ハイパフォーマンス
コンピューティング 48－1
（１９９３．８．１９）

物理PEのデータ構造を併用した 並列記述言語ADETRANのプログラミングとその実効性能

尾林 善正　金子 克幸　岡本 理　　冨田 貞文　石川 隆

松下電器産業(株)半導体研究センター　(株)松下ソフトリサーチ

並列計算機ADENART上にインプリメントされている高級言語ADETRANのプログラミングにおいて、仮想PEに対応するデータ構造であるスラッシュ付きの2/3次元配列だけではなく、物理PEに対応するデータ構造のスカラー変数/1次元配列を併用することで、使用メモリ量の節約、プログラムの高速化等が実現できることを、ベクトルと行列の積・密行列の積の並列プログラミングを例に紹介する。

Programming and Performance of Parallel Programming Language ADETRAN with Data Structures corresponding to Physical Processing Element

Yoshimasa OBAYASHI　Katsuyuki KANEKO　Sadafumi TOMIDA　Takashi ISHIKAWA
Tadashi OKAMOTO

Matsushida Soft-Research, Inc.

Semiconductor Research Center
Matsushita Electric Industrial Co.,Ltd.

In programming on ADETRAN implemented on parallel computer ADENART, saving the amount of memory used or/and improvement of performance is achieved by using scalar and 1 dimensional array, that are corresponding to physical processing element in ADETRAN, not only 2/3 dimensional array with slashes, that are corresponding to logical processing element.

1 はじめに

当研究所では京都大学工学部と共同で分散メモリ型の並列計算機 ADENART を開発した。([1],[4])

現在、社内外で試作機が稼働しており、ADENART 上の高級言語 ADETRAN([2],[3]) で記述された応用プログラムでさまざまな分野の大規模シミュレーション等が行なわれている。([5], [6],[7], [8])

ここでは、ADETRAN によるプログラミングにおいて、仮想 PE に対応するデータ構造であるスラッシュ付きの 2/3 次元配列だけではなく、物理 PE に対応するデータ構造のスカラー変数/1 次元配列を併用することで、

1. 使用メモリ量の節約
2. プログラムの高速化

が実現できることを、ベクトルと行列の積・密行列の積を例に紹介する。

2 並列記述言語 ADETRAN のデータ構造

2.1 スラッシュ付きの 2/3 次元配列

ADETRAN では主に 2 次元あるいは 3 次元の配列を並列処理の対象とする。それらを 1 次元配列の「束」に分けて各プロセッサエレメント (以下、PE と略記) に割り当て、各 PE 内では分けられた束に相当する 1 次元配列を処理の対象とする。

例えば、2 次元配列を扱う場合、FORTRAN では a(i,j) と記述するが、ADETRAN では PE へのデータの分散方法によって a(i,/j/), a(/i/,j) の 2 種類があり、スラッシュにはさまれたインデックスがそのデータを持つ PE の番号を表す。

```
parameter( N = 1024 )
region(N,N)
real a(/*/,N)
real a(N,/*/)
```

ADETRAN ではハードウェアとして実際に存在する PE(以下、物理 PE と略記する) の個数[1]を越えて、region 文で指定した任意の個数の PE が仮想的にある (以下、仮想 PE と略記する) としてプログラミングできる。

[1]現在、物理 PE 数が 256 と 64 のシステムがある。以下の説明は全て 256PE を前提とする。

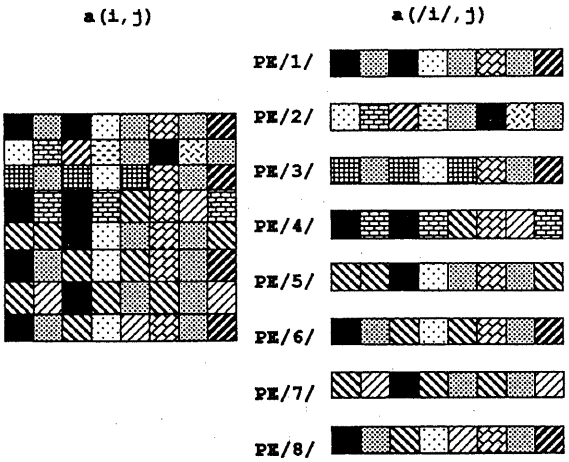

図 1:ADETRAN における 2 次元配列の分散

以下、仮想 PE の番号を/i/ とスラッシュではさんで表し、物理 PE の番号を//I// と二重スラッシュではさんで表す。

そのとき、a(/i/,j) を演算する仮想 PE/i/の処理を実際に行なうのは、番号//I// の物理 PE(但し、I =1+((i-1) mod 256)) であり、この物理 PE は、この他にも仮想 PE/i-256/,/i+256/,...等の処理も行なっている。これを (物理 PE による仮想 PE の) 多重処理と呼んでいる。

多重処理のために、番号//I//の物理 PE は、配列 a(/i/,j) のうち、a(/I/,j),a(/I+256/,j),a(/I+512/,j),... のデータを割り当てられて持っている。(図 2)

このような意味で、スラッシュ付きの配列は、仮想 PE に対応したデータ構造だということができる。

このようなデータ構造は、High Performance Fortran[9] における DISTRIBUTE された配列、Data Parallel C[10] における poly データと同様のものである。

2.2 スカラー変数/1 次元配列

前節の 2/3 次元スラッシュ付き配列以外に、ADETRAN ではスカラー変数と 1 次元配列が使用できる。

これは、変数や配列の宣言に対して 1 個の物理 PE 当たり 1 個分のスカラー変数または 1 次元配列の領域を割り当てたデータ構造である。

```
parameter( n = 1024 )
real sp(n)
integer itmp
```

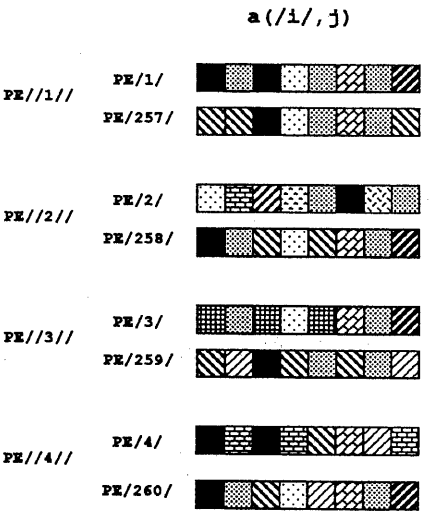

図 2:物理 PE による仮想 PE の多重処理

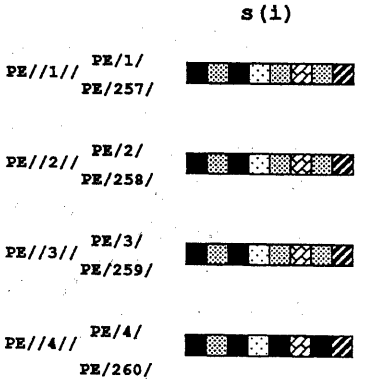

図 3:ADETRAN の 1 次元配列

つまり、256 個の物理 PE 上に、スカラー変数/1 次元配列の領域が 1 個ずつとられる。仮想 PE がこれを使用する場合には、番号/I/, 番号/I+256/, 番号/I+512/,... (I = 1, ..., 256) の仮想 PE が同一のデータを共用することになる。(図 3)

このように、スラッシュ付きの配列が仮想 PE に対応したデータ構造であるのに対し、スカラー変数/1 次元配列は、物理 PE に対応したデータ構造になっている。

このような物理 PE に対応したデータ構造は、High Performance Fortran における DISTRIBUTE されていない配列、Data Parallel C における mono データと類似のものであるが、HPF や DPC ではこれらのデータを変更した場合には、変更後のデータのコヒーレンシーを保証するためのデータ転送等を行なうのに対し、ADETRAN ではそれを保証しないとしている点が異なっている。

3 プログラム例

3.1 ベクトルと行列の積

行列 A の行ベクトル a_i と、ベクトル x の第 i 成分 x_i が、仮想 PE/i/に割り当てられているとする。

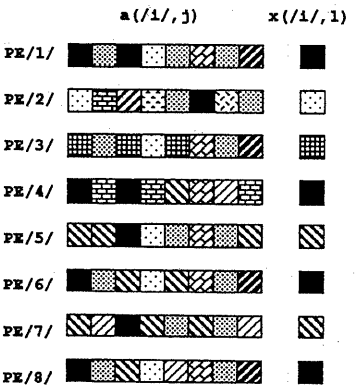

図 4:分散された行列 A とベクトル x

N 次元ベクトル x と (N,N) 行列 A の積を N 次元ベクトル y に求める計算

$$y = Ax$$

を、ベクトル y の各成分 y_i がベクトル x と行列 A の行ベクトル a_i との内積であるとして、これらの内積

を各 PE で並列に求める場合を考える。

$$y_i = (a_i, x) \quad (i = 1, ..., N)$$

この計算を並列に行なうためには、各 PE 毎に x のコピーが必要になる。

例えば、このコピーを格納する領域としてスラッシュ付き配列を用いて、全ての仮想 PE に x のコピーを用意すればよい。

```
全仮想PE/i/にxの全要素をコピー;
以下をPE/i/で並列に実行 {
  Aの第i行ベクトルとxとの内積を計算し、
   yの第i成分とする;
}
```

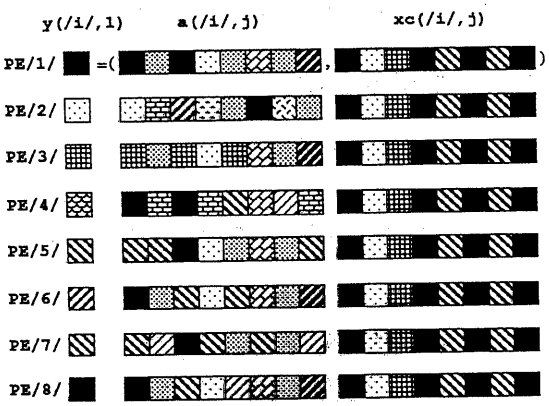

図 5:y=Ax の並列計算 (I)

しかし、これでは各物理 PE 内に、参照しかされない x のデータが多重処理の分だけ重複して、メモリの領域を占めてしまう。

また、x のコピーも仮想 PE の個数だけ作ることになるので、仮想 PE の数が大きくなれば、その実行時間 (PE 内の演算や PE 間のデータ転送の実行時間) も大きくなる。

そこで、1 次元配列に x のコピーを作れば、各物理 PE 内に 1 つだけ x のコピーがあることになって、使用メモリ量を減らすことができ、また、コピーを作る実行時間も減少することが期待できる。

このような用途のために、ADETRAN では、2 次元のスラッシュ付き配列として各仮想 PE に割り当てられているデータを、全物理 PE の 1 次元配列に変換する操作 (データ転送) を指示する構文:PCAST 文を備えている。

```
real xs(N)
real x(/*/,1)
pcast i=1,N
  xs(i) = x(/i/,1)
pend
```

上記の例では、仮想 PE/i/ (i=1,...,n) に分散させて持っていたデータ x(/i/,1) を 256 個の全物理 PE 上の 1 次元配列 xs(i) (i=1,...,n) にコピーしている。

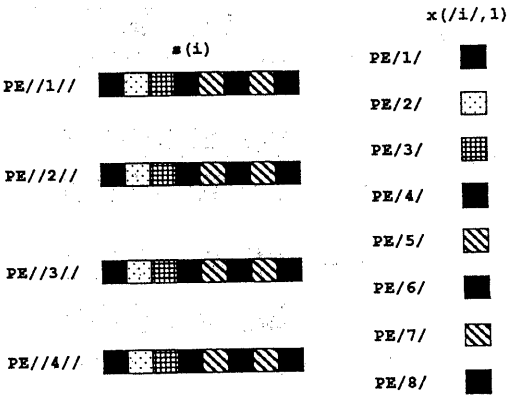

図 6:PCAST 文による x のコピー

これを使用して、内積計算に必要な x のコピーを各物理 PE 内の 1 次元配列に作れば、前述のものよりも使用メモリ量を減らしたベクトルと行列の積の並列計算が実現できる。

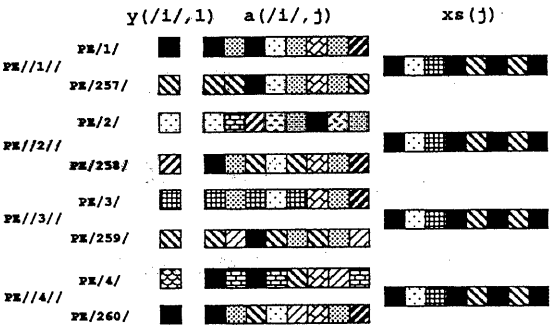

図 7:y=Ax の並列計算 (II)

これらの場合のシステム全体でのメモリ使用量は、ceil(r) を、r 以上の最大の整数として、

- 全仮想 PE に x をコピーした場合 :
  $((2 * N + 2) * ceil(N/256) * 256) * 8$ Byte(倍精度)

- 全物理 PE に x をコピーした場合：
  $((N+2) * ceil(N/256) * 256 + 256 * N) * 8$ Byte(倍精度)

であり、例えば、N=2048 の場合、前者が約 64MByte、後者が約 32.5MByte となり約半分強のメモリ使用量になる。

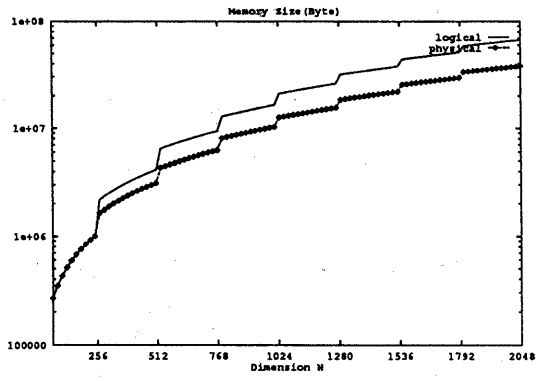

図 8:ベクトルと行列の積の使用メモリ量

また、サイズ N を変えた時の、これらのプログラムの実行時間を図 9に示す。

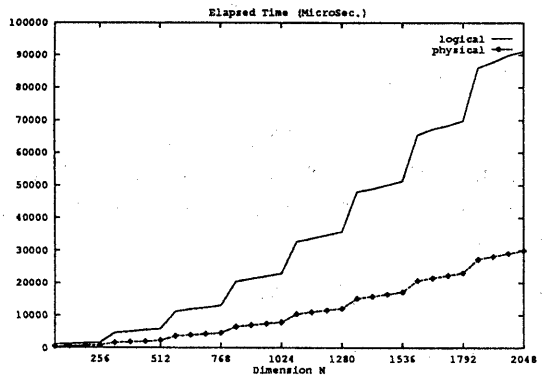

図 9:ベクトルと行列の積の実行時間

このグラフで、

- logical は仮想 PE 全てに x のコピーを作った場合
- physical は PCAST 文で物理 PE だけに x のコピーを作った場合

の実行時間である。N=2048 では、実行時間が約 3 分の 1 になっている。

3.2 密行列の積

密行列の積の場合にも、前節のように物理 PE を意識したデータ構造を用いた並列化を考える。

$$C = AB$$

いま、行列 A、B、Cは、前節と同様に、各第 i 行ベクトルが仮想 PE/i/に割り当てられているとする。このとき、アルゴリズムとして、以下の 2 つを考える。

1. 物理 PE 全てに行列 Bのコピーを (長さ N*N の 1 次元配列として) 持ち
   各 PE では、A の行ベクトルと行列 Bとの積を行なう。
2. 物理 PE 全てに行列 Bの列ベクトルのコピーを持ち、
   前節のベクトルと行列の積を N 回繰り返す。
   繰り返し毎に列ベクトルのコピーは上書きする。

前者は、行列 A、Cの行ベクトルを a_i、c_iとして、A の行ベクトル a_iと行列 Bの積

$$c_i = a_i B$$

を PE/i/内で行なうものであり、
後者は、行列 Cの第 (i,j) 成分を c_{ij}、Bの第 j 列ベクトルを b^jとして、a_iと b^jの内積

$$c_{ij} = (a_i, b^j)$$

を PE/i/内で行ない、これを j について 1 から N まで繰り返す。

前者は、行列 Bのコピーを 1 度行なっておけば、あとは各 PE が全く独立に計算が行なえるのに対し、後者は j についての繰り返し毎に、行列 Bの第 j 列ベクトルのコピーを、データ転送で実現しなければならないので、実行時間に関しては、前者のアルゴリズムの方が優れていると予想される。

しかし、使用メモリ量の点では、後者のアルゴリズムに比べて、前者のアルゴリズムは膨大なメモリを使用する。システム全体でのメモリ使用量は、ceil(r) を r 以上の最大の整数として、

- 全物理 PE に Bをコピーした場合：
  $(256 * N^2 + (3 * N) * ceil(N/256) * 256) * 8$ Byte(倍精度)

- 全物理 PE に b^i をコピーした場合：
  $((3*N)*ceil(N/256)*256+256*N)*8$ Byte(倍精度)

であり、例えば、N=2048 の場合、前者が約 8GByte、後者が約 104MByte となり、N が大きくなれば、メモリ量の点で前者を選択できなくなってくる。

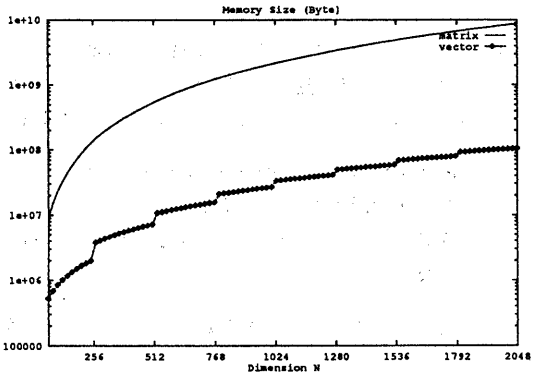

図 10:密行列の積の使用メモリ量

サイズ N を変えた時の、これらのプログラムの実行時間を図 11に、MFLOPS 値で表した性能を図 12に示す。

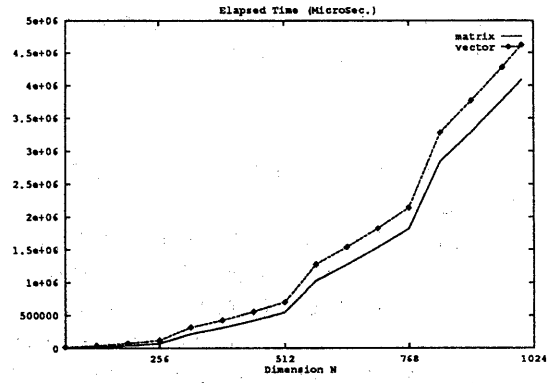

図 11:密行列の積の実行時間

これらのグラフで、

- matrix は全物理 PE に B をコピーした場合
- vector は全物理 PE に b^i をコピーした場合

の実行時間・性能である。ただし、全物理 PE に B をコピーするアルゴリズムに関しては全物理 PE に B をコピーの時間を含まない実行時間・性能である。

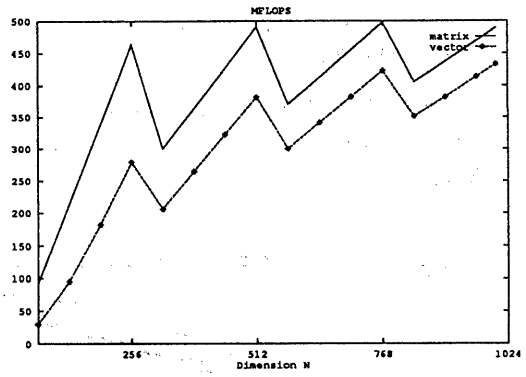

図 12:密行列の積の性能

ここで注目すべきは、後者の実行時間は b^i を全物理プロセッサにコピーする時間を計 N 回も含んでいるにも関わらず、N が大きくなれば、両者の実行時間の差がかなり小さくなっていることである。

つまり、メモリ節約のために B のコピーを持っておくアルゴリズムから、b^i を N 回コピーしながら計算するアルゴリズムにかえても、その際の実行時間の増加は、例えば、N:960 で 13.4%、N:1000 で 12.9%程度に留めることができている。

4 まとめ

分散メモリ型並列計算機 ADENART の高級言語 ADETRAN のプログラミングにおいて、物理 PE に対応するデータ構造を併用したプログラミングの有用性を

- ベクトルと行列の積
- 密行列の積

を例として紹介し、その効果として

1. 使用メモリ量の節約 ⇔ 実メモリで計算可能な問題サイズの拡大
2. プログラムの高速化

などが実現できることを示した。

このようなプログラミングの方法は、メモリ節約に関しては、ADENART/ADETRAN に限らず HighPerformance Fortran や Data Parallel C 等でも有効であると考えられる。しかし、性能面から、ADETRAN における PCAST 文のような、全対全通

信が高速に (あるいは、演算とオーバーラップして) 実行可能かどうかが問題である。

今後の課題としては、

- 言語仕様を拡張し、物理 PE に対応する多次元配列等をサポートする。
- PCAST 文によるデータ転送をさらに高速化するための検討。
- PCAST 文によるデータ転送と演算とのオーバーラップによる高速化の検討。

等が挙げられる。

5 謝辞

本研究を進める上で有益な示唆を頂いた、京都大学工学部野木助教授、及び本研究を進める上で御指導頂いた、半導体研究センター竹本センター長、同センター超 LSI デバイス研究所間野所長、廉田主幹技師に感謝致します。

参考文献

[1] T.Nogi, "Parallel Computation," Patterns and Waves, Studies in Mathematics and Its Applications, No.18, North-Holland, Amsterdam, pp.279-318, 1986.

[2] T.Nogi, "Parallel Programming Language ADETRAN," Memoirs of the Faculty of Engineering, Kyoto University, 51, part 4, 1989.

[3] 若谷 他., " 並列処理言語 ADETRAN の実装," 通信学会ソフトウェア、コンピューテーション、情報処理学会ソフトウェア、プログラミング言語、合同研究会, 1990.

[4] H.Kadota et al., "VLSI parallel computer with data transfer network:ADENA," ICPP, pp.I-319-322, 1989.

[5] H.Kadota et al., "Parallel Computer ADENART - Its Architecture and Application," ICS, pp.1-8, 1991

[6] 小田中 他., " 分割作用素法を用いた超並列計算による 3 次元デバイスシミュレーション," 信学技報, SDM92-91, VLD92-66(1992-10).

[7] A.Hiroki et al., "Massively Parallel Computation for Monte Carlo Device Simulation ADENART," International Workshop on VLSI Process and Device Modeling, pp.18-19, 1993.

[8] 岡本 他., " 並列計算機 ADENART のシミュレーション分野での応用," National Technical Report, Vol.39, No.1(Feb.1993)

[9] High Performace Fortran Forum., "High Performance Fortran Language Specification (Version 1.0), Technical Report CRPC-TR92225, Center for Research on Parallel Computation, Rice University, Houston, TX, 1993

[10] P.J.Hatcher, M.J.Quinn., "Data-Parallel Programming on MIMD Computers," The MIT Press, 1991.

[11] J.M.Ortega., "Introduction to Parallel and Vector Solution of Linear Systems," Plenum Press, NY, 1988.

[12] 内野 他, " 分散メモリ型並列計算機 AP1000 上への疎行列用 BLAS-3 の実装," 情報処理学会第 46 回全国大会, pp.131-132, 1993.