ハイパフォーマンス
コンピューティング 82−18
（2000. 8. 4）

ウェーブフロント型計算における性能予測モデルの構築

齋藤剛博† 窪田昌史†† 津田孝夫††

分散メモリ型並列計算機では通信性能が全体の性能に大きく影響する。そのため通信の最適化は不可欠なものである。しかしながら、並列性と通信量がトレードオフの関係にあるウェーブフロント型計算では最適なデータ分割方法や、通信パラメータを決めることは困難である。そこで、本研究ではウェーブフロント型計算のモデル化を行ない、最適なパラメータを決定する手法について述べる。

ウェーブフロント型計算としてSOR法のプログラムを用いた結果モデルからほぼ最適なプロセッサ台数、データ分割方法が求められる確認できた。

Construction of Performance Prediction Models in Wavefront Computation

TAKAHIRO SAITO ,† ATSUSHI KUBOTA †† and TAKAO TSUDA ††

In distributed memory parallel machines, communication performance has large effect on total performance. Thus, optimization of communication is essential. However, it is difficult to determine the best data distribution and communication parameter in the wavefront method that has tread off relation between parallelism and transferred messages. In this paper, we show a method to determine the best parameter by constructing the model for wavefront computation. We could ascertain the approximately best number of processors and data distribution based on the model for wavefront method.

1. はじめに

近年、流体力学、熱力学、天文学などの科学技術計算分野では数TFLOPSの性能が要求されている。この要求を満たしえるシステムとしては現在、並列計算機が主に使用される。そのため、大規模なデータを扱い並列性を内在するプログラムを高速に実行するために分散メモリ型並列計算機を使用することが一般的になってきている。

大規模なデータを扱うプログラムを高い性能を持つ並列プログラムへと変換する際には、プロセッサ間の通信を最適化して、通信に伴うオーバーヘッドを減少させることが重要となる。プロセッサ間通信の最適化としては、メッセージの一括送信などさまざまな手法が知られているが、その最適化手法を容易に適用できるとは限らない。メッセージの一括化と並列性の抽出とがトレードオフの関係にあるような、ウェーブフロント型計算では、一括化すべきメッセージの最適な大きさを決定することは困難である。

たとえば、SOR法では2重ループ内でループ運搬依存があり、これを単純に並列化すると要素ごとに通信が必要となり、通信のオーバーヘッドが問題となる。そのため、ブロック化を行なうことで通信回数を減少させることができる。また、データの分割方法やブロックサイズ、メッセージ長の違いによって通信回数などが変わり、プログラムの性能が変化する。メッセージ長を長くすれば、計算開始時での他のプロセッサの待ち時間が増加する。しかし、メッセージ長を短くすれば、通信回数が増え、通信のオーバーヘッドが増加する。ブロックサイズを増やせば、計算量に対する通信量の割合が増え、台数に見合うだけの効果が得られない可能性がある。

これらの最適なブロックサイズやメッセージ長は並列計算機の通信性能や演算性能からある程度割り出されると考えられる。本研究では、データのサイズ、計算アルゴリズムなどから計算モデルを構築し最適なデータ分割方法、プロセッサ数などのパラメータを決定することを目標とする。

本稿の構成は以下のとおりである。2章ではSOR法のアルゴリズムとその並列化について述べる。3章でSOR法の性能モデルについて述べる。4章で並列化されたプログラムを分散メモリ型並列計算機上で実行し、実行時間や通信時間を測定した結果を示す。5章では3章で述べた性能モデルと4章で示した測定結果とを比べ、考察を行なう。最後に、6章でまとめとする。

2. 計算アルゴリズム

本研究では、問題として熱伝導方程式を解く際に出てくる規則的な2次元格子領域上で差分法によって定式化

† 広島市立大学 大学院 情報科学研究科
Graduate school of Information sciences,Hiroshima city University
†† 広島市立大学 情報科学部
Faculty of Information sciences, Hiroshima city University

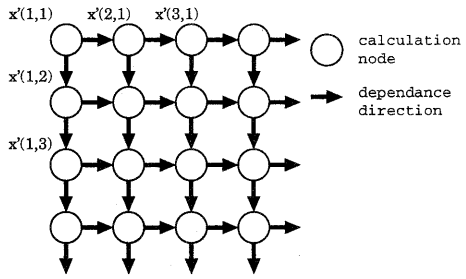

図1 SOR法の依存

された偏微分方程式[1]を例題として用いる。

この偏微分方程式を解く過程には連立1次方程式が存在する。これを $Ax = b$ と表し、A を行列、b, x をベクトルとする。対象となる行列 A の成分はほとんどが0の疎行列となりその非零成分は1行、1列あたり最大5つしかなく、規則的に並んでいる。

SOR法は連立1次方程式の反復解法の1つであり、SOR法を用いて連立1次方程式を解く場合、ループ運搬依存が2箇所存在する。この依存をウェーブフロント状に抽出することで並列計算機においても効率良く問題を解くことができる。

行列 A を $n^2 \times n^2$、b, x を長さ n^2 とし x を $n \times n$ の2次元行列に並び換えた行列を x' とする。また、疎行列 A を並び換え $n \times n$ の密行列 A' を作る。第k次解が与えられている場合、ω を緩和パラメータとして第k+1次解は式(1)の様になる。

$$\begin{aligned} x'(i,j)^{(k+1)} = & (x'(i-1,j)^{(k+1)} * A'(i-1,j) \\ & + x'(i,j-1)^{(k+1)} * A'(i,j-1) \\ & + x'(i+1,j)^{(k)} * A'(i+1,j) \\ & + x'(i,j+1)^{(k)} * A'(i,j+1)) \\ & * \omega / A'(i,i) \\ & + (1-\omega) * x'(i,j)^{(k)} \end{aligned} \quad (1)$$

ここで第k次解から第k+1次解を求めるときに $x'(i,j)^{(k+1)}$ は $x'(i-1,j), x'(i,j-1)$ については第k+1次解、$x'(i+1,j), x'(i,j+1)$ については第k次解について依存があることが分かる。この式から分かるループ運搬依存は図1の様になる。そのため、単純に並列化することはできない。

2.1 SOR法の並列化

SOR法を並列化した場合、計算は図1左上の $x'(1,1)$ から開始され、$x'(1,2)$,$x'(2,1)$ へと右下方向へ順次伝わって行く。SOR法ではウェーブフロントが右下まで伝わりきるまでは解が求まらないがウェーブフロントが伝わりきると、単位時間ごとに解が求まっていく。

2.1.1 1次元ブロック分割

1次元ブロック分割を行なった場合ブロックの形状としては長方形になり、通信はブロック単位で行なわれる。計算は左端のプロセッサから開始される。プロセッサは1つ前のプロセッサからデータを受けとると、計算を開始しすべての要素について計算が終了すると次のプロセッサにデータを送る。

メッセージ長を変化させる場合では、プロセッサは1ブロック単位で通信を行なわず、数要素計算が終った時点で通信を行なう。そのため、次のプロセッサはあまり待たされることなく計算を開始することができる。

2.1.2 2次元ブロック分割

2次元ブロック分割では通信はブロック単位で行なわれる。この場合、1次元ブロック分割とは違いプロセッサは周囲4プロセッサと通信を行なう必要がある。また、上方向、左方向の両方からデータを受信しなければ計算を開始することができない。

3. 性能モデル

本章では、SOR法の性能モデルについて述べる。分散メモリ型並列計算機において、性能モデルは通信パラメータと計算機の演算性能から導き出されると考えられる。そのため、この2種類のパラメータが分かっていれば実行にかかる時間をある程度見積もることができる。具体的には通信パラメータとしては受信、送信のためのオーバヘッドや通信速度、演算性能としては、プロセッサの性能などが考えられる。

計算アルゴリズムをモデル化することで実行時間をこれらパラメータで表すことができ、これにより、より効率的な分割方法や分割サイズ、メッセージ長などを求めることができると考えられる。

本研究ではLogPモデル[2]や文献3)と同様、以下の様に定義した記号を用いる。

n 2次元配列の1辺のサイズ
N_B 2次元配列の1辺のブロックの個数
n_b ブロックのサイズ
n_{msg} メッセージ長
N_{MSG} ブロック当たりのメッセージを送る回数
T_{comm} クリティカルパスの通信時間の総和
T_{comp} クリティカルパスの計算時間の総和
T_{cb} クリティカルパスの全実行時間
α n_b 個のデータを送信するのにかかる時間
O_s n_b 個のデータを送信するのにかかる送信オーバーヘッド
t_{reg} 1ブロックあたりの計算時間
t_{next} 第k次解から第k+1次解を求めるために必要な時間

ただし、n/N_B=n_b、$n/N_{MSG} = n_{msg}$ である。
ここで α は送信プロセッサがデータを送信し、受信プロセッサが受信するまでの時間である。Os は送信プロセッサが送信を完了するまでの時間である。

3.1 1次元ブロック分割

1次元ブロック分割ではメッセージ長とブロックの縦のサイズを等しいとする。1次元ブロック分割のクリティカルパスを図2に示す。1次元ブロック分割ではクリティカルパスはすべてのプロセッサを通る。つまり、

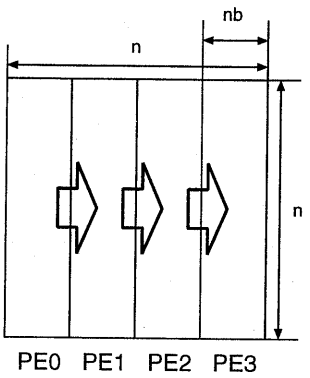

図2 1次元ブロック分割のクリティカルパス

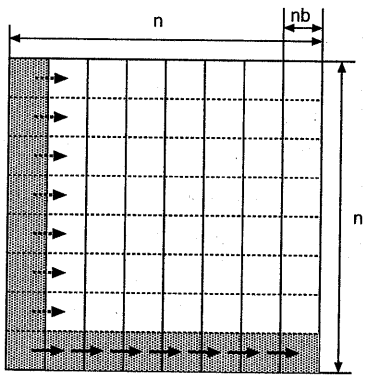

図3 メッセージ長を変化させた場合のクリティカルパス

PE0 から PE(N_B-1) までクリティカルパスが続く。このとき、クリティカルパスの全実行時間 T_{cb} はクリティカルパスの通信時間 T_{comm} とクリティカルパスの計算時間 T_{comp} の和で与えられる。

$$T_{cb} = T_{comm} + T_{comp} \quad (2)$$

クリティカルパス中の通信時間の総和は次の式で表される。

$$T_{comm} = \alpha(N_B - 1) \quad (3)$$

これは最後のプロセッサを除く全プロセッサの通信時間の和となる。クリティカルパス中の計算時間の総和は次の式で表される。

$$T_{comp} = t_{reg} N_B \quad (4)$$

これは単純に全プロセッサの計算量の和となる。

SOR 法では実際には解が収束するまで計算を行なう。ここでは 2.2.2 節でのべたように第 k 次解から第 k+1 次解を計算を開始し、その結果を隣接するプロセッサに送信するのに要する時間 t_{next} は以下の様になる。

$$t_{next} = t_{reg} + \alpha \quad (5)$$

理想的なパイプライン実行が行なわれると、第 M 次解を求めるために必要な全実行時間 T_M は次の様になる。

$$T_M = t_{next}(M - N_B - 1) + 2T_{cb} \quad (6)$$

右辺第 1 項は全プロセッサが演算、通信を行なっている時間であり、右辺第 2 項はパイプライン型実行の開始時（プレリュード）と終了時（ポストリュード）でかかる時間である。

3.2 メッセージ長を変化させた場合

SOR 法の 1 次元分割においてメッセージ長を変化させた場合クリティカルパスは PE0 の全領域の実行と PE1 から PE(N_B-1) までの 1 領域分の実行および N_B-1 回の通信であり、図 3の様になる。

クリティカルパス中の通信時間の総和は次の式で表される。

$$T_{comm} = (N_{MSG} - 1)Os + (N_B - 1)\alpha \quad (7)$$

右辺第 1 項はクリティカルパス内からクリティカルパス外への送信のオーバーヘッドである。右辺第 2 項はクリティカルパス内での通信時間である。メッセージ長を短くするとデータの通信回数か増えるために、クリティカルパス中での通信のオーバーヘッドが増える。しかし、クリティカルパス全体の通信量は減る。

クリティカルパス中の計算時間の総和は次の式で表される。

$$T_{comp} = (N_{MSG} + N_B - 1)t_{reg} \quad (8)$$

t_{reg} は n_{msg} と n_b の関数として表すことができると考えられる。

ここで、t_{next} は 1 プロセッサの全領域の計算時間と次のプロセッサへの通信時間の和となり、次の様に表すことができる。

$$t_{next} = N_{MSG}(\alpha + t_{reg}) \quad (9)$$

第 M 次解を求めるために必要な時間は次の式になる。

$$T_M = t_{next}(M - N_B - 1) + 2T_{cb} \quad (10)$$

3.3 2次元ブロック分割

2 次元ブロック分割ではブロックの縦とブロックの横のサイズを等しいとする。また、メッセージ長はブロックサイズと等しいものとする。

SOR 法の 2 次元分割においてブロック間の依存関係から考えられる理想的なクリティカルパスは図 4の (a) または (b) の灰色の部分（濃淡とも）になる。

クリティカルパスの通信時間の総和は次の式で表される。

$$T_{comm} = 2(N_B-1)\alpha + (2(N_B-2)+1)O_s \quad (N_B \geq 2) \quad (11)$$

右辺第 1 項はクリティカルパス中のブロック同士の通信時間 (図 4実線矢印) であり、これはプロセッサが送信を開始し次のプロセッサがデータを受信するまでの時間となる。第 2 項はクリティカルパス中からクリティカルパス外への通信時間 (図 4破線矢印) であり、これはプロセッサの送信のオーバーヘッドのみとなる。

クリティカルパス中の計算時間の総和は次の式で表される。

$$T_{comp} = (2N_B - 1)t_{reg} \quad (12)$$

t_{reg} は n_b の 2 次関数として表すことができると考えられる。クリティカルパスの全実行時間の下限はこれらの式により求めることができる。

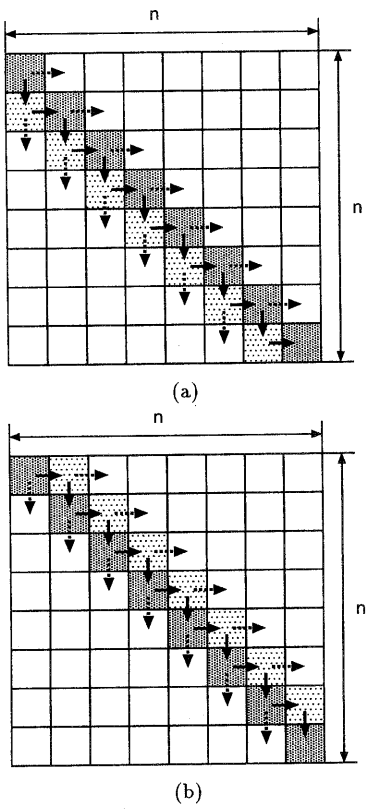

図4　2次元ブロック分割でのクリティカルパス

また、t_{next} は次の様になる。これは、計算時間と右方向、下方向への通信時間の和となる。

$$t_{next} = t_{reg} + 2\alpha \qquad (13)$$

第 M 次解を求めるために必要な全実行時間 T_M は次の様になる。

$$T_M = t_{next}(M-2N_B)+2T_{cb} \qquad (N_B \geq 2)(14)$$

4. 評　　価

本章では実際に測定された性能と性能モデルから計算できる理論値を示す。

測定には NEC Cenju-3,Cenju-4, 日立 SR2201 を使用した。通信ライブラリはいずれも MPI を使用した。また、理論式での α、O_s の値はあらかじめ各計算機でメッセージ長を変え計測しておいたものを使用した。t_{reg} の値については 1 ブロックの通信を除く計算のみの実行時間を各計算機の 1PE で計測したものを使用した。

また、第１００次解まで求めた場合での１次元ブロック分割、２次元ブロック分割における各計算機の台数効果を図 5に示す。以下それぞれの分割でクリティカルパスの実行時間と全体の実行時間を示し、最適なプロセッサ数とメッセージ長について議論する。

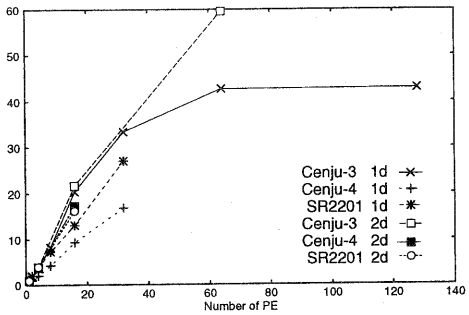

図5　1次元分割、2次元分割における台数効果

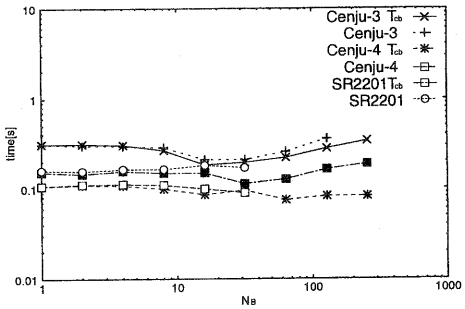

図6　1次元のクリティカルパス実行時間

4.1　1次元ブロック分割

問題サイズを $n = 512$ の密行列とし、Cenju-3 ではブロックサイズ N_B、つまりプロセッサ数を 1 から 128 まで、Cenju-4,SR2201 では N_B を 1 から 32 まで変化させて測定した。図 6にブロック数を変化させた場合のクリティカルパスの実行時間の実測値および理論値 (T_{cb}) 関係を示す。また、第 100 次解まで求めた場合の全実行時間の実測値および理論値 (T_M) を図 7に示す。

どの計算機においても理論値と実測値はほぼ一致している。また、クリティカルパス実行時間はほぼ一定になっておりプロセッサ台数を増加させても効果は薄い。

クリティカルパス実行時間が最も少ないのはブロック数が 16 または 32 あたりであり、それ以上プロセッサ数を増やすとクリティカルパス実行時間は長くなると考えられる。また、全実行時間も理論値と実測値はほぼ一致しており、プロセッサ台数の増加とともに性能が向上すると考えられる。

4.2　メッセージ長を変化させた場合

問題サイズを $n = 512$ の密行列とし、ブロック数を 4 （プロセッサ数を 4 ）、メッセージ送信回数を 1,2,4, …,256,512 とし、各計算機で測定した。図 8にメッセージ送信回数とクリティカルパスの実行時間の関係を示す。また、第 100 次解まで求めた場合の全実行時間を図 9に示す。メッセージ送信回数が 1 の場合では $n_{msg} = 512$ となり 1 次元ブロック分割でメッセージ長を変化させない場合とほぼ同じになる。どの計算機でも

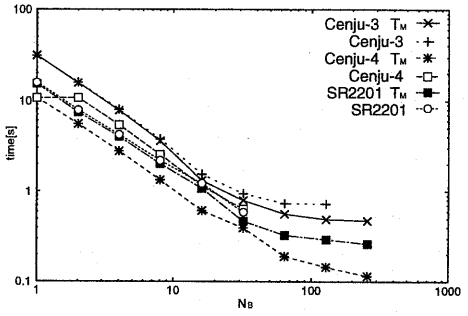

図7　1次元ブロックサイズと全実行時間の関係

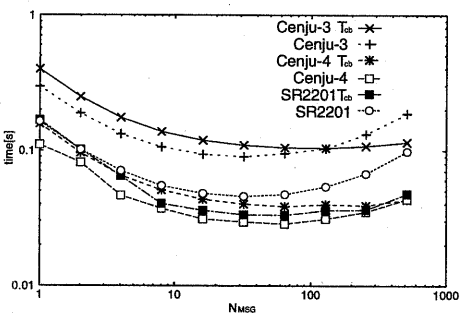

図8　メッセージ長を変化させた場合のクリティカルパス実行時間

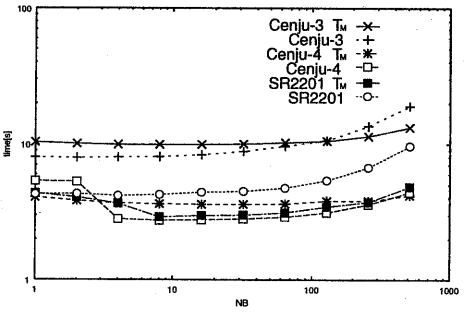

図9　メッセージ長と全実行時間の関係

$N_{MSG} = 32$ あたりでクリティカルパス実行時間が最小になっている。

4.3　2次元ブロック分割

問題サイズを $n = 512$ の密行列とし、Cenju-3では N_B を $1,2,4,8$(プロセッサ数を $1,4,16,64$) とし、Cenju-4,SR2201では N_B を $1,2,4$(プロセッサ数を $1,4,16$) として計測した。図10にブロック数と実行時間の関係を示す。また、第100次解まで求めた場合の全実行時間を図11に示す。どの計算機においても性能モデルの理論値と実測値は近い値になっている。また、理論値ではどの計算機においても N_B が32の時に最適になることを予測している。また、全実行時間は N_B が64の辺りで最適になることを予測している。

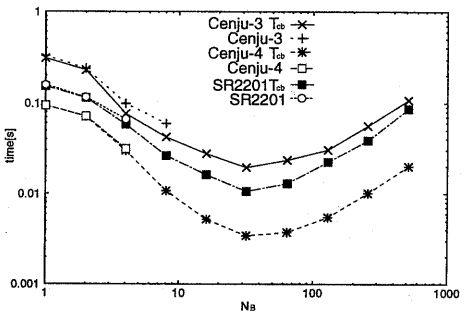

図10　2次元でのクリティカルパス実行時間

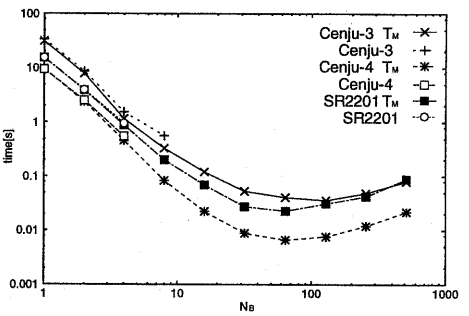

図11　2次元ブロックサイズと全実行時間の関係

5. 考　　察

5.1　1次元ブロック分割

式(6)に示すように1次元ブロック分割では N_B が増加すると t_{reg} は N_B に反比例する。そのため、計算時間 T_{comp} は N_B に関わらずほぼ一定になる。しかし、式(5)に示すように T_{comm} は N_B の増加とともに増えるため、T_{cb} は N_B の増加とともに増えることになる。実際は N_B の増加とともに1プロセッサあたりの保持するデータ量は減りキャッシュヒット率が向上するため T_{comp} が減少すると考えられる。さらに N_B が増加すると式(2)において T_{comm} が支配的になると考えられる。

5.2　メッセージ長を変化させた場合

メッセージ長を変化させた場合、クリティカルパスの実行時間は N_{MSG} が16,32,64あたりが最も速い。N_B 大きくなるにつれてクリティカルパス実行時間が短くなり、再び長くなるのは次の様に考えられる。式(8)を書き直すと次のようになる。

$$T_{comp} = (N_{MSG} + N_B - 1)t_{reg} = N_{MSG}t_{reg} + (N_B - 1)t_{reg} \qquad (15)$$

t_{reg} は N_{MSG} に反比例する。そのため、式(15)の右辺第1項はほぼ変わらず右辺第2項は N_{MSG} の増加とともに t_{reg} が減少するため、クリティカルパスの計算量は減り T_{comp} は減少する。しかし、さらに N_{MSG} が増加すると通信回数が増加し通信オーバーヘッドが増加す

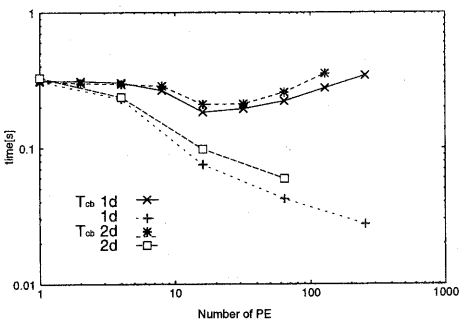

図12 Cenju-3での1次元と2次元の比較

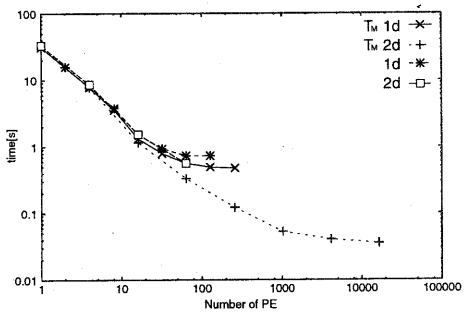

図13 Cenju-3において第100次解を求める場合の比較

るため T_{comm} は増加することになる。また、ループのオーバヘッドの増加も考えられる。

また、第M次解を求める場合では式(9)において α、t_{reg} ともに N_{MSG} にほぼ反比例するため、t_{next} はほぼ一定になる。そのため、、第M次解を求める場合ではメッセージ長は全実行時間にほとんど影響しないと考えられる。

5.3 2次元ブロック分割

2次元ブロック分割の場合どの計算機も同じような傾向を示している。クリティカルパス実行時間はどの計算機でも N_B が32の時最も速い。t_{reg} は N_B が減少するとともに2次関数的に増加すると考えられるため、t_{cb} 内で t_{reg} が支配的になり大きく時間が増加する。N_B が増加すると通信回数が増加し t_{cb} 内で t_{comm} の割合が大きくなる。また、ループのオーバーヘッドの増加も考えられる。

5.4 1次元と2次元の比較

Cenju-3での1次元ブロック分割と2次元ブロック分割のクリティカルパス実行時間を比較したものを図12に示す。1次元ブロック分割のクリティカルパスには並列性がないため、2次元ブロック分割の方が並列度が高く実行時間は短くて済む。Cenju-3での第100次解を求める場合での1次元ブロック分割と2次元ブロック分割を比較したものを図13に示す。ブロック数が少ない場合には1次元ブロック分割、2次元ブロック分割ともにほとんど差はない。しかし、ブロック数の増加とともに差が開いており、2次元分割の方が性能がよいことがわかる。また、図5からも同様に2次元ブロック分割の方が性能がよいことが分かる。これは2次元ブロック分割の方が総通信量が少ないためだと考えられる。第k次解から第k+1解を求めるために必要な総通信量はプロセッサ数を P^2、2次元配列1辺の長さの通信量を C とすると1次元ブロック分割の通信量は次の様になる。

$$(P^2-1)C \tag{16}$$

2次元ブロック分割の通信量は次の様になる。

$$2(P-1)C \tag{17}$$

すると式(16)-式(17)は以下の様になる。

$$(P^2-1)C-2(P-1)C = CP^2-2CP+C = C(P-1)^2 \tag{18}$$

よって同プロセッサ数の場合、常に2次元ブロック分割の方が通信量が少ないことが分かる。また、1次元ブロック分割は2次元ブロック分割よりプレリュード、ポストリュードが長くなり最大の並列度を保っていられるが短い。以上の様な理由から2次元ブロック分割の方が1次元ブロック分割よりよいと分かる。

6. おわりに

本研究ではウェーブフロント型計算としてSOR法を分散メモリ型並列計算機に実装し、それぞれのアルゴリズムのモデル化を行なった。その結果、通信速度、送信オーバーヘッド、ブロックあたりの計算時間などから最適な分割方法やパラメータを求めることができた。分割方法としては1次元ブロック分割より2次元ブロック分割の方が並列性も高く通信量も少ないため優れていることが分かった。ほぼ最適なメッセージ長を求めることができた。

謝辞 本研究を行なうにあたり、NEC並列処理センターのCenju-3,Cenju-4、日本原子力研究所計算科学技術推進センターのSR-2201、および東京大学情報基盤センターのSR-2201を使用させていただいた。ここに感謝の意を表する。

参考文献

1) 森 正武 “数値計算プログラミング”,pp.302-311, 岩波書店,1991
2) Culler, D.E.,Karp,R.,Patterson,D.,Sahay,A. ,Schauser,K.,Santos,E.,Subramonian,R.and Eichken, T.V.: “LogP: Towards a Realistic Model of Parallel Computation”, Proc. 4th ACM SIGPLAN PPoPP, pp.1-12(1993).
3) 坂根 広史, 児玉 祐悦, 建部 修見, 小池 汎平, 山名 早人, 山口 喜教, 弓場 敏嗣 “ウェーブフロント型並列処理における分散メモリ型並列計算機の通信機構の評価”,pp2281-2293, 情報処理学会論文誌,Vol.40,No.5,1999