JAERI-Data/Code
2000-020

高並列可視化処理時における Zバッファ画像合成処理

2000年3月

金子　勇・村松一弘

日本原子力研究所
Japan Atomic Energy Research Institute

本レポートは，日本原子力研究所が不定期に公刊している研究報告書です。
入手の問合わせは，日本原子力研究所研究情報部研究情報課（〒319-1195　茨城県那珂郡東海村）あて，お申し越しください。なお，このほかに財団法人原子力弘済会資料センター（〒319-1195　茨城県那珂郡東海村日本原子力研究所内）で複写による実費頒布をおこなっております。

This report is issued irregularly.
Inquiries about availability of the reports should be addressed to Research Information Division, Department of Intellectual Resources, Japan Atomic Energy Research Institute, Tokai-mura, Naka-gun, Ibaraki-ken, 319-1195, Japan.

編集兼発行　　日本原子力研究所

JAERI-Data/Code 2000-020

高並列可視化処理時におけるZバッファ画像合成処理

日本原子力研究所計算科学技術推進センター
金子 勇 ・ 村松 一弘

(2000年2月9日 受理)

多数のプロセッサを有する高並列計算機上においては，領域分割による数値計算手法が用いられる．ここで，解析規模の増大してきている現状では，これらの解析結果の可視化を計算本体と同時に各プロセッサで同時に行う手法が現実的なものとなりつつある．特に，リアルタイムに計算結果を可視化するという，実時間可視化を行う際には，この手法は必須のものである．

そして，このような多数のプロセッサを用いて画像レンダリングを行う手法の場合，各プロセッサのレンダリング結果を最終的に一つの画像に合成する処理を行う必要があり，これは，Z バッファを用いた画像合成手法で処理が行われる．しかし，数十プロセッサを超える規模でこの Z バッファ画像合成処理を行うと，処理の低速化ならびに各プロセッサでのローカルメモリ不足という問題が発生する．

そこで本報告では，Zバッファ画像合成処理を演算子と考え，これがReduceオペレータという特殊な演算子であることを用いた並列化手法，ならびに背景情報の削除によるバッファ圧縮という，この問題に対する二つの解決法を新たに提案する．また実際に，並列計算機Paragonで評価を行った結果を示し，これらの手法の有効性を検証する．

日本原子力研究所：〒153-0061 東京都目黒区中目黒2-2-54

JAERI-Data/Code 2000-020

Z-Buffer Image Assembly Processing in High Parallel Visualization Processing

Isamu KANEKO and Kazuhiro MURAMATSU

Center for Promotion of Computational Science and Engineering
Japan Atomic Energy Research Institute
Nakameguro, Meguro-ku, Tokyo

(Received February 9, 2000)

On the platform of the parallel computer with many processors, the domain decomposition method is used as a popular means of parallel processing. In these days when the simulation scale becomes much larger and takes a lot of time, the simultaneous visualization processing with the actual computation is much more needed, and especially in case of a real-time visualization, the domain decomposition technique is indispensable.

In case of parallel rendering processing, the rendered results must be gathered to one processor to compose the integrated picture in the last stage. This integration is usually conducted by the method using Z-buffer values. This process, however, induces the crucial problems of much lower speed processing and local memory shortage in case of parallel processing exceeding more than several tens of processors.

In this report, the two new solutions are proposed. The one is the adoption of a special operator (Reduce operator) in the parallelization process, and the other is a buffer compression by deleting the background informations. This report includes the performance results of these new techniques to investigate their effect with use of the parallel computer Paragon.

Keywords: Z-buffer, Image Assembly, Image Rendering, Parallel, Real-time Visualization, Reduce Operation, Binary Tree, Buffer Compression, Background Information, Paragon

目 次

Contents

1 はじめに

近年の計算科学技術の発展には目覚ましいものがあり，これに対する可視化技術もより重要なものとなりつつある．ここで，この可視化において，現状では計算規模の増大により，全計算結果を保存後の可視化が困難になりつつある．解析の結果生成される計算結果も膨大なものとなり，磁気ディスクに納まりきらないのが現状である．

そこで，解析と同時に可視化を行う実時間可視化手法が，大規模解析における現実的な手法となりつつある．それは，解析に用いる格子が基本的に三次元であるのに対して，可視化結果は常に二次元画像であり，可視化処理の結果生成させる画像データの大きさは，解析規模にかかわらず常に同じ大きさとなるためである．この実時間可視化手法では，解析計算を行っている各プロセッサを，可視化のためのレンダリングプロセッサとしても用いることで，解析と可視化を同時に行う．

そしてこの解析と同時に可視化を行う可視化システムとして日本原子力研究所計算科学技術推進センターが開発しているシステムがPATRAS(実時間可視化システム)である[1]．これは，並列計算機上での解析結果を，ネットワークで接続されたクライアント上で解析と同時に可視化(トラッキング)し，可視化のための種々のパラメータをクライアントの GUI で制御(ステアリング) するものである．この PATRAS は，以下のような特徴を持っている．

- 可視化の図種としては，等高線，ベクトル図，等値面，流線およびトレーサをサポート
- サーバ側で生成された画像データは画像圧縮を施してクライントに転送されるため，ネットワークへの負荷が少なく，高速な実時間可視化が可能
- 機種に依存しない Web ベースの GUI であり，クライアント側では Web ブラウザを用いる
- ライブラリ形式を採用しているため，ユーザの並列解析コードに対して，少数のサブルーチンを呼び出すだけで利用可能である

ここで，この PATRAS は数千プロセッサを有する地球シミュレータ上でも用いる予定であり，現在そのために新たなシステムを開発中であるが，この地球シミュレータのような数千プロセッサ下での PATRAS 使用では，従来，問題とならなかった画像合成処理部において，処理時間やメモリ使用量が問題となる．つまり，複数プロセッサを用いた領域分割による解析結果の可視化の際に，可視化モジュール側も領域分割でレンダリングを行う必要があるが，複数で並列レンダリングした画像を3次元的な重ね合わせを考慮して画像合成処理を行う必要があり，この画像合成処理を行うZバッファ画像合成処理が，レンダリングの際の使用プロセッサ数が増大した際に問題となる．

そのため本報告では，この問題に関する考察と具体的な解決策を示す．また，この手法に関して実際に日本原子力研究所関西研究所の並列計算機 Paragon により測定を行った検証結果を報告する．

2 並列Zバッファ処理の概要とその問題点

2.1 画像バッファとZバッファ

二次元画像を複数のプロセッサで描画(レンダリング)する場合は，画像領域を領域分割し，それぞれのプロセッサで描画した後に単純に合成を行えば良い．それに対して，三次元格子などを用いて行われる科学技術計算の可視化においては，生成させる画像が三次元的な画像であり，隠面処理の問題により，二次元画像のように各プロセッサでレンダリングされた画像を単純に組み合わせただけでは最終的な画像を得ることはできない(図1)．

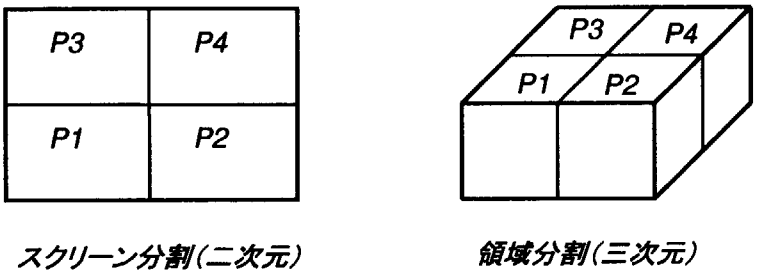


図1 分散レンダリング時における，スクリーン分割と領域分割

そのために，PATRAS などの複数プロセッサを用いて画像を生成する際に採用されている手法が，画像バッファ(各画素に対応したメモリ)として，従来のカラーバッファ以外にZバッファを用いることである．

現在の CG 方式の主流であるビットマップ方式のコンピュータグラフィックスにおいては，スクリーンの各画素それぞれにメモリを確保し，色コードを割り当てる．なお，本報告ではこの色コードデータを保持するメモリ領域のことをカラーバッファと呼ぶこととする．ここで，このカラーバッファに加えて，スクリーンの各画素にスクリーン奥行き方向の値(Z 値)を保持する画像バッファが Z バッファである(図 2)．

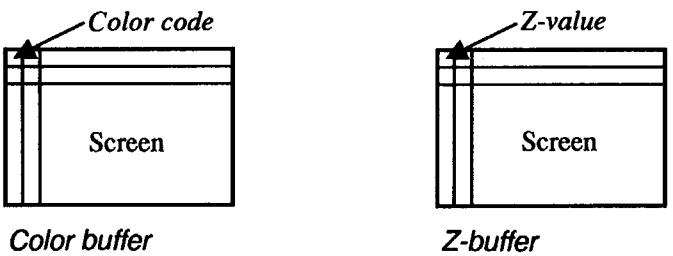


図2 カラーバッファとZ-バッファ

Zバッファはピクセル単位の隠面処理を行う際に使用され，各ピクセルにレンダリングする際に先立ってそのピクセルに対応するZ値を求め，従来のZ値よりも小さい場合(＝前方にある場合)のみレンダリングを行うことで，ピクセル単位の隠面処理を行う(図 3)．

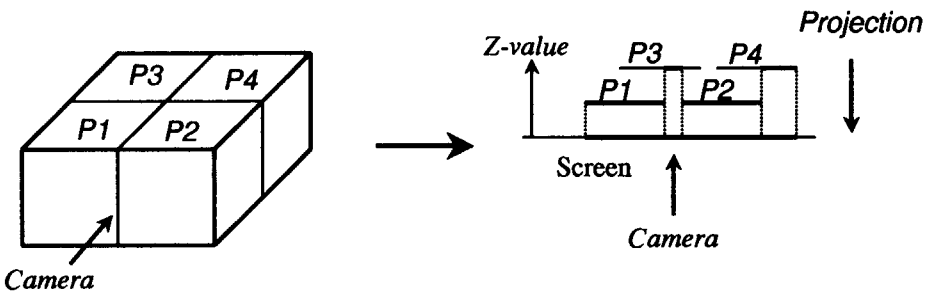


図 3 Zバッファによる隠面処理

このZバッファと従来の画像バッファであるカラーバッファを各プロセッサに用意し，各プロセッサで分散レンダリングを行った後に，各ピクセルのZ値を比較し，最もZ値の小さいカラー値を採用する．この手順により，複数プロセッサで三次元画像を分散レンダリング後，一つの画像に合成する処理が可能となる．

2.2 従来の処理方法とその問題点

従来のPATRASでの分散レンダリングにおいては，画像合成専用のプロセッサを用意し，そこに全レンダリングプロセッサのレンダリング結果を集約し，全データが揃った時点で，各ピクセルに最小となるZ値を持つカラー値を検索し，最終画像を合成するという手法を採用していた(図 4)．

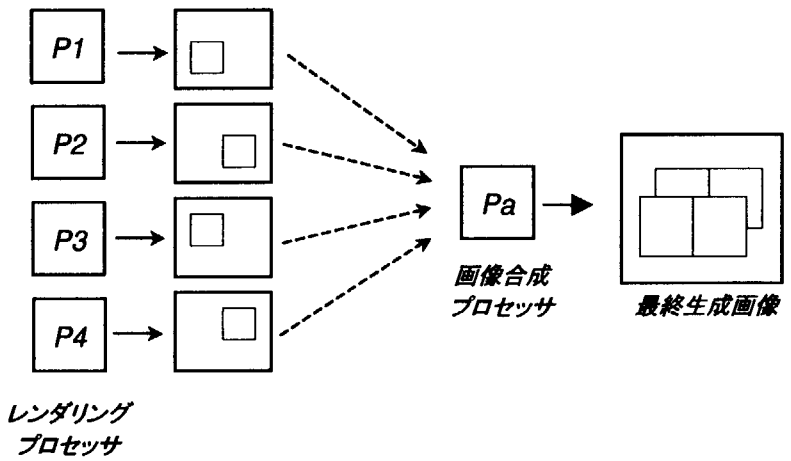


図 4 画像合成プロセッサを用いたZバッファ画像合成処理

しかし，レンダリングプロセッサが数プロセッサから数十プロセッサであればこの方式でも問題はないが，レンダリングプロセッサ数が増加した場合に，この方式では，消費メモリや処理時間の点で問題が発生する．

まず，各レンダリングプロセッサで処理，および，レンダリングプロセッサ間で転送されるデータは色データとZ値データの組である．これを1ピクセル毎に色情報として32ビット，Z値情報として64ビットと仮定すると，レンダリング画像の解像度が 512×512 時に各レンダリングプロセッサに 約3Mbyte の画像バッファが必要となる．そしてこれを合成するプロセッサでは，各レンダリングプロセッサそれぞれの画像を保持する必要上，レンダリングプロセッサ数×3Mbyte のローカルメモリが必要となり，これだけのデータを1プロセッサに集約させるためのデータ転送処理も問題となる．

そのため，レンダリングの並列度が増大すると，このZバッファによる画像合成処理に関しても分散・並列処理で行う必要性が生じる．

2.3 本報告で示す改良案

本報告では，この高並列レンダリング時における Z バッファ画像合成処理に対して次の二つの提案を行う．

- Zバッファ処理をオペレータ演算とみなし分散処理を行う手法
- カラーバッファとZバッファを圧縮し，各プロセッサ間のデータ転送量を削減する手法

以下，3章でReduceオペレータ処理による分散処理手法を，4章でバッファ圧縮手法を示す．

3 Reduce 処理方式による分散 Z バッファ合成処理

Z バッファ画像合成処理を分散処理する場合，レンダリングプロセッサを二分木構造に配置し，画像を順に合成すれば良いことは直感的に分かる．しかし，より厳密には，この Z バッファ処理というものが，Reduce オペレータという特殊な演算に分類されるために，二分木構造で処理を行うことが可能になっていることを指摘することができる．この並列 Z バッファ合成処理が Reduce オペレータであるということは，本報告で初めて指摘される事柄である．

3.1 Reduce オペレータ

各ピクセルに対する Z バッファ処理をオペレータ [Z] とみなすと，Z バッファによる二つのバッファ(B1 と B2 間)をバッファ B0 に合成する処理を次の様に記述することができる．

$$B0(C0, Z0) = B1\,(C1, Z1)\ [Z]\ B2\,(C2, Z2) \qquad \text{C:色情報 Z:Z 値情報}$$

ここで，オペレータ[Z]は

$$\text{If}\quad Z1 < Z2\quad \text{then}\quad B0 = B1\quad \text{else}\quad B0 = B2$$

となる演算子である．

そして，このオペレータ[Z]が Reduce オペレータであることが指摘できる．Reduce オペレータとは，交換則・結合則を満たすオペレータであり，代表的なものとして，加算や積算，最大・最小値演算などである．

(交換則) $A+B = B+A$

(結合則) $(A+B)+C = A+(B+C)$

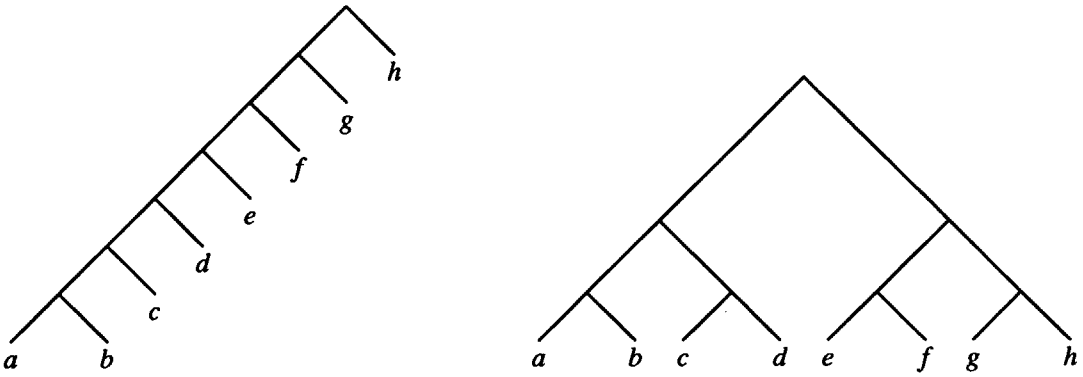
Z バッファ合成処理において，同一スクリーン上における各レンダリングプロセッサが生成した画像情報それぞれの処理に関しても，交換則，結合則を満たす．すなわち，R1, R2, R3 という3つのレンダリングプロセッサによるレンダリング画像を生成後，R1 と R2 の画像情報を合成した後に，それと R3 の画像情報と合成しても，R2 と R3 の合成結果と R1 との合成処理を行っても同じ結果となる．また，R1 と R2 の合成結果と，R2 と R1 の合成結果も同様である．このことから，Z バッファ処理をオペレータであると考えると，Reduce オペレータという特殊な演算子に分類できることが分かる．

そして，この Reduce オペレータの並列処理に関しては，既に多くの効率的な処理方式が提案されており，Z バッファ処理に対しても，この Reduce オペレータの並列処理に準じることで，効率の良い処理が可能となる[2].

3.2 Reduce オペレータの並列処理

一般的にオペレータの逐次処理ではn個の要素があった場合に *n-1* 回の演算が必要である. これに対して, Reduce オペレータでは結合則が成り立つため, 演算順番を最適化することで, 演算回数は *n-1* 回でも, 並列処理においては $\log_2 n$ ステップで処理が可能である.

Reduce オペレータの例として加算を挙げる. 結合則が成り立つため *(((a+b)+c)+d) = ((a+b)+(c+d))* であり, 並列処理であれば演算順番を交換することで, *(a+b)*と*(c+d)*の演算は並列に行うことができる(図5). そのため, 並列処理における最終的な処理時間は図5に示したような各ツリーにおけるツリーのレベル(階層)と比例する. これにより, ツリーのレベル(階層)が最小となるように演算順番を並列処理に向いた完全二分木構造へ変更することで, 並列計算における Reduce オペレータ処理時間を最短とすることができる.



逐次処理(ツリーレベル 7) *並列処理(ツリーレベル 3)*

図5 Reduce オペレータの逐次処理と並列処理

3.3 MPI による実装方針

実際に Z バッファ画像合成処理がどの程度問題となり, 本報告の提案でどの程度解決できるのかをサンプルの測定結果で示す. なお, 本報告では分散システム API として MPI を仮定している.

ここで, 前節で示したように, 二分木を各プロセッサ間のメッセージ送信で実現することも可能であるが, より効率の良い, 各システム最適な手法とするためには, 初めからReduceオペレータとして処理を行うべきであることを指摘できる.

MPI には, Reduce オペレータ用関数 MPI_Reduce があり, 代表的な Reduce オペレータである加算や総和処理などに対応している. そして, この Reduce オペレータに対して, ユーザ定義の

Reduce オペレータというものを作成することが可能であり，Z バッファ処理専用の新たな Reduce オペレータを定義することができる．この実装方法により，各マシンのアーキテクチャに合わせた最適化された Z バッファ処理を行うことができる．

ユーザ定義の Reduce オペレータが MPI によってどの様に処理されるかは MPI 実装に依存するので，一概には言うことができないが，一般的には前節で書いた完全二分木構造へのプロセッサ配置による最適化を期待できる．

3.4 サンプルによる実測結果

今回，画像合成部の問題点をより明確にするために，この Z バッファ画像合成をメインとするサンプルプログラムを製作し，これにより実測を行った．Z バッファ処理測定目的のため，レンダリング部分には，簡易的で高速な処理方式を採用し，レンダリングに関する処理負荷がほとんど発生しないようにしている．測定環境としては，日本原子力研究所関西研究所の Paragon(800 Processor)を用いた．測定条件は以下の通りである．

- テクスチャマップされた球体を並列レンダリングするサンプル例を使用
- 画像合成に要する時間を 100 ステップ分の平均値として測定
- 処理画素数は 128x128, 256x256, 512x512

この実測結果を図 6 に示す．このグラフでは，Z バッファ画像合成処理に要した各プロセッサの平均時間を示している．なお，図 6 中の gather が Z バッファを1プロセッサで集約処理したものであり，reduce が画像バッファ合成を分散処理したものである．ここで，横軸はレンダリングに用いたプロセッサ数であり，縦軸は画像合成処理に要した時間(レンダリングに要する時間は除く)である．また，メモリ不足により測定が行えなかった個所は値を省略している．

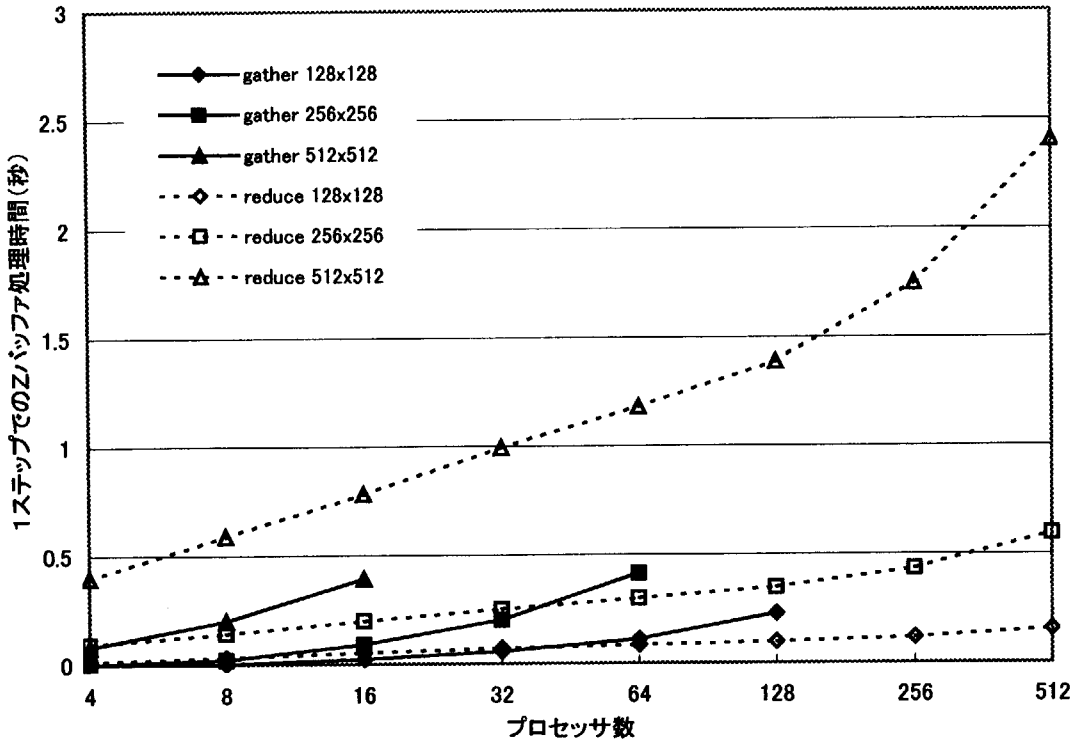


図6 従来手法とReduce手法によるZバッファ処理時間の比較

3.5 Reduce処理方式に関する考察

Zバッファ処理時間はgatherが*O(n)*, reduceが*O(log2n)*であり, この測定結果より40プロセッサを越えるとreduce処理の方が高速である. プロセッサ数が少ない場合に並列合成処理の方が集中合成処理より時間がかかっているのは, MPI_Gatherよりユーザ定義のMPI_Reduceの方が処理負荷が高いためである.

ここで, 従来の分散処理を行わない手法では, 画像合成プロセッサでローカルメモリが不足するため, 使用プロセッサ数が百程度となると, 測定が行えなかった(Paragonのローカルメモリは128Mである). しかし, 使用メモリ量に関しても分散が行われるreduce処理ではこの問題は発生しない. また, もしローカルメモリが充分であったとしても, 従来のgather方式では, 画像合成プロセッサでプロセッサ数に比例した処理時間が必要であり, 従来の手法では百プロセッサを超える並列処理時に処理時間が無視できなくなる.

この様に, 従来方式では, 40プロセッサ程度までは高速であるが, 百プロセッサを超えた場合ローカルメモリ不足という問題が発生し, また同時に処理時間が問題となるが, 本報告のReduce処理方式であれば, 数百プロセッサを越えてもローカルメモリの消費量に関して問題が発生せず, 処理速度の点でも問題にならないことが分かる.

4 バッファ圧縮による高速化

3章では，Zバッファ処理をReduceオペレータとみなすことによる並列処理方式を示したが，この手法でも512プロセッサ程度のプロセッサ数となると画像合成処理のみで3秒以上の処理時間が必要となる．

ここで，この処理時間は主にデータの転送時間である．画像合成処理自身は単純なピクセル単位でのZ値比較と置換であり，プロセッサ数が増加しても処理量に変化はないが，プロセッサ間のデータ転送に関しては，全プロセッサで3Mbyte以上のバッファを有し，これらの全プロセッサ間の転送が行われるため，プロセッサ数が増えるほど問題となる．そこで，この各プロセッサ間で転送される転送データの削減と，その効果に関しても考察を行った．

4.1 画像バッファ圧縮

転送データ量を削減するためには，画像バッファ，すなわち，カラーバッファとZバッファを削減すれば良い．ここで，多数のプロセッサで領域分割レンダリングされる画像の特徴として，背景の占める領域が大きいということが挙げられる．画像バッファは，各ピクセル毎のデータの集合であるため，何もレンダリングしていない個所(＝背景部)でも，レンダリングを行った個所と同じだけのメモリを消費する．同様，分散レンダリングの際にも，この背景領域がそのまま他プロセッサに転送されている．

そして，この背景領域は何もレンダリングされていない領域であり，これを省略することで，転送データ量を大幅に減少させることが可能である．この背景領域の削除手法としては，次の方式が存在する．

- 矩形による描画領域切り出し
- ブロック化手法
- マスク情報の使用
- 画像圧縮技術の応用

ここで，Zバッファを含んだ画像バッファにおける背景削除の場合，画像圧縮系の技術により，背景領域の削除を容易に行うことができる．そこで本報告では，Zバッファに対するランレングスバッファ圧縮手法を新たに提案する．

なお，ランレングス画像圧縮とは，画像圧縮の最も基本となる手法であり，画像の横方向をスキャンし，同じ値を有する連続した画素についてはその個数を記録することで圧縮する手法である．ここで，通常のランレングス画像圧縮はZバッファを有しないカラーバッファのみ有する画像バッファに対して適用されるが，本報告での手法では，カラー値ではなく，Z値を参照することで，カラーバ

ッファとZバッファ両方に対してランレングス圧縮を行う.

4.2 Zバッファを含んだ画像バッファのランレングス圧縮による背景削除

背景情報を削除するには, まず画像バッファにおける背景領域の検出を行う必要があるが, 本報告の手法では, Zバッファを参照することで, 情報圧縮と同時に背景検出を実現する.

透視変換とZバッファを用いる一般的な3DCG手法においては, 台形領域によるクリッピングボリュームが利用される. クリッピングボリュームとは, 視野領域におけるレンダリング範囲を制限する領域であり, この領域に含まれるオブジェクトのみがレンダリングの対象となる.

このクリッピングボリュームにおいて, スクリーン座標系におけるZ値をZバッファに格納する都合上, クリップ領域を Z 方向(スクリーン垂直方向)にも設定する. スクリーンに近い方のクリップ面がnear clipであり, 最遠面がfar clipである. 各ピクセルのZ値はこの間に丸められる. そのため, 特定のピクセルが背景かどうかは, このZ値の値をfar clip 値と比較することで判定することができる(図7).

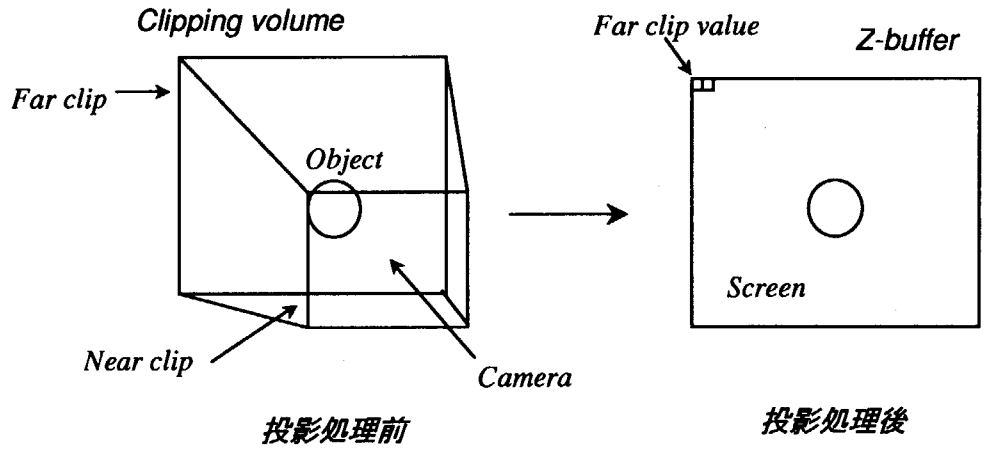


図7 Clipping Volumeと Far Clip

この背景検出法とZ値に対するランレングス圧縮で, 背景を削除する手法を図8に示す. Zバッファへのランレングス圧縮の適用により, Z 値>=Z_{far}となっている画素数を水平方向へカウントし, 背景ピクセルの場合, カラーバッファをランレングス個数バッファとして利用する. これにより, 圧縮率最低の場合(背景領域が存在しない場合)でも, 圧縮処理後データ量が増えることはない. また, この手法の場合, 高速に圧縮・展開が可能であり, 画像合成処理時に同時に圧縮展開処理を行うこともできる.

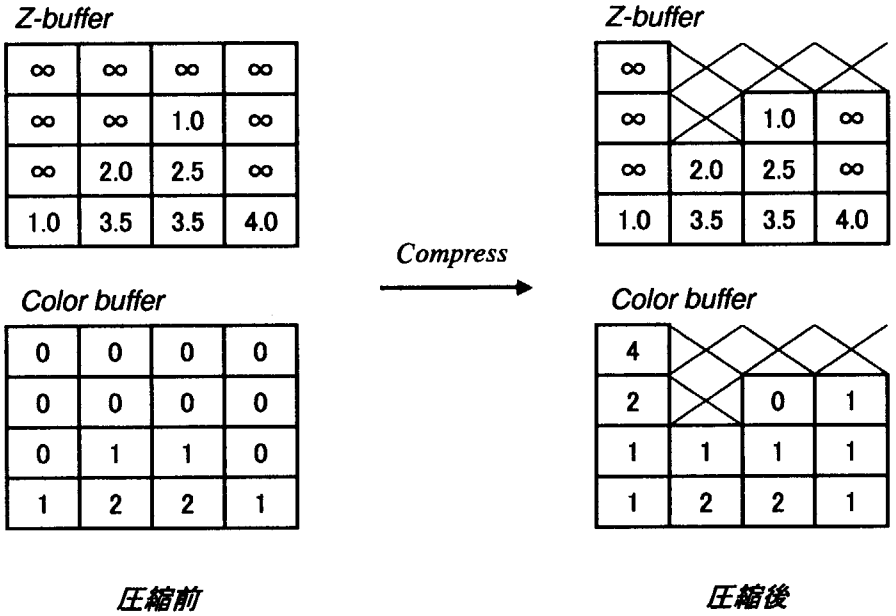


図 8 Z バッファを含んだ画像バッファに対するランレングス圧縮

4.3 バッファ圧縮を考慮した画像合成処理

前節で示したバッファ圧縮手法の効果を検証するために，サンプルを作成し実際に測定を行った．ここで，実装は 3 章のサンプル同様，測定環境として Paragon を用い，並列化ライブラリとして MPI を用いたが，MPI の Reduce オペレータ用関数(MPI_Reduce)では，可変長サイズのデータは扱えないため，MPI のノード間通信による論理的な二分木で Reduce オペレータを擬似的に処理することとした．そのため，プロセッサ配置に関しては，最適な処理が行われない可能性がある．

ここで，Reduce オペレータではなく二分木で処理するとどの程度の処理時間となるのかも合わせて示すために，圧縮処理を行わず，二分木で処理する方式の性能評価も合わせて行った．このノード間通信による二分木と圧縮処理関連の処理内容を図 9 に示す．

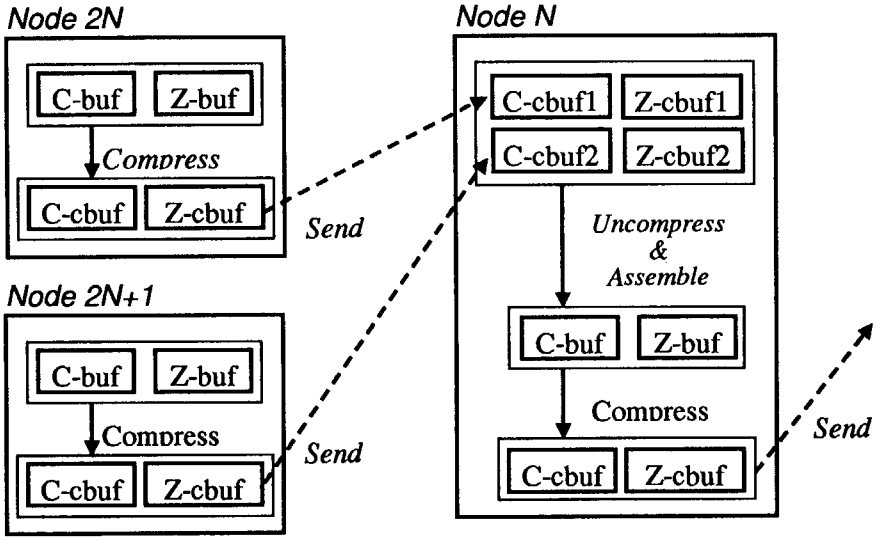


図 9 バッファ圧縮を考慮した画像合成処理

処理手順は以下の通りである.

1. 各レンダリングプロセッサを論理的な二分木構成に配置する
2. 各レンダリングプロセッサにおいて, レンダリングを行う
3. 二分木の葉に対応する各レンダリングプロセッサでレンダリング結果を圧縮する
4. 圧縮バッファを二分木の親プロセッサに送信
5. 受信した親側のプロセッサでは, 子プロセッサからの二つの圧縮受信データに対して, 展開と同時に Z バッファ画像合成処理を行う
6. 合成画像を再び圧縮し二分木の親プロセッサに送信
7. 再帰的にこれを繰り返すことで, 二分木の根プロセッサに最終的な画像が合成される

4.4 実測結果

3.4 節同様, バッファ圧縮検証のためのサンプルプログラムを製作し, これに関する性能評価を行った. 測定条件は以下の通りである.

- 3 章同様, テクスチャマップされた球体を並列レンダリングするサンプル例を使用
- 日本原子力研究所関西研の Paragon を測定に使用
- 1 ステップでの平均処理時間を測定(100 サンプル平均)

・ 画素数は 512x512

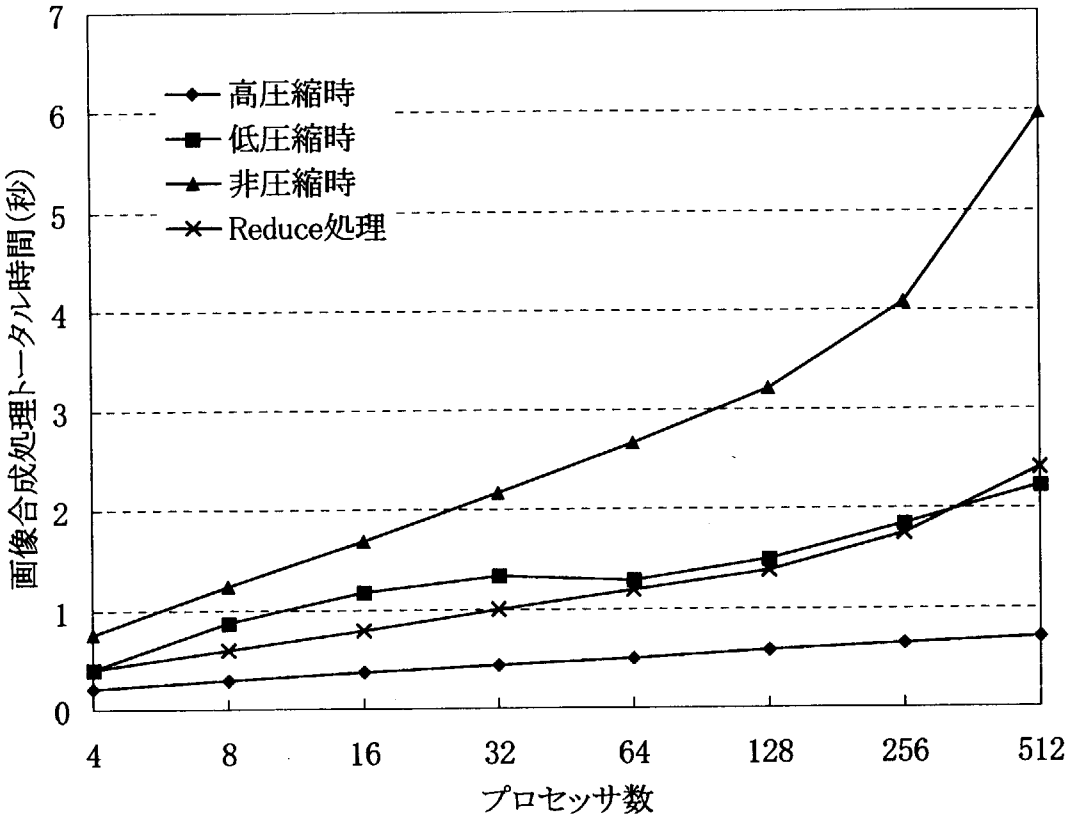


図 10 バッファ圧縮に関する測定結果

図 10 にその結果を示す. 図中の高圧縮率時とは, 画面小領域への描画を行った場合で, 背景面積が大で, 4 プロセッサ分散レンダリング時のバッファ圧縮率が 98.7%の場合である. 低圧縮率時とは, スクリーン全体に描画されるような, 背景削除の効果が少ない場合であり, 4 プロセッサレンダリング時のバッファ圧縮率が 40.7%である. また, 全プロセッサで背景が存在しないという, 最悪な条件として, 非圧縮時(圧縮 OFF 時)も示した. この非圧縮時は, 3 章の Reduce 処理を, 明示的なプロセス間通信で擬似的に処理した場合に相当する. また, 3 章の MPI_Reduce 処理結果(圧縮処理なし)も図中に示した.

4.5 バッファ圧縮に関する考察

領域分割によるレンダリングでは, 背景削除によるバッファ圧縮が有効的に働くことがこの結果より分かる. 1 プロセッサ毎の画像圧縮処理時間は多少長くなる(実測で約 2 倍)が, レンダリングの並列度が増加すると, 各レンダリングプロセッサがレンダリングする画素領域が小さくなるため, この処

理時間も短縮される.

ここで, 非圧縮時は, 単純な Reduce 処理時よりも低速となるが, 一般的な分散レンダリング時ではこのようになることはまず考えられず, 今回の低圧縮時の例が, 最も処理時間のかかる場合に相当する. そのため, ほとんどの場合, 単純な Reduce 処理よりも, 背景領域圧縮処理を行ったほうが高速に画像合成処理を行うことができる.

ここで, 非圧縮(論理的な二分木による画像合成処理)時に Reduce 処理より低速な理由は, 論理的な二分木で処理しているため, ノード配置が物理的な二分木構造に最適化されていないためと, Reduce では画像合成とデータ受信が同時に処理されるためである.

この実測に用いた Paragon は二次元メッシュ構成をとる並列機でありプロセッサ間の転送速度には送受信のプロセッサの組み合わせにより差が生じる. これに関して, MPI_Reduce を用いた場合, アーキテクチャに応じた最適化が期待できる.

また, 自明的にバッファ合成と転送処理を行っている場合, Reduce 処理と異なり, バッファ合成処理終了後に転送が行われる(MPI_Reduce では同時に行われる). MPI_Reduce のユーザ定義関数の形で Z バッファリング処理を記述した場合, 各ピクセル単位で, 画像バッファの受信の時点でZバッファリング処理が行われる. そのため, 合成処理後, 交互に転送する方式よりも高速となる. この様に, Zバッファ処理を Reduce オペレータで実装する事によるメリットも多数存在する.

なお, Reduce 処理とバッファ圧縮処理を組み合わせることが最も理想的であるが, MPI の Reduce 処理関数は転送データ量が固定であるため, この方式を実装することは今回できなかった. 全プロセッサで共通の背景領域を先立って検出し, その部分を削除した後に Reduce 処理を行うという手法も考えられるが, 高並列時に背景領域が増加するという特徴を用いたバッファ圧縮手法なので, この方式では, それほど圧縮率は良くならず, 共通部分の検出と全レンダリングプロセッサへの転送処理を補うほどの高速性は得られない.

今回の測定結果から, 多くの場合, バッファ圧縮と二分木による擬似 Reduce オペレーション処理が高速であるが, 最悪の場合を考慮すると, 単純に Reduce で処理した方が高速な場合もあり, 各アーキテクチャに応じて使い分けるのが最適であろうとの結論を得ることができる.

プロセッサ数として百を越える計算機として今回は Paragon のみが利用可能であったため具体的な測定は行えなかったが, プロセッサの配置として二次元メッシュやトーラス構造を採用した高並列機では Reduce 処理が, クロスバ方式に基づく高並列機では, 二分木による擬似 Reduce オペレーション処理が有利であろうとの推測がなりたつ. また, 数十プロセッサ以下の並列度であれば, 従来の PATRAS で用いていたような, 特定のプロセッサを画像合成専用プロセッサとして用い集中処理する方式が最も高速に処理できるものと推測できる.

5 PATRAS への適用

前章までの結果を踏まえ, PATRAS における画像合成部を実際に本提案の方式に変更し駆動を行った. これを前述の Paragon で駆動し, 数百プロセッサ下でも問題なく分散レンダリングが可能

となることを確認した．従来の手法では主にレンダリングプロセッサでのメモリ不足とデータ転送の集中により問題が生じていた部分であり，PATRAS の他の個所はプロセッサ数が多くなるほど逆に負荷は軽くなる．そのため，高並列駆動時に問題となるのは画像合成部のみであった．これにより，数百プロセッサからなる高並列機でも PATRAS を問題なく稼動できるようになった．

また，本報告で取り上げた問題は，実時間可視化に限らず，数百プロセッサ以上で分散レンダリングする際には必須となる事柄であり，ポストプロセッシングとして可視化を行った場合にも同じ議論が当てはまる．その点で，実際に Paragon により数百プロセッサで分散レンダリングが可能であることを確認できたことは，将来に向けての予備検証としての重要な意味があるといえる．

6 他研究との比較

本報告のベースとなっている PATRAS[1]において，Zバッファ画像合成処理自身は既に並列に処理が行われてきていた．このZバッファ画像合成処理自身は，はじめにで書いたように領域分割で三次元画像をレンダリグする際には必要となる処理であり，他研究でも使われている手法である[3][4][5][6].

しかし本報告は，百プロセッサを超える場合にこのZバッファ画像合成処理自身が問題となることを新たに示し，これに対する解決策を提案するものである．ここで，百プロセッサを超えるプロセッサを用いてレンダリングをする必要性は現在のところそれほど見当たらず，PATRAS を地球シミュレータ上で駆動した場合に問題になるであろうとの観点から行われた研究である．ここで取り上げた処理方式自体は一般的なものであるが，問題の提案自体に新規性がある．

ここで，文献[3]ではレンダリングを高速に行うという目的で百プロセッサを越える並列レンダリングを行っているが，本研究が仮定している任意オブジェクトの隠面処理とは異なり，ポリゴンレンダリングに特化した手法であり，隠面アルゴリズムとしてスキャンライン法（ポリゴンを走査線毎に水平に分割する手法）を用いている．また，画像バッファを全てのプロセッサでスクリーン全体分保持するのではなく，画像バッファ自体もスクリーン領域分割しており，他のプロセッサが保持する画像バッファへのレンダリングの際には，スキャンライン情報を他のプロセッサにノード間通信で送信するという手法を採用している．そのため，本報告のような大量のZバッファリング処理が必要とはならず，この例では画像合成処理部が特に問題とはなっていない．

また，本報告ではボリュームレンダリングのような半透明処理は考慮していないが，この半透明を用いた場合の分散レンダリングに関する画像合成については，また別の特有の問題が存在し，本報告の例よりもプロセッサ数が少ない場合でも問題となる．そのため，これに関する研究も既に行われているが，こちらはZバッファを用いる手法ではないため本報告とは異なるものである．

7 まとめ

現PATRASの問題点である画像合成処理部の並列化を行った．Zバッファを用いた画像合成処

理を Reduce オペレータと考えることで対処し，他にバッファの圧縮による高速化を考慮した．その結果，プロセッサ数が増加した際に，従来方式より高速になること，画像合成プロセッサでのメモリ消費量増大が解消されることを確認した．

最終的に本報告で新たに提案されたことは，Z バッファ画像合成処理が Reduce オペレータであることを利用した並列化手法と Z バッファに対するランレングス圧縮による背景削除という手法である．今後は更なる高速化技法の検証を行う予定である．

参考文献

[1] 村松一弘，他：並列計算機上での流体解析のための実時間可視化システム，計算工学講演会論文集，Vol.2, No.1, pp.109-112 (1997).

[2] 梅尾博司：超並列計算機アーキテクチャとそのアルゴリズム，共立出版 (1991).

[3] Thomas W. Crockett : Design Consideration for Parallel Graphics Libraries, *NASA Constructor Report 194935* (1994).

[4] Ned Greene, Michael Kass, Gavin Miller : Hierarchical Z-Buffer Visibility, *SIGGRAPH Conference (20th)*, pp. 231-238 (1993).

[5] Michael E. Palmer and Stephen Taylor : Rotation Invariant Partition for Concurrent Scientific Visualization, *Proc. Parallel CFD'94*, North-Holland, pp. 409-416 (1995).

[6] John E. Dorband : Scan Line Graphics Generation on the Massively Parallel Processor, *Proc. 2nd Symposium Front Massively Parallel Compute*, pp. 327-329 (1988).

国際単位系（SI）と換算表

表1　SI基本単位および補助単位

量	名称	記号
長さ	メートル	m
質量	キログラム	kg
時間	秒	s
電流	アンペア	A
熱力学温度	ケルビン	K
物質量	モル	mol
光度	カンデラ	cd
平面角	ラジアン	rad
立体角	ステラジアン	sr

表2　SIと併用される単位

名称	記号
分，時，日	min，h，d
度，分，秒	°，′，″
リットル	l，L
トン	t
電子ボルト	eV
原子質量単位	u

1 eV=1.60218×10^{-19}J

1 u=1.66054×10^{-27}kg

表3　固有の名称をもつSI組立単位

量	名称	記号	他のSI単位による表現
周波数	ヘルツ	Hz	s^{-1}
力	ニュートン	N	m·kg/s^2
圧力，応力	パスカル	Pa	N/m^2
エネルギー，仕事，熱量	ジュール	J	N·m
工率，放射束	ワット	W	J/s
電気量，電荷	クーロン	C	A·s
電位，電圧，起電力	ボルト	V	W/A
静電容量	ファラド	F	C/V
電気抵抗	オーム	Ω	V/A
コンダクタンス	ジーメンス	S	A/V
磁束	ウェーバ	Wb	V·s
磁束密度	テスラ	T	Wb/m^2
インダクタンス	ヘンリー	H	Wb/A
セルシウス温度	セルシウス度	℃	
光束	ルーメン	lm	cd·sr
照度	ルクス	lx	lm/m^2
放射能	ベクレル	Bq	s^{-1}
吸収線量	グレイ	Gy	J/kg
線量等量	シーベルト	Sv	J/kg

表4　SIと共に暫定的に維持される単位

名称	記号
オングストローム	Å
バーン	b
バール	bar
ガル	Gal
キュリー	Ci
レントゲン	R
ラド	rad
レム	rem

1 Å=0.1nm=10^{-10}m

1 b=100fm^2=10^{-28}m^2

1 bar=0.1MPa=10^{5}Pa

1 Gal=1cm/s^2=10^{-2}m/s^2

1 Ci=3.7×10^{10}Bq

1 R=2.58×10^{-4}C/kg

1 rad=1cGy=10^{-2}Gy

1 rem=1cSv=10^{-2}Sv

表5　SI接頭語

倍数	接頭語	記号
10^{18}	エクサ	E
10^{15}	ペタ	P
10^{12}	テラ	T
10^9	ギガ	G
10^6	メガ	M
10^3	キロ	k
10^2	ヘクト	h
10^1	デカ	da
10^{-1}	デシ	d
10^{-2}	センチ	c
10^{-3}	ミリ	m
10^{-6}	マイクロ	μ
10^{-9}	ナノ	n
10^{-12}	ピコ	p
10^{-15}	フェムト	f
10^{-18}	アト	a

（注）

1．表1－5は「国際単位系」第5版，国際度量衡局1985年刊行による。ただし，1 eVおよび1 uの値はCODATAの1986年推奨値によった。

2．表4には海里，ノット，アール，ヘクタールも含まれているが日常の単位なのでここでは省略した。

3．barは，JISでは流体の圧力を表わす場合に限り表2のカテゴリーに分類されている。

4．EC閣僚理事会指令ではbar，barnおよび「血圧の単位」mmHgを表2のカテゴリーに入れている。

換算表

力	N(=10^{5}dyn)	kgf	lbf
	1	0.101972	0.224809
	9.80665	1	2.20462
	4.44822	0.453592	1

粘度　1 Pa·s(N·s/m^2)=10 P(ポアズ)(g/(cm·s))

動粘度　1m^2/s=10^{4}St(ストークス)(cm^2/s)

圧力	MPa(=10bar)	kgf/cm^2	atm	mmHg(Torr)	lbf/in^2(psi)
	1	10.1972	9.86923	7.50062×10^3	145.038
	0.0980665	1	0.967841	735.559	14.2233
	0.101325	1.03323	1	760	14.6959
	1.33322×10^{-4}	1.35951×10^{-3}	1.31579×10^{-3}	1	1.93368×10^{-2}
	6.89476×10^{-3}	7.03070×10^{-2}	6.80460×10^{-2}	51.7149	1

エネルギー・仕事・熱量	J(=10^7 erg)	kgf·m	kW·h	cal(計量法)	Btu	ft·lbf	eV
	1	0.101972	2.77778×10^{-7}	0.238889	9.47813×10^{-4}	0.737562	6.24150×10^{18}
	9.80665	1	2.72407×10^{-6}	2.34270	9.29487×10^{-3}	7.23301	6.12082×10^{19}
	3.6×10^6	3.67098×10^5	1	8.59999×10^5	3412.13	2.65522×10^6	2.24694×10^{25}
	4.18605	0.426858	1.16279×10^{-6}	1	3.96759×10^{-3}	3.08747	2.61272×10^{19}
	1055.06	107.586	2.93072×10^{-4}	252.042	1	778.172	6.58515×10^{21}
	1.35582	0.138255	3.76616×10^{-7}	0.323890	1.28506×10^{-3}	1	8.46233×10^{18}
	1.60218×10^{-19}	1.63377×10^{-20}	4.45050×10^{-26}	3.82743×10^{-20}	1.51857×10^{-22}	1.18171×10^{-19}	1

1 cal=4.18605J（計量法）

=4.184J（熱化学）

=4.1855J（15℃）

=4.1868J（国際蒸気表）

仕事率　1 PS(仏馬力)

=75 kgf·m/s

=735.499W

放射能	Bq	Ci
	1	2.70270×10^{-11}
	3.7×10^{10}	1

吸収線量	Gy	rad
	1	100
	0.01	1

照射線量	C/kg	R
	1	3876
	2.58×10^{-4}	1

線量当量	Sv	rem
	1	100
	0.01	1

（86年12月26日現在）

高並列可視化処理時におけるZバッファ画像合成処理