ハイパフォーマンス
コンピューティング 79－7
（１９９９．１２．３）

JAHPFベンチマークプログラムを用いたHPFの性能評価

水見 俊介 (株)日立製作所 電力・電機開発研究所
高橋 俊 (株)日立製作所 電力・電機開発研究所

JAHPFで選定された7つのベンチマークプログラムを日立Parallel FORTRANを用いて並列化し、現行のHPFコンパイラに対する問題点の抽出と、MPIとの比較を含む性能評価を行った。ここで使用したプログラムはコンパイラ評価用に簡略化されたものではあるが、全て実際の技術計算プログラム(流体差分法，粒子コード，第1原理分子動力学コード等)から抜出したものである。現行のHPFによる並列化とMPIによる並列化の性能差は1.0～2.2倍程度であり、これらのプログラムに対しては実用に耐える程度の性能を実現してはいるが、性能を引き出すためには、指示行挿入以外のプログラムの変更が必要となる。

Performance and Applicability Evaluation of High Performance Fortran on JAHPF Benchmark Programs

Shunsuke Mizumi Power& Industrial Systems R&D Laboratory Hitachi Ltd
Shun Takahashi Power& Industrial Systems R&D Laboratory Hitachi Ltd

Through the JAHPF activity, we have so far developed and tested seven benchmark programs in the field of real-world scientific computation; molecular dynamics, electro-static plasma, fluid dynamics and so on. These programs are trimmed to be compact, but they still contain the core part of the calculation. Since the performance differences between MPI and HPF programs are within the ratio of 1.0 to 2.2, current HPF compiler seems to be already satisfactorily practicable. However, in order to derive a good performance, not a few modifications to a program seem to be inevitable.

１．はじめに

今日、科学時術計算に要求される計算規模はデータ量、計算時間ともに膨大になってきている。一方、計算機の単一プロセッサ性能を見た場合、PC、WS の性能向上は目覚しいものの、かつてのスーパーコンピューターの代名詞であったベクトルチップの性能は、頭うちになってきている。そのため、現在のスーパーコンピューターは複数のプロセッサを擁する並列計算機の形態をとっている。基本的に並列計算機には、主記憶を共有して使用する共有メモリ型と個々のプロセッサが個別の主記憶を持つ分散メモリ型、およびそれらの複合型が存在する。

分散メモリ型の並列計算機を使用した場合、プログラムの並列化には通常、MPI 等の通信関数ライブラリが用いられている。しかし、この MPI による並列化はあまりに煩雑であるため、分散型並列機の普及を阻害する要因となっていた。そこで、従来の Fortran 言語に最小限の 付加的指示で並列化を可能にすることを目指した言語、HPF(High Performance Fortran)が、米国を中心とした産学の代表者による HPFF(High Performance Fortran Forum)[１] によって検討され、現在

HPF2 およびその拡張仕様が公開されている。しかし、 ここで策定された HPF の言語仕様は、実装上の問題を十分に検討されることなく策定され、ユーザによる並列化指示も限定されたものとなっており、実用に耐えうるほどのものではなかった。一方日本では、実装上の問題とユーザのニーズとを両立させるべく、国内の大学、国立研究機関等のアプリケーション研究者・開発者及び、富士通、日立製作所、日本電気 3 社の計算機言語・コンパイラ開発者が集まり「HPF 合同検討会」 (JAHPF：Japan Association for HPF)[2] が発足した。ここでは、HPF2 をベースとし改良を加えた HPF/JA が検討され、現在 HPF/JA1.0 の仕様が策定されている。JAHPF では、HPF 言語を真に使いやすいものにするためには、ユーザ側からの検討も不可欠であると考え、言語評価のための幾つかのベンチマークプログラムを選定した。本稿はそのベンチマークプログラムに対して行った言語評価である。

2．ベンチマークプログラムの概要

現在までに、以下の 7 つのベンチマークコードが用意されている。これらは計算上あまり主要でないと思われる個所、ほぼ同じ処理をしていると思われる個所などは削られ、数百から千ステップ程度の比較的扱いやすいサイズに簡略化されている。

表 1　JAHPF の HPF ベンチマークプログラム

プログラム名	説明
impact3d	3 次元圧縮性流体 TVD 差分法
cip2d	2 次元圧縮性流体 CIP 差分法
espac2d	2 次元プラズマ粒子コード(粒子-格子)
carparri	第 1 原理分子動力学のベンチマーク用簡略化コード
maxwell	3 次元電磁場解析差分法
pc3d	3 次元プラズマ粒子コードの粒子関連部分
miccg	MICCG 法による行列ソルバーのカーネル

3．HPF 化方針

ここでは、各プログラムについての簡単な特徴と HPF による並列化方針について述べる。

3.1　impact3d コード

本プログラムは、TVD 陽解法による 3 次元圧縮性流体の差分法プログラムである。構造格子を用いた差分法プログラムなので、並列化方針は明確であり、3 次元なので図に示すように、i,j,k のいずれでも分散可能であるが、ここでは最も単純な第 3 次元だけの分散とする。

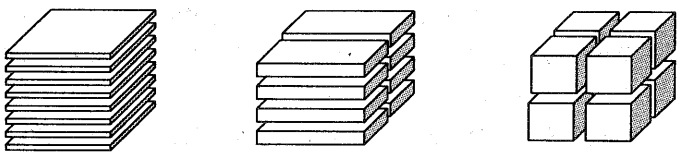

図 1　3 次元差分法の領域分割方法

これを HPF 指示行であらわすには、分散対象配列を以下の文の「::」以降に書き並べればよい。

!hpf$ distribute (*,*,block) onto P ::

ただし、我々の使用したHPF(日立Parallel FORTRAN)には隣接境界のデータ通信を最適化するための拡張仕様があるため、(i,j,k+1)や(i,j,k-1)を参照する配列に対しては次の行を追加した。

!pfd$ overlap(*,*,1:1) ::

この指示行は、通信に必要な個所を明示することにより、コンパイラが不必要に大きなデータを確保したり、送信したりすることを防ぐためのもので、HPF/JAではshadow指示行に相当する。

3．2　cip2dコード

本プログラムはCIP法を用いた2次元圧縮性流体の差分法プログラムである。並列化方針はimpact3dコードとほぼ同様であり、静的なブロック分散で並列化する。分散対象とする配列は、(*,block)，(block,*)，(block,block)のいずれの分散でも並列化できるが、ここでは(*,block)のように第2次元目をブロック分散した。上記2つの例でも分かるように、構造格子差分法プログラムを並列化する場合、HPFは非常に明確な並列化手段を提供してくれる。

3．3　espac2dコード

本プログラムは2次元静電プラズマ粒子コードである。3次元化しても、磁場の効果を入れても、プログラムの基本的な構造は変わらないので、ここでの議論は一般的なプラズマ粒子コード全てに共通に当てはまると考えられる。プラズマの粒子コードを並列化する場合、領域分割と粒子番号分割の2通りが考えられる。しかし、一つの境界格子点に対して周囲の隣接粒子の効果を取り入れる必要があるため、領域分割で並列化を行う場合、コーディングは非常に複雑になる。そこで、ここでは図2に示すような粒子番号で分割する並列化を行った。ただし、図のREDUCTIONとは個々のプロセッサで行った演算結果の集計を、REAMPはデータのプロセッサ間に渡る並び替えを、また、REPLICATEは各プロセッサ上の個別データを元にしたデータの複製を意味し、自動的にあるいは簡単な指示行を加えることにより、コンパイラが行ってくれるものである。

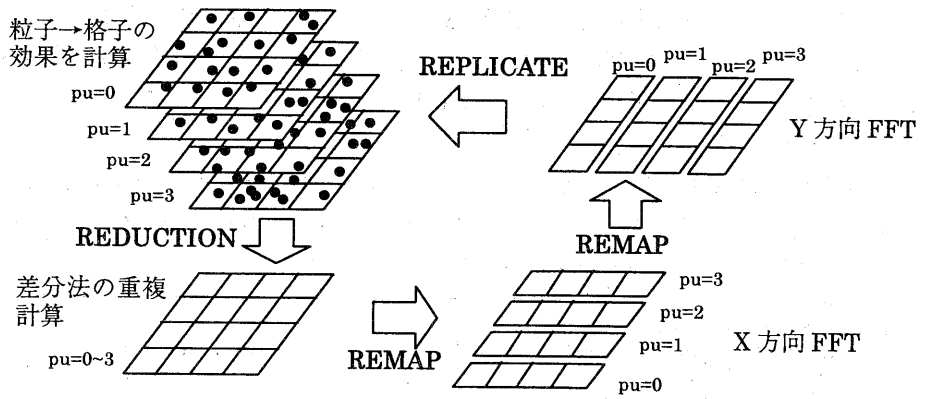

図2　粒子コードのHPF化概要

3．4　carparriコード

本プログラムは第1原理分子動力学法(Car-Parrinello法)のプログラムである。主要なサブルーチンはschmit、toreal、evnlの3つで、それぞれ順に、ベクトルの直交化、各ベクトルに対するFFT、各ベクトルからのスカラ量算出を行っている(図3)。

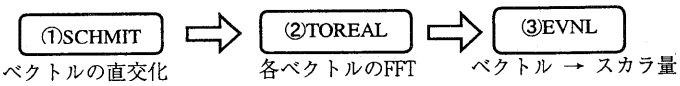

図3　Car-Parrinello法主要部の概念図

ここで、主要な配列(ベクトル配列)の各サブルーチンでのデータ依存関係を調べると図4のようになる。

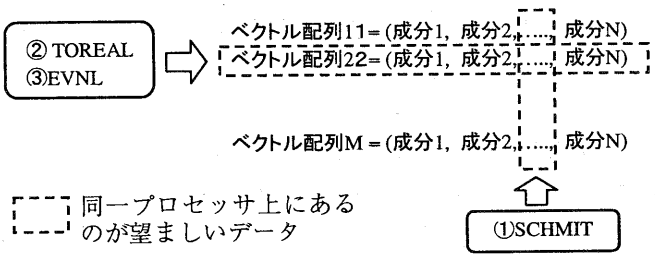

図4　Car-Parrinello法主要部の概念図

この図から分かるように、toreal と evnl ルーチンはベクトル番号をブロック分散し、schmit ではベクトル成分をブロック分散するのが適当と考えられる。従って、schmit 実行後、ベクトル群に相当する配列を再分散もしくは REMAP する必要がある。

3．5　maxwell コード

本プログラムは、2 次元の差分法による電磁場コードで、計算の主要部分は FFT 計算である。基本的には差分計算なので、HPF 化は配列をブロック分散するだけでよい。ただし、ベクトル計算機向けに配列の 1 次元化などがされているため、実際の並列化のためには一部配列の非ベクトル化(1 次元→2 次元)などの、プログラム修正の必要があった。

3．6　pc3d コード

本プログラムは 3 次元プラズマ粒子コードである。従って、基本的な並列化方針は ESPAC2D と同様である。ただし、簡略化のために電場を解くポアソン方程式部分は削除されている。具体的には、粒子関連配列(粒子の位置と速度)をブロック分散し、格子関連配列（電場と磁場と電荷密度と電流密度）は、分散せず各プロセッサで重複して持たせるようにした。

3．7　miccg コード

本プログラムは MICCG 法による行列ソルバーのカーネルである。ICCG 法は前処理部のデータ依存性が強いため、元々アルゴリズム的に並列化効率はよくならない。そこで、このプログラムでは HPF の並列化記述性についてのみ評価した。

```
!hpf$ processors Proc(Npe)
!hpf$ distribute(block(Npart1h)) onto Proc :: a0,b,r, a1,a2,x,p,q, a3,d
      do 720 k=1,Nz
        l1 = 1-k+1
        do 710 j = max(1,l1+1-Nx), min(l1,Ny)
          i = 1-k-j+2
          h = i + (j-1)*Nx + (k-1)*Nxy
          q(h) = d(h)*(r(h) - a1(h)*q(h-1) - a2(h)*q(h-Nx) )
710     continue
720   continue
```

図5　ベクトル化コードの HPF 化

原作では、3次元であるべき配列が1次元化されている（図5）。このように、ループカウンタjおよびkと、配列添字hが一致しない場合、そのままHPF指示行を書き加えてもデータは分散されない。そこで、参照される配列も更新される配列もすべて3次元配列にし、ループカウンタkと配列添字kが一致するようにするための修正が必要となる。

なお、MPIでこのプログラムを並列化する場合、プログラムを大幅に書き換え、パイプライン的に並列化を行うことになるが、現状のHPFの文法では、これを記述する方法はない。ただし、日立Parallel FORTRANは、プログラムを完全に元の3次元の状態にし、依存関係が明確な形に記述すれば、正しく解析して自動的にほぼ同様の並列化ができるようになっている。

4．HPFコンパイラの評価

4．1　HPF化の作業コスト

HPFは本来、煩雑なMPIプログラミングに変わるものとして、実行性能を多少犠牲にしつつも、指示行追加だけで並列化ができる言語として策定されたものである。しかし、今回HPF化において実施したソース変更の多くは、指示行追加ではなく、並列化しやすいように原作プログラムを変更することであった。表2にこれらのHPF化に伴うソース変更量を示す。なお、一部プログラムはMPIで並列化し、その変更行数もあわせて示した。

表2　HPFベンチマークのプログラムの変更行数

プログラム名	原作行数	変更行数	HPF指示行数	MPI変更行数
impact3d	1193	175	24	107
cip2d	336	79	45	95
espac2d	947	112	42	257
carparri	566	429	38	196
maxwell	491	201	35	-
pc3d	547	9	4	-

HPF指示行追加量についてみて見ると、原作プログラム行数と比較して1～5%であり、十分に許容できる程度と考えられる。以下ではHPF指示行追加以外に行ったソース変更点について述べる。

（1）interfaceブロック(手続き呼び出しの並列化のため)

Interfaceブロックは Fortran90 から新設された文法で、呼び出すサブルーチンの仮引数情報を記述するものである。サブルーチンの分割コンパイル時にも、サブルーチン間に渡る大域最適化を可能にするための情報を、コンパイラに与えることが目的で新設されたものである。今回のHPF化では、分散したDOループ内からサブルーチンを呼び出す場合等に、不必要なREMAP通信を抑制させる目的で、interfaceブロックを記述した。

（2）commonブロック（COMMON配列はメモリ分散対象とならないため）

commonに確保された配列は、それがBlock Dataで初期化されている可能性があるため、メモリ分散されない。このcommon代わりには、Fortran90のmoduleを使用した。これ内部の配列は、Block Dataで初期化されることがないので、メモリ分散が行える。

（3）HPF化チューニング

3章で触れたように、たとえHPFにより、ループの分散を記述できたとしても必ずしも並列化

が行われるわけではない、そのため、コンパイラの出力する並列化メッセージを見ながら、分散対象のDOループが正しく分散されるように、プログラムの修正を行った。

4.2 実行性能

日立Parallel FORTRANで測定した場合の加速率および、MPIとの性能比較を表3に示す。なお、HPF、MPIプログラムとも、特に最適化は行っていない。MPI版のESPAC2Dは通信量を最小化してあるため、大きな差がついてはいるが、その他のプログラムに関しては大差ない。

表3 HPFベンチマーク課題の加速率

	1pu	2pu	4pu	MPIとHPFの比(4cpu)
IMPACT3D	1.00	2.02	3.90	1.15
CIP2D	1.00	1.94	3.73	0.98
ESPAC2D	1.00	1.52	2.05	2.29
CARPARRI	1.00	1.76	2.61	1.00

ここで、MPI版とHPF版の性能差を考えると、主に、以下のような並列化に伴うオーバヘッドの違いがあげられる（表4）。またそれらの主因は、分割コンパイルされるサブルーチン間の、並列化に伴う最適化を全てコンパイラにゆだねているところにあると考えられる。この点に関してHPF/JAでは、データ通信周りの制御においては、ユーザにもある程度の自由度を与えることにより解決を図ろうとしている。

表4 MPIユーザとHPFの最適化の違い

	MPIユーザによる最適化	HPFコンパイラによる最適化
最適化（アルゴリズム）	大域(サブルーチン間)最適化	局所(サブルーチン内)最適化
パラメータ算出（ループの始点/終点、配列、通信バッファサイズ）	特定ルーチンでまとめて計算	各サブルーチンで毎回計算
	同じ形状の配列・ループは同じパラメータを流用	個々の配列・ループごとにパラメータを算出
通信（量、タイミング）	サブルーチン間で通信を最適化	サブルーチン内で通信を最適化

5. おわりに

JAHPF会議で選定された7つのベンチマークプログラムについて現行のHPFで並列化し、コンパイラの評価を行った。その結果、HPFの実行性能はMPIに比してそれほど遜色ない性能が達成可能であることがわかった。しかし、プログラムの修正量はMPIによる並列化よりも少なくなるわけではなく、修正量の面では並列化の煩瑣さを解決するものにはなっていない。

今後、JAHFPの活動を通じで、より完成されたコンパイラへの仕様提案を行っていく。

参考文献

[1] http://www.crpc.rice.edu/cgi-bin/HPFF/redirect.pl

[2] http://www.tokyo.rist.or.jp/jahpf/index-j.html