解　説

3. 超並列マシンの応用と評価

3.1 科学技術計算シミュレーション超並列計算機 ADENA†

野　木　達　夫††

1. はじめに

科学技術計算シミュレーションではどんな計算ができればよいかを考えていくとき，並列計算機なら自然に ADENA に行き着くことを明らかにしたい[1]．

基本的なデータ構造といえば配列である．物理現象をシミュレーションする場合，考えている物理空間内に規則正しく配置した代表点網（格子点網）における未知量は配列で表すのが自然である（連続体モデル）．あるいは連続体を粒子の集合とみなす場合もある（粒子モデル）．粒子に番号付けをすると粒子の物理量はやはり配列になる．

配列データは，配列添数について DO 処理という単純な制御を許す．配列は DO の逐次処理だけでなく，並列処理を促す魅力をもっている．ADENA はこの点を積極的に活かすように構想したものである．

2. 分散メモリシステム

2.1 メモリシステム

通常の計算機では，どのようなデータであっても，1次元的にアドレッシングされた物理的メモリ配列に蓄えられる．2次元，3次元の配列データも1次元メモリ配列にマッピングされる．このことでなんら不自由でないのは，単一の CPU はメモリ全体にアクセスでき，アドレスの簡単な計算だけでことがすむからである．

ベクトル計算機でもその事情は変わらない．ただし1次元データ（ベクトル）に高速にアクセスできるよう複数メモリブロックを設けた．これで，多数のブロックにわたるデータの組をほとんど一度にレジスタにもってきたり，その逆の動作もできるようになっている．ところが一つのブロック内のデータを組にしたものにアクセスする場合，一度で済まなくなる．メモリコンフリクトがおこるのである．

*メモリコンフリクトの問題は，共有メモリをもつ並列計算機ではもっと深刻な問題として現れる．これを直接的に回避するためには*分散メモリ方式をとらなければならない．ADENA も各演算プロセッサに直結したローカルメモリをもたせたシステムである．

分散メモリ方式を選べば，もはやいずれの演算プロセッサもほかのプロセッサのローカルメモリには直接アクセスできなくなる．そうすると当然プロセッサ（のローカルメモリ）間のデータ転送の問題が浮上してくる．これが余分な時間のロス（*オーバヘッド）を生み，並列計算効率を下げる要因であり，それを克服することが分散メモリ並列計算機の最大の問題である．

2.2 配列の処理様式

転送問題の解決の糸口をもとめて，冒頭で指摘したように科学技術シミュレーション計算で扱う主要なデータが2次元や3次元の配列であることにもどって考えてみる．配列は，一方で DO 文による逐次処理に便利であるとともに，他方で並列処理を許す場合も多い，そういう両面性をもっている．どちらが重みをもつかはいろいろな問題やアルゴリズムによって違ってくる．

二つのベクトルの内積を計算する場合，対応する成分同士の積を決める部分はまさに並列処理に向いたものであるが，それらの積の和を求める部分は基本的には逐次処理になる．二つの行列の和を求めることはまったく並列処理可能であるが，一つの行列の最大要素を決める（ピボット選択）アルゴリズムは基本的には逐次処理になる．

通常の問題やアルゴリズムでは両方が混合して

† Super-parallel Computer ADENA for Scientific Simulation by Tatsuo NOGI (Division of Applied Systems Sceience, Faculty of Engineering, Kyoto University)

†† 京都大学工学部応用システム科学

*：用語解説にあることを示す記号

いる場合が多い．一般には洗練された解法ほど逐次処理の要求度が高くなる．素朴な方法ほど並列処理しやすい．

このような事情をみると，完全な並列処理をねらうよりも，はじめから適度な逐次処理を含むことを勘定にいれた並列処理を基本とするのが妥当である．さて2次元や3次元配列ならその1次元部分配列を逐次処理することが自然である．それは直接 DO 処理になじむからである．それをセグメントと呼ぼう．個々のセグメント処理には結構複雑なアルゴリズムを盛り込めるものである．当然セグメントの組全体については並列処理することを想定しているわけである．この全体をセグメント配列と呼ぶことにしよう．

2.3 分散メモリ上のセグメント配列

個々のセグメントは逐次処理の対象になるので，それが一つの分散メモリユニット内に格納されて，それに直結するプロセッサがアクセスして処理するようになっていると都合よい．

行列を想起しよう．セグメントといっても「行」もあれば「列」もある．当然これらを区別してどちらも扱えなければならないし，また上記のような逐次—並列処理の対象でなければならない（**図-1**）．ここでその区別の表記方法を与えよう．行列 $\{U(I,J), I=1,N, J=1,M\}$ に対して第 I 行のセグメントを

$\{U(/I/,1), U(/I/,2), \cdots, U(/I/,M)\}$

あるいは $U(/I/,)$

と表し，第 J 列のセグメントを

$\{U(1,/J/), U(2,/J/), \cdots, U(N,/J/)\}$

あるいは $U(,/J/)$

と表す．ここに見るとおり，スラッシュ付きの添字変数がセグメントを特定する．裸の添字変数が各セグメントの要素を特定する．またいずれの添字変数がスラッシュ付きかの違いによって「行」と「列」の区別を与える．これをセグメント配列の「方向属性」による区別ということにする．

当然スラッシュ付き添字変数はその値で特定されるセグメントを確保するべき分散メモリユニットをも特定しているとみなす．しかもいずれのセグメント配列であってもその $/I/$ 番目のセグメントはそれの方向属性のいかんにかかわらず同じ $/I/$ 番目の分散メモリユニットに確保するものとする．

3次元セグメント配列も同様に考えることができる．通常なら $\{U(I,J,K), I=1,N, J=1,M, K=1,L\}$ と表される3次元配列は，方向属性の違いによって3種類のセグメント配列として捉えることができる（**図-2**）．

$\{U(/I,J/,K)\}$，$\{U(I/,J,/K)\}$，

及び $\{U(I,/J,K/)\}$

ここでは分散メモリユニットが2次元配列になっていると想定し，二つの添字変数がスラッシュ対で囲まれている．ただし $U(I/,J,/K)$ においては K と I が囲まれていると見る．残り一つの裸の添字変数は各セグメントの要素を指定するものである．

なお，スラッシュ対で囲まれた添字変数がプロセッサや分散メモリユニットを特定するといったがこれはあくまで論理的な事柄であり，実際のシステムではプロセッサやメモリユニットの個数に限りがある（現有のものでは256）．したがって論理的に複数個のものが物理的にある1台のものに対応づけられる．しかしユーザはそのことに気を止めることなく，宣言しただけのプロセッサやメモリユニットが存在するものとしておけばよ

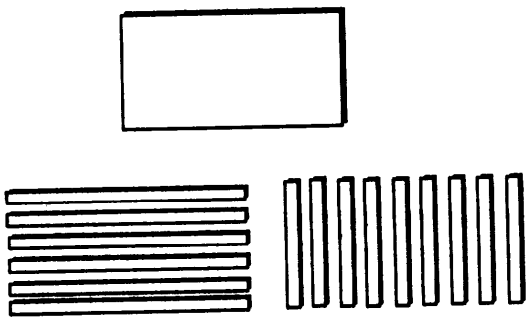

図-1 もとの行列（上）と行セグメント（左）と列セグメント

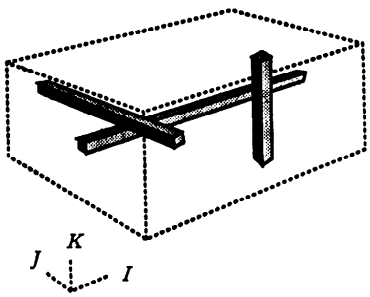

図-2 3方向セグメント

い．言わば仮想プロセッサ・メモリユニット方式をとっている．

3. 並列処理の基本形

3.1 セグメント配列の並列処理

いくつかの同一次元，同一方向属性のセグメント配列に対して，セグメントごとに斉一の処理を行わせる並列処理は簡単に実現できる．それがPDO構文である．たとえば行列 $\{A(/I/,J)\}$ と行列 $\{B(/I/,J)\}$ の和 $\{C(/I/,J)\}$ を求めるという場合，次のようなプログラムで表せばよい．

```
PDO I=1, N
DO 10 J=1, M
10 C(/I/, J)=A(/I/, J)+B(/I/, J)
PEND
```

PDO と PEND ではさんだ部分（PDO パラグラフという）が一つの並列処理単位になる．PDO文に現れる整数変数 I が並列処理に係るセグメントを特定するためのものであるが，変数範囲指定 $I=1,N$ でもって対象となるセグメントの範囲を与えている．PDO パラグラフ内部では各セグメントに対する処理が FORTRAN 表現される．各セグメントは1次元配列であるので，通常はその要素を順次処理する DO ループが登場することになる．

もしもとの行列が3次元配列の「断面」として $A(/I,J/,1)$ と $B(/I,J/,1)$ の形で与えてあり，和も同形で求めるなら次のようなプログラムで表してもよい．

```
PDO I=1, N, J=1, M
C(/I, J/, 1)=A(/I, J/, 1)+B(/I, J/, 1)
PEND
```

この場合には処理内容は単独の代入文だけである．行列の和を求めるということだけに限れば，後者のプログラムのほうが完全に並列処理を実現していると言える．一般にいずれを選択するかは，当の計算がどういうものか，あるいはどういうところに必要となったかという状況によって左右されるものである．たとえば二つの行列の乗算をとりあげると，後者のデータ表現よりも前者のもののほうがよい．

3.2 データ編集

今度は方向属性が異なる二つの行列 $\{A(/I/,J),I=1,N,J=1,M\}$ と $\{B(I,/J/),I=1,N,J=1,M\}$ が与えられているものとして，その和 $\{C(/I/,J),I=1,N,J=1,M\}$ を求めることにする．この場合，いきなり PDO 処理に入るわけにはいかない．A が行ベクトルごとに対応する分散メモリに確保されているのに対し，Bは列ベクトルごとに分散メモリに確保されているからである．結果の C を A と同じ方向属性のもので得たいのであるから B の代わりにもとの B と同じ値のセットをもつが方向属性が変更されたものを準備する必要がある．これはもとの列ベクトルの組から行ベクトルの組へのデータ編集になっているので，その操作を ADE (Alternating Direction Edition) とよぶ．いまその実現を指示するものとして PASS 構文を導入する．

```
PASS I=1, N, J=1, M
B(/I/, J)=B(I, /J/)
PEND
```

この PASS パラグラフにおかれた置換の式では，等号の両辺に異なる方向属性のものを書き，その右辺に編集のもとになるセグメント配列，左辺に編集後のものを置く．この編集が終わっていると行列の和を求める計算は前項で述べたとおりになる．

実は ADENA では，これまでに述べたセグメント配列とそれに対する ADE が最も基本的な概念である．これでもって多様な「データ転送」を実現する．たとえば，行列の乗算で必要になる列を行に「転置」して「放送（同一データの配送）」すること (POUR 構文) やピボット行の放送 (PAD 構文) も PASS の応用になる．

3.3 言語処理系

方向属性をもつセグメント配列を基本的データ構造として，それに対する並列計算と ADE を基本的操作とする言語体系及びその処理系を *ADETRAN と名付けた[2),4),5)]．

プログラミング言語としてはサブプログラム構造も重要である．詳しく述べる余裕がないが，基本的区分だけを紹介しておく．一般にホストプログラムと*スレーブプログラムを用意する．*ホストプログラムは通常の FORTRAN プログラムにほかならない．スレーブプログラムにはGサブルーチン，L サブルーチンそして L 関数などがある．

Gサブルーチンは ADENA 本体部分のための

サブプログラムであり，既述のセグメント配列に対する並列処理構文をふくむ．ホストサブプログラムがこのGサブルーチンを CALL することでADENA 本体が駆動される．Gサブルーチンの中の PDO パラグラフ中には個々のセグメントに対する処理が記述されるが，その内容の全部あるいは一部を一つのサブプログラムにまとめたのがL サブルーチンやL関数である．それらを PDO パラグラフ内で CALL したり，引用することができる．LサブルーチンやL関数の形態は FORTRAN そのものである．

4. ADENA ネットワーク

4.1 ADE 実現の論理的構成 1

ADENA では個々のプロセッサがそれにつながる分散メモリ上のセグメント処理を分担するものであるから，プロセッサ自身は通常のプロセッサ機能をもっておればよい．現在のシステムではそれぞれがオブジェクトプログラムをもち制御することにしている．したがってシステム全体としては MIMD タイプになっている．

ハードウェアとしての特徴は ADE というプロセッサ間のデータ転送を首尾よく実現するネットワークにある．物理的構造を述べるまえに，論理的な構成を与える．2次元セグメント配列の ADE に対しては図-3 にあるように，プロセッサの1次元並びを「座標軸」上に配置する．ただし x 軸上にも y 軸上にも「時をかえて」配置できるとする．そして両座標軸にそれぞれ垂直に交差するバスの線分の組からなるクロスネットを備える．当然各プロセッサは両軸上のそれぞれの位置から一つのバスセグメント，あわせて二つのバスセグメントにつながる．クロスネットのノードのところには FIFO（先入れ先出し）メモリからなるバッファをおく．各プロセッサはそれにつながるバスセグメント上のノードバッファにアクセスできるものとする．各バッファは互いに「直交」する二つのバスセグメントの交点に置かれているので二つのプロセッサからアクセスされることになる．

上のようなクロスネットがあれば，ADE の実現は容易である．基本的な PASS は次のように行われる．一つの方向属性をもった配列 $\{A(I,/J/)\}$ があれば，$/J/$ 番目のプロセッサがそれがアクセスできるセグメントの要素 $A(I,/J/)$ を I 番目のノードバッファに書き入れる，それをすべての J と I について行う．この操作自体は J について並列に，I については逐次的に行う．次にプロセッサ並びの位置をかえ（実際には同時アクセスするバスセグメントの組を取り替えるだけ），今度はそれぞれのプロセッサがそれにつながるバスセグメント上のバッファメモリから順次データを読みだし，やはりそれにつながる分散メモリユニットにセグメントデータとして書き込む．そうすると自然に方向属性の変更されたセグメント配列 $\{A(/I/,J)\}$ ができあがる．ここでバッファから読んで分散メモリに書き込む動作は I について並列に，J について逐次にとなる．逆方向の PASS についても逆順になるがまったく同様である．

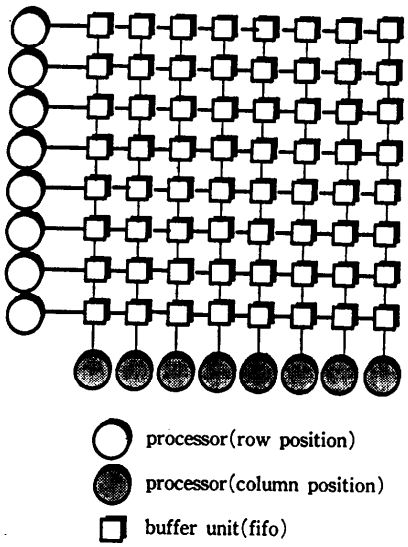

図-3 ADENA の論理的構成 1

ところが上の論理的構成はすべてのプロセッサがお互いにそれぞれ一つのノードバッファを介してつながるという「完全結合」を意味し，それをそのまま物理的構成にもっていくことは一般に困難である．というのは，プロセッサの台数が N というときノードバッファの個数は N^2 となり，N が大きくなるとそのハードウェア量は莫大になる．そこで物理的構成のためには，3次元の ADE（PASS）用の論理的構成を土台にする．

4.2 ADE 実現の論理的構成2

3次元の PASS にふさわしい論理的構成といえば，平面的に配置したプロセッサならびに「時をかえ」三つの「座標面」上を占めるようにする．そして互いに「垂直」な3方向のバスセグメントの組を考えて，三つのバスセグメントの交点にあたるノードにそれぞれバッファメモリをおいたクロスネットを備える．このとき各プロセッサは，三つのバスセグメント上にあるノードバッファにアクセスできる．また各ノードバッファは三つの異なるバスセグメントを通じて三つのプロセッサからアクセスされる（図-4）．このようなクロスネットがあれば3方向の属性をもったセグメント配列の間での ADE が容易に実現されることは2次元の場合の説明から明らかであろう．一方ハードウェアの負担も大分軽くなっている．というのは N 台のプロセッサを $\sqrt{N}\times\sqrt{N}$ の2次元配列に置いたとき，全ノードバッファの個数は $\sqrt{N^3}$ となり，N^2 に比べて $\sqrt{N}$ 分だけオーダ低下をみる．ただしバスセグメントの本数は $3N$ となり，2次元の場合の $2N$ より多い．このネットによれば各プロセッサは一つずつのノードバッファを介して $2\sqrt{N}$ 個のプロセッサとデータ交換できる．当然2次元ネットが任意のもの（N 個）とデータ交換できることと比較すれば劣っている．

4.3 クロスネットの物理的構成

物理的構成のためにもう少しハードウェアの軽減をはかるのである．論理的構成では平面的プロセッサ並びが三つの座標面を占めたのに対して，物理的構成では二つの座標面だけを占めるようにする．上述のクロスネットを設けることで3次元

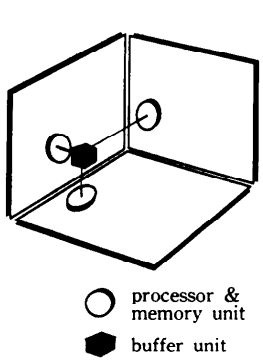

図-4 ADENA の論理的構成2（単一バッファユニット＋3プロセッサユニットのみ）

セグメント配列のいずれか二つの異なる方向属性をもつものの間では ADE が簡単に実現できる．では残る一つの方向属性のものはどうなるか．

これについてはプロセッサ・クロスネット複合体を「回転」させることにすれば可能である．いままで論理的構成を考えている場合に，イメージとしては，扱うセグメント配列（それは通常シミュレーションの実対象領域を反映している）に「固定した関係」でその複合体を設置してきた．しかし ADE とは「時をかえて」二つの方向属性間の操作であり，一時には三つのうち二つの方向属性間で ADE ができればよい．

プロセッサ並びを置く位置について言うと，あるときは x 座標面（$x=0$ の面）と y 座標面（$y=0$ の面）の間で，あるときは y 座標面と z 座標面（$z=0$ の面）との間で，またあるときは z 座標面と x 座標面との間で ADE が可能であればよい．だから「時をかえて」プロセッサ・クロスネット複合体をそれぞれの間に配置しかえることにする．たとえば x—y 間でことを済ませたあと y—z 間に移行する際，先に x 座標面にあったプロセッサ並びを y 座標面に，y 座標面にあったもの（実際は別ものでないが論理的に想定している）を z 座標面にもってくる．ほかの移行も同様とする（図-5）．

この「回転」で大事なことは，セグメント配列の個々のセグメントとプロセッサの同定（アイデンティフィケーション）がどの位置にあっても一様な決まり方になっていることである．そうでないとソフトウェアの負担は大きくなり，分散メモリも無駄になり，もっと悪いことに x—y—z—x と1回転したとき各プロセッサは前に扱ってきたのとは異なる番号のセグメントを扱うはめになる．その一様な対応付けとは，たとえばセグメント $A(,/J,K/)$ も $B(K/,,/J)$ も $C(/J,K/,)$ もすべて一定の物理的平面的プロセッサ並びの $/J,K/$ 番目のプロセッサがアクセスできる分散メモリユニットに確保することである．

そのためにはハードとしてのプロセッサ配列 $\{P[J,K],J,K=1,M\}$ とノードバッファ配列 $\{B[I,J,K],I,J,K=1,M\}$ の結合関係を次のようにする．

$$P[J,K]\cdots B[I,J,K]\cdots P[K,I]$$

この関係はノードバッファの立方配列を固定して

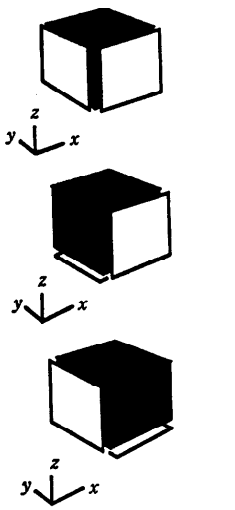

図-5　3様の配置

みたとき，プロセッサ配列面が片方の位置から他方の位置に変わる場合，その面を I, J, K 軸のうち1軸のまわりだけでなく，2軸のまわりをそれぞれ直角分回転させたことになっている（図-6）．このことから ADENA クロスネットのことをツイステドクロスネット（あるいはハイパクロスネット）と呼ぶ．この物理的構成によれば，プロセッサ台数 $N(=M^2)$ に対してノードバッファ数が $\sqrt{N^3}$ であることは前に述べたとおりであるが，バスセグメントの本数は2次元の場合と同じく $2N$ である．

そのようなハード量を備えることで，任意の二つのプロセッサ間のデータ転送もプロセッサ番号

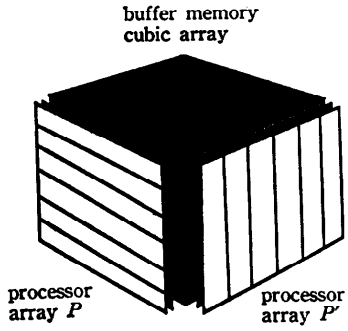

図-6　ADENA の物理的構成

のいかんを問わず一様に2パスで実現できるのである．ここで1パスとは一つのノードバッファを使った転送を意味する．実際任意の二つのプロセッサ $P[I,J]$ と $P[K,L]$ の間には

$P[I,J]\cdots B[K,I,J]\cdots P[J,K]\cdots$
$B[L,J,K]\cdots P[K,L]$

あるいは

$P[I,J]\cdots B[J,L,I]\cdots P[L,I]\cdots$
$B[I,K,L]\cdots P[K,L]$

などの結合が成立している．すなわち途中にもう一つ別のプロセッサの「読み書き」を介在させることでつないでいるのである．これをシステム全体で行えば，完全結合の2次元クロスネットをシミュレートできるのである．実際に，ADENA では2次元 ADE をこのシミュレーションによって実現している．

4.4　並行転送動作

ADE に関係する事例でどうしても触れておかなければならないことが，転送に係るオーバヘッドである．これに対する方策も実は ADENA で基本データ構造をセグメント配列としたこと自身から案出できているのである．それは，各プロセッサが自分の担当するセグメントを逐次処理し，結果も逐次決めていくという姿を標準としているので，その時間に結果が出てきしだいノードバッファにも書き入れ，また違う方向からノードバッファの内容を読むことを並行して行うようにする．すなわち PDO と PASS を可能なかぎり並行処理するのである．もちろん，計算に関係なく転送できるデータは先に送るようにする．その判断をあらかじめコンパイラが行っておく．こういう並行処理のことをSスキームと呼んでいる．このSスキームのために分散メモリユニットをスチールアクセスしたり，バッファへのアクセスを制御する高度な *DMA（ダイレクトメモリアクセス）装置を各プロセッサに併置する．

3次元の ADE では，Sスキームは直接的に効果を示す．しかし2次元の ADE では2パスになっているので，1パス部分は並行に処理できても残りの1パス分はまったくのオーバヘッドにならざるをえない．

5. 応用問題

5.1 典型的問題

ADENA で解くときにさまざまな計算様式を含む典型的問題を一つ選ぼう．矩形あるいは直方体領域におけるヘルムホルツ方程式の境界値問題を差分化して生じる連立一次方程式を前処理付き*CG 法でもって反復的に解く過程をとりあげる．

*ヘルムホルツ方程式は

$$-\Delta u+pu=f \qquad (1)$$

の形をしている．不均一幅をもつ格子点領域上で標準的な5点あるいは7点の*差分スキームを構成する．差分方程式の形が

$$a_{ij}u_{ij}-b_{i-1j}u_{i-1j}-b_{ij}u_{i+1j}-c_{ij-1}u_{ij-1}-c_{ij}u_{ij+1}=f_{ij}\quad (i,j=1,2,\cdots,M) \qquad (2)$$

あるいは

$$a_{ijk}u_{ijk}-b_{i-1jk}u_{i-1jk}-b_{ijk}u_{i+1jk}-c_{ij-1k}u_{ij-1k}-c_{ijk}u_{ij+1k}-d_{ijk-1}u_{ijk-1}-d_{ijk}u_{ijk+1}=f_{ijk}\quad (i,j,k=1,2,\cdots,M) \qquad (3)$$

であるとする．ここで係数 a, b, c, d および右辺 f は与えられた格子点上の関数，u が未知関数である．格子点の座標番号を与える添字変数 i, j, k の値が0あるいは $M+1$ となる点は境界上にあり，そこでの u の値は境界条件によって与えられているものとする．領域内部の格子点上の u の値を自然な順序で並べたものを成分にもつ長さ M^2 あるいは M^3 のベクトル（それも u と表す）に対する問題として連立一次方程式 $Au=f$ の形に表すことができる．係数行列 A は，2次元問題ではブロック3重対角形で各ブロックも対角形か3重対角形である．3次元問題では，そのようなブロック3重のものを単位ブロックにする大3重ブロック対角形になる．右辺の f にはもとの f に境界値が加わったものが登場している．

5.2 前処理行列問題

いま2次元の場合に限って説明する．係数行列を省略形で表すことにして

$$A=(0\ -c_{ij-1}\ 0\,|\,-b_{i-1j}\ a_{ij}\ -b_{ij}\,|\,0\ -c_{ij}\ 0)$$

とおく．これは水平に並ぶ3重対角ブロックをとりだし，それぞれのブロックの水平に並ぶ3重対角成分だけを表示したものである．そこで並列処理しやすい前処理行列 C を作ることにする．問題のもつ「方向属性」に基づいて抽出した行列

$$X=(0\ 0\ 0\,|\,-b_{i-1j}\ 0\ -b_{ij}\,|\,0\ 0\ 0) \quad と$$

$$Y=(0\ -c_{ij-1}\ 0\,|\,0\ 0\ 0\,|\,0\ -c_{ij}\ 0)$$

および対角行列 $D=(0\ 0\ 0\,|\,0\ a_{ij}\ 0\,|\,0\ 0\ 0)$ を用いて

$$C=(D+X)D^{-1}(D+Y)$$

とおく．3次元問題でも同様な作り方をする．この「分割作用素型」前処理行列は単純でありながら比較的高速の収束性を与える．これを用いたCG 反復法を SPCG 法と呼んでおく．

CG 法の反復過程では，毎回 $Cv=w$（w が既知で v が未知）の形の連立一次方程式を解く必要がある．それには三つの方程式

$$(D+X)v_1=w,\quad D^{-1}v_2=v_1,\quad (D+Y)v=v_2$$

を順次解いていけばよい．既知データ w が x 方向セグメント配列 $\{W(I,/J/)\}$ として与えられているとすると，v_1 も同種の $\{V1(I,/J/)\}$ として求めるのが都合よい．このとき係数行列 $D+X$ の構造は各 J ごとのセグメントに対して独立した形の部分問題の組を与えるものになっている．したがってそれは並列に処理できる．しかも個々の部分問題は3重対角形の係数行列をもつものでガウス消去法によって簡単に解ける．

```
PDO J=1,M
Q(M)=B(M-1,/J/)/A(M,/J/)
R(M)=F(M,/J/)/A(M,/J/)
DO 10 I=M-1,1,-1
TEMP=A(I,/J/)-B(I,/J/)*Q(I+1)
Q(I)=B(I-1,/J/)/TEMP
10 R(I)=(F(I,/J/)+B(I,/J/)*R(I+1))/TEMP
V1(0,/J/)=0.0
DO 20 I=1,M
20 V1(I,/J/)=Q(I)*V1(I-1,/J/)+R(I)
PEND
```

さらに v_2 も $\{V2(I,/J/)\}$ の形でもとめられる．それは D が対角行列であるから，$\{V1(I,/J/)\}$ の各要素ごとに対応する $D(I,/J/)$ を乗ずることでよい．

```
PDO J=1,M
DO 30 I=1,M
30 V2(I,/J/)=D(I,/J/)*V1(I,/J/)
PEND
```

最後に v_2 から v を求めようとするとき，セグメントごとにガウス消去法を適用するには右辺の

データになる $V2$ に対して前以て ADE を行っておく必要がある.

```
PASS I, J=1, M
V2(/I/, J)=V2(I, /J/)
PEND
```

v の方向属性を ADE 後の v_2 と同種と考えると係数行列 $D+Y$ は各セグメント $V(/I/,)$ に対して部分3重対角形行列を係数にするようなブロック対角行列になる. したがってセグメント配列 $V(/I/,)$ に対して並列処理できる.

```
PDO I=1, M
Q(M)=C(/I/, M-1)/A(/I/, M)
R(M)=V2(/I/, M)/A(/I/, M)
DO 40 J=M-1, 1, -1
TEMP=A(/I/, J)-B(/I/, J)*Q(J+1)
Q(J)=B(/I/, J-1)/TEMP
40 R(J)=(V2(/I/, J)+B(/I/, J)*R(J+1))/TEMP
V(/I/, 0)=0.0
DO 50 J=1, M
50 V(/I/, J)=Q(J)*V(/I/, J-1)+R(J)
PEND
```

これらのサンプルプログラムで, $Q(I)$ や $R(I)$ などの1次元配列や TEMP などの変数が用いられているが, これらは暗黙で方向属性には無関係に物理的分散メモリユニットにそれぞれ確保されるものである. 通常はテンポラリな値の保持のために利用する.

5.3 A内積計算

CG 法の反復過程の中でもう一つの典型的な処理として A 内積すなわち (Au, u) を決める計算をとりあげる. そのためにまず $w=Au$ の計算を行う. やはり2次元の場合だけを表示してみる. Au の中身は(2)式の左辺そのものである. u は初め $\{U(I, /J/)\}$ の形で与えられているとする. (2)式左辺の中で前半部分の $a_{ij}u_{ij}-b_{i-1j}u_{i-1j}-b_{ij}u_{i+1j}$ についてはすべてのデータを /J/ 番目の分散メモリユニットに確保しているとみなせるので, PDO 処理の対象になる. しかし後半部分の $-c_{ij-1}u_{ij-1}-c_{ij}u_{ij+1}$ については /J-1/ 番目から /J+1/ 番目にわたるので直接 PDO 処理の対象にはならない. その部分は方向属性を変更して I についての PDO としなければならない.

```
PDO J=1, M
DO 60 I=1, M
60 W(I, /J/)=A(I, /J/)*U(I, /J/)-B(I-1, /J/)
 &*U(I-1, /J/)-B(I, /J/)*U(I+1, /J/)
PEND
PASS I, J=1, M
U(/I/, J)=U(I, /J/)
PEND
PDO I=1, M
DO 70 J=1, M
70 V(/I/, J)=C(/I/, J-1)*U(/I/, J-1)
 &+C(/I/, J)*U(/I/, J+1)
PASS I, J=1, M
V(I, /J/)=V(/I/, J)
PEND
PDO J=1, N
DO 80 I=1, N
80 W(I, /J/)=W(I, /J/)-V(I, /J/)
PEND
```

そして最後に内積計算 (w, u) を行う. いったん

```
PDO J=1, M
S(1, /J/)=0, 0
DO 90 I=1, M
90 S(1, /J/)=S(1, /J/)+W(I, /J/)*U(I, /J/)
PEND
```

のように部分積和（セグメント対の内積）を求めた後 J についての和をとるが, 事前に ADE を行っておく.

```
PASS I=1, 1, J=1, M
S(/I/, J)=S(I, /J/)
PEND
PDO I=1, 1
DO 100 J=2, M
100 S(/I/, 1)=S(/I/, 1)+S(/I/, J)
PEND
```

5.4 性能評価例

前項までに扱った問題で, 2次元の場合にその解が

$$u(x, y)=\mathrm{sech}(5(x-0.5))\mathrm{sech}(10(y-0.25))$$
$$(0<x<1.0, 0<y<0.5)$$

あるいは3次元の場合に

$$u(x, y, z)=\mathrm{sech}(5(x-0.5))\mathrm{sech}(10(y-0.25))\mathrm{sech}(20(z-0.125))$$
$$(0<x<1.0, 0<y<0.5, 0<z<0.25)$$

となるように関数 p と f, ならびに境界条件を設

定するものとする．これらの解は領域の中央で最大値をとる山形の関数であるから，中央部で密になる小さな矩形あるいは直方体格子目に分けた格子点領域をとる．この場合に，前処理付き CG 法によって最大残差が一定の小さな値 (10^{-6}) 以下になるまで繰り返し計算を行う．測定は当初京大に設置された単独プロセッサ性能が 10 MIPS (64 ビット語長)，プロセッサ台数が 256 台，総データ記憶容量 512 MB のものの上で行った．

（1） 2 次元問題で 256×256 の内部格子点をもつ場合

反復法	反復回数	1 回あたり	全反復	MFLOPS
SPCG 法	143	27.46 ms	3.9 s	87
ICCG 法	61	2905.0 ms	177 s	

SPCG 法の実質計算部分を差し引いた残りのデータ転送に係るものの割合を測定すると，26.4% にもなっていた．これは，2 次元の場合に 2 パス転送によらねばならないことが効いていることを示すものである．なお，同じ問題を標準的に利用されている ICCG 法を用いた場合の測定結果をあげておく．反復回数では優秀なアルゴリズムであるが，前処理問題の計算を自然なデータ構造，標準的処理に委ねるだけだと逐次的部分が多く，あまり並列処理効果がでていない．

（2） 3 次元問題では 80×80×80 の内部格子点の場合

SPCG 法	反復回数	1 回あたり	全反復	MFLOPS
S スキーム無	76	207 ms	15.7 s	123
S スキーム有	76	178 ms	13.5 s	143

2 次元の場合と比較しても MFLOPS 値が大幅に向上していることが分かる．転送に係る割合が S スキームなしの場合で 13% におさまっている．S スキームありの場合では転送に係るものはほとんど並行処理の中にかくれてしまった．

上に得た性能は現在のベクトル型スーパコンに匹敵するものである．それが空冷で，標準消費電力 3〜4 kW，そしてデスクサイドタイプで実現できているのである[3]．

6. おわりに

この解説では ADENA の基本構想を明らかにするために，標準的利用法の原形のようなものについてのみ述べてきた．しかし，方向属性をもったセグメント配列は大方の科学技術計算の並列処理の基本的データ構造として十分汎用的な役割を果たすものであり，ADE も素朴で汎用的なデータ編集である．SPCG 法の前処理問題でみた利用法は偏微分方程式を解くときの多くの部分的インプリシット法に共通である．多次元問題に対して，ある次元方向にスペクトル法を適用し FFT 処理をする場合の形態でもある．現在までにも，いくつかの数値流体力学，気体力学，MHD，プラズマ粒子，気象力学，相変化問題，分子動力学，LSI デバイス解析あるいは乱流画像解析などのコードをインプリメントしてきた．

最後に現在のマシンで問題となる点を拾っておく．

（1） データ入出力チャンネルが細く，頻繁に結果を出す処理では，超高速演算性能を相殺してしまう．

（2） 並列プログラムだけに，デバッグがなかなか困難であり，プログラムの完成に時間がかかる．

京都大学工学部に設置した試作機は，いろいろな分野の研究者に開放し共同利用に供している．大型のシミュレーションおよび新しい並列システムやアルゴリズムの研究に係わる問題をかかえている人たちのご要望には可能なかぎり応えていきたいと思っている．そして利用者の間で経験交流などをはかるための「ADENA 並列計算研究会」も発足させた．

ADENA の共同開発に多大な努力を傾注してきた松下電器産業(株)半導体研究センターでは，引き続き超高性能の LSI チップの製作，一層コンパクトな実装あるいは最適化コンパイラの高度化，FORTRAN から ADETRAN への自動翻訳などの研究を推進している．計画どおり進めば応用プログラムにおいて実効速度性能が GFLOPS が普通になるのも間もなくとなっている．大学ではいろいろな応用の可能性を広げること，さらにまったく新しい使い方などの研究を始動させている．

謝辞　日頃松下電器産業(株)半導体研究センターの ADENA 開発チームの皆さんにはいろいろとご援助をいただいている．ここにあらためて感謝を述べます．

参考文献

1) Nogi, T.: Parallel Computation, Pattern and Waves (Studies in Mathematics and Its Applications 18) Kinokuniya-NorthHolland, pp. 279-318 (1966).
2) Nogi, T.: Parallel Language ADETRAN, Memoirs of Faculty of Engineering, Kyoto University, Vol. 51, Part 4, pp. 235-290 (1989).
3) Kadota, H., Kaneko, K., Tanikawa, Y. and Nogi, T.: VLSI Parallel Computer with Data Transfer Network: ADENA, 1989 International Conference on Parallel Processing, pp. I-319-I-322 (1989).
4) 野木達夫：並列計算機 ADENA, bit, Vol. 21, No. 2, 共立出版, pp. 1264-1274 (1989).
5) 若谷彰良, 森 康治, 佐々木真司, 岡本 理：並列処理言語 ADETRAN の実装, 電子情報通信学会第7回コンピュテーション研究会, 九大 (1990).

(平成3年1月7日受付)

用語解説

ADETRAN

並列計算機 ADENA 用の並列プログラミング言語とその処理系（コンパイラなど）のことを言う．字面は ADENA FORTRAN を略したものである．言語としては FORTRAN の拡張版の形を採っている．

オーバヘッド

並列計算機では複数のプロセッサが計算に関与することで処理スピードの向上を図ろうとするが，通常プロセッサ間でのデータ交換並びに同期待ちを伴う．これらに時間を要すると計算効率の劣化を招くことになる．一般に実質計算部分に対して余分にかかる負荷のことをオーバヘッドと言う．

差分スキーム

通常物理モデルは連続体として偏微分方程式などを用いて表される．計算のためにはさらに空間時間を離散化し微分を差分で置き換えてモデルを書き直す必要がある．この離散化計算モデルのことを差分スキームという．

CG 法

Conjugate Gradient 法（共役勾配法）の略称である．連立一次方程式の代表的反復解法の一つである．もとの方程式の問題を，対応する2次形式の最小化問題として定式化し，最急降下法のように最小値探索を行う．後者の収束性の遅さを大幅に改善する方法となっている．

スレイブプログラム

フロントエンド用のホスト計算機とバックエンド用のスレイブシステムからなる計算機システムでは，通常ホスト計算機用のプログラムとスレイブシステム用のプログラムを与えることになる．後者をスレイブプログラムと呼ぶ．例えば ADETRAN の場合，Gサブルーチン（Gサブコルーチン），Lサブルーチン及びL関数の種別がある．前者はスレイブシステム全体に対するプログラムであり，これがホストプログラムから呼ばれる．後2者は個々のプロセッサのプログラムであり，Gサブ(コ)ルーチンの並列処理構文 (PDO) のなかで呼ばれる形で利用される．

DMA

Direct Memory Access の略語である．一般にデータの入出力を行う際，CPU がバスを用いていない時に CPU を介さずに適当なハードウェア（DMA コントローラ）でもって直接主記憶にアクセスすることを言う．

分散メモリ方式

並列計算機においては，メモリ構成は最も重要な事柄の一つである．一方に複数のプロセッサからアクセスできる共有のメモリブロックを設けてプロセッサ間のデータ転送問題を軽減する共有メモリ方式がある．他方，共有メモリ方式に生じるメモリコンフリクト問題を回避するために，個々のプロセッサごとに直接アクセスできるローカルメモリを備える方式を分散メモリ方式と言う．

ヘルムホルツ方程式

波動方程式

$$\frac{\partial^2 v}{\partial t^2}=\Delta v-av+g$$

の強制項 g が時間的周期関数 $g=f(x,y,z)\exp(i\omega t)$ で与えられる場合の周期解 $v=u(x,y,z)\exp(i\omega t)$ の振幅関数 u が満たすべきものがヘルムホルツ方程式である：

$$-\Delta u+pu=f \qquad (p=a-\omega^2)$$

ホストプログラム

フロントエンド用のホスト計算機とバックエンド用のスレイブシステムからなる計算機システムでは，通常ホスト計算機用のプログラムとスレイブシステム用のプログラムを与えることになる．前者をホストプログラムと呼ぶ．例え

ば ADETRAN の場合，ホストプログラムを独立した FORTRAN プログラムとして書き下せる．スレイブシステムを駆動するにはスレイブプログラムを CALL するだけでよい．

メモリコンフリクト

同一メモリブロックを複数のプロセッサが同時にアクセスしようとした時のことをいう．一度には一台のプロセッサしかメモリにアクセスできないため，ほかのプロセッサのメモリアクセス時間が大きくなり，その分処理時間が遅くなる．並列計算機で共有メモリ方式をとるかぎり常に生じる問題である．それを避けるために分散メモリ方式が採られる．

野木　達夫

昭和 16 年生．京都大学大学院修士課程卒業．京都大学工学博士．京都大学工学部勤務．計算数学，並列計算機「数値解析の基礎」(共著，共立)「発展方程式の数値解析」(共著，岩波)「ステファン問題」(共著，産業図書)，「基礎工業数学」(朝倉)，日本数学会会員．