計算機アーキテクチャ 48－4
（1983. 3. 18）

シミュレーションによるデータフロープロセッサアレイ計算機の評価

高橋直久[+] 雨宮真人[+] 大渕竜太郎[++]
（[+]日本電信電話公社 [++]電気通信大学）

1. はじめに

近年，高性能並列計算機としてデータフローマシンが注目され，数値計算をはじめ記号処理，画像処理，交換処理など多くの分野において研究が活発に進められている。[1,2] データフローマシンは，原理的には柔軟性に富み，プログラムの持つ並列性を自然な形で引き出すことができる。しかし，実際にデータフローマシンを実用システムへ適用するためには，通信オーバヘッドをどのように軽減するか，或いは，応用問題をどのように演算器に割付けるかなど多くの課題が残されている。また，これまでに提案されたシステムについても性能の評価はまだ十分ではない。

先に提案したデータフロープロセッサアレイ計算機は，科学技術計算の分野において十分に汎用性を確保してプログラムの作成を容易にするとともに超高速処理を実現することを目差している。この方式のねらいは，プロセッサ間転送制御とプロセッサ内の実行制御をデータ駆動概念で統一して高度の非同期並列処理を実現することにある。方式上の特徴は，通信の局所性を保つように結合系を構成してデータ転送遅延を小さくしたこと，及び，実行時の適応的な資源割付と実行前の静的な資源割付とを組み合わせて柔軟性が高くオーバヘッドの少ない資源割付を実現したことである。

我々は，これまで，提案方式に関して要素プロセッサや結合系の構成法や高級言語の検討を行ない，これらの検討のためのツールとして実験システムを開発した。[3,4,5,6,7] 本稿では，提案方式の特性評価の第一段階として楕円型偏微分方程式を差分法により反復計算する典型的な3つのデータフロープログラムを実行させた時の動作特性を実験システムにより測定し評価を加えたので報告する。

本稿では，まず第2章でデータフロープロセッサアレイ方式のアーキテクチャの特徴と，データフローマシンで性能上問題となる点を示す。次に，第3章で実験システムの構成を示す。ここでは，特に，実験システムによりデータフロープロセッサアレイ計算機をシミュレートする際の要素プロセッサのモデルと特性パラメータについて述べる。第4章では，シミュレーションの各種条件と実験結果を示す。この結果に従い，第5章で，データフロープロセッサアレイ方式の特性について考察を加える。

2. データフロープロセッサアレイ計算機のアーキテクチャ

2.1. アーキテクチャの特徴[4,5]

データフロープロセッサアレイ計算機はホストプロセッサと2次元アレイ状に結合した多数の要素プロセッサ（PE）とから成る。PEは，図1に示すような，命令メモリ（IM），オペランドメモリ（OM），演算ユニット，通信制御部，リンクメモリ（LM），から成るサーキュラパイプライン型のデータフロープロセッサである。以下に動作概要を示す。

データフローグラフの各ノードは，演算を指定するオペレーションノードと演算結果の分配先を指定するリンクノードに分解され，それぞれIM，LM

に格納される。IMは，データトークンが到着するとオペレーションノードをフェッチして，トークンとノードをOMに送る。OMでは，送られたトークンにより オペレーションノードの入力オペランドが全て揃うと，トークンとノードに待ち合わせていたオペランドデータを付与して演算ユニットに送る。他方，入力オペランドが揃わない場合には，送られたトークンをOMに格納して待ち合わせる。

演算ユニットは，複数の演算器から成り，実行可能なオペレーションノードを並列実行し，その結果を通信制御部に送る。通信制御部は，演算ユニット或いは隣接PEから演算結果を受け取りLMに送る。但し，演算結果の送り先が自PEでない場合には送り先PEへの転送経路となる隣接PEへ送り出す。LMでは，リンクノードをフェッチして分配先のオペレーションノード名を求めて，各ノードに対して演算結果を分配する。

データフロープロセッサアレイ方式の主な特徴は次の通りである。

（1）高並列処理

自律的に実行を進める多数のPEの並列実行による高度なMIMDシステムを実現する。PE内では複数の演算器による並列処理を行ないシステム全体では多数のPEが並列動作する。

（2）アレイ結合方式

通信の局所性を活かすように転送路を分散化したデータフローマルチプロセッサの構成である。また，流体シミュレーションや行列計算など多くの科学技術計算で有効なアレイ結合を実現して，近傍PE間の転送を特に高速にする。

（3）半動的な資源割付

実行時に適応的に行なう動的な資源割付と実行前の静的な資源割付とを組合わせることにより柔軟性が高く，オーバヘッドの少ない資源割付方式を実現する。たとえば，配列処理のプログラムでは，図2のように実行前に配列要素を規則的なマッピング法に従いPEに割付け，実行時に個々の演算命令にPE内の演算器群の中の一つの演算器をデータ駆動原理に従い動的に割付ける。

（4）非同期制御

関数呼出しや繰返し処理及び配列データの処理に対してタグ付きトークンの概念[8]を適用してプログラムの非同期並列実行を実現する。[4,5] 特に，配列データの処理では，各配列要素に対してユニークな名前を付与して配列デ

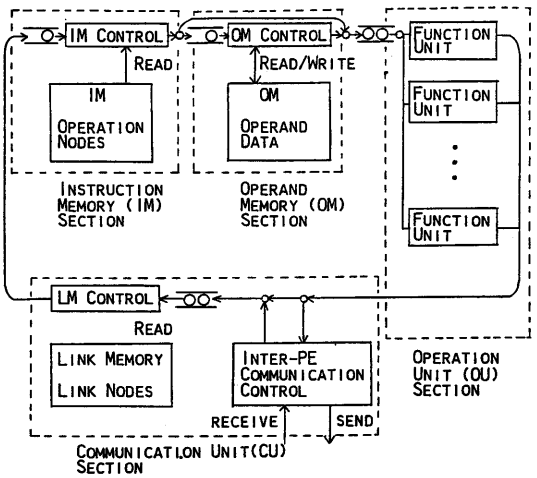


図1．PEの論理構成

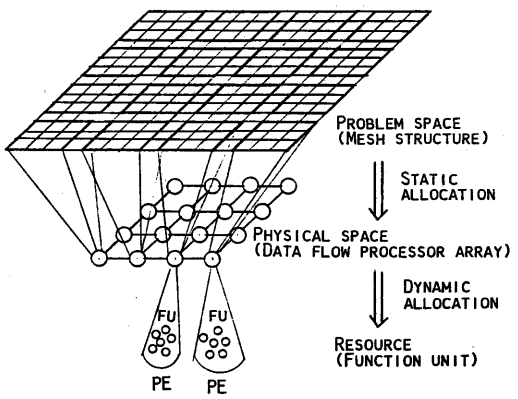


図2．資源割付方式

ータの値の計算と参照を配列の各要素間で完全に非同期に行ない並列性を高める。また，配列データに対してもポインタを用いないで値そのものをオペレーションノードに送ることを基本とすることによりポインタ操作のオーバヘッドやメモリのアクセス競合をなくす。

（5）状態の保存

通常のオペランドは演算を行なうと値が不要となるのでOMから消去される。しかし，値を一定期間（たとえば，関数の実行中或いはループの実行中）OMに保持してオペランドが状態を持つことができる。これにより，オーバヘッドなく特定データを複数の実行環境下の演算で使用できる。

2.2. データフローマシンに内在する問題点

超高速データフローマシンを実現する場合に性能上特に考慮すべき問題を以下に示す。

（1）通信路の構成と通信遅延

データフローマシンでは，演算間のデータの引き渡しのために「通信」が必須となる。この通信の遅延による性能の劣化をいかに抑えるかが大きな問題である。通信の局所性を活かしたり，パイプライン効果を引き出したりする方法などが考えられるが，これらの方法が実際の応用においてどの程度有効であるかを明らかにする必要がある。また，大規模なシステムでは，VLSI化が前提となるので，ピンネックを生じさせないためにデータトークンの分割転送が必要となろう。このために生じる転送路の実質的な速度低下に対しても高性能を維持できなければならない。

（2）実行時の負荷の偏り

マルチプロセッサ構成でプログラムの持つ並列性を十分に引き出すためには，時間，空間的に負荷を分散するようにプログラムをPEに割付ける必要がある。本方式のようにPEの割付を静的に行なうと割付けのオーバヘッドは小さいが実行時に負荷の偏りが生じて並列性を十分に引き出せない恐れがある。

（3）制御命令のオーバヘッド

タグ付きトークンの概念を実現してプログラムの持つ並列性を十分に引き出すためには多数の制御命令が必要となる。たとえば，繰返し処理の場合には，逐次型計算機ではコントロールフローを制御するために数命令要するだけであるが，データフローマシンでは繰返して使う変数それぞれについてデータフローを制御する命令と新しい実行環境名を付与する制御命令が必要である。

3. 実験システム

3.1. 実験システムの構成[4]

次のような目的により実験システムEddy（Experimental system for data driven processor array）を開発した。

（1）データフロープロセッサアレイ方式の有効性と問題点を定量的に明らかにする。

（2）種々のシミュレーションパラメータを変更してシステム特性を評価することによりハードウェアシステムの詳細設計の指針を得る。

（3）関数型高級言語Valid[7]の実験環境を与えてValidの有効性を検証する。

Eddyのハードウェアは図3に示すように4×4のPEと2個のブロードキャスト制御装置（BCU）から成る。各PEは2個のマイクロプロセッサから成り，8隣接PEと結合されている。各PEのソフトウェアが次節に示すモデルに従いデータ駆動型プロセッサを

詳細にシミュレートして特性データを発生する。

3．2．PEのシミュレーションモデル

PEのサーキュラパイプラインを流れるパケットのスループットを最大にすることを狙って，次のような観点から図1に示したPEの各ブロックを詳細なモジュールに分解すると図4のシミュレーションモデルが得られる。

（1）各モジュールの処理時間を小さくするためにできる限り機能を分割する。但し，複数のモジュールから同一メモリにアクセスすることは禁じる。

（2）パケットの調整・分配はそれぞれ1モジュールとして独立させる。その他の機能では，メモリをアクセスしない場合には独立のモジュールにしないで他のモジュールに含めることによってパイプライン段数をできるだけ小さくする。

ここで，各モジュールをパイプラインの1段と考え，各モジュールはパケットを逐次的に処理して後段のモジュールに渡す。また，各モジュールは，論理的なシミュレーションクロック（以下単にクロックと呼ぶ）に従ってそれぞれの機能を進める。この時，各モジュールの稼動時間（クロック数）やキューの長さ等の特性が得られる。これらの測定データにより次のような性能指標が求められる。

（a）実行時間 Te ： プログラムの実行開始から終るまでに要するクロック数。

（b）稼動率 ρ ： 演算器が実際に稼動している割合を示すもので，次式で与えられる。

$$\rho = \frac{\{\text{演算器モジュールの稼動時間の総和}\}}{Te \times \{\text{演算器モジュールの総数}\}}$$

（c）並列度 P ： 平均して何個の演算器が同時に稼動しているかを示すもので次式で与えられる。

$$P = \rho \times \{\text{演算器モジュールの総数}\}$$

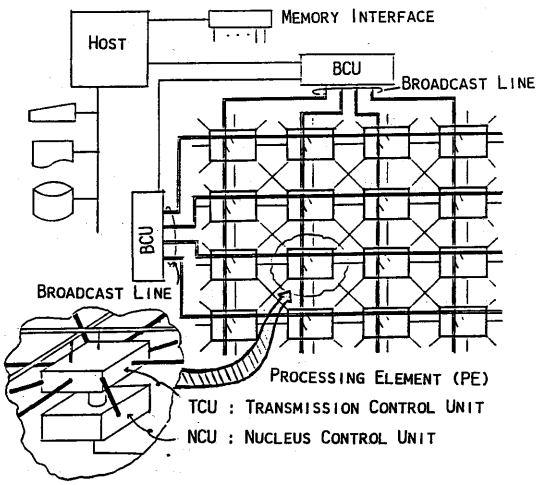


図3．実験システム Eddy の構成

4．評価実験[9]

4．1．例題プログラム

楕円型偏微分方程式を差分法により反復計算する問題を例題とした。二次元問題では空間座標を x，y 方向の格子に分割し各格子点の解を X_{ij} とすると，次のような差分方程式の組みが得られる。

$$a_{ij}X_{ij-1} + b_{ij}X_{i-1,j} + c_{ij}X_{ij} + d_{ij}X_{i+1,j} + e_{ij}X_{ij+1} = g_{ij} \quad (i=1,2,\cdots,N_x,\ j=1,2,\cdots,N_y) \cdots (1)$$

ここでは，次の3つの典型的な解法についてその動作特性を得るためにデータフロープログラムを作成し，一定

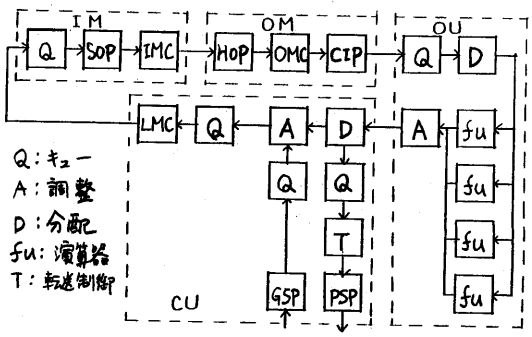


図4．PEのシミュレーションモデル

回数だけ反復させて特性データを得た。なお，ヤコビ法とADI法では上式の係数 a_{ij}～e_{ij} を1とし g_{ij} を0として各係数を用いた演算は排除したプログラムとした。

（1）ヤコビ法

第k近似 $X_{ij}^{(k)}$ から第(k+1)近似 $X_{ij}^{(k+1)}$ を次式により求めていく。データの従属関係が非常に単純であり全格子点の値を並列に求めることが可能である。

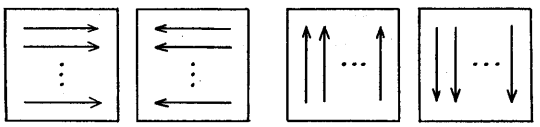
$$X_{ij}^{(k+1)} = (X_{i+1,j}^{(k)} + X_{i-1,j}^{(k)} + X_{i,j+1}^{(k)} + X_{i,j-1}^{(k)})/4 \quad \cdots(2)$$

（2）red-black SOR法

格子点の座標を(i,j)とすると，まず i+j＝偶数となる格子点について第k近似 $X_{ij}^{(k)}$ から第(k+1)近似 $X_{ij}^{(k+1)}$ を求める。次に i+j＝奇数となる格子点について第(k+1)近似 $X_{ij}^{(k+1)}$ から第(k+2)近似 $X_{ij}^{(k+2)}$ を求める。このように2回の反復で全格子点が計算され，全格子点の1/2が並列に処理される。

（3）ADI法

ADI法は陰解法であり，(1),(2)の解法に比べてデータの従属関係が複雑である。計算は，まず各行並列に実行でき図5-(a)のように水平方向に往復するように進められる。次に，各列の計算が並列に，図5-(b)のように垂直方向を往復するように進められる。



(a) 水平方向のスイープ　(b) 垂直方向のスイープ

図5. ADI法の計算の流れ

4.2. シミュレーション条件

次のような条件でシミュレーションを行なった。

（1）マシン条件

- PE台数 --- n×n （n=1～4）
- PE当りの演算器数 --- f （f=1～4）
- 各モジュールでの1パケット当りの処理時間（クロック数）
  - 演算器 --- T_f（T_f=5～80）
  - LM制御部 --- l（lは分配数であり，1クロック毎に1パケット分配する）
  - 転送制御部 --- T_t（T_t=1～50）
  - その他のモジュール --- 1
- キューの最大制限長
  - LM用キュー --- Q_{LM}（Q_{LM}=16～∞）
  - 演算器用キュー --- Q_{FU}（Q_{FU}=16～∞）
  - その他のキュー --- 無限長

（2）プログラム条件

- 格子の大きさ --- M×M （M=4～32）
- 反復回数
  - ヤコビ法 --- 32回
  - red-black SOR法 --- 16回
  - ADI法 --- 16回

（3）マッピング法

次の2つの規則的なマッピング法に従い格子点 (i,j) を PE_{kl} に割付ける。

（a）隣接マップ

互いに隣接する「M/N」×「M/N」の格子点を1つのPEに割付ける。

（b）モジュロマップ

$k = i \bmod n$，$l = j \bmod n$ を満す格子点 (i,j) を PE_{kl} に割付ける。

4.3. 命令ミックスの解析

例題プログラムについて命令の動的な出現頻度を調べ，演算の型及び命令種別に関して整理する。

（1）演算の型と出現頻度

演算は，オペランド数により単項演算と二項演算に分けられる。更に，二項演算は，オペランドの属性に従ってV-V型（両方とも変数），C-V型（定数と変数），L-V型（ループ定数と変数）に分類される。ここで，ループ定数とはループ内で値が変わらずに繰返し用いられるものであり，状態の保存によりループの実行中OMに保持される。

表1. 演算の型で分類したノード数の割合

		点ヤコビ法	red-black SOR法	ADI法
単項演算		29%	19%	19%
二項演算	V-V型	31%	38%	31%
	C-V型	20%	10%	29%
	L-V型	20%	33%	21%

表1.は各プログラムの実行命令を演算の型に従い分類したものである。表より，L-V型の演算が比較的多いことがわかる。この結果，状態の保存制御が性能向上に大きく寄与すると考えられる。

（2）命令の種別と出現頻度

図6，7は，点ヤコビ法とADI法のプログラムの命令について動的出現頻度の高い順に並べて累積分布を表わしたものである。いずれのプログラムでも，Gate，SW，Sendなどの制御命令の比率が高い。ここで，Send命令は，隣接格子点にデータを送るsend_e，send_w，send_s，send_n命令[4]の総和である。これらの命令は逐次型計算機でのインデクス計算に対応するものと考えられる。

4.4. 動作特性の解析

（1）問題の大きさと並列度の関係

一般に問題の大きさによって，得られる並列度は変化する。特に，ここでとりあげたプログラムでは格子が大きくなるにつれてプログラムの持つ並列度が増大する。そこで，各解法のプログラムについて格子の大きさがどの程度になればシステムの並列演算機能を十分に働かせることができるかを調べた。その結果を図8～10に示す。

図8，9より，点ヤコビ法とred-black SOR法では格子が256（16x16）程度で並列度が飽和することがわかる。特に，$T_S=10$の時には，並列度がシステムの演算器総数64たまで達する。

ADI法では，図10のように，ここで測定した範囲（MxM≦24x24）では，並列度の飽和は見られない。これは，システムに潜在する並列度がプログラムの持つ並列度よりも大きく，資源にまだ十分な余裕があることを示している。

（2）キューの長さと実行時間の関係

PEをハードウェアで実現する際には，各キューは有限の長さを持つことになる。この時，パケットの到着が集中すると行列が一杯となりブロッキングが発生して性能が低下する恐れがある。そこで各キューの長さを制限した時のシステムの動作特性を調べた。

図11に，LMのキューの制限値と実行時間の関係を示す。図より，LMのキューの制限値が16の時には，キューが十分に長い場合に比べて8%程度実行時間が延びることがわかる。また，LMのキューの長さを256程度以上にすれば実行時間はキューが十分に長い

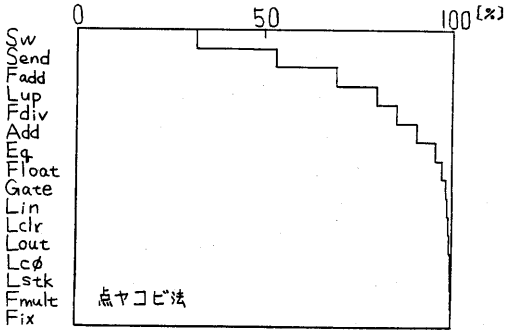


図6 命令の動的出現頻度（1）

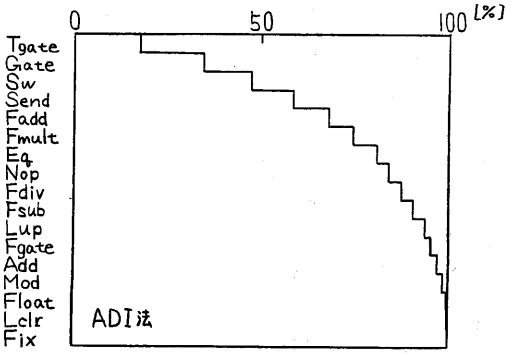


図7. 命令の動的出現頻度（2）

時の値に落ち着くことがわかる。

なお，演算ユニットのキューの制限値を16程度に小さくするとブロッキングは発生するが実行時間は延びないことが確認された。

（3）パイプライン遅れと稼動率の関係

ADI法のように格子点間のデータ従属関係が強いプログラムでは，パイプライン遅れが大きいと演算器に遊びが生じる恐れがある。そこで，パイプライン遅れと演算器の遊びの関係を調べるためにパイプライン遅延比 Rp（Rp = 演算器以外のパイプライン遅延/Tf）と稼動率 P の関係を，マッピング法と1PE当りの演算器個数 f をパラメータとして求めた。その結果を図12に示す。図より，モジュロマップの時には，Rp が変化すると P が大きく変化することがわかる。

（4）PE間転送遅延と実行時間の関係

通信の遅延によりどの程度性能劣化が生じるかを調べるために，PE間転送遅延の大きさ Tt を変えた時の実行時間 Te の変化を各マッピング法について求めた。図13のように点ヤコビ法では，隣接マップの時には Te はほとんど変化

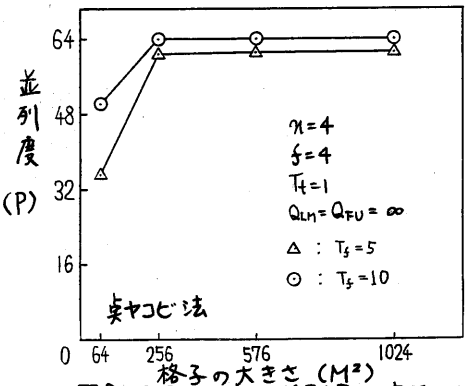


図8 問題の大きさと並列度の関係(1)

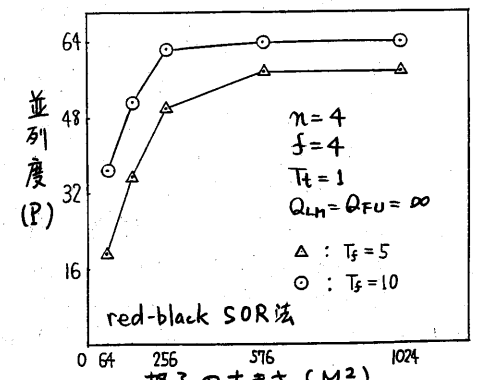


図9. 問題の大きさと並列度の関係(2)

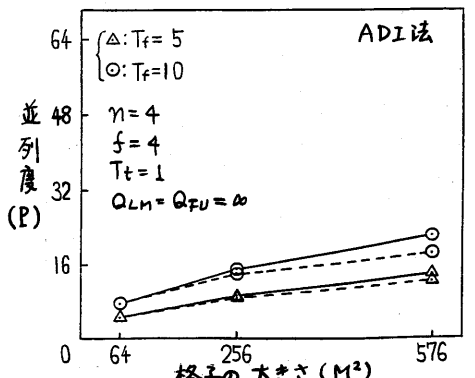


図10. 問題の大きさと並列度の関係(3)

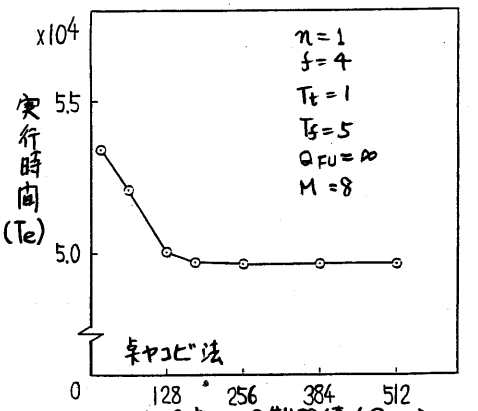


図11. キューの長さと実行時間の関係

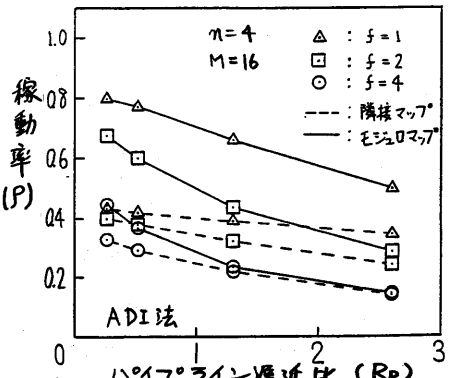


図12. パイプライン遅れと実行時間の関係

しないが，モジュロマップの時にはTeが10程度以上になるとTeが急激に増大することが確認された。また，ADI法では，図14のようにいずれのマッピング法でもTeの急激な変化は見られない。

5. 考察

2章で述べたデータフロープロセッサアレイ方式の特徴と問題点に関して実験結果に基づき考察を加える。

5.1. 並列処理の効果

(1) データ従属性の弱いプログラムの場合

点ヤコビ法やred-black SOR法など従属性の弱いプログラムの実行では，

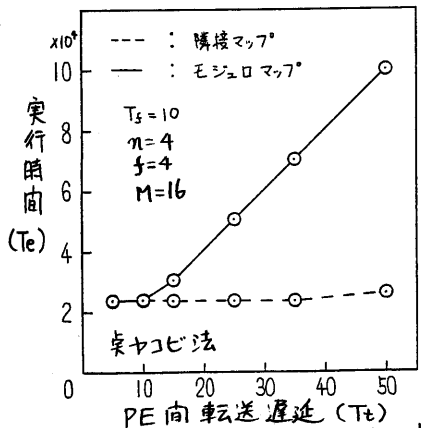


図13. PE間転送遅延と実行時間の関係(1)

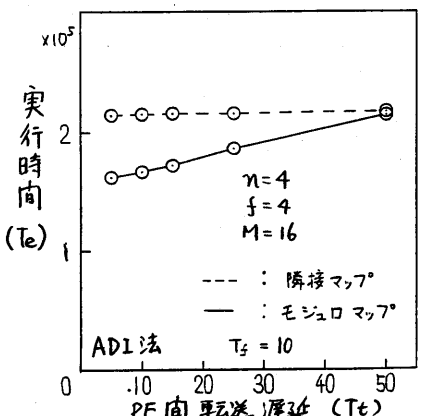


図14. PE間転送遅延と実行時間の関係(2)

システムの持つ並列処理能力を最大限発揮している。即ち，問題の大きさが16格子点/PE程度以上の時には，プログラムの実行時間が1/{演算器総数}となる。

(2) データ従属性の強いプログラムの場合

図15のように格子の一片の長さMと実行時間Teとの関係を表わすと，ADI法ではモジュロマップの時にTeがMにほぼ比例して増大することが示される。このことから，ADI法では処理量が$\Theta(M^2)$で増大するのに対して実行時間を$\Theta(M)$程度の増加で抑えられることがわかる。従って，ADI法のようにデータ従属性が強いプログラムでもプログラムに内在する並列度を十分に引き出しているといえる。

5.2. 分散化の効果と問題点

PE間の転送遅延とパイプラインの遅れに関して次のことを指摘できる。

(1) 図13, 14より，PE間転送遅延が10クロック程度まではマッピング法によらず実行時間はほとんど変化しないことがわかる。いいかえると，PE間転送用パケットを1/10程度に分割転送しても性能が劣化しない。このようにPE間の転送路幅を狭くすることができることは，VLSI化して小型化し

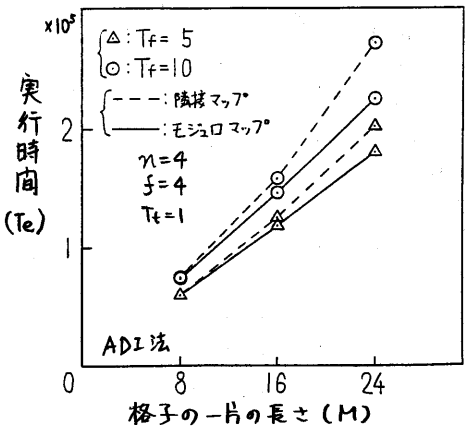


図15. 格子の一片の長さと実行時間の関係

たPEを多数結合したシステムを構成する上で大きな利点となる。

（2）図12に示したように，ADI法では，稼動率はパイプライン遅れの割合に大きく左右される。従って，データ従属性の比較的強いプログラムに対してもより高速に処理するためには，パイプライン遅れの割合をいかに小さくするかということが大きな課題である。

5．3．マッピング方式

ここで用いたデータフロープログラムでは，実行前に格子点を規則的にPEに割付けても高い並列度が得られることが確認された。また，図5に示したようなデータ従属関係をもつADI法プログラムでも，マジュロマップを行なうと負荷を時間，空間的に分散させて負荷が特定PEに偏らないようにすることができる。このことは，図16のように各マッピング法についてPE内の各演算器の稼動率を調べると明らかとなる。即ち，隣接マップでは大きな負荷がまとまって各PEを巡回するように流れるので各PEでは大きな負荷が短期間に集中する。このためPE内の4個の演算器の稼動率は互いにほぼ等しい低い値となる。他方モジュロマップでは，負荷が全PEに分散されるので各PEには比較的小さい負荷が連続的にかかると考えられる。PE内では演算器1から4の順に空き状態の演算器を探して実行可能な命令を割付けるので，

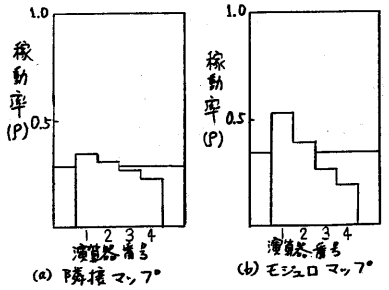


図16．PE内の各演算器の稼動率(ADI法)

図16-(b)のように演算器1の稼動率だけが高くなる。

5．4．非同期制御の効果と問題点

（1）非同期制御の効果

PE間転送遅延が大きくなってもプログラムの実行時間が増大しない原因のひとつとして各PEが非同期処理を行なっていることが考えられる。即ち，PEは各格子点をデータ駆動メカニズムに従い非同期に処理するので，各PEに割付けられた格子点のうち実行可能なものを動的に選択して実行している。たとえば，隣接マップの場合に，各PEに割付けられた格子点のうち，隣接PEからの転送データを使う必要のある外側の格子点の計算が遅れてもその間に内側の格子点の計算が進められるので演算器に遊びが生じない。

（2）オーバヘッド

偏微分方程式の解析では浮動小数点演算が本質的に必要なものであり，繰返し制御命令やインデクス計算命令は二次的なものである。そこで，浮動小数点演算以外の命令をすべてオーバヘッドと考え，全命令の実行回数に対する浮動小数点命令の実行回数の割合を調べて逐次型計算機の場合と比較すると表2が得られる[9]。ここで，逐次型計算機としては3オペランド演算命令と，各オペランドに対してインデクス修飾を同時に使えるような比較的強力なアドレッシングモードを持つプロセッサ（たとえばIntel社のiAPX-432）を想定した。

表2では，本方式と逐次型計算機の間で大きな差が見られない。本方式のように色付きトークンの概念を配列要

表2．浮動小数点演算命令の割合

	ヤコビ	ADI
本方式	21.4%	22.8%
逐次型 計算機	20.3%	23.4%

素にまで適用し，更に状態の保存を可能にすると，複数の配列要素に同一操作を施す場合にもベクタプロセッサのように1命令の実行で済むのに対して，逐次型計算機では複数の命令により繰返し制御やインデクス計算を行なう必要がある。他方，ループ処理を実現する場合にはこれと逆に逐次型計算機よりもデータフローマシンの方が命令数が増加する。これらの効果が相殺して両者の間に大きな差が生じないと考えられる。

6. おわりに

超高速科学技術計算向きデータフロープロセッサアレイ計算機の動作特性をシミュレーションにより解析し，該方式の評価を行なった。その結果，ヤコビ法やred-black SOR法などのプログラムのように格子点間のデータ従属性が弱い場合にはシステムに潜在する並列処理能力が最大限働くことを確認した。また，ADI法プログラムのように格子点間のデータ従属関係が強い場合でもプログラムの並列性を十分引き出し得ることがわかった。更に，プロセッサ間のデータ転送遅延がある程度大きくなってもプログラムの実行時間が増大しないことを示してプロセッサ間でパケットの分割転送が可能であることを明らかにした。

今後の課題としては，LU分解や有限要素法などデータ従属関係が複雑で遠隔プロセッサ間のデータ転送を含むようなプログラムに対して本方式を適用した場合の特性評価を行なうこと，パイプライン遅れの割合をどのようにして小さくするかなどがある。

謝辞

本研究に関して御指導頂いた山下鉱一郎研究室長，ならびに実験システムEddyのハードウェアを作成された吉田雅治主任に深く感謝します。また日頃御指導御討論頂く塚本克治室長はじめ第八研究室の皆様に感謝します。

参考文献

[1] Treleaven, P.C., Brownbridge, D.R., and Hopkins, R.P., "Data-Driven and Demand-Driven Computer Architecture," Computing Surveys, Vol. 14, No. 1, PP. 93－143, (March 1982).

[2] 秋山，山下，「データフローアーキテクチャの通信システムへの応用」信学誌 Vol. 66, No. 2, PP 191－193, (1983).

[3] 高橋，雨宮，「超高速科学技術計算を指向したデータフロープロセッサアレイ計算機の提案」，信学技報 EC 80-24, PP. 67-78 (1980).

[4] 高橋，吉田，雨宮，「データフロープロセッサアレイ計算機の構成と実験システム」信学技報 EC 81-73, PP 29-40, (1982).

[5] Amamiya, M., Takahashi, N., Naruse, T., and Yoshida, M., "A Data Flow Processor Array for Solving Partial Differential Equations," Proc. of Inter. Symp. on Applied Mathematics and Information Science, Kyoto Univ. (1982).

[6] 成瀬，雨宮，「科学技術計算向きデータフロー計算機に用いる相互結合ネットワークの性能評価」信学技報 EC82-36, PP31-42 (1982).

[7] 雨宮，尾内，「データフローマシン用高級言語 Valid について」信学技報 EC82-9, PP 35-64 (1982).

[8] Arvind, Gostelow, K., Plouffe, W., "An Asynchronous Programming Language and Computing Machine," Information and Computer Science, University of California, Irvine, TR 114a, (Dec. 1978).

[9] 大渕，「データフロープロセッサアレイ計算機アーキテクチャのシミュレーションによる評価」昭和57年度電気通信大学修士論文，(1983).