率} = \frac{\text{単一 PE による画像生成時間}}{\text{システムの画像生成時間} \times \text{システムの PE 数}} \tag{23}$$

静的負荷分散は定義空間を PE 数に比べ多く分割し、離れた位置にある複数の部分空間を各 PE に割付けることによって行なわれる。静的負荷分散の効果のみを見るために図 26,27,28 ではレイプロセッシングブロック数を 1 に固定した。これらの図が示すように PE 数が増加するに従い、性能の向上が見られ、各次元の $(M\pi)^2$ はそれぞれ良いスケーラビリティを有することがわかる。よって、本論文で提案したオブジェクト空間分割型並列計算モデルに基づくマルチパスレンダリング法のラジオシティ法の並列化が有効であることが示された。現存する超並列計算機においてはシステムの稼働率が数パーセントになることも珍しくないが、静的負荷分散の効果により PE を 256 持つ 2 次元の $(M\pi)^2$ は 6 割を越える実効稼働率を示している。ブロック型の部分空間割付法によって静的負荷分散を行わない場合の同規模の $(M\pi)^2$ の実効稼働率が 2 割程度であることを考えると非常に高い実効稼働率である。

また、静的負荷分散の方式別に見るとブロック型の割付法よりも分散型の割付法の方が常に良い性能を収めており、かつまた分散型の割付法よりもスキュー化分散割付法の方が常に性能が良いことがわかる。よって、本論文で提案したスキュ化静的負荷分散法の有効性が示された。

各次元での速度向上率と実効稼働率を見ると、PE 数が少ない方が速度向上率が理想に近く、効率が高いことがわかる。これは空間分割数を固定したために PE 数が増大するにつれて静的負荷分散の効果が低下しているためであると思われる。

さらに、3 次元の $(M\pi)^2$ における分散型の割付法と 2 次元の $(M\pi)^2$ におけるスキュー化分散型の割付法を図 29 において比較してみた。2 次元の $(M\pi)^2$ においてスキュー化分散型割付法を適用した場合の性能が 3 次元の $(M\pi)^2$ における分散型割付法よりも良いことは注目に値する。次数の低いシステムは高い次数のシステムに比較して低コストで実装可能であることを考えるとこのスキュー化分散型割付法の有効性はさらに高い。これはスキュー化分散型割付法の方が分散型の割付法よりも柔軟性が高いことが要因であると考えられる。なぜなら、3 次元の分散型の割付法では、PE の担当する部分空間が軸方向に固定されてしまう傾向があるが、スキュー化分散型の割付法では PE の担当する部分空間が軸方向には固定されず、分配されるからである。多くの物体が軸方向のコヒーレンスを持つことを考えると、スキュー化

によって PE の担当する部分空間を軸方向に対し分散させることは有効な静的負荷分散法であると考えられる。

次に並列ラジオシティ法における動的負荷分散の効果をみることにする。図 30 にスキュー化分散型割付法を適用時の動的負荷分散の効果を示す。動的負荷分散法を適用した場合の速度向上率と実行効率の定義をそれぞれ式 (24),(25) に示す。

$$\text{速度向上率} = \frac{\text{単一レイプロセッシングブロックによる画像生成時間}}{\text{システムの画像生成時間}} \quad (24)$$

$$\text{実効稼働率} = \frac{\text{単一レイプロセッシングブロックによる画像生成時間}}{\text{システムの画像生成時間} \times \text{システム全体でのレイプロセッシングブロック数}} \quad (25)$$

動的負荷分散は、PE 数を増加させる代わりにレイプロセッシングブロックを 1 つの PE 中に複数持たせ、各レイプロセッシングブロックにレイを動的に割り当てることによって達成される。3.3.3 節で述べたように動的負荷分散の効果は静的負荷分散の効果がある場合に発揮されると考えられる。そのためここでは最も静的負荷分散の効果のあった 2 次元の $(M\pi)^2$ にスキュー化分散型割付法を適用した場合の性能について示した。

PE 数を一定にした場合、レイプロセッシングブロック数を増加させると、レイプロセッシングブロック数の増加につれて性能が向上していくことがわかる。これにより、レイプロセッシングブロック数を増加させたことによる性能向上法は効果があることがわかる。ただし、どの PE 数の場合でもレイプロセッシングブロック数が 8 を越えるあたりから性能の向上がみられなくなる。これは PE がレイプロセッシングブロックを計算要素としてもつ共有メモリ型の計算機であり、物体情報が格納される共有メモリへのアクセス競合やレイの動的割り当てを行うレイディストリビュータの性能がレイプロセッシングブロック数の増大によって飽和してしまうことによると考えられる。

レイプロセッシングブロック数を増加させた場合に性能向上があるということだけでは動的な負荷分散が行なわれたかを判別することは難しいため、次に動的負荷分散の効果を見ることにする。図 30-b には、レイプロセッシングブロック数が 1024 の位置に破線を付加してある。これは同数のレイプロセッシングブロックを持つシステム間で性能がどのように分布するかを見るためである。この線上に注目すると、性能飽和が起きていない範囲では PE 数が小さい方が性能が良いことがわかる。つまり同一数のレイプロセッシングブロックのシステムであっても PE 数をおさえてレイプロセッシングブロック数を増加させた場合に性能が良くなることがわかる。これが動的負荷分散の効果である。この線上の、64 PE × 16 レイプロセッシングブロックの時には 6 割以上の実効稼働率を達成し、256 PE × 4 レイプロセッシングブロックの場合にも 5 割を越える実効稼働率を達成している。これは静的負荷分散法だけでは達成することのできない稼働率である。以上により静的負荷分散

法と動的負荷分散法は組み合わせて利用することで相乗的な効果を生むことがわかる。

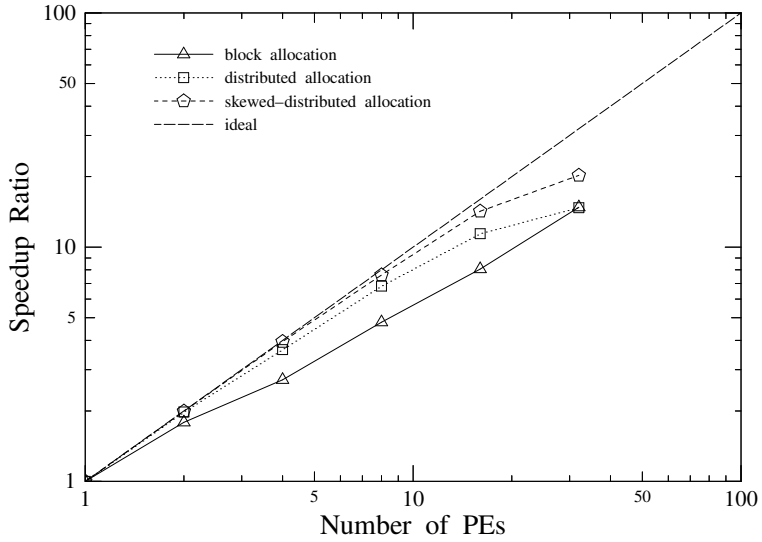

図 26-a: PE 数と速度向上率の関係

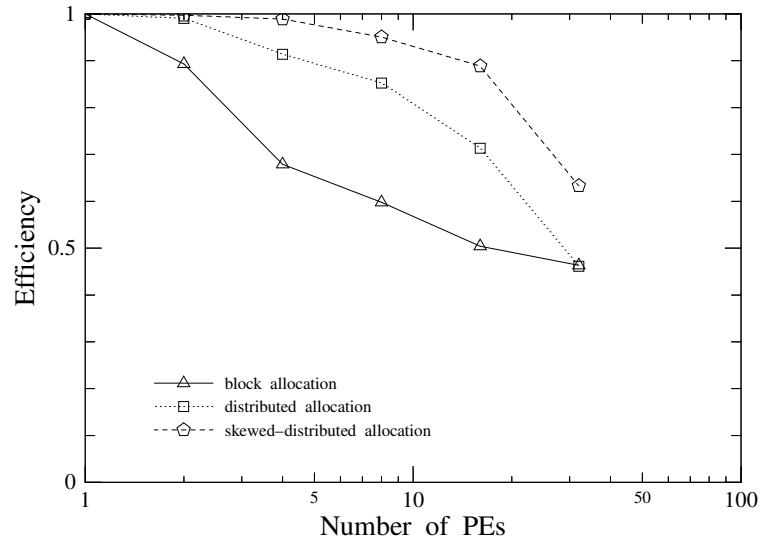

図 26-b: PE 数と実効稼働率の関係

図 26: 1 次元の $(M\pi)^2$ システムの性能 (並列ラジオシティ法)

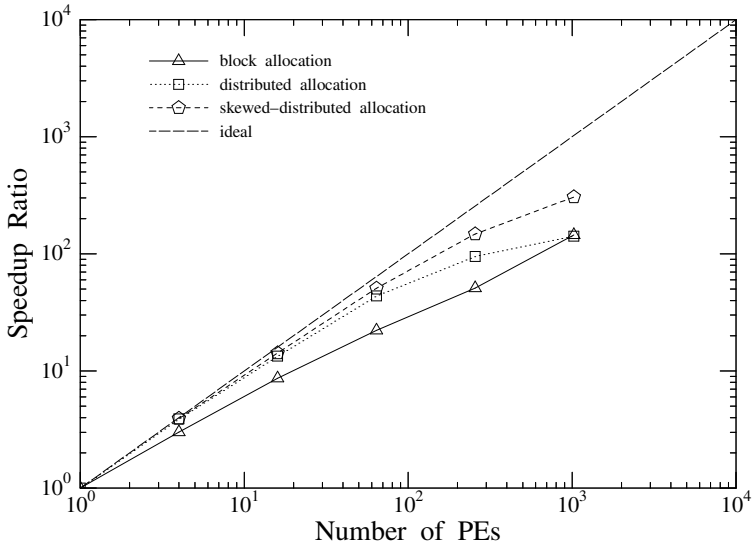

図 27-a: PE 数と速度向上率の関係

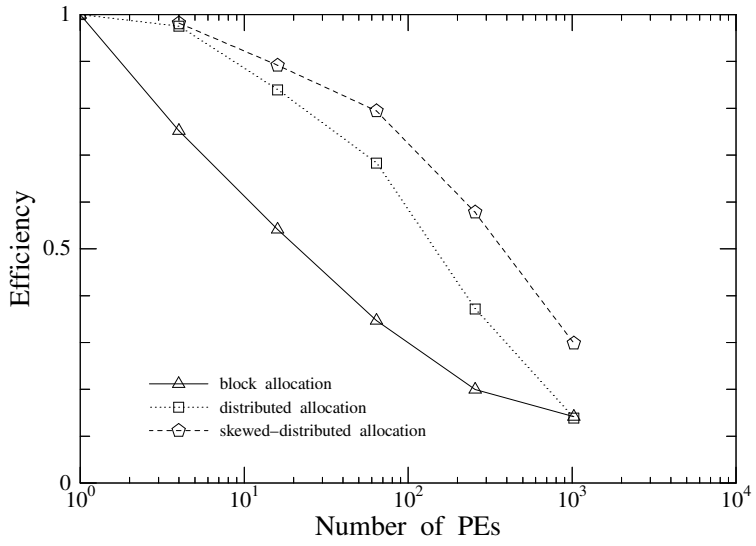

図 27-b: PE 数と実効稼働率の関係

図 27: 2 次元の $(M\pi)^2$ システムの性能 (並列ラジオシティ法)

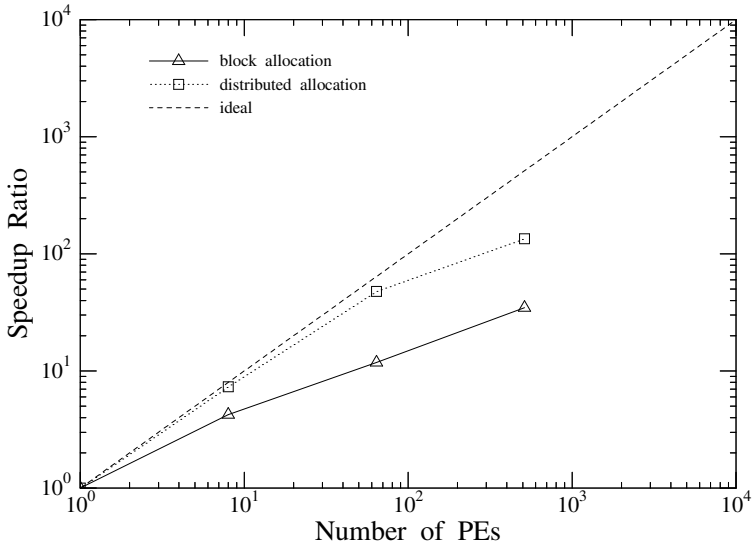

図 28-a: PE 数と速度向上率の関係

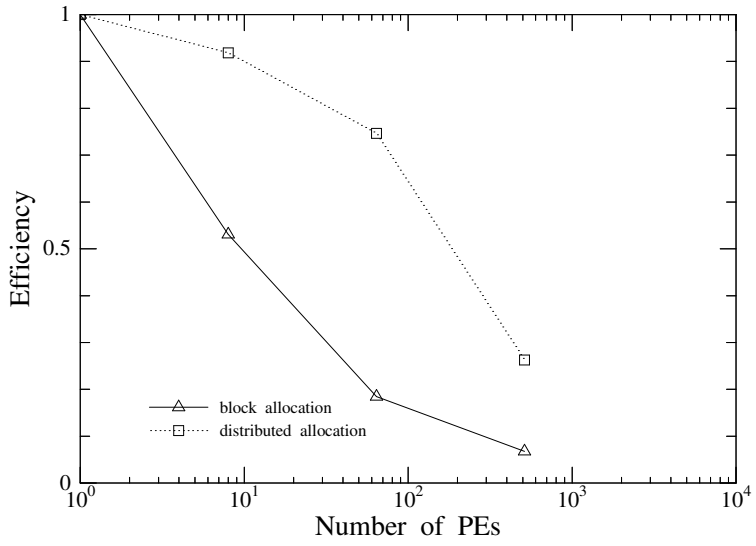

図 28-b: PE 数と実効稼働率の関係

図 28: 3 次元の $(M\pi)^2$ システムの性能 (並列ラジオシティ法)

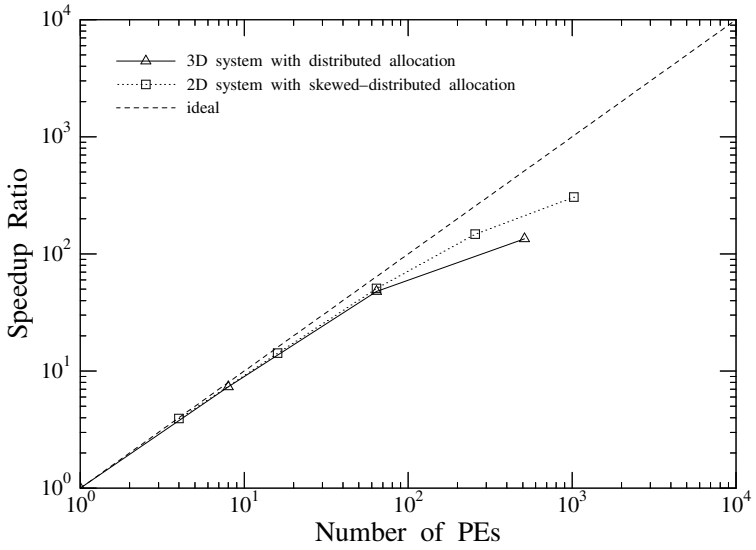

図 29-a: PE 数と速度向上率の関係

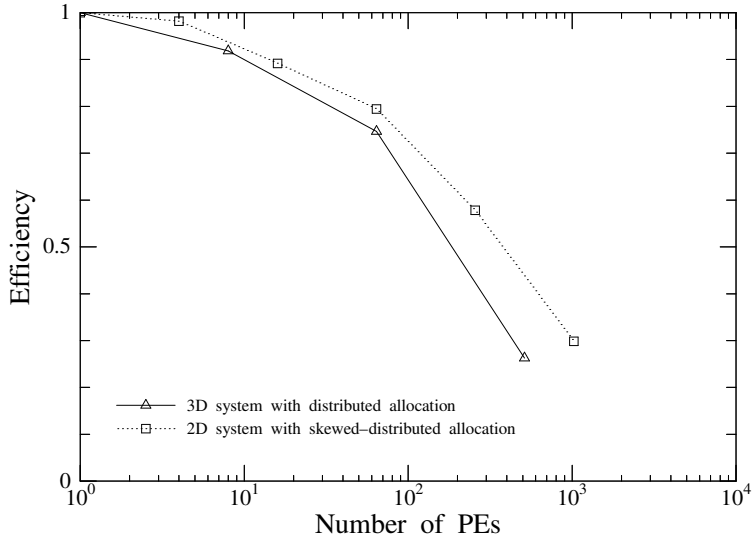

図 29-b: PE 数と実効稼働率の関係

図 29: 2 次元の $(M\pi)^2$ におけるスキュー化分散型割付法と 3 次元の $(M\pi)^2$ における分散型割付法の比較 (並列ラジオシティ法)

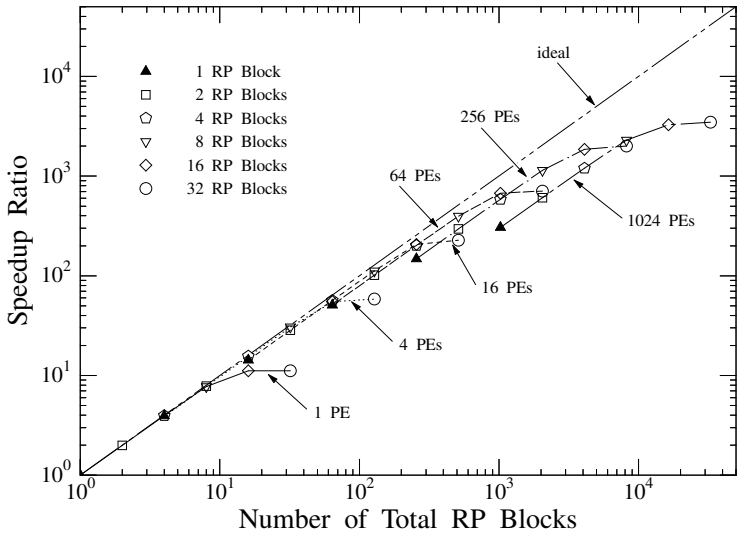

図 30-a: レイプロセッシングブロック数と速度向上率の関係

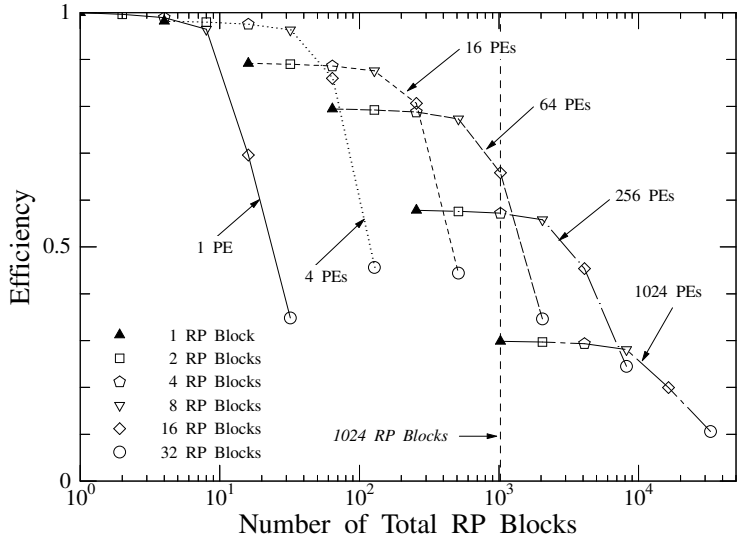

図 30-b: レイプロセッシングブロック数と実効稼働率の関係

図 30: 2 次元の $(M\pi)^2$ におけるスキュー化分散型割付法適用時の動的負荷分散の効果 (並列ラジオシティ法)

並列レイトレーシング法の性能評価　図 31, 32, 33 にそれぞれ 1, 2, 3 次元の $(M\pi)^2$ での並列レイトレーシング法における PE 数と速度向上率及び実行効率の関係を示す。これらの図では静的負荷分散法の効果を見るためにレイプロセッシングブロック数を 1 に固定してある。並列ラジオシティ法と同様、PE 数が増大するに従って性能の向上がみられ、各次元の $(M\pi)^2$ はそれぞれ良いスケーラビリティを有することがわかる。並列ラジオシティ法と同様、2 次元の構成の 256 PE の $(M\pi)^2$ の稼働率は 6 割を越している。また、図 31, 32, 33 をそれぞれ図 26, 27, 28 と比較するとほぼ同様の傾向がみえることがわかる。これは、オブジェクト空間分割型並列計算モデルに基づく並列化方式は、ラジオシティ法、レイトレーシング法によらず、大域照明モデルに基づく画像生成アルゴリズムを効果的に並列化可能であることを示している。

静的負荷分散法の違いをみると、並列レイトレーシング法では並列ラジオシティ法ほとブロック型の割付法と分散型の割付法に顕著な差はみられないが、並列ラジオシティ法同様に、ブロック型の割付法、分散型の割付法、スキュー化分散割付法の順に良い性能を示していることがわかる。並列レイトレーシング法は、並列ラジオシティ法に比べると良い性能を示す傾向があるが、これは並列レイトレーシング法では並列ラジオシティ法に比べ、一次レイが一点から放射されるため、初期レイに強いコヒーレンスが存在しその影響がでているものと思われる。たとえば視点から発っせられたレイはラジオシティ法と異なり、最初は空間に対しほぼ同一方向に伝播していく。PE が光線の伝播方向に対し垂直に並べられていた場合には、ブロック型と分散型の割付法では担当する光線が各 PE に分散しやすいという利点がある。つまり配置によって光線の負荷が分散されやすい。これは担当する物体の負荷分散ではないことと視点に依存するために良い性質の負荷分散ではないが、一次レイの持つコヒーレンスを利用した負荷分散となっている。今回のシミュレーションでは PE を画面と同方向の配置になるよう、つまりレイの伝播方向とは垂直になるように並べたため、このような影響がブロック型の割付法と分散型の割付法に対して表われ、良い性能を示したと考えられる。また、このように PE が光線の伝播方向に対し垂直に並べられていた場合にはブロック型の割付法と分散型の割付法は、各レイの伝播の様子をみるとほとんど違いがないため、並列ラジオシティ法に比べて二者の差が表れにくく、1 次元のシステムにおいて一部ブロック型が分散型を上まわる結果を得たと考えられる。しかし、スキュー化分散方式ではこのような PE の配置と画面の位置による影響が少ないため、たまたま存在した一次レイの負荷分散に、スキュー化分散型の割付け法による負荷分散の効果が表れ、スキュー化分散型割付方式は常に分散型とブロック型の割付法をはっきりと凌駕する性能を示している。つまり、スキュー化分散型割付法は視点の位置などの影響を受けにくいロバストな静的負荷分散法であると言える。以上からスキュー化分散割付法

がオブジェクト空間分割型並列計算モデルに基づく並列計算における静的負荷分散法として有効であるということが言える。

また、並列ラジオシティ法と同様、3 次元の $(M\pi)^2$ における分散型の割付法と 2 次元の $(M\pi)^2$ におけるスキュー化分散型の割付法を図 34 において比較した。やはり、2 次元の $(M\pi)^2$ においてスキュー化分散型割付法を適用した場合の性能が 3 次元の $(M\pi)^2$ における分散型割付法よりも良いことがわかる。これにより、レイトレーシング法においても分散型の割付法より、スキュー化分散型割付法が優れていることが示された。

図 35 に 2 次元の $(M\pi)^2$ においてスキュー化分散型の割付法に動的負荷分散法を適用した場合の並列レイトレーシング法の性能を示す。256 PE で 8 レイプロセッシングブロックの時、つまり 4096 レイプロセッシングブロックの部分であっても実効稼働率が 5 割以上を保持しており非常に良い結果を得ている。

図 30 と図 35 は並列ラジオシティ法と並列レイトレーシング法という差はあるが、同じ 2 次元の $(M\pi)^2$ におけるスキュー化分散型の割付法適用下における動的負荷分散のシミュレーション結果を示した図である。この 2 つの図を比較すると速度向上率と実効稼働率について並列ラジオシティ法と並列レイトレーシング法では、ほぼ同じ傾向が見られることがわかる。これは、オブジェクト空間分割型並列計算モデルに基づく並列化手法がレイトレーシング法とラジオシティ法という異なる画像生成アルゴリズムを統一的に並列化可能であることを示していると考えられる。また、この図 30 と図 35 を詳細に比較するとレイプロセッシングブロック数が大きい部分で並列レイトレーシング法の速度向上率の方が並列ラジオシティ法のものよりわずかに上回っていることがわかる。これは先程述べたように並列レイトレーシング法では一次レイのコヒーレンスの利用によって並列ラジオシティ法を上回る性能を得たと予測される。

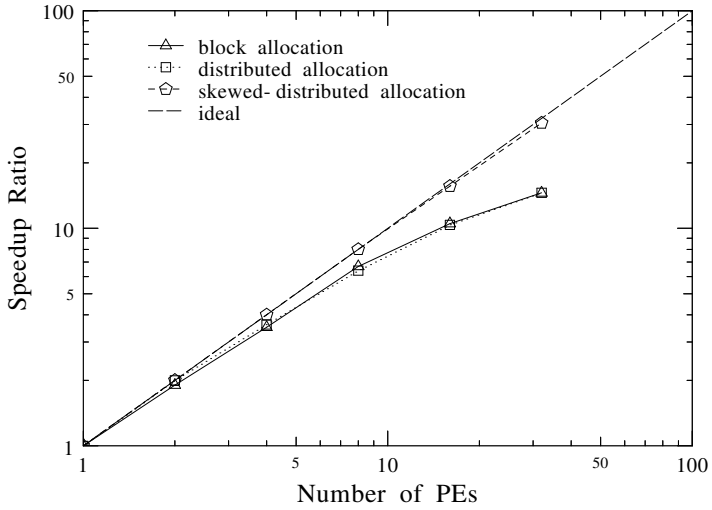

図 31-a: PE 数と速度向上率の関係

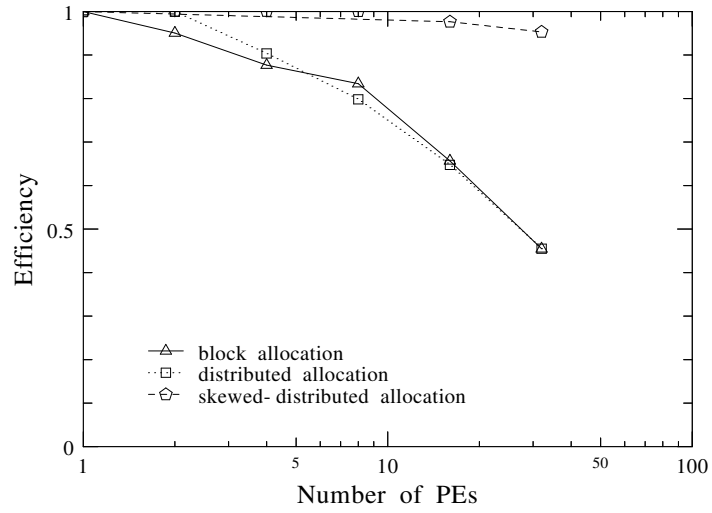

図 31-b: PE 数と実効稼働率の関係

図 31: 1 次元の $(M\pi)^2$ システムの性能 (並列レイトレーシング法)

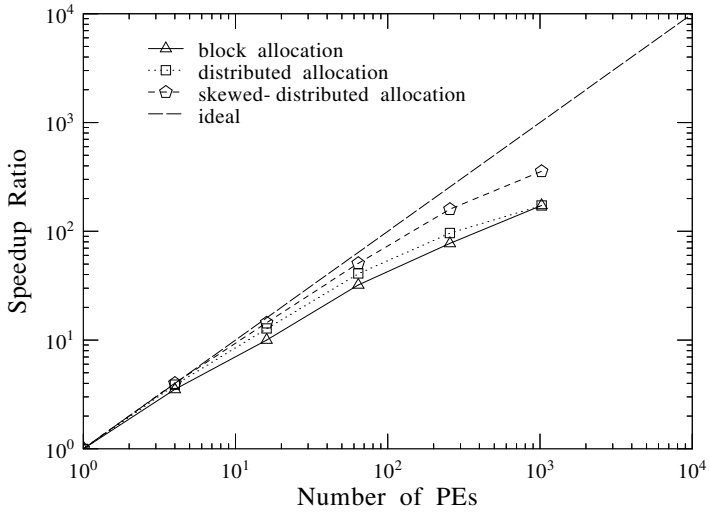

図 32-a: PE 数と速度向上率の関係

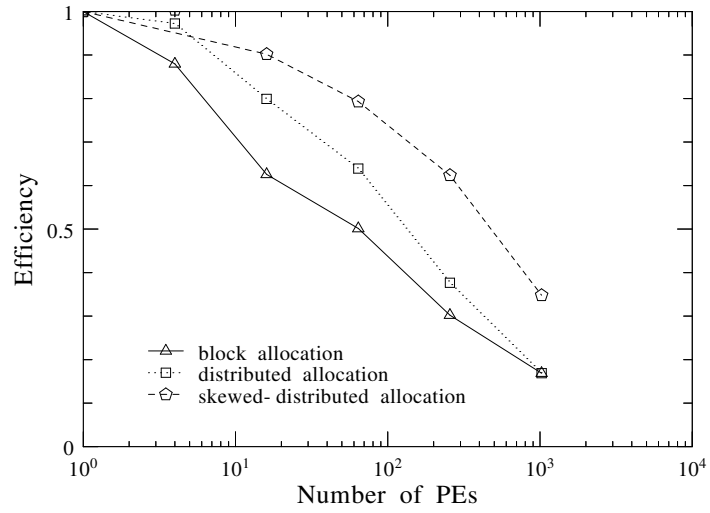

図 32-b: PE 数と実効稼働率の関係

図 32: 2 次元の $(M\pi)^2$ システムの性能 (並列レイトレーシング法)

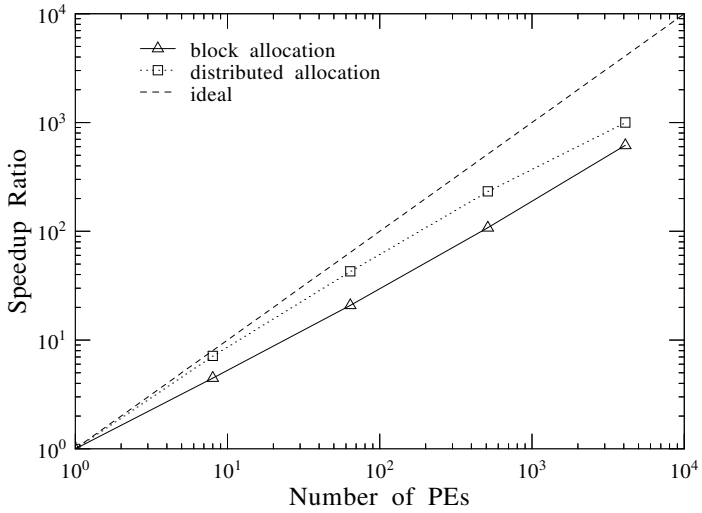

図 33-a: PE 数と速度向上率の関係

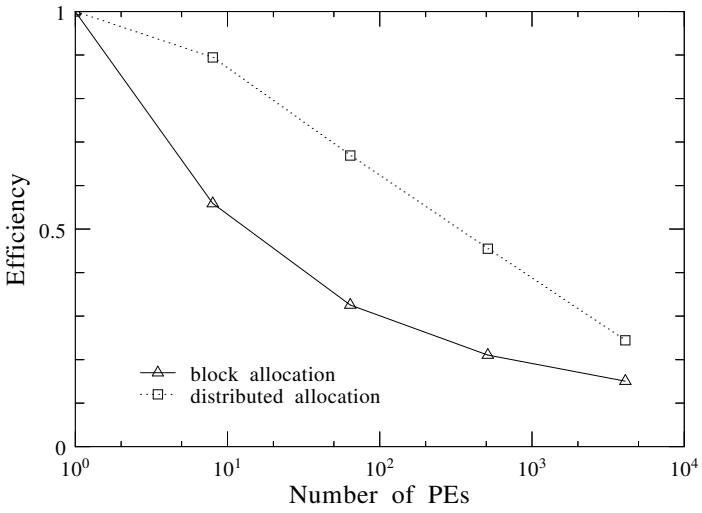

図 33-b: PE 数と実効稼働率の関係

図 33: 3 次元の $(M\pi)^2$ システムの性能 (並列レイトレーシング法)

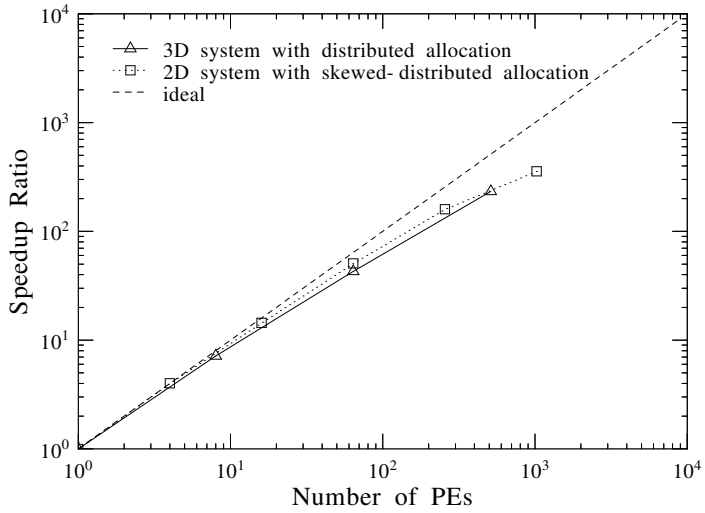

図 34-a: PE 数と速度向上率の関係

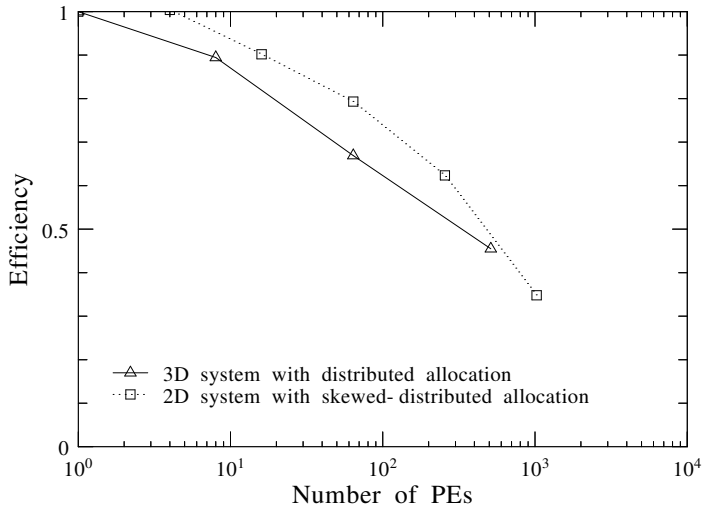

図 34-b: PE 数と実効稼働率の関係

図 34: 2 次元の $(M\pi)^2$ におけるスキュー化分散型割付法と 3 次元の $(M\pi)^2$ における分散型割付法の比較 (並列レイトレーシング法)

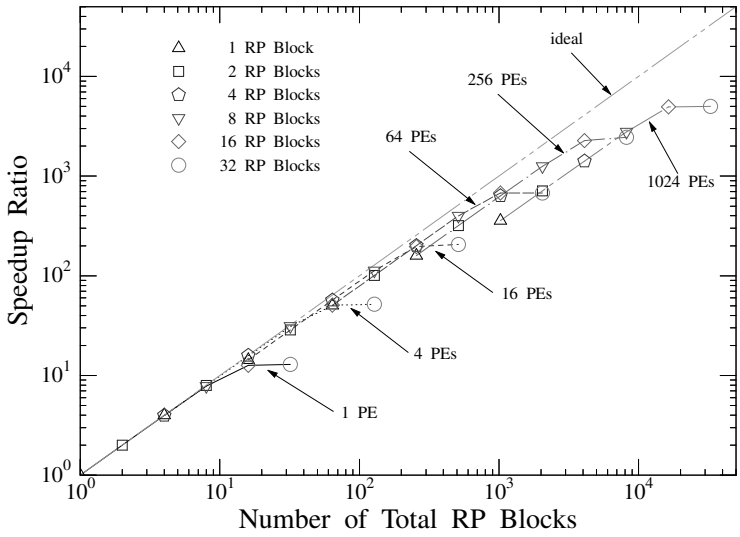

図 35-a: レイプロセッシングブロック数と速度向上率の関係

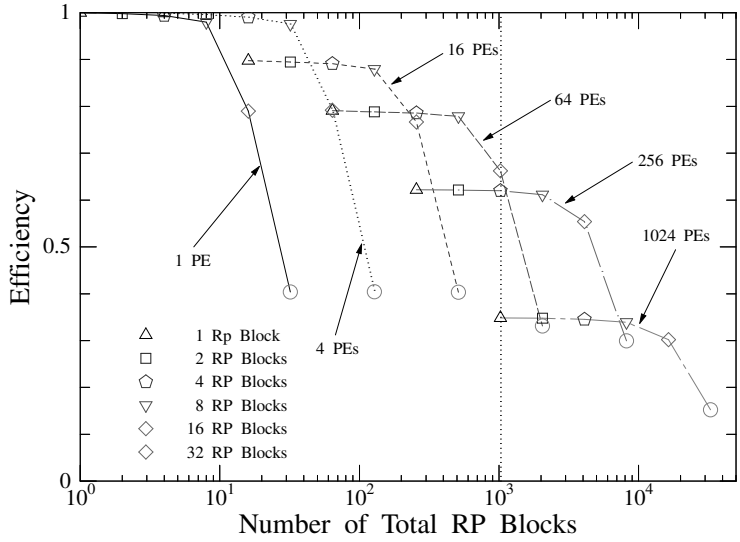

図 35-b: レイプロセッシングブロック数と実効稼働率の関係

図 35: 2 次元の $(M\pi)^2$ におけるスキュー化分散型割付法適用時の動的負荷分散の効果 (並列レイトレーシング法)

4.3 キャッシュドフレームバッファシステムの性能評価

4.3.1 キャッシュドフレームバッファシステムのランダムシミュレーションによる性能評価

3.4 節において、キャッシュドフレームバッファシステムの挙動を待ち行列理論によって解析した。その際、PE からの輝度情報の発生間隔をポアソン分布と仮定した。また、フレームバッファキャッシュ要素は M/M/1、通信要素は M/D/1 のふるまいをし、M/D/1 への入力がポアソン分布である場合の退去過程をポアソン分布で近似できると仮定した。しかしながら、本来 M/D/1 では、ポアソン分布間隔の入力があっても退去間隔はポアソン分布とはならないなど、このモデルには仮定に問題が存在する可能性がある。そこでこの仮定を用いての解析がどの程度信頼できるか、離散時間イベントドリブン [19] のシミュレータを構築し、シミュレーションを行なった。使用したパラメータは 3.4.3 節の表 3 と同じである。輝度発生は乱数を用いて指数分布に従うように行なった。乱数の発生には、GNU の libg++の Random クラス (ACG, NegativeExpntl) を用いた。

図 36 は 4-ary 5-stage のキャッシュドフレームバッファシステムの平均レイテンシをシミュレーションにより求めた結果である。シミュレーションにおいては 1000×1000 のサイズの画像の生成を想定し、レイを 100 万本発生させた。この条件により、シミュレーション開始時の過渡的な状態の影響などはほぼ無視できると考えられる。理論解析の結果である図 22, 23 に対応するランダムシミュレーションの結果が図 36, 37 に示されている。また、理論解析と同様に、ミス率の変化と平均滞在時間の関係を図 38 に示す。これらを比較すると、理論値とシミュレーション結果が同様の傾向を示していることがわかる。平均滞在時間が飽和点に達する平均輝度発生率も理論値とシミュレーションの差はおよそ一桁程度におさまっている。これは、通信要素の退去間隔をポアソン分布に近似するという仮定をしても傾向を見るには十分であるということを示していると考えられる。また、シミュレーションの結果ではキャッシュドフレームバッファシステムの性能が飽和点に関しては一桁程度良く、飽和しない範囲での平均性能も良い結果となっていることがわかる。よって、待ち行列理論によるキャッシュドフレームバッファシステムの理論解析はキャッシュドフレームバッファシステムの設計における基礎的なデータとして採用することが可能であると考えられる。

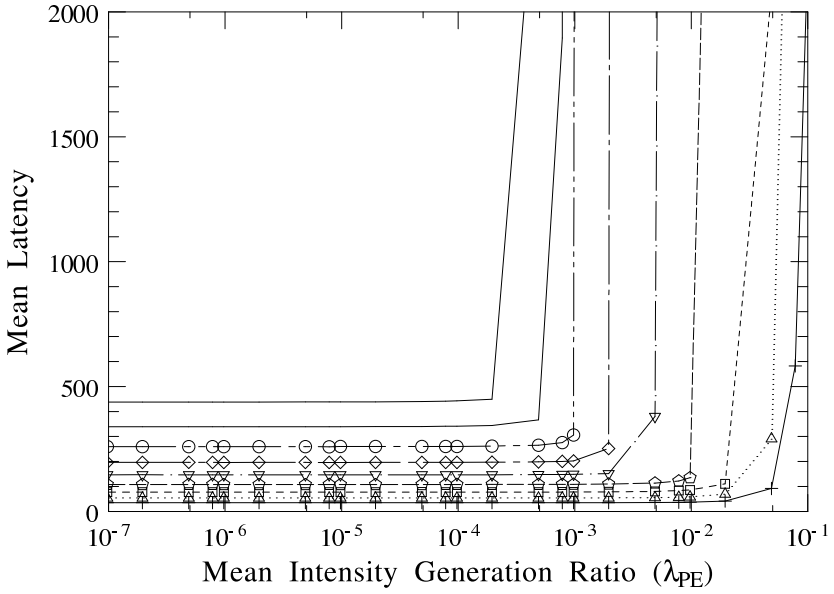

図 36: ランダムシミュレーションによるキャッシュドフレームバッファシステムのキャッシュミス率と平均レイテンシの関係 (4-ary 5-stage)

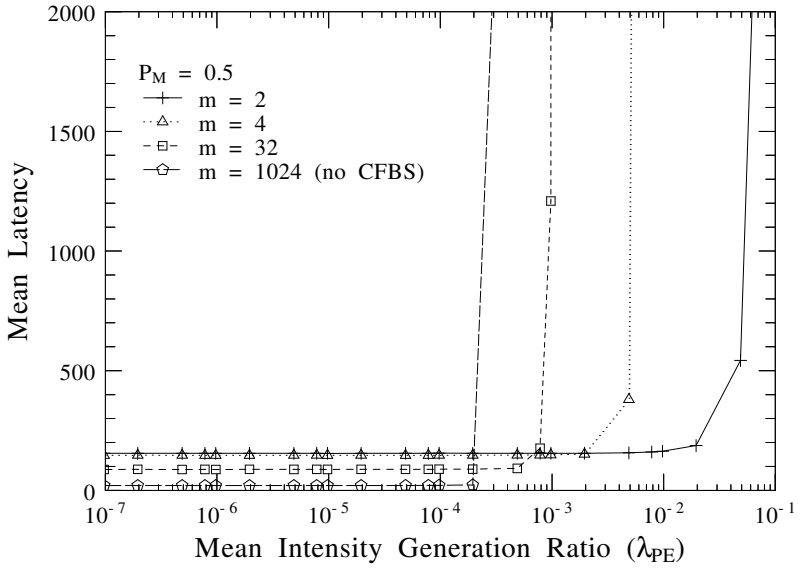

図 37: ランダムシミュレーションによるキャッシュドフレームバッファシステムの構成と平均レイテンシの関係 (ミス率 = 0.5)

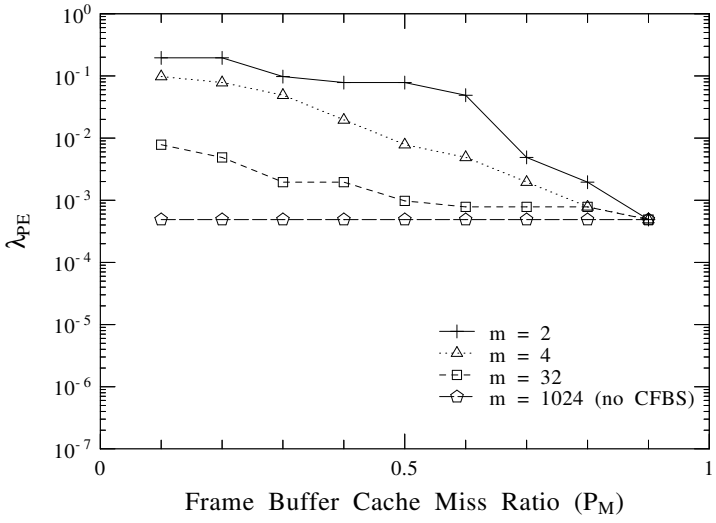

図 38: ランダムシミュレーションによるキャッシュドフレームバッファシステムのミス率と性能飽和点の関係

4.3.2 キャッシュドフレームバッファシステムを含めた $(M\pi)^2$ の性能評価

第 3.4 節でキャッシュドフレームバッファシステムの性能を待ち行列理論により理論的に解析し、前節ではランダムシミュレーションによりキャッシュドフレームバッファシステムの性能を評価した。ここではキャッシュドフレームバッファシステムを $(M\pi)^2$ と組み合わせ、並列レイトレーシング法のステージで実際に画像を生成しながらの実行駆動型シミュレーションによりキャッシュドフレームバッファシステムの評価を行う。

図 39,40 にフレームバッファへのアクセス競合を考慮した $(M\pi)^2$ のシミュレーション結果を示す。PE 数が 256 の場合には 16-ary 2-stage のキャッシュドフレームバッファシステムを用い、PE 数が 1024 の場合には 32-ary 2-stage のキャッシュドフレームバッファシステムを用いた。それぞれ、レイプロセッシンブロック数が 1 の場合の速度を基準にしてレイプロセッシングブロック数を増加させた時の速度向上率を、フレームバッファあり、無し、フレームバッファの代わりにライトバッファを用いた場合で比較した。図 39 より PE 数が 256 の時にはフレームバッファキャッシュの効果がほとんどないことがわかる。しかし、図 40 で PE 数が 1024 の場合にレイプロセッシングブロック数が増加した時点でキャッシュドフレームバッファシステムの効果があることがわかる。またライトバッファの場合よりもキャッシュドフレームバッファシステムの方が効果があることからキャッシュでのアキュムレートが行

なわわれていることがわかる。

理論解析においてもランダムシミュレーションにおいてもキャッシュドフレームバッファシステムの評価に際して重要なパラメータとして輝度の発生確率があった。実際に $(M\pi)^2$ でどの程度の輝度の発生確率があるかを測定してみると、1024 PE で 1 レイプロセッシングブロックの時には約 5.9×10^{-6} であり、1024 PE で 32 レイプロセッシングブロックの時には 8.3×10^{-5} であった。また今回の条件ではフレームバッファキャッシュへのヒット率は 6 割程度であった。理論解析とランダムシミュレーションの結果からはあと一桁程度大きな輝度発生率でないとキャッシュドフレームバッファシステムが不必要であるという結論になる。しかし、実際には PE 数が 1024 の場合にレイプロセッシングブロック数が 16 あたりでキャッシュドフレームバッファシステムの効果が見られる。これは理論解析とランダムシミュレーションで輝度の発生はポアソン分布であるという仮定と実際の発生分布にずれがあるためと考えられる。実際の輝度の発生間隔の分散は平均値に比べて 100 から 1000 倍の大きさを持っており、実際には少なくともポアソン分布とは異なった分布で輝度は発生していると考えられる。

では、キャッシュドフレームバッファシステムは無用の存在なのであろうか。近年のマイクロプロセッサの速度の向上は目をみはるものがあり、それに伴なって、レイプロセッシングブロックの速度が今後急速に向上していくものと考えられる。それに比べ、フレームバッファのようなメモリや、ネットワークの速度の向上率は低く抑えられている。また、今回仮定したユニットの速度の基準となる Sparc Chip 40MHz は既に現在、遅い部類のプロセッサとなっている。そこで、プロセッサの速度とフレームバッファ等のメモリ要素との速度の比をプロセッサメモリサイクル比 (PMCR : Processor Memory Cycle Ratio) とし、これをパラメータとして変化させた場合の $(M\pi)^2$ の性能評価を行なってみた。その結果が図 41 である。この図には、1024 PE の $(M\pi)^2$ において、プロセッサの速度を向上させた、つまりプロセッサメモリサイクル比を増大させた場合の性能評価を行なった。これを見ると、プロセッサメモリサイクル比が向上した時点でキャッシュドフレームバッファシステムの効果がみられるようになる。また、レイトレーシング法にディストリビューテッドレイトレーシング法を採用すれば、一つのピクセルに複数のレイを発するため局所性が増加し、さらに性能向上が見られると予想される。以上からキャッシュドフレームバッファシステムは今後有効になっていく技術であると考えられる。今後の超並列画像生成計算機の設計にはこのような階層的なフレームバッファを考慮したものになるであることが予想できる。

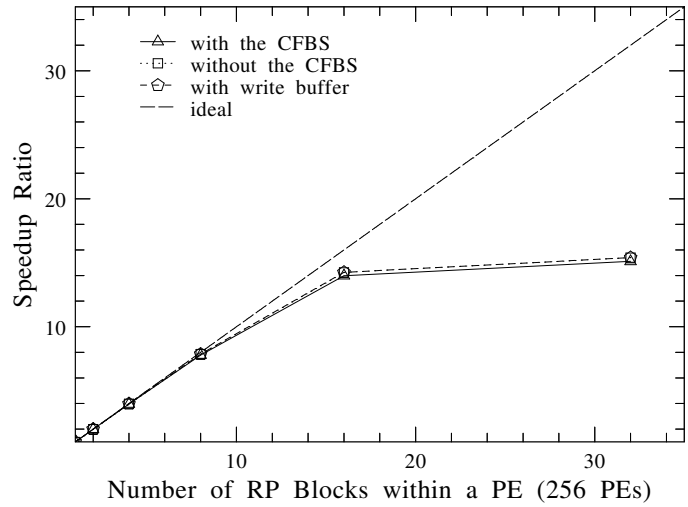

図 39-a: 速度向上率

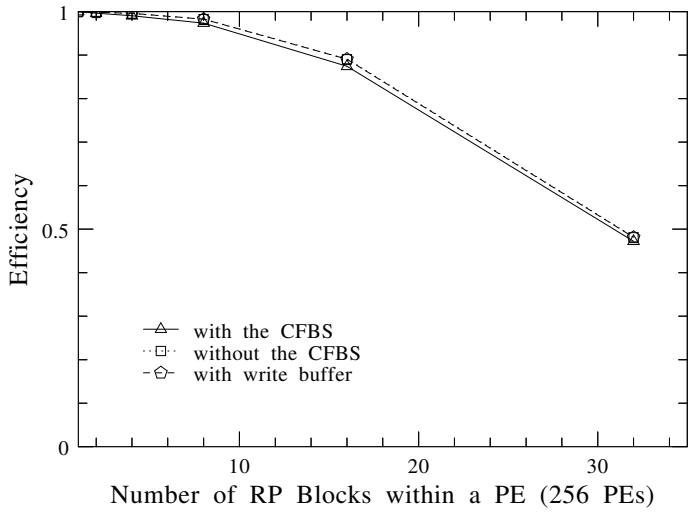

図 39-b: 実効効率

図 39: キャッシュドフレームバッファシステムを考慮した場合の $(M\pi)^2$ の性能 (PE 数 256)

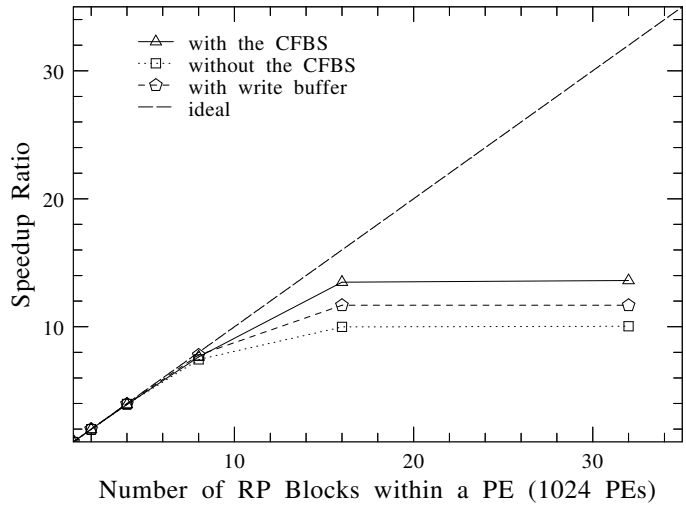

図 40-a: 速度向上率

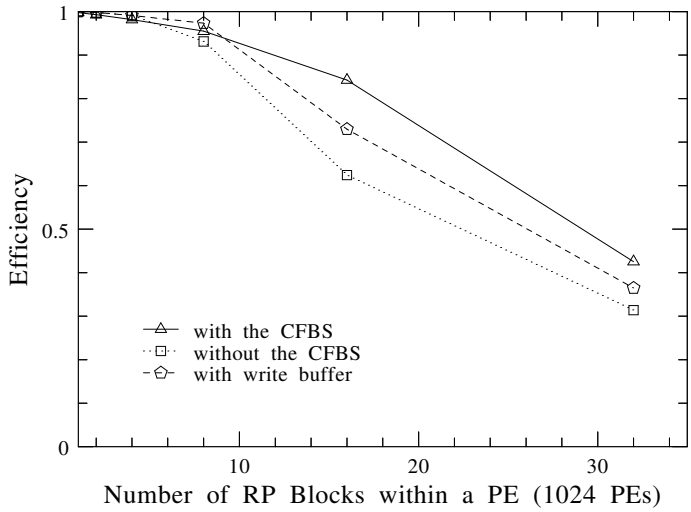

図 40-b: 実効効率

図 40: キャッシュドフレームバッファシステムを考慮した場合の $(M\pi)^2$ の性能 (PE 数 1024)

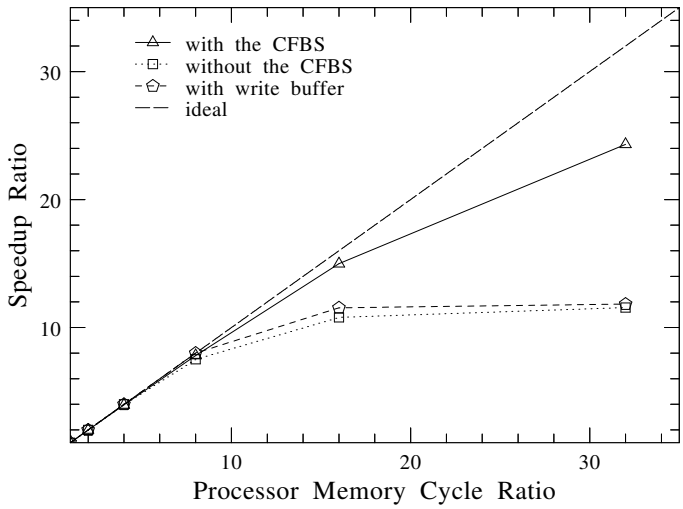

図 41: キャッシュドフレームバッファシステムを考慮した $(M\pi)^2$ においてプロセッサメモリサイクル比を変化させた場合の性能評価 (PE 数 1024)

4.4 結言

本章ではこれまでに提案してきたオブジェクト空間分割型並列計算モデルに基づく並列マルチパスレンダリング法の $(M\pi)^2$ 上での性能について考察した。

まずは、ソフトウェアシミュレーションにより並列マルチパスレンダリング法を $(M\pi)^2$ 上で実行した際の性能評価を行なった。最初にシミュレーションのモデルを示し、次に静的負荷分散法と動的負荷分散法がマルチパスレンダリング法の並列ラジオシティ法と並列レイトレーシング法で有効に働くことを確認し、オブジェクト空間分割型並列計算モデルに基づくマルチパスレンダリング法の並列化が有効であることを示した。並列ラジオシティ法においては静的な負荷分散法と動的な負荷分散法を組み合わせた時に 1024 レイプロセッシングブロックで 6 割を越える実効稼働率を達成した。同様に並列レイトレーシング法においては 4096 レイプロセッシングブロックで 6 割を越える稼働率を達成し、並列計算により 2500 倍近くの高速化を達成した。また、スキュー化分散型の割付法は 2 次元のシステムにおいて 3 次元の分散型割付法よりも効果的であることを示した。より低次のシステムの方がシステム構築のコストが低いことを考えると、スキュー化分散型割付法は静的負荷分散に対し有効な手法であると言える。これらの結果は現在渇望されている写実的画像生成の高速化に対する一つの解を与えるものである。また、ランダムに輝度を発生させた場合のキャッシュドフレームバッファシステムのふるまいを離散時間シミュレーションにより調査した。離散時間シミュレーションによるキャッシュドフレームバッファシステムの動作は理論解析に近いものであり、この理論解析がキャッシュドフレームバッファシステムの設計に有効であることがわかった。

次にキャッシュドフレームバッファシステムを含めた $(M\pi)^2$ の並列レイトレーシング法の総合的な性能評価をソフトウェアシミュレーションにより行なった。その結果 PE 数が 1000 を越える所でキャッシュドフレームバッファシステムを用いた場合のみにキャッシュドフレームバッファシステムの効果が見られた。全体的に、本性能評価で用いたテスト画像では、$(M\pi)^2$ からの輝度情報の発生間隔が長いことが判明し、キャッシュドフレームバッファシステムの効果があまりみられなかった。しかしながら、今後の超並列計算機の規模の増加と近年のプロセッサ速度とメモリ速度の差の広がりを考えると、フレームバッファに対する負荷がより重くなることが考えられる。これらを考慮して計算要素の速度を向上させたシミュレーションを行なったところ、キャッシュドフレームバッファシステムの効果がみられた。よって、キャッシュドフレームバッファシステムは将来重要になる技術であると考えられる。今後超並列画像生成システムにおいてはこのような階層的なフレームバッファシステムが採用されると考えられる。

5　結論

本論文は写実的画像生成法であるマルチパスレンダリング法をオブジェクト空間分割型並列計算モデルに基づいて並列化し、並列化したマルチパスレンダリング法を効率良く実行するための超並列画像生成システム $(M\pi)^2$ を提案し、その評価を行なった。以下にその内容を総括する。

第 2 章では、本論文で着目した写実的画像生成法であるマルチパスレンダリング法とその並列化手法について述べた。マルチパスレンダリング法はラジオシティ法とレイトレーシング法の逐次的な組合せであり、現在最も写実的な画像を生成可能であると考えられている。この画像生成法は、生成する画像の質は十分と考えられるが、その反面膨大な計算時間を要するという欠点がある。本論文はこの膨大な計算時間を超並列処理によって短縮しようとするものである。まず、これまでの並列化手法が問題の並列化に拘泥するあまり、実際の物理的な並列計算機からかけはなれた並列計算モデルを利用していることを指摘した。そしてレイトレーシング法とラジオシティ法に共通して存在する並列性として物体と光の相互作用を単位とする処理に着目した。この処理単位は物理的なモデルに基づくものであり、これを基礎として現実の並列計算機を考慮した並列計算モデルを構築可能である。本論文で提案したオブジェクト空間分割型並列計算モデルはこの物体と光の相互作用を並列処理の単位とする並列計算モデルであり、レイトレーシング法とラジオシティ法という 2 つの異なる画像生成法から構成されているマルチパスレンダリング法を統一的に並列化可能な並列計算モデルである。ここではさらにこのオブジェクト空間分割型並列計算モデルに基づいてマルチパスレンダリング法を並列化し、そのアルゴリズムを示した。

第 3 章では、第 2 章で提案したオブジェクト空間分割型並列計算モデルに基づく並列マルチパスレンダリング法を高速に実行するための並列計算機アーキテクチャ $(M\pi)^2$ を提案し、その詳細について述べた。通常の並列処理方式ではひとたび並列処理単位を決定するとそれを基準に並列処理が行なわれる。しかし、抽出された並列処理単位内にもさらに並列処理可能な単位が存在し、並列処理単位自身が階層構造を持つことがある。この構造を有効に利用するために、並列処理単位の粒度に応じた並列処理を行うことが考えられる。マルチパスレンダリング法には、部分空間ごとの光の伝播という並列処理単位中に、光と物体の相互作用というさらに細い並列処理単位が存在し、その処理単位もまたより細い並列処理単位へと分解される。このような階層的な並列処理単位を効率良く処理するため、本論文では階層的構造を持った超並列画像生成システム $(M\pi)^2$ を提案した。$(M\pi)^2$ は分散メモリ型の超並列計算機であるシステムレベル、共有メモリ型の並列計算機である PE レベル、そしてパイプライン処理を行なうレイプロセッシングブロックレベルという 3 種の階層を持つ超並列画像生成システムである。また、$(M\pi)^2$ のような超並列画像生成システムでは、画像生成の際のフレームバッファへのアクセス競

合による性能低下が考えられるため、フレームバッファへのアクセス競合を緩和するキャッシュドフレームバッファシステムを提案し、その性能についての理論解析を行なった。そして、並列計算機の性能を十分に引き出すための負荷分散法として静的な方法と動的な方法を提案した。静的な負荷分散法は物体の定義空間を PE 数に対してより細く分割し、分散型の割付法により各 PE に割り当てられる物体数を均一に保つ方法である。この方法は低次の $(M\pi)^2$ においては不十分な面があるため、低次のシステムにおいてより有効なスキュー化分散割付法を提案した。しかしながら静的な負荷分散だけでは解決できない負荷の不均衡も存在する。これを解決するため、動的な負荷分散法を提案した。この動的負荷分散法は、$(M\pi)^2$ の階層的な構造を利用し、静的負荷分散法と組みあわせることで局所的に負荷分散を解決するものであり、通信のコストが必要ない点と計算要素の増加などによってオーバヘッドが増大することがないという点が画期的な動的負荷分散方法である。今後は $(M\pi)^2$ のような階層的な並列処理を行なう超並列計算機が多く開発されると予想される。

第 4 章ではこれまでに提案したオブジェクト空間分割型並列計算モデルに基づく並列マルチパスレンダリング法の $(M\pi)^2$ 上の性能を調べるために、離散時間実行駆動方式のソフトウェアシミュレーションによる性能評価を行なった。最初にソフトウェアシミュレーションによって並列ラジオシティ法と並列レイトレーシング法のそれぞれについて静的負荷分散を適用した場合と動的負荷分散を組み合わせた場合についての評価を行なった。その結果、並列ラジオシティ法では静的な負荷分散法と動的な負荷分散法を組み合わせた場合に 1024 レイプロセッシングブロックで実に 6 割を越える稼働率を達成した。同様に並列レイトレーシング法においては 4096 レイプロセッシングブロックのシステムで 6 割を越える稼働率を達成し、並列計算を適用することによってマルチパスレンダリング法による画像生成が最大 2500 倍近く高速化されることを示した。また、スキュー化分散型の割付法による静的負荷分散法を 2 次元の $(M\pi)^2$ に適用した場合の性能は、3 次元のシステムに分散型の割付法を適用した場合を上まわることがわかった。システムの構築のコストはより低次のシステム方が低いことを考えると、スキュー化分散型の割付法が有効であることがわかる。次に、キャッシュドフレームバッファシステムを考慮した $(M\pi)^2$ についての評価を行なった。キャッシュドフレームバッファシステムについては第 3 章において理論的な解析を行なっているが、まずは理論解析の妥当性を検証するため、輝度をランダムに発生する方法による離散時間イベントドリブン型のシミュレーションを行なった。その結果、理論解析と同様の傾向を得ることができ、モデルの妥当性が示された。さらに $(M\pi)^2$ の並列レイトレーシング法の実行駆動形式のシミュレータにキャッシュドフレームバッファシステムを組み込んでシミュレーションを行い、システムの振舞いを調べた。その結果 PE 数が最大の部分、つまり最も並列計

算機の潜在的な性能が高い部分でキャッシュドフレームバッファシステムによる性能向上があることがわかった。そこで今後の計算要素の性能向上や超並列システムの規模の増大をみこして計算要素の速度を恣意的に向上させたシミュレーションを行なった結果、キャッシュドフレームバッファシステムの効果がより顕著にみられた。従って、今後の超並列画像生成計算機の設計にはこのような階層的なフレームバッファを必要としたものになると思われる。

以上のことから本論文で提案したオブジェクト空間分割型並列計算モデルに基づくマルチパスレンダリング法の並列化手法は、レイトレーシング法やラジオシティ法などの大域照明問題一般を統一して並列化可能な手法であり、画像生成問題の高速化に有効な並列化手法であると言える。さらに本論文で提案した $(M\pi)^2$ とキャッシュドフレームバッファシステムは本論文の並列化手法を施した並列マルチパスレンダリング法を効率良く実行可能なシステムであると結論づけられると考えられる。

今後の課題としては

- より多くのシーンによる $(M\pi)^2$ の詳細な解析
- 並列マルチパスレンダリング法の商用計算機上への実装
- $(M\pi)^2$ の実装
- オブジェクト空間分割型並列計算モデルを光学現象以外の分野に適用する研究
- 3 次元空間を 2 次元へ投影する現在の表示装置とは異なるオブジェクト空間分割型並列計算モデルによる画像生成を直接行なう表示装置の研究

などがあげられる。

謝辞

本研究を進めるに当たり、終始懇切なる御指導、御鞭撻を賜わりました中村維男教授に心より感謝致します。そして本論文の審査に当り、有意義な御指導、御助言を賜わりました根元義章教授、静谷啓樹教授、小柳光正教授に心より感謝致します。また、本研究を進めるに当り、小林広明助教授におかれましては終始丁寧かつ細部にわたった御指導、御助言を賜わり、深く感謝致します。

スタンフォード大学の Michael J. Flynn 教授におかれましては本研究を進めるに当り、有意義な御助言を賜わり深く感謝の意を表します。

中村研究室の片平昌幸助手、沈紅助手におかれましては本論文をまとめるに当り、有意義な御指摘、御助言を賜わり深く感謝致します。

本研究に御協力頂きましたソニー株式会社の藤勇一郎氏、東北大学大学院修士課程在学中の前田隆之氏に深く感謝致します。また、日本 IBM 東京基礎研究所の大庭信之博士におかれましては多くの有意義な御助言を賜わり深く感謝致します。そして中村研究室の皆様には種々の面でご支援を頂きました。ここに厚く御礼申し上げます。

最後に父正美、母智子、妹美穂に感謝致します。

発表論文リスト

1. 学術論文

[電子情報通信学会英文論文誌]

Hiroaki Kobayashi, Hitoshi Yamauchi, Yuichiro Toh and Tadao Nakamura : "$(M\pi)^2$: A Hierarchical Parallel Processing System for the Multipass Rendering Method", IEICE Trans. on Info. & Syst. E79-D, No.8, pp.1055-1064 (1996)

第 2, 3 章, 4.2 節

[電子情報通信学会英文論文誌]

Hitoshi Yamauchi, Takayuki Maeda, Hiroaki Kobayashi and Tadao Nakamura : "The Object-Space Parallel Processing of the Multipass Rendering Method by the $(M\pi)^2$ with a Distributed-Frame Buffer System"

IEICE 投稿中

第 3 章, 第 4 章

2. 国際会議論文

[ISPAN '94]

Hiroaki Kobayashi, Hitoshi Yamauchi, Yuichiro Toh and Tadao Nakamura : "$(M\pi)^2$: A Hierarchical Parallel Processing System for a Global Illumination Model", Proc. of International Symposium on Parallel Architectures Algorithms and Networks, pp.157-164 (1994)

第 2 章, 3.3.2 節, 4.2 節

[IPPS '96]

Hiroaki Kobayashi, Hitoshi Yamauchi, Yuichiro Toh and Tadao Nakamura : "A Hierarchical Parallel Processing System for the Multipass-Rendering Method", Proc. of IEEE International Parallel Processing Symposium, pp.62-67 (1996)

第 2 章, 3.3 節, 4.1 節

[CGI '97]

Hiroaki Kobayashi, Takayuki Maeda, Hitoshi Yamauchi and Tadao Nakamura : "A Cached Frame Buffer System for Object-Space Parallel Processing Systems"

CGI '97 採録決定

3.4 節, 4.2 節

3. 研究会，ワークショップ論文

[情報処理学会全国大会]

山内 斉、礒田 隆男、小林 広明、中村 維男 : 画像生成の並列処理に関する一検討, 第 46 回情報処理学会全国大会, pp369-370 (1993)

第 2 章

[電子情報通信学会研究会]

前田 隆之、山内 斉、小林 広明、中村 維男 : フレームバッファキャッシュを持つ画像生成用超並列システム $(\mathrm{M}\pi)^2$ の性能評価, コンピュータシステム (CPSY) 研究会

3.4 節, 4.2 節

[電気通信学会秋季大会]

藤 勇一郎、山内 斉、小林 広明、中村 維男 : 並列画像生成システム $(M\pi)^2$ の性能評価 (1994)

2.4.2 節, 4.2 節

[情報処理学会東北支部研究会]

藤勇一郎、山内斉、小林広明、中村維男 : 分散レイトレーシング法の空間分割型並列処理による高速化, 情報処理学会東北支部研究会 (1995) 3.3, 3.4 節

[電気関係学会東北支部連合大会]

前田隆之、藤勇一郎、山内斉、小林広明、中村 維男 : $(M\pi)^2$ のレイトレーシングステージにおける性能評価, 平成 8 年度電気関係学会東北支部連合大会, pp.100 (1996) 2.4.2 節, 3.4.2 節

4. 一般講演論文

[第 6 回計算力学講演会]

山内 斉、小林 広明、中村 維男 : 超高速グラフィクス, 第 6 回計算力学講演会 (1993)

第 2 章

参考文献

[1] Kurt Akeley. Reality engine graphics. *Computer Graphics (Proc. SIGGRAPH)*, pp. 109–116, 1993.

[2] Fujimoto Akira, Takayuki Tanaka, and Kansei Iwata. Arts: Accelerated ray-tracing system. *IEEE CG & Applications*, pp. 148–157, 1986.

[3] D. Badouel, K. Bouatouch, and T. Priol. Distributing data and control for ray tracing in parallel. *IEEE CG & Application*, Vol. 14, No. 4, pp. 69–77, July 1994.

[4] Shenchang Eric Chen, Holly E. Rushmeier, Gavin Miller, and Douglass Turner. A progressive multi-pass method for global illumination. *Computer Graphics (SIGGRAPH '91 Proceedings)*, Vol. 25, No. 4, pp. 165–174, July 1991.

[5] Michael F. Cohen and Donald P. Greenberg. The hemi-cube: A radiosity solution for complex environments. *Proceedings of SIGGRAPH*, Vol. 19, No. 3, pp. 254–263, 1985.

[6] Michael F. Cohen and John R. Wallace. *Radiosity and Realistic Image Synthesis*. Academic Press Professional, 1993.

[7] Michael Cohen, Shenchang Eric Chen, John R. Wallace, and Donald P. Greenberg. A progressive refinement approach to fast radiosity image generation. *Computer Graphics (SIGGRAPH '88 Proceedings)*, Vol. 22, No. 4, pp. 75–84, Aug. 1988.

[8] H. Nishimura et al. Links-1 : A parallel pipelined multimicrocomputer system for image creation. *ACM Computer Architecture*, pp. 387–394, July 1983.

[9] Foley, van Dam, Feiner, and Hughes. *Computer Graphics Principles and Practice 2nd Edition*. Addison-Wesley publishing company, 1990.

[10] Andrew S. Glassner. Space subdivision for fast ray tracing. *IEEE Computer Graphics and Applications*, pp. 160–167, 1984.

[11] Pat Hanrahan, David Salzman, and Larry Aupperle. A rapid hierarchical radiosity algorithm. *Computer Graphics (SIGGRAPH '91 Proceedings)*, Vol. 25, No. 4, pp. 197–206, July 1991.

[12] Chandlee B. Harrell and Farhad Fouladi. Graphics rendering architecture for a high performance desktop workstation. *Computer Graphics (Proc. SIGGRAPH)*, pp. 93–100, 1993.

[13] James T. Kajiya. The rendering equation. *Computer Graphics (Proc. SIGGRAPHS)*, Vol. 20, No. 4, pp. 143–150, 1986.

[14] L. Kleinrock. *Queueing System, Volume I: Theory.* John Wiley & Sons, Inc., 1975.

[15] Hiroaki Kobayashi, Hitoshi Yamauchi, Yuichiro Toh, and Tadao Nakamura. $(M\pi)^2$: A hierarchical parallel processing system for a global illumination model. *International Symposium on Parallel Architectures Algorithms and Networks*, pp. 157–164, December 1994.

[16] Hiroaki Kobayashi, Hitoshi Yamauchi, Yuichiro Toh, and Tadao Nakamura. A hierarchical parallel processing system for the multipass-rendering method. *IEEE International Parallel Processing Symposium*, pp. 62–67, April 1996.

[17] Hiroaki Kobayashi, Hitoshi Yamauchi, Yuichiro Toh, and Tadao Nakamura. $(M\pi)^2$: A hierarchical parallel processing system for the multipass rendering method. *IEICE Trans. on Info. & Syst.*, Vol. E79-D, No. 8, pp. 1055–1064, August 1996.

[18] F. Thomson Leighton. *Introduction to PARALLEL ALGORITHMS AND ARCHITECTURES.* Morgan Kaufmann Publishers, 1992.

[19] M. H. MacDougall. *Simulation Computer Systems: Techniques and Tools.* The MIT Press, 1987.

[20] M.F.Deering, S.A.Schlapp, and M.G.Lavelle. FBRAM: A new form of memory optimized for 3d graphics. *Proc. of SIGGRAPH*, pp. 167–174, 1994.

[21] Derek Paddon and Alan Chalmers. Parallel processing of the radiosity method. *Computer-Aided Design*, Vol. 26, No. 12, pp. 917–927, December 1994.

[22] J. P. Singh, A. Gupta, and M. Levoy. Parallel visualization algorithms: Performance and architectural implications. *IEEE Computer*, pp. 45–55, July 1994.

[23] S.Nishimura and T.L.Kunii. VC-1: A scalable graphics computer with virtual local frame buffers. *Proc. of SIGGRAPH*, pp. 365–372, 1996.

[24] Whitted T. An improved illumination model for shaded display. *CACM*, Vol. 23, No. 6, pp. 343–349, 1980.

[25] John R. Wallace, Michael F. Cohen, and Donald P. Greenberg. A two-pass solution to the rendering equation: A synthesis of ray tracing and radiosity methods. *Computer Graphics (SIGGRAPH '87 Proceedings)*, Vol. 21, No. 4, pp. 311–320, July 1987.

[26] Masaharu YOSHIDA, Tadashi NARUSE, Tokiichiro TAKAHASHI, and Seiichiro NAITO. Design of graphics computer – sight –. 情報処理学会研究会報告, Vol. 85, No. 60-5, pp. 1–8, December 1985.

[27] 天野英晴. 並列コンピュータ. 昭晃堂, 1996.

[28] 小林広明. 3 次元図形生成のための並列処理機構に関する研究. PhD thesis, 東北大学, 1988.

[29] 山内斉, 小林広明, 中村維男. 超高速グラフィクス. 第 6 回計算力学講演会, 1993.

[30] 山内斉, 礒田隆男, 小林広明, 中村維男. 画像生成の並列処理に関する一検討. 第 46 回情報処理学会全国大会, 1993.

索 引