計算機アーキテクチャ 106－3
（１９９４．６．１３）

並列計算機EM-4におけるループ間Doacross方式の自動最適化

山名早人　佐藤三久　児玉祐悦　坂根広史　坂井修一‡　山口喜教

電子技術総合研究所　‡新情報処理開発機構

本報告では，分散メモリ型の並列計算機において，イテレーション間にデータ依存があるループを高速実行するための手法として，ループ間Doacross方式を提案した．本方式は，通信ディレイ，リモートメモリライト・リード時間，及びデータ通信準備時間が全体の実行時間に与える影響を小さくする方式である．並列計算機EM-4を用いた評価の結果，Doacrossの1.8～3.6倍，Pipeliningの3.3～5.7倍，data owner computes rule に基づく実行方式の1.8～3.6倍の処理速度向上を確認した．また，実行時間を最小にするためには，部分ブロック化係数ｋの決定が重要であることを示し，並列計算機が持つ各種パラメータとプログラム情報からコンパイル段階で決定できることを示す．

Automatic Tuning of Loop-Doacross Execution Scheme on the EM-4 Multiprocessor

Hayato YAMANA　Mitsuhisa SATO　Yuetsu KODAMA　Hirohumi SAKANE
Shuichi SAKAI ‡　Yoshinori YAMAGUCHI

Electrotechnical Laboratory　‡Real World Computing Partnership
1-1-4 Umezono, Tsukuba, Ibaraki 305, Japan

The purpose of this paper is to propose a new loop execution scheme, called Loop-Doacross, on distributed memory machines. A Loop-Doacross execution scheme shortens the total execution time by excluding the time of communication delay, remote memory writting&reading, and data sending preparation from the main execution stream. An experimental evaluation shows that the measured speedup ratio is 180%～570% of the the conventional execution schemes. This paper also describes its tuning scheme on the EM-4 multiprocessor at compile time.

1．まえがき

従来，Doall型以外のループを並列計算機上で実行する方式としてDoacross[Cytr86]やPipelining[PaKL80]が提案されている．しかし，これらの方式は，元々，密結合型の並列計算機を対象としたものであり，メッセージ通信によりプロセッサ間のデータ交換を行う分散メモリ型の並列計算機では，十分な処理性能を引き出すことができない．

分散メモリ型の並列計算機では，以下に述べる理由により，Doacross及びPipeliningの効果が出ない．

(1) Doacrossでは，ループの繰り返し回数をNとすると，プロセッサ間の通信ディレイδが(N－1)回分，全体の実行時間に加算される．この為，δが十分に小さくないと処理速度の向上が得られない．しかし，分散メモリ型並列計算機ではδを無視できない．また，プロセッサ間通信に伴う通信準備時間も(N－1)に比例して，全体の実行時間に加算される．

(2) Pipeliningでは，各文の実行時間TsがN回分，全体の実行時間に加算されるため，Tsが十分に小さくないと処理速度の向上が得られない．しかし，分散メモリ型並列計算機でPlipeliningを実現とようとすると，Tsにリモートメモリライト時間(定義される配列要素を他のプロセッサが持つメモリに書くための時間)やリモートメモリリード時間(参照される配列要素を他のプロセッサから読むための時間)が含まれTsを小さくできない．

一方，分散メモリ型計算機でのループの実行方式としては，data owner computes ruleに基づいて実行を行う方法が有名である．data owner computes ruleに基づく実行では，リモートメモリライト時間を無視することができても，対象とするループがDoall型以外のループの場合，以下に述べる理由により，十分な性能を引き出せない．

(3) Doall型以外のループ，すなわち，イテレーション間にデータ依存関係が存在するループを実行する場合，計算を実行するプロセッサ間(data owner computes ruleにより決定)で，プロセッサ間通信が必要となる．今，データが各プロセッサにサイクリックに割当られているとすると，上述のDoacross同様，(N－1)回のプロセッサ間通信が必要となり，通信ディレイδ及び通信準備時間が十分に小さくないと処理速度の向上が得られない．

これに対して，我々の提案するループ間Doacross[YSKS94]は，プロセッサ間の通信ディレイδ及びリモートメモリライト／リード時間が全体の実行時間に与える影響を小さくすると共に，プロセッサ間の通信回数を減らすことにより，通信準備時間が全体の実行時間に与える影響も小さくする方式である．

本方式を並列計算機EM-4[SYHK89]上において，評価した結果，Doacrossの1.8～3.6倍，Pipeliningの3.3～5.7倍，data owner computes ruleに基づく実行の1.8～3.6倍の処理速度向上を確認した．

以下では，まず，ループ間Doacross方式について述べた後，並列計算機EM-4上での評価結果を示す．次に，ループ間Doacross方式を最適化するための手法と，実際にその手法をいくつかのループに適用した結果を報告し，コンパイル段階での最適化が可能であることを示す．

2．ループ間依存関係の定義

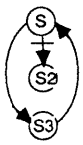
2.1 前処理

ループ間の依存関係をπブロック[BCKT79]間のデータ依存関係として定義する．πブロックとは，データ依存関係のある文集合中，最大のデータ依存サイクルを持つ文集合に分割したものと定義される．

まず，対象とするループをπブロックに分割する．πブロックを用いることにより，DOループを最小の単位に分割することができる．例えば，図1に示すDOループは，2つのπブロックから構成される．

```
      DO 10 I=2,N
        A(I)=B(I-1)*3+C(I)  ...S1
        C(I)=A(I+1)*3       ...S2
        B(I)=A(I)*B(I)      ...S3
10 CONTINUE
```

π1={S1,S3}
π2={S2}




```
      DO 10 I=2,N
        C(I)=A(I+1)*3       ...S2
10 CONTINUE
      DO 20 I=2,N
        A(I)=B(I-1)*3+C(I)  ...S1
        B(I)=A(I)*B(I)      ...S3
20 CONTINUE
```

図1　対象ループ

2.2 ループ間のデータ依存関係の定義

ループ間のデータ依存関係を以下のように定義する．2つのπブロック $\{\pi_1, \pi_2\}$ が存在する時，これら2つのπブロック間には，以下に示す4つのデータ依存関係が存在する．

フロー依存($\pi_1 \rightarrow \pi_2$)
$\exists S_k \in \pi_1, \exists S_l \in \pi_2, S_k \delta_f S_l$

出力依存($\pi_1 \dashv\!\rightarrow \pi_2$)
$\exists S_k \in \pi_1, \exists S_l \in \pi_2, S_k \delta_o S_l$

逆依存($\pi_1 \ominus \pi_2$)
$\exists S_k \in \pi_1, \exists S_l \in \pi_2, S_k \delta_a S_l$

入力依存

$$^{\exists}S_K \in \pi_1, {}^{\exists}S_l \in \pi_2, S_k \delta_I S_l$$

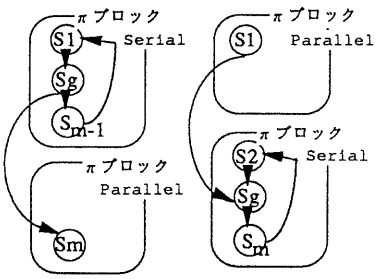
ここで，S_k,S_lは文を，δ_f,δ_o,δ_a,δ_Iは，それぞれ，フロー依存，出力依存，逆依存，入力依存を示すものとする．これら，４つの依存関係の内，入力依存を除く３つの依存のうちの少なくとも１つが，π_1，π_2の間に成立する場合，π_1,π_2の実行順序関係を変えることができない．

3．ループ間Doacross対象ループ

ループ間Doacrossを適用するループは，同一ネストレベルにあるπブロックに分割されたDOループであり，条件分岐を含まないものとする．そして，πブロックに分割されたDoallループ[Poly88](π_P)とDoserialループ[Poly88](π_S)との間に，($\pi_S \delta_f \pi_P$)あるいは($\pi_P \delta_f \pi_S$)のデータ依存関係があるとき，これら２つのループを融合してできたループを対象とする．ただし，δ_fはフロー依存を表す．

```
DO 10 I=1,N
      S1
      S2
      ⋮
      Sm
10 CONTINUE
```

(a)対象ループ



(b)文間のデータ依存関係

図２　ループ間Doacross対象ループ

図２に，対象ループを示す．図２に示すように，融合後のループの総文数はm（Doallループ(π_P)が1文，Doserial型ループ(π_S)がm−1文），繰り返し回数はNであるとする．

Doall型ループ(π_P)の文数が1であるのは，πブロックの定義（データ依存関係のある文集合中，最大のデータ依存サイクルを持つ文集合に分割したものをπブロックと定義）より，πブロックであるDoallループは，1文になるからである．また，($\pi_S \delta_f \pi_P$)や($\pi_P \delta_f \pi_S$)におけるフロー依存は，単一のフロー依存であるとする．すなわち，ある1つの配列データについてのみフロー依存しているものとする．

次に，実際のプログラムにおいて，どの程度のDoall型ループが，3.1で述べた($\pi_S \delta_f \pi_P$)や($\pi_P \delta_f \pi_S$)の形になっているかを評価する．

サンプルプログラムとして，表１に示す科学技術計算でよく用いられるサブルーチン６本を用いた．

表1　科学技術計算用サブルーチン

LAX	実行列の連立一次方程式
BLN	実行列の平衡化
LAL	実行列の最小二乗解
HES	実行列の実ﾍｯｾﾝﾍﾞﾙｸﾞ行列への変換
HBK	実行列の固有ベクトルへの変換と正規化
LTX	実３項行列の連立一次（ガウスの消去法）

表２にDoall型ループ数及び($\pi_S \delta_f \pi_P$)や($\pi_P \delta_f \pi_S$)の形となるDoall型ループ数を示した．ただし，完全にネストしたループは1と数えた．表２の結果より，プログラムの性質に左右されるが，25～50％程度のDoall型ループが対象となることがわかる．

表2　ループ間Doacross適用ループ数

プログラム	Doall型ループ数	対象となるDoall型ループ数
LAX	18	9 (50%)
BLN	2	1 (50%)
LAL	18	7 (39%)
HES	15	5 (33%)
HBK	19	5 (26%)
LTX	4	1 (25%)

4．従来の実行方式の実行時間

図２に示したループをPipelining，Doacross，そしてπ_Sとπ_PをそれぞれDoserial／Doallとして実行した場合，data owner computes ruleを適用した場合の実行時間を比較し表３に示す．なお以下の議論では，図２に示した２種類の依存関係($\pi_S \delta_f \pi_P$)及び($\pi_P \delta_f \pi_S$)の内，($\pi_S \delta_f \pi_P$)の場合について実行時間を求める．($\pi_P \delta_f \pi_S$)についても同様にして求めることができるが，ここでは結果のみを下欄に示す．実行時間算出にあたっての仮定は以下の通りである．

- 図２に示すように，π_S内にm−1個の文，π_P内に1個の文が存在する．
- π_Sからπ_Pへのデータ依存は，文Sg∈π_S（1≦g≦m-1）から文Smに対しての１つのフロー依存である．
- 文Siの実行開始から文Sjの実行終了までの実行時間をT(Si,Sj)で表す．
- π_Sからπ_Pへのデータ通信は，π_Sでデータが定義された時点で直ちに行われる．
- data owner computes ruleによる実行時間算出においては，簡単のため，同一イテレーション内の各文で定義される配列要素は全て同一のPE内に格納されるとする．

表３に示した実行時間は，文の実行時間を単にT(Si,Sj)で記述しているが，実計算機上でこれらの方式を実現する場合，T(Si,Sj)は，次の３種類の時間から構成される．

(1)加算等の演算時間（ｔe）
(2)配列のアドレス計算時間及び配列要素のリモートメモリライト／リード時間（ｔa）
(3)他プロセッサへのデータ出力／入力に伴う通信準備時間（ｔc）

先に述べたように，上記に加えて，プロセッサ間のデータ通信ディレイδを含めた合計４つのパラメータにより各実行方式の実行時間が求まる．

表３に示した３種類の実行方式を用いた場合について，上記に示した４項目が全体の実行時間に与える影響を比較し，表４に示す．ただし，表４では，簡単のため，T(S1,Sm-1)≧T(Sm,Sm)と仮定する．

表４　te, ta, tc, δが実行時間に与える影響

	Pipelining	Doacross	Doserial / Doall	data owner computes rule
te	O(N)	O(N)	O(N)	O(N)
ta	O(N)	none	O(N)	none
tc	O(N)	O(N)	O(N)	O(N)
δ	O(1)	O(N)	O(1)	O(N)

まず，Doacross時のｔａを見かけ上，0とすることができる理由を示す．Doacross実行では，表３におけるT(S1,Sm-1)の実行時間が全体の実行時間を決定する主要因となる．従って，できる限りT(S1,Sm-1)を小さくすべきである．ここで，S1～Sm-1実行時には予め定められたローカルPE内のメモリ上に一時的にデータをストアし，後に(Smを実行する際に)，ローカルPE内のメモリから取り出し，実際書き込むべきプロセッサに対してリモートメモリライトを行う．

これにより，見かけ上，リモートメモリライトの時間を０にすることができる．また，リモートメモリリードについては，隣接のプロセッサからデータが届く前に先行リードすることが出来る．これにより，Doacrossでは，ｔａを見かけ上，０とすることができる．

また，data owner computes rule適用時には，定義される配列要素が存在するPE上で実行が行われるため，ｔaの時間を０にできる．さらに，リモートメモリリードに対しては，Doacrossと同様，先行リードすることができるため，ｔaを見かけ上，０とすることができる．

表４より，PipeliningやDoserial / Doallによる実行時間は，δの影響をほとんど受けないが，逆に，ｔaによる実行時間がそのまま全体の実行時間に加算される．これに対して，Doacrossやdata owner computes ruleによる実行時間は，δの影響が大きいが，逆に，ｔaによる実行時間の増大を回避できる．

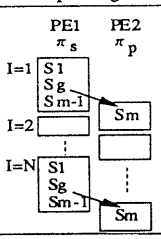
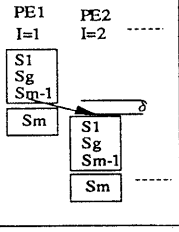
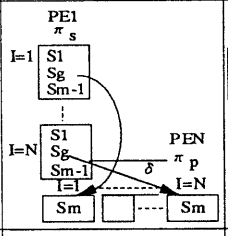
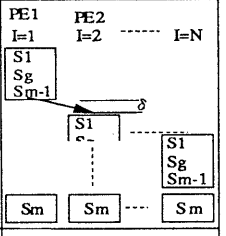
このように従来の実行方式は，各々，一長一短を持つ．

５．ループ間Doacross方式

ループ間Doacrossは，πブロックに分割されたDoallループとSerialループの間のデータ依存関係に着目し，これらのループ間にフロー依存が存在する時，これらを融合し一つのループとする．そして，融合されたループをｋ個のイテレーション毎に部分ループ化し，部分ループを１つのプロセッサに割り当てる．これにより，表４のｔａをDoacrossと同様に０にできると共に，ｔcをO(N／k)，δをO(N／k)にすることができる．図３に変換例を示す．

図３は，(π_s δ π_p)の依存関係にあるループを融合後，インデックスｋ個単位(部分ブロック化係数と呼ぶ)で内部の２文を部分ループ化したものである．部分ループ化できる条件は，全てのインデックス集合Ｉに属する任意の部分集合{i}で各々のπブロックを分割した際に，分割された部分πブロック間のデータ依存関係が，分割前後で変わらないことである．

表３　従来の実行方式適用時の実行時間

Pipelining	Doacross	Doserial / Doall	data owner computes rule
			
●T(S1,Sm-1)≧T(Sm,Sm)時 T(S1,Sm-1)*(N-1)+T(S1,Sg) +Max(T(Sg,Sm-1), δ+T(Sm,Sm)) ●T(S1,Sm-1)＜T(Sm,Sm)時 T(S1,Sg)+δ+T(Sm,Sm)*N	(T(S1,Sm-1)+δ)*(N-1) +T(S1,Sm)	T(S1,Sm-1)*N +T(Sm,Sm) +Max(0,δ-T(Sg,Sm-1))	(T(S1,Sm-1)+δ)*(N-1) +T(S1,Sm)

※(π_p δ_f π_s)の時の実行時間は，上式において，T(S1,Sm-1)をT(S2,Sm)，T(S1,Sg)をT(Sg,Sm)，T(Sg,Sm-1)をT(S1,Sg)，T(Sm,Sm)をT(S1,S1)で置き換える．

例えば，図3の例では，部分ループ化後も(L1 δ_f L2)となりデータ依存関係が保たれる．ループ間Doacross化後の実行時間は，図2の例を用いると，

$$T(S1,Sm-1)\times N+\delta\times N/k+k\times T(Sm,Sm)$$

となる．

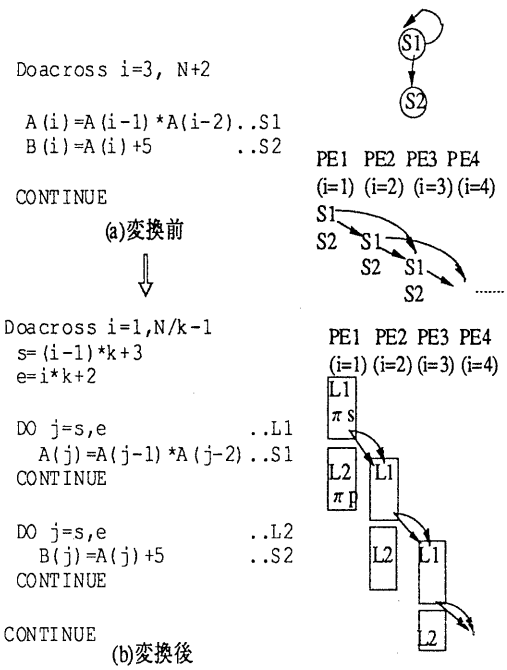


図3　ループ間Doacross方式

表3のDoacrossの実行時間に比較すると，δによる実行時間の増大が，1／kになり，最後の項T(m,m)の係数がk倍になっていることがわかる．すなわち，kが増大するにつれ，ループ間Doacrossの実行時間は減少し，その後増大する．これは，k×T(S1,Sm)の項がkの増加と共に増加するからである．従って，ループ間Doacrossを用いて実行時間を短縮するためには，kの決定が重要な要素となる．

5. ループ間Doacrossの評価

5.1　並列計算機EM-4

並列計算機EM-4[SYHK89]上で，図4に示す3つのプログラムについて，従来の4方式及びループ間Doacross方式を適用した場合の各々の実行時間を求めた．

EM-4は，80台のPEからなるデータ駆動機構を持つ並列計算機である．データ駆動機構を用いてスレッド間の高速な通信同期が可能になっている．

また，EM-4は，12.5MHzのクロックで動作しており，メモリ参照命令を除く大部分の命令は1クロックで実行される．ネットワークの性能はPEのポート当たり60.9Mbytes/secである．

```
DO   I=3, 3+1024
     A(I) = A(I-1) + A(I-2)
     B(I) = A(I) + 5
CONTINUE
     (1) Program -A

DO I=3, 3+1024
     A(I) = A(I-1) + A(I-2)
     B(I) = B(I-1) + A(I)
     B(I) = B(I) +5
CONTINUE
     (2) Program -B

DO I=2, 2+1024
     A(I) = A(I) × A(I-1)
     B(I) = A(I) + C(I)
CONTINUE
     (3) Program -C
```

図4　評価に用いたプログラム

図4に示すように各プログラムは，イテレーション回数を1024回とし，80台全てのPEを用いて実行を行った．配列要素は，80台のプロセッサが持つローカルメモリにサイクリックに割当てる．

プログラムは，EM-4用のプログラミング言語の一つであるEM-C[SKSY94]を用いて記述後，アセンブラコードに変換し，アセンブラレベルでの最適化を手作業で行った．また，プロセッサ間のデータ通信は，直接データ待ち合わせ機構[YaSK91]を用い，データの到着と同時にそのデータを必要とするスレッドが起動されるようにプログラムした．

5.2　評価結果

各々の実行方式で図4のプログラムを実行した時の実行時間を部分ブロック化係数kをパラメータとして図5(1)～(3)に示す．

図5に示すように，kの値によって，ループ間Doacross時の実行時間が変化することがわかる．実行時間を最小とするkを採用した場合，ループ間Doacross適用時，Doacrossの1.8～3.6倍，Pipeliningの3.3～5.7倍，data owner computes ruleに基づく実行の1.8～3.6倍の処理速度向上を確認できた．

5.3　評価結果考察

・Serial-DoallとPipelining

・Doacrossとdata owner computes rule

これら2組の実行時間はほぼ等しい．これは，表4に示したように，Serial-DoallとPipeling，Doacrossとdata owner computes ruleでは，それぞれ，te, ta, tc, δが全体の実行時間に与える影響が等しいからである．唯一の違いは，表3におけるSmの実行時間であり，イテレーション回数の増大に伴ってそれぞれ，PipeliningとDoacrossの方が有利となる．しかし，本評価では，イテレーション回数を1024に固定した為，その傾向はみられない．

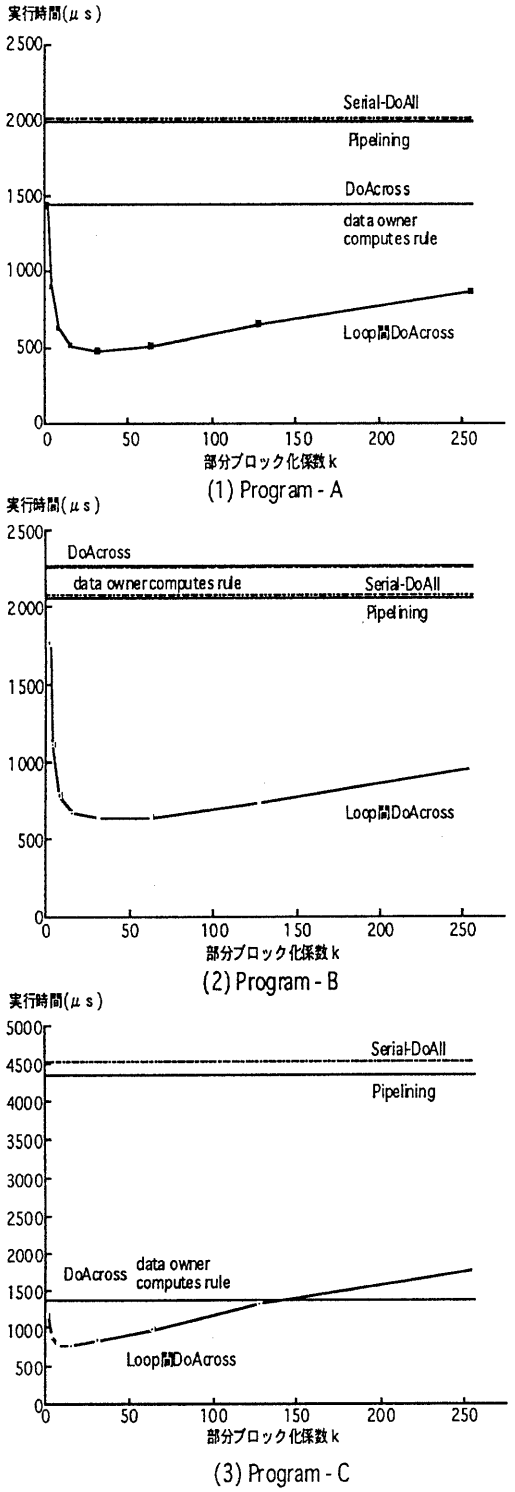


図5　各実行方式による実行時間比較

・Program-AとProgram-BでのDoacrossの実行時間

Program-A, Program-Bにおいて，Pipeliningによる実行時間はほとんど変わっていないのに対して，Doacrossの実行時間が大きく異なる．これは，プロセッサ間通信の回数の違いに起因する．Program-Aでは，1イテレーションに付きプロセッサ間通信が2回であるのに対し，Program-Bでは，3回必要である．Doacrossでは，表3に示すT(S1,Sm-1)の実行時間に通信準備時間が含まれるので，プロセッサ間通信が増大すると実行時間に大きな影響が出る．

・Program-CでのPipelining, Serial-Doallの実行時間

Program-Cでは，Pipelining及びSerial-Doallの実行時間が長くなっている．これは，表4に示したように，PipeliningやSerial-Doallでは，リモートメモリライト/リードにかかる時間taを無視することができないからである．Program-Cのπsの部分では，A(I)のリモートメモリリード及びリモートメモリライトが必要となる．これに対してProgam-A,Bでは，参照されるべきA(I-1),A(I-2)はプロセッサ間通信により受け取ることができるので，A(I)のリモートメモリライトのみでよい．

・ループ間Doacrossの実行時間

ループ間Doacrossでは，何れのプログラムの場合も部分ブロック化係数kをうまく選ぶことによって，他の既存の手法に比較して高速な実行が可能となることがわかる．

6．部分ブロック化係数kの決定手法

本節では，最適な部分ブロック化係数kを求める手法を示す．本手法により，コンパイル段階でのkの決定が可能となる．

具体的には，実行に用いる並列計算機が持つパラメータ，及び，対象とするプログラムが持つ性質からkをパラメータとしたループ間Doacrossの理論実行時間を求め，最適なkを導く．ただし，ループのイテレーション回数はコンパイル時に既知であるとする．

6.1　並列計算機のパラメータ化

対象とする並列計算機から以下のパラメータを抽出する．

(1)通信準備時間 tc
(2)演算時間 te
(3)ローカルメモリへのデータのロード/ストア時間 tlm
(4)プロセッサ間通信時の通信ディレイ δ
(5)アドレス計算及びリモートメモリライト時間 taw
(6)アドレス計算及びリモートメモリリード時間 tar
(7)ループ制御時間　tlp

上記のパラメータにおいて，ネットワークを用いるもの等，値が一意に定まらないものについては平均値を用いる．例えば，EM-4の場合，多段結合網であるサーキュラオメガ網[YaSK91]を用いているため，プロセッサ間通信のディレイ等の値は，平均の段数を用いて求めた．

EM-4上での上記パラメータは，以下のようになる．

$$
\begin{aligned}
tc &= 0.32 \quad (\mu s)\\
te &= 0.16 \quad (\mu s)\\
tlm &= 0.16 \quad (\mu s)\\
\delta &= 0.8 \quad (\mu s)\\
taw &= 1.04 \quad (\mu s)\\
tar &= 2.8 \quad (\mu s)\\
tlp &= 0.24 \quad (\mu s)
\end{aligned}
$$

6.2 プログラムのパラメータ化

対象とするプログラムから以下のパラメータを抽出する．

(1) π sにおけるイテレーション間に渡るデータ依存数 Nd
(2) π s内の参照配列数 Nrs
(3) π s内の定義配列数 Nws
(4) π s内の演算数 Nes
(5) π p内の参照配列数 Nrp
(6) π p内の定義配列数 Nwp
(7) π p内の演算数 Nep

上記のパラメータにおいて，参照配列数には，プロセッサ間通信によって受け渡される配列要素は含まない．すなわち，リモートメモリリードが必要な配列数を示す．図4のプログラムについて上記パラメータを求めた結果を表5に示す．Prg-BのNwsはA(I)とB(I)の2つであるが，π pにおいてB(I)を再定義している為，A(I)の1つのみとしている．

表5 プログラムのパラメータ化

	Nd	Nrs	Nws	Nes	Nrp	Nwp	Nep
Prog-A	2	0	1	1	0	1	1
Prog-B	3	0	1	2	0	1	1
Prog-C	1	1	1	1	1	1	1

6.3 ループ間Doacrossの理論実行時間算出

上記のパラメータを用いてループ間Doacrossの理論実行時間を求める．理論実行時間を，図6を用いて説明する．ループ間Doacrossの実行時間は，図6の❶の部分と❷の部分の実行時間の合計で求めることができる．

まず，❶と❷の部分の部分の実行時間を求める．イテレーション回数をNとすると，図より明らかなように，

$$❶ = \pi s \times (N/k) + \delta \times (N/k - 1) \quad \text{.....(1)}$$
$$❷ = \pi p + (\text{先行リード}) \quad \text{.....(2)}$$

となる．

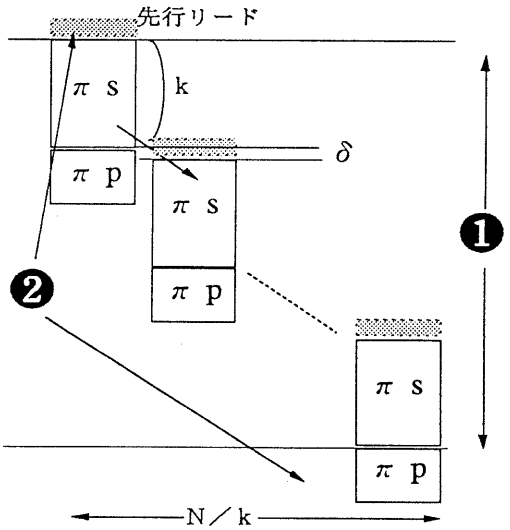


図6 理論時間説明図

(先行リード)は，π sで必要な配列（イテレーション間でデータ依存関係を伴わない配列）を前もってリモートリードしローカルメモリ上にストアするための時間であり，

$$(\text{先行リード}) = \{Nrs \times (tar + tlm)\} \times k + tlp \quad \text{.....(3)}$$

π sの時間は，π s内で定義された配列要素を一時的にローカルメモリ上にストアすることにより，

$$\pi s = \{(Nrs + Nws) \times tlm + Nes \times te\} \times k + Nd \times tc + tlp \quad \text{.....(4)}$$

π pの時間は，π s内でローカルメモリ内にストアされた配列要素をもリモートライトするので，

$$\pi p = \{Nws \times (taw + tlm) + Nwp \times taw + Nrp \times tar + Nep \times te\} \times k + tlp. \quad \text{.....(5)}$$

以上の(1)～(5)式を用いて実行時間を算出することができる．

6.4 評価結果

6.3の結果得られた式に各種パラメータの値を代入すると，理論実行時間が得られる．例えば，EM-4でProgram-A,B,Cを実行する場合の実行時間は，

Program-Aの実行時間(μs)
$=(1.92/k+0.32)\times N+2.4k$

Program-Bの実行時間(μs)
$=(2.24/k+0.48)\times N+2.4k$

Program-Cの実行時間(μs)
$=(1.6/k+0.48)\times N+5.36k$

となる．実測値と比較した結果を図7に示す．図7より，理論値と実測値がほぼ等しくなっていることがわかる．また，理論式から求められた値から最適なkを選択した場合と，実測値から得られた最適なkの値を比較した表を表6に示す．表中，○印のついたところが実行時間最小となる点である．表より，Program-A,B,C共に，理論値と実測値の両方で求められる最適なkの値が等しいことがわかる．

このように，コンパイル段階で理論式を計算することにより，実行時間を最小にする最適なkを求めることができることがわかった．

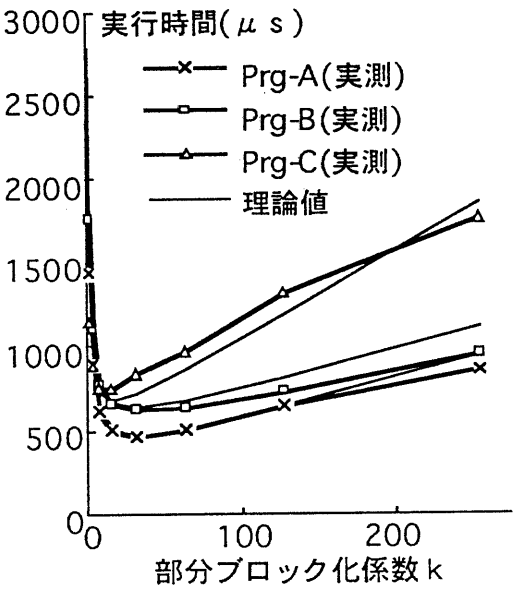


図7　ループ間Doacrossの実測値と理論値の比較

表6　部分ブロック化係数kの選択(○印)と実測値比較(単位μs)

k	Prg-A 実測	Prg-A 理論値	Prg-B 実測	Prg-B 理論値	Prg-C 実測	Prg-C 理論値
8	627	593	776	798	768	740
16	512	489	664	674	(760)	(680)
32	(474)	(466)	(632)	(640)	840	715
64	506	512	640	681	976	861

7．まとめ

本報告では，ループ間Doacrossを提案し，通信ディレイδ及びリモートメモリライト/リード時間ta，データ通信準備時間tcが全体の実行時間に与える影響を小さくし，分散メモリ型の並列計算機上でのループの実行時間を短縮できることを示した．並列計算機EM-4を用いた評価の結果，ループ間Doacrossを適用することによって，Doacrossの1.8～3.6倍，Pipeliningの3.3～5.7倍，data owner computes ruleに基づく実行の1.8～3.6倍の処理速度向上を確認できた．

また，ループ間Doacrossを実行時間を最小にするためには，部分ブロック化係数kの決定が重要であることを示すと共に，対象とする並列計算機が持つ各種パラメータとプログラム情報からコンパイル段階で決定できることを示した．

今後は，本方式による並列化手法及び投機的実行を取り入れた並列化コンパイラの開発を行って行く予定である．

謝辞

本研究を遂行するにあたり御指導，御討論いただいた太田情報アーキテクチャ部部長ならびに計算機方式研究室の同僚諸氏に感謝いたします．

文献

[BCKT79] U.Banerjee, S.Chen, D.J.Kuck, R.A.Towl. "Time and Parallel Processor Bounds for Frortan-like Loops," IEEE Trans. on Comp., C-28, 9, 660-670 (1979).

[Cytr86] Ron Cytron:"Doacross: Beyond Vectrization for Multiprocessors ", Proc. of Int. Conf. on Parallel Processing '86,pp.836-844(1986).

[PaKL80] D.A.Padua, D.J.Kuck, D.H.Lawrie. "High-Speed Multiprocessors and Compilation Techniques," IEEE Trans. on Comp., C-29, 9, 763-776 (1980)

[Poly88] C.D.Polychronopoulos:"Parallel Programming and Compilers," Kluwer Academic Publishers (1988).

[S KSY 94]佐藤,児玉,坂井,山口:"並列計算機EM-4の並列プログラミング言語EM-C",情報処理学会論文誌, Vol.35, No.4, PP.551-560(1994)

[SYHK89] S.Sakai,Y.Yamaguchi,K.Hiraki, Y.Kodama,T.Yuba :"An Architecture of a Dataflow Single Chip Processor", Proc. of 16th Ann. Symp. on Comp. Arch., pp.46-53(1989)

[YaSK91] Y.Yamaguchi,S.Sakai,Y.Kodama : "Synchronization Mechanisms of a highly parallel dataflow machine EM-4", IEICE Trans.,E74,1 (1991).

[YSKS94] 山名,佐藤,児玉,坂根,坂井,山口:"ループ間Doacross方式の並列計算機EM-4上での評価",第48回情処全大,2B-1,(1994)