ハイパフォーマンス
コンピューティング 87－13
（２００１．７．２６）

国産 HPF コンパイラの性能評価と互換性検証

坂上仁志，水野貴夫
姫路工業大学工学部情報工学科

従来ベンダの実装依存による非互換性および得られる並列性能の低さがHPFの普及を妨げていたが，最近になってようやく安定したHPF処理系が提供され，HPFを実際に利用できる環境が整いつつある．そこで，実用コードである3次元流体コードおよび2次元静電粒子コードを国産3社の並列計算機上（NEC SX-4,5，富士通VPP800，日立SR-8000）でHPF化し，並列性能およびソースコード互換性について調べた．3次元流体コードでは，良好な並列性能が得られた．2次元静電粒子コードでも同様に良好な結果が得られたが，並列化のために若干のコード修正が必要であった．また，互換性については，ほぼ満足な検証結果が得られた．

Performance Evaluation and Compatibility Verification of Japanese HPF Compilers

Hitoshi Sakagami and Takao Mizuno
Computer Engineering, Himeji Institute of Technology

Many incompatibilities of HPF depending on the vender implementations and terrible parallel performances using HPF have been discouraging scientific application users to develop portable programs. We have evaluated the compatibility and capability of Japanese HPF compilers (NEC, Fujitsu and Hitachi) for two actual applications, 3D fluid code and 2D particle code. We have found that HPF could achieve a good performance for 3D fluid code with the almost same source code. We could also get a good result for 2D particle code, but some minor changes in the source code would be required. Source level compatibility between Japanese HPF compilers has been almost preserved.

１．はじめに

超高速演算を達成する並列計算機が多数商用化されており，これらの並列計算機上で並列に動作するプログラムを開発するためには，一般にMPI（Message Passing Interface）と呼ばれるプログラミングインターフェイスを用いる[1,2]．しかし，MPIではプロセッサ間のデータ転送をプログラム実行のフローを意識した上でユーザが明示的に記述しなければならず，一般ユーザが利用するには敷居が高く煩雑なものとなっている．そのため，共有メモリ型の並列計算機では，並列プログラミングを簡略化するためOpenMPと呼ばれるプログラミングインターフェイスが提案されている[3]．OpenMPでは，ソースコード中に指示行を挿入するだけで並列処理が実現でき，一般ユーザにも非常に利用しやすいが，分散メモリ型のハイエンドクラスの超並列計算機では利用できない．

そこで，従来のFortran77/90で書かれたプログラムに最小限の付加的な指示行を追加するだけで，分散メモリ型の並列計算機上でプログラムの並列実行を可能とするデータ並列言語High Performance Fortran（HPF）が提案されている[4-6]．基本的にHPFでは，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ユーザが明示的に指示し，データ転送等のそれ以外のことはすべて処理系に任せる．プログラムの制御は単一のセマンティックスで記述されデータは大域的に扱われるため，従来のFortranを用いたプログラミングスタイルと非常に親和性が高い．このため，ユーザは煩わしいプロセッサ間の通信管理や実行制御を記述するプログラミングから解放され，比較的容易に並列計算機の高性能が享受できる．また，HPF指示行は通常のFortranコンパイラではコメントとして扱われるため，HPFプログラムは通常のFortranソースとしてもコンパイル，実行ができ，ソースレベルでの可用性が高い．

しかしながら，ベンダの実装依存による非互換性および得られる並列性能の低さがHPFの普及を妨げていたが，最近になって並列性能が期待できる安定したHPF処理系がハイエンドクラスの並列計算機上で提供され，HPFを実際に利用できる環境がようやく整いつつある．そこで，実用コードである3次元流体コードおよび2次元静電粒子コードを実際に国産3社の並列計算機上（NEC SX-4,5，富士通VPP800，日立SR-8000）でHPF化して実行し，得られる並列性能およびHPFソースコードのコンパイラ間における互換性について調べる．

表1に，実際に使った並列計算機と利用可能な最大プロセッサ数xノード数，HPFコンパイラのバージョンおよびコンパイルオプションを示す．

２．３次元流体コードの並列化

3次元非粘性圧縮性流体方程式を固定グリッドのオイラー法を用いて離散化し，理想気体の状態方程式と共に解く3次元流体コードを考える．このコードでは，カーテシアン座標系（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は陽的解法で解いているため，計算には近距離の絡合いしかなく，シミュレーション空間を部分領域に分割して，各々の部分領域を別々のプロセッサに割り振って並列計算を行うことが原則として容易である[7]．NEC SX-4,5および富士通VPP800は，各プロセッサがベクトル演算機構を持っているので，ベクトル処理の効率を考え，ベクトル長が確保できるZ方向のみの領域分割が望ましい．このため，HPFで定義する抽象プロセッサは1次元とし，計算に使われる3次元配列の各変数を3次元目だけで分散する．

表１：並列計算機とコンパイラの一覧

NEC SX-4 (32x2 台)	HPF/SX V2 hpf driver:Rev.1.1 -O -Moverlap=size:0
NEC SX-5 (16x1 台)	HPF/SX V2 hpf driver:Rev.1.2 -O -Moverlap=size:0
富士通 VPP800 (1x32 台)	UXP/V HPF V20L20 L00121 -Wh,-Lt,-Owl -Wv,-m3
日立 SR-8000 (8x1 台)	Parallel FORTRAN V01-01 (pre) -Wt,'comlib(mpi),opt(scope(0))' OFORT90 V01-01-/B (post) -Oss -noparallel

HPF化は，(1)DIST，(2)IND，(3)SHADおよび(4)LOCの4つのレベルに分けて順次行う．まず，DISTレベルでは，配列の分散指定だけをDISTRIBUTE文で行う．次のINDレベルでは，配列分散に加えて並列実行すべきループを明示的にINDEPENDENT文で指示する．このコードの並列実行に主として必要な通信は，隣接プロセッサ間での境界データの交換だけである．この通信は，JAHPF[8]で規定されたHPF/JA拡張仕様[6]のSHADOW文およびREFLECT文によって最適化できるので，SHADレベルでこの指示文を導入する．更に，コンパイラの変数依存関係の解析が不十分で無駄な通信が発生することを防ぐため，LOCレベルでは，明示的に並列実行に通信の必要がないことをLOCAL文で指示する．図1に各レベルで挿入したHPF指示文の行数を示す．実際には，分散指示すべき3次元配列の変数はFortranのincludeファイルに書かれているので，DISTRIBUTE文やSHADOW文の挿入は，includeファイルに対してのみ行えばよく非常に容易にできる．

計算領域のシステムサイズは，NECと富士通ではLx=Ly=Lz=128とし，日立ではキャッシュミスによる深刻な性能低下を避けるため127とした．また，並列性能は初期化を除いた主計算部のみの実行時間より評価した．

NEC SX-4における実行時間を表2に示す．プロセッサ数が16以下のときはノード内通信のみを，32以上のときはノード間通信を行っている．NEC SX-4では，コンパイラが並列実行可能ループを検出できれば自動的に並列実行し，シフト型の通信を検出できれば，SHADOW+REFLECT文と同様に非効率な通信を行わず，LOCAL文と同様に無駄な通信も行わない．このコードでは，コンパイ

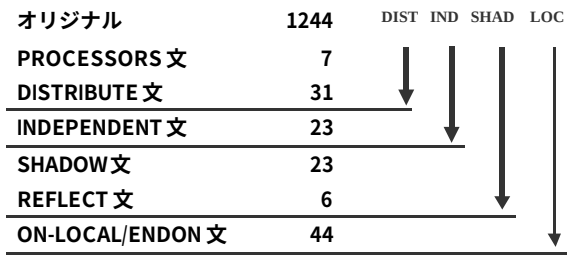

図１：各レベルにおいて挿入したHPF指示文

表2：NEC SX-4における並列実行時間 [秒]
ただし，*はノード間通信を示す．

プロセッサ数	DIST	IND	SHAD	LOC
1	82.69	82.71	83.16	81.40
2	41.50	41.53	41.85	40.92
4	20.93	20.94	21.23	20.64
8	10.59	10.58	10.87	10.47
16	5.45	5.45	5.74	5.54
*32	2.85	2.89	3.25	2.92
*64	2.01	1.71	2.39	2.06
*32(256)	18.59	18.19	19.17	18.61
*64(256)	11.11	9.80	11.65	10.29

表3：NEC SX-5における並列実行時間 [秒]

プロセッサ数	DIST	IND	SHAD	LOC
1	26.32	27.90	26.69	27.38
2	13.25	13.79	13.55	13.30
4	6.76	6.81	6.90	6.78
8	3.52	3.49	3.58	3.50
16	1.92	1.91	2.04	1.95

ラがすべての自動並列化に成功し，4つのレベルにおける並列効率には，ほとんど差がない結果となった．なお，SHADレベルの結果が若干悪いのは，コンパイラが自動的に挿入したシフト通信とREFLECT文による通信とが2重に行われたためである．LOCレベルでは，LOCAL文によりコンパイラによる通信の自動生成が抑制されるので，並列性能がSHADレベルより改善されている．また，64台では並列性能が低下しているが，これはサイズが小さすぎてプロセッサあたりz方向に2メッシュしか割り当てられないためと考えられる．表2に示すように，サイズを256にして32台と64台の性能を比較すると顕著な性能低下が見られないことから，この推測は裏付けられる．

NEC SX-5における実行時間を表3に示す．すべてノード内通信である．NEC SX-4と同様な結果が得られている．

表4：富士通VPP800における並列実行時間 [秒]

プロセッサ数	LOC
1	31.33
2	15.85
4	8.12
8	4.13
16	2.30
32	1.58
16(256)	11.82
32(256)	6.47

表5：日立SR-8000における並列実行時間 [秒]

プロセッサ数	DIST	IND	SHAD
1	209.79	210.06	209.99
2	106.46	106.52	106.50
4	56.02	55.98	55.97
6	45.53	45.17	44.91
8	46.59	45.91	44.13

富士通VPP800におけるLOCレベルの実行時間を表4に示す．NECとは対照的に，富士通のコンパイラは自動並列ループの検出に失敗しており，INDEPENDENT文による指示が必要である．更に，SHADOW+REFLECT文による効率のよいシフト型の指示とLOCAL文による不必要通信の抑制を指示しなければ，コンパイラが自動生成する非効率な通信により実効的な性能が得られなかった．このため，富士通VPP800では，LOCAL文までの記述が事実上必須となる．また，富士通VPP800のプロセッサは，NEC SX-4より2.6倍の性能を持っているため，32台では計算粒度が小さくなりすぎて並列性能が低下している．このため，表4に示すようにサイズを256にすると，顕著な性能低下は見られない．

日立SR-8000における実行時間を表5に示す．すべてノード内通信である．日立のコンパイラもNECと同様に，すべての自動並列化に成功してるため，レベルによる並列性能の差異はほとんど見

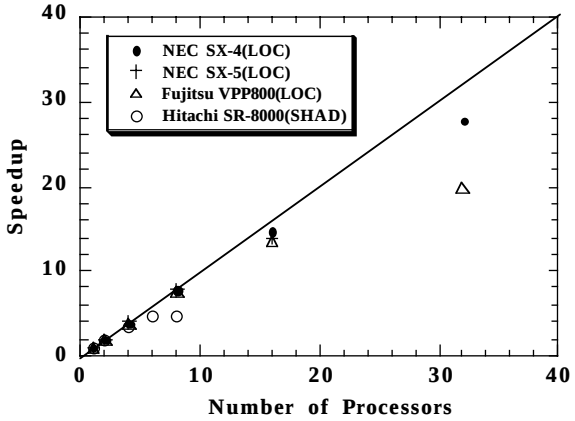

図2：3次元流体コードのスピードアップ

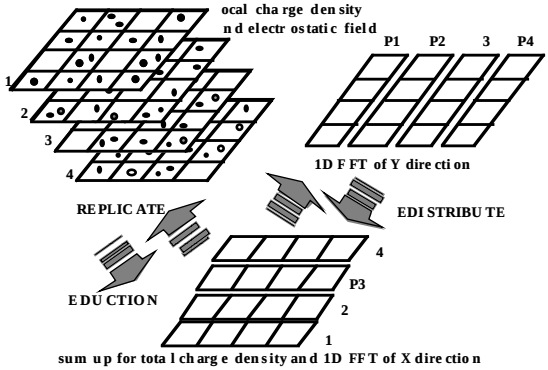

図3：2次元粒子コードにおける場変数の動的再分散と複写

られない．また，8台では著しい並列効率の低下が見られる．日立は，HPFをノード間の並列化に用いノード内は自動並列化を用いるという設計方針を取っているので，ノード内のみのHPF並列化は推奨していないことを付記しておく．

なお，日立のコンパイラはLOCAL文をサポートしていないため，LOCレベルのコンパイルはできない．

すべての並列計算機における並列実行によるスピードアップを図2に示す．非常に良好な並列性能が得られている．32台における並列効率の低下は，計算粒度が小さくプロセッサあたりz方向に2メッシュしか割り当てられないためである．

３．２次元静電粒子コードの並列化

次に，古典的Particle-In-Cell法を用いた2次元静電粒子コードを考える．このコードでは，2次元FFTを用いてポアソン方程式を解いて静電場を求め，その静電場内における電子と陽子の運動をすべての粒子について計算している．電荷密度を計算する部分に不規則な配列参照／代入があり，一般にはベクトル型スーパーコンピュータでも高い性能は得にくい．

並列化の手法として領域分割法を用いると，各プロセッサは自身が受け持つ領域内に存在する粒子に関する計算のみを行えばよいことになる．この場合，粒子がある領域から別の領域に移動すると，その粒子の計算を受け持つプロセッサが替わるので，その粒子の情報をプロセッサ間でやり取りしなければならない．しかし，このような通信パターンをHPFで記述するためには，かなりトリッキーなコーディングとなり，付加的な指示行を追加するだけでプログラムの並列化を行うHPF本来の趣旨から逸脱する．そこで，領域分割法ではなく，粒子情報を納めている配列を単純にBLOCK分散し，各プロセッサが受け持つ粒子群を固定して並列化を行う．このため，粒子の運動を計算するDOループは，無理なく効率よく並列化される．しかし，粒子は2次元のシミュレーション空間を自由に移動するため，各プロセッサは場の情報を納めている配列に関しては，全空間について重複して持つ必要がある．各プロセッサは，自身が受け持つ粒子の位置情報から全空間に分布するローカルな電荷密度を計算し，そのローカル電荷密度に対してREDUCTION演算を行うことにより，トータルな電荷密度を求める．トータルな電荷密度が求まると，それを2次元FFTして電荷密度のフーリエ係数を求めるが，2次元FFTは1次元FFTの重ね合わせで行っている．そこで，FFTを並列実行するため，1次元FFTの方向にあわせてFFTされる配列を再分散する．電荷密度のフーリエ係数が求まると，それにフォームファクタを乗じて電位のフーリエ係数を計算し，それを2次元逆フーリエ変換することで電位を求める．このときも，1次元逆フーリエ変換の方向にあわせて同様に配列を再分散する．各プロセッサは，電位について全空間の情報が必要なため，求まった電位を複写する．この様子を図3に示す．ローカルな電荷密度の計算は，通信の必要がないので効率よく並列化できるが，それ以外の計算には，REDUCTION，再分散，複写に伴う多くの通信が必要となる[9,10].

図3に示されているREDUCTION演算では，プロセッサ方向に分散された3次元配列を計算対象として結果をy方向に分散された2次元配列に格納することを前提としている．このため，効率のよ

表6：すべての計算機における並列実行時間 [秒]

プロセッサ数	SX-4	SX-5	VPP800	SR-8000
1	40.25	20.36	24.82	27.65
2	21.43	9.84	12.57	13.84
4	12.09	5.54	6.37	7.33
8	7.66	3.37	3.34	3.97
16	78.09	2.69	1.92	

表7：REDUCTION 文における記述法の差異

SX-4	!HPF$ independent,reduction(var)
SX-5	!HPF$ independent,reduction(max:var)
VPP800	!HPF$ independent,reduction(max:var)
SR-8000	!PFD$ independent,reduction:max(var)

オリジナル	**1591**
PROCESSORS 文	**11**
DISTRIBUTE 文	**19**
INDEPENDENT 文	**19**
ON-LOCAL/ENDON 文	**12**

図4：挿入したHPF指示文の行数

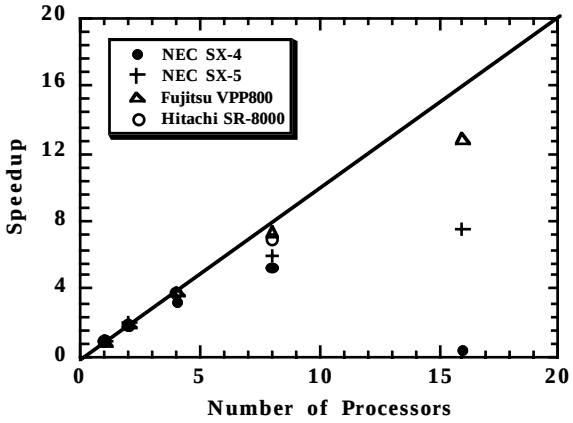

図5：2次元静電粒子コードのスピードアップ

い通信を行うには複雑な通信パターンを実装した並列コードの生成が求められるが，現状ではコンパイラの解析能力不足から非常に効率の悪い通信コードが生成され，並列実行によるスピードアップが得られなかった．そこで，REDUCTION演算の結果をプロセッサ毎に独立に確保された2次元一時配列にいったん格納し，REDUCTION演算終了後にトータル電荷密度をｙ方向に分散された2次元配列に代入するようソースコードを修正した．この結果，最適な通信パターンとは言えないものの，並列実行によるスピードアップが得られるようになった．

2次元FFTの並列実行は，1次元FFTルーチンの並列呼び出しで行うが，サブルーチンを並列呼び出しするためには，呼び出し側のINTERFACE BLOCK内にEXTRINSIC属性を新たに書く必要があった．またこの場合，実引数は非分散の配列でなければならない．2次元FFTルーチンとして受けた仮引数は分散されているため，その仮引数をそのまま1次元FFTルーチンの実引数としては使えず，いったん非分散の一時配列に複写するようソースコードを修正する必要もあった．図4に挿入したHPF指示文の行数を示す．

Lx=Ly=128, メッシュ当たりの粒子数を200としたときの，すべての並列計算機における実行時間を表6に，並列実行によるスピードアップを図5に示す．なお，並列性能は初期化を除いた主計算部のみの実行時間より評価した．通信が多いにもかかわらず，8台まででは，おおむね良好な結果が得られている．16台ではNEC SX-4の性能低下が著しいが，これはREDUCTION演算の効率が悪いためである．NEC SX-5では，コンパイラのバージョンが上がったので改善されている．

４．コンパイラの互換性

HPF指示文の記述については，国産3社でソースレベルでの互換性を保証しようとしているが[8]，現状ではREDUCTION文において，表7に示すように若干記述法が異なる部分があった．サブルーチンの並列呼び出しを可能とするためには，INTERFACE BLOCK内にEXTRINSIC属性を記述しなければならないが，この記述方法にも図6に示すように若干の差異があった．

また，実引数と仮引数の分散が異なる場合，富士通と日立では呼び出し元にINTERFACE BLOCKを書かなければならなかったが，NECでは必要なかった．

LOCAL文等のHPF/JA拡張仕様についても，3社でサポートの足並みが完全には揃っていない．日立は，SHADOW文と同等の機能をOVERLAP文で実現していたりLOCAL文をサポートしていないため，NECや富士通でコンパイルできたソースコードを日立でコンパイルしようとすると，ソースレベルでのコード修正が必要となる場合がある．

```
      interface
      extrinsic('FORTRAN','LOCAL') pure subroutine rfftf(n,a,b,c,j)
      dimension a(n),b(n),c(n),j(15)
      intent (in) :: n,b,c,j
      intent (inout) :: a
      end subroutine
      end interface          (a) NEC SX-4, SX-5

      interface
      extrinsic('FORTRAN','LOCAL') subroutine rfftf(n,a,b,c,j)
      dimension a(:),b(:),c(:),j(15)
      intent (in) :: n,b,c,j
      intent (inout) :: a
      end subroutine
      end interface          (b) 富士通 VPP800

      interface
      extrinsic(FORTRAN_LOCAL) pure subroutine rfftf(n,a,b,c,j)
!PFD$ LOCAL_PURE
      dimension a(n),b(n),c(n),j(15)
      intent (in) :: n,b,c,j
      intent (inout) :: a
      end subroutine
      end interface          (c) 日立 SR-8000
```

図6：EXTRINSIC属性における記述法の差異

以上より国産のHPFコンパイラについて，ソースレベルでは100%互換性があるとは言えないが，十分な互換性があることは検証できた．

５．むすび

並列計算機を一部の人のみが利用できる特殊な機械から一般ユーザでも活用できる高性能なツールとするためには，HPFのような高レベルで並列性を記述できる言語が不可欠である．また，HPFが広く普及するためには，各社の並列計算機上でソースレベルの互換性が保証されていることも重要である．

そこで，国産3社の並列計算機上のHPFコンパイラによる実用コードの並列性能を評価し互換性を検証するため，陽的解法を用いた3次元流体コードとPIC法を用いた2次元静電粒子コードを取り上げた．3次元流体コードでは，高い並列性能が得られることがわかった．2次元静電粒子コードでも良好な結果が得られたが，並列化のために若干のコード修正が必要であった．また，ソースコードの互換性については，100%とは言えないがほぼ満足な検証結果が得られた．

謝辞

本研究の実施において，並列計算機の利用環境を提供いただいた大阪大学サイバーメディアセンター（NEC SX-4,5），京都大学データ処理センター（富士通VPP800），大阪大学レーザー核融合研究センター（日立SR-8000）に謝意を表する．

参考文献

[1]Message Passing Interface Forum:A Message Passing Interface Standard, Int. J. Supercomputing Applications and High Performance Computing, Vol. 8, No. 3/4, pp. 165-416 (1994).

[2]Message Passing Interface Forum:Extensions to the Message-Passing Interface (1997).

[3]http://www/openmp.org/.

[4]Koelbel, C. et al.: The High Performance Fortran Handbook, The MIT Press, Cambridge, MA (1992).

[5]High Performance Fortran Forum:High Performance Fortran language specification, version 2.0 (1996).

[6]HPFF:HPF2.0公式マニュアル, スプリンガーフェラーグ東京 (1999).

[7]坂上仁志, 高橋豐:３次元流体コードの領域分割法における並列効率, 電情通学論(D-I), Vol. J80-D-I, pp.683-690 (1997).

[8]http://www.tokyo.rist.or.jp/jahpf/.

[9]H.Sakagami and Y.Ogawa, Proc. of 3rd HPF User Group meeting, Redondo, USA, Aug. 1-2 (1999).

[10]H.Sakagami, T.Mizuno and H.Murai, Proc. of 4th HPF User Group meeting, Tokyo, Japan, Oct. 19-20 (2000).