ハイパフォーマンス
コンピューティング 77－26
（１９９９．８．３）

HPFによる並列プログラミングに関する研究

小川　雄三，坂上仁志
姫路工業大学工学部情報工学科

概要

並列計算機言語HPF（High Performance Fortran）では，ユーザはデータを各プロセッサ上にどのように配置するかのみを記述し，データ転送等の処理はコンパイラに任せるので，容易に並列計算機の並列性能を引き出すことができる．本研究では3次元流体プログラムと2次元静電粒子プログラムの並列化をHPFで行った．3次元流体プログラムでは全ソース行数の約5％のHPF指示文を挿入することで，プロセッサ数64で51倍の高いスピードアップが得られた．2次元静電粒子プログラムでは，若干のソース変更と全ソース行数の約3％のHPF指示文を挿入するだけで，プロセッサ8台で4.1倍のスピードアップが得られた．

Performance of the parallel programming using HPF

Yuzo OGAWA, Hitoshi SAKAGAMI
Department of cmputer Engineering, Faculty of Technology,
Himeji Institute of the Technology

HPF is the parallel programming language with which users can easily get the high performance of the parallel computer only writing the data allocation to processors. Data transmissions are automatically performed by the system. In this papar, we have parallelized 3D fluid program and 2D electrostatic particle program. In the former, we achieved 51 times speedup with 64 processors. In this case, the number of HPF directives is about 5% of the total number of source. In the latter, 4.1 times speedup with 8 processors, and the HPF directives is about 3% of the total lines.

１．はじめに

並列計算機上で並列実行を可能にするためには，一般にハードウェアアーキテクチャに依存したライブラリをプログラムに組み込まなければならない．これはアーキテクチャごとに仕様が異なり，互換性がないのでプログラムの可搬性を著しく損ねてしまう．そこで，最近になってMPI（Message Passing Interface）と呼ばれる並列プログラミングインターフェースが提案され，多くの並列計算機に実装されている．しかし，MPIでは複雑なプロセッサ間のデータ転送の記述をユーザが行わなければならず，一般のユーザには利用しにくい．

そこで，従来のFortran77/90に最小限の指示行を付加するだけでプログラムの並列実

行を可能にする言語HPF（High Performance Fortran）[1]が提案されている．HPFでは，ユーザはデータを各プロセッサ上にどのように配置するかのみ記述する．並列実行に伴うデータ転送等の処理コードの出力はシステムが自動的に行うので，ユーザは複雑な記述を行うことなく容易に並列計算機の並列性能を引き出すことが可能となる[2]．本研究では，HPFで実用プログラムの並列化を行い，その性能とHPFの利便性について評価する．

2．3次元流体プログラムの並列化

3次元流体プログラムの並列化を考える．このプログラムでは，直交座標系を用いた立方格子状の計算グリッドを導入し，離散化された方程式の時間発展を陽的解法で解いている[3]．このため，計算には近距離の絡み合いしかなく，シミュレーション空間を部分領域に分割して，各部分領域を別々のプロセッサに割り当てて並列計算を行うことが可能である．多次元の時間発展は1次元問題の重ね合わせで解いており，空間の微分には5点差分スキームを使用しているので，両隣2要素ずつのデータが必要になる．このため，隣接した部分領域を持つプロセッサ間で境界のデータを交換する必要が生じる．

2．1　シミュレーション空間の分割方法

シミュレーション空間を分割するのに，まず図1に示されるような方法を考える．この方法は，最後まで分散方向を変えないので固定分散と呼ぶ．固定分散では，領域の境界面においてプロセッサ間でデータ交換が必要になる．分割次元数が大きくなると境界面が分割されるため，データ通信量が小さくなるが，通信回数は増加する[3]．

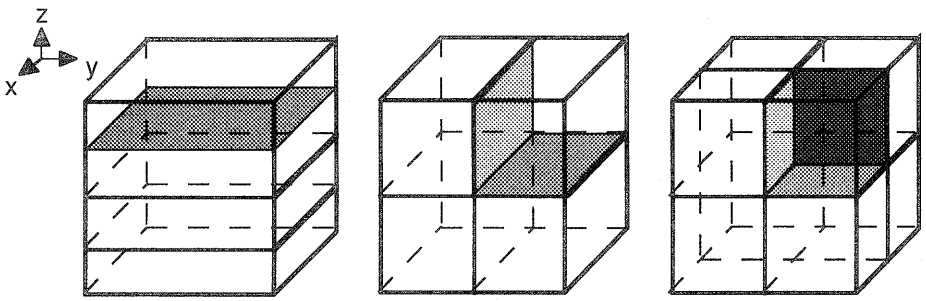

図1　固定分散における空間の分割方法

3次元配列arrayが次のように宣言されているとする．

```
real array(lx,ly,lz)
```

固定分散の場合，この配列arrayに対するHPF指示文は次のようになる．

(a) z 方向分割

```
!hpf$ PROCESSORS cj1(64)
!hpf$ DISTRIBUTE array(*,*,block) onto cj1
```

(b) y z 方向分割

```
!hpf$ PROCESSORS cj2(8,8)
```

```
!hpf$ DISTRIBUTE array(*,block,block) onto cj2
```

(c) x y z 方向分割

```
!hpf$ PROCESSORS cj3(4,4,4)
!hpf$ DISTRIBUTE array(block,block,block) onto cj3
```

この方法の場合，どの方向に分割してもコメント行を除く全体のソース1050行に対して44行のHPF指示文を挿入するだけで並列化ができた．実際はinclude文を使って配列の宣言とHPF指示文による分散の指示を展開するので，エディタで挿入する行はもっと少ない．

このプログラムでは，各方向の時間発展を解くモジュールは複数のdoループで構成されている．固定分散した場合，ある配列がこれらのループごとに参照されていたら，ループごとにその配列に対してデータ通信が発生することが考えられる．そこで，図2に示されるような分散方法を考える．この方法では，計算方向ごとに分散方向を変えるので再分散と呼ぶ．図2では，x方向，y方向計算時にはシミュレーション空間をz方向に分割し，z方向計算時にはy方向に分割する．この方法では，各方向の時間発展を解くモジュールを呼び出す段階で配列の再分散が生じ，このとき大量のデータ通信が発生するが，モジュール内の計算では通信が発生しない．固定分散ではデータ通信量は少ないが通信回数が多い．これに対して，再分散ではデータ通信量は大きいが通信回数が1回ですむので，プログラム全体でデータ通信にかかる時間が減少する可能性がある．

再分散の場合，全体のソース1050行に対して51行のHPF指示文を挿入するだけで並列化ができた．実際には，固定分散の時に書いたDISTRIBUTE指示文を若干書き換えるだけで，容易に変更できる．

```
      real  array(lx,ly,lz)
!hpf$ DISTRIBUTE array(TYPE,TYPE,TYPE) onto linear
```

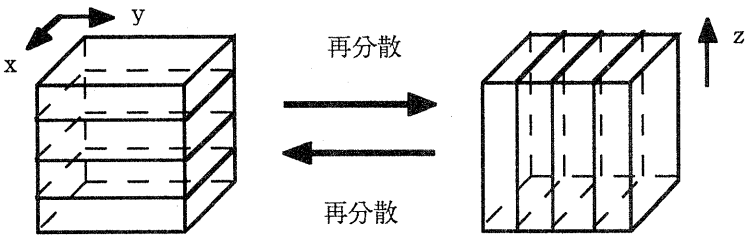

図2　再分散における空間の分割方法

2．2　並列性能の評価

NEC並列処理センターのCenju-3に実装されているHPFを用いて3次元流体コードの並列性能を評価した．$L_x = L_y = L_z = 64$としたときの並列実行によるスピードアップ率を図3に示す．

図3より，z固定分散では32台でスピードアップ27倍，yzおよびxyz固定分散においては64台でスピードアップ51倍という高い並列性能が得られた．再分散は固定分散よりスピードアップ率が落ちるが，これは結果的に通信回数を1回にしたことによるオーバーヘッ

ドの減少より，再分散によるデータ通信量の増加の方が大きかったことによる．また，3つの固定分散方法にも，通信量と通信回数についてトレードオフが存在し，方法ごとの並列実行性能はプロセッサ間通信のハードウェアに依存する[4][5]．z方向のみに分割した場合は，演算時間はプロセッサ数にほぼ反比例して小さくなるが，データ量はプロセッサ数に関係なく一定なので，全体の実行時間に対する通信時間の占める割合が急速に大きくなる．yz方向，xyz方向に分割した場合は，プロセッサ数をnとするとデータ通信量はそれぞれ～$n^{-1/2}$，～$n^{-2/3}$と減少し，通信回数はz方向のみの分割に対してそれぞれ2倍，3倍となる．これより，プロセッサ数が大きくなると多方向に分割した方が通信時間が短くなり高並列性能が期待できるが，Cenju-3ではその傾向は見られない．これは，Cenju-3のプロセッサ間通信のハードウェアが多段接続網のため[6]，通信一回当たりに発生するオーバーヘッドが通信回数の増加に伴って無視できなくなるためと考えられる．

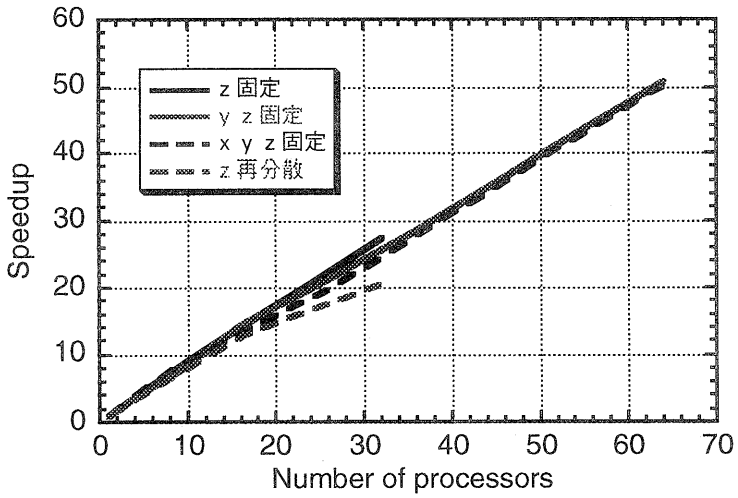

図3　各分散方法によるスピードアップ率

3．2次元静電粒子プログラムの並列化

2次元静電粒子プログラムの並列化を考える．このプログラムにおいて，電位計算は電荷密度をフーリエ変換したものにフォームファクターを乗算し，それを逆フーリエ変換することで求めている．このとき，フーリエ変換には2次元FFTを用いているが，x方向とy方向で独立に行っている．また，このプログラムでは粒子データを単純に分割するだけで並列化することが可能である．

3．1　電位計算の並列化

電位計算を行うモジュールでは，粒子データと電荷密度データの依存関係で並列化されないことを防ぐため，電荷密度の配列を次元拡張して各プロセッサに渡し，それぞれ計算した後それらの総和を計算している．FFTの計算においては分散方向がx方向（y方向）固定で

ある場合，y方向（x方向）計算時に全プロセッサ間で通信が発生し，並列効率を大きく下げてしまうので，動的に分散方向を変更する必要がある．このとき，分散を変える際にデータ通信が発生するが，計算は並列化して実行することが可能である．この様子を図4に示す．また，このときのHPF指示文を以下に示す．

```
      subroutine rfftfx(fdata,kx,ky)
      dimension fdata(kx,ky),ftmpx(kx)
!HPF$ PROCESSORS cj(4)
!HPF$ DISTRIBUTE fdata(*,BLOCK) onto cj
      do iy=1,ky
        ftmpx(*) <= fdata(*,iy)
        call rfftf(ftmpx)
        fdata(*,iy) <= ftmpx(*)
      enddo
      return
subroutine rfftfx(fdata,kx,ky)
      dimension fdata(kx,ky),ftmpy(ky)
!HPF$ PROCESSORS cj(4)
!HPF$ DISTRIBUTE fdata(BLOCK,*) onto cj
      do ix=1,kx
        ftmpy(*) <= fdata(ix,*)
        call rfftf(ftmpx)
        fdata(ix,*) <= ftmpx(*)
      enddo
      return
```

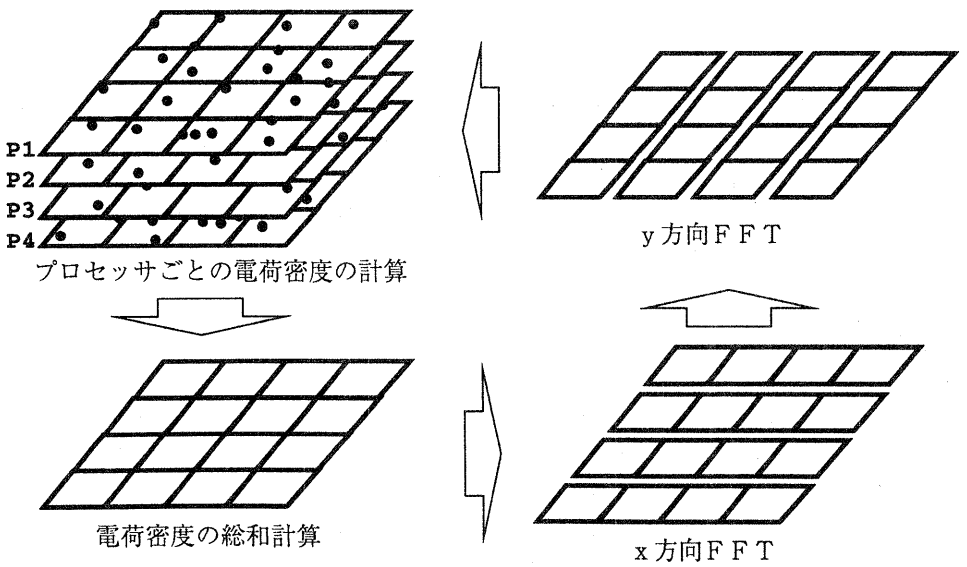

図4　格子データの動的分散

このプログラムの並列化では，再分散するためオリジナルソースを若干修正する必要があった．また，HPF指示文に関しては全体のソース1295行に対して39行挿入するだけで並列化できた．

3．2　並列性能の評価

2次元静電粒子プログラムをCenju-3で並列実行し，その性能を評価した．領域のサイズを64×64，1メッシュ当りの粒子数を25としたときのスピードアップ率を図5に示す．この結果，このプログラムは，アルゴリズム的に3次元流体プログラムより並列性能は出ないので，スピードアップ率では劣っているものの並列性能は引き出すことができた．

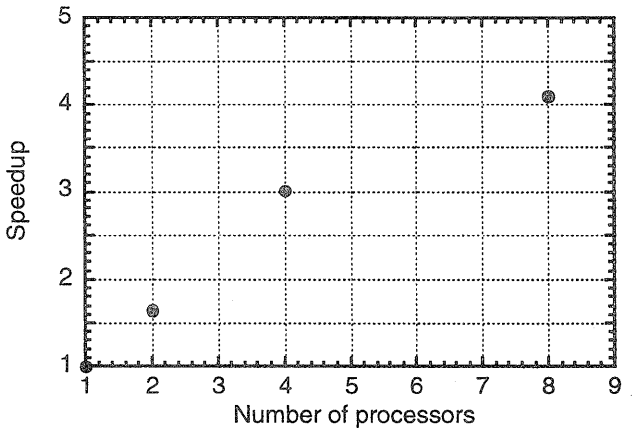

図5　2次元静電粒子プログラムのスピードアップ率

4．むすび

HPFを用いて3次元流体プログラムの並列化を行った．その結果，全体のソースに対して5%程度の行数のHPF指示文を挿入するだけで，プロセッサ64台で51倍のスピードアップという高い並列性能が得られた．また，2次元静電粒子プログラムの並列化を行った．その結果，若干のソースの変更と，全体のソースに対して3%程度の行数のHPF指示文を挿入するだけで，プロセッサ8台で4.1倍のスピードアップが得られた．

参考文献

[1] 妹尾義樹：“HPF言語の現状と将来”，情報処理学会誌第38巻2号，p.30（1997）
[2] 坂上仁志：“3次元流体コードの領域分割における並列効率”，電子情報通信学会 論文誌 Vol.J80-D-I No.8（1997）
[3] 村岡洋一：“並列処理”，昭光堂（1988）
[4] Steven Brawer：“並列プログラミングの基礎”，丸善株式会社（1990）
[5] R.W.Hockney, C.R.Jesshope：“並列計算機”，共立出版株式会社（1984）
[6] NEC並列処理センター:"http://www.ppc.nec.co.jp/search/docj/heiretu/Cenju-3.html"