解　説　HPF言語の動向

5. Cenju-3におけるHPF処理系の開発と評価

Design and Implementation of an HPF Compiler for Cenju-3 by Tsunehiko KAMACHI, Kenji SUEHIRO, Kazuhiro KUSANO and Yoshiki SEO (C & C Research Laboratories, NEC Corporation).

蒲池恒彦[1]　末広謙二[1]　草野和寛[1]　妹尾義樹[1]

1 NEC C & C研究所

1. はじめに

本稿では，我々が開発したHPF処理系[1]の実現方法について，とくにHPF[2]の現状での問題点を解決するべく我々が独自に拡張した言語仕様を中心に説明する．我々の拡張仕様では，HPFでは記述できない計算処理のプロセッサへのマッピングや手続き呼び出しを含むループの並列化などが容易に記述することができる．また，NECの並列コンピュータCenju-3[3]上での実験結果を元に拡張言語仕様の評価を行う．

2. HPF言語の拡張

HPFで並列プログラムを開発する場合，プログラマが記述するのはデータのマッピング方法であり，実際の並列化作業，つまり計算処理のマッピングやプロセッサ間通信の生成などは言語処理系に委ねられている．しかし，残念ながら現在のコンパイル技術では必ずしも処理系が最適な並列化を行えるとは限らない．そこで，我々は，以下の制御を明示的に記述できるようにHPF言語を拡張し，プログラマがより効率のよい並列プログラムを容易に開発できるようにした．

1)計算処理のマッピング
2)手続き呼び出しを含むループの並列化を可能とする独自EXTRINSICインタフェース
3)オーバラップ領域を含むデータマッピング
4)プロセッサ間通信の抑制

以下，本稿では計算処理のマッピングと並列化制御にかかわる上記1)，2)について詳しく説明する．

2.1 計算処理マッピングの指示[4]

ここで扱う計算処理マッピングとは，ループの各繰返しをマッピングするイタレーションマッピングと，プログラム中のある実行区間を特定のプロセッサ群にマッピングする逐次ブロックマッピングの2種類である．

イタレーションマッピングの指示方法としては，ON節による指示方法も提案されているが[5),6)]，本HPF処理系での指示方法は，HPFにおけるデータマッピングの枠組みを利用しているところに特長がある．**図-1**はベンチマークプログラムPDE 1におけるイタレーションマッピング指示の適用例である（"!DIR$"で始まる行が我々が拡張した指示行である）．まず，3重ループのイタレーション空間を配列にみたてた3次元のイタレーションテンプレートITをITEMPLATE指示行で宣言し，これをITERATIONS指示行でループと対応づける．さらに，IALIGN指示行を用いて，ITを配列を整列させる方法と同じ方法で配列REDに整列させる．配列REDはDISTRIBUTE指示行で抽象プロセッサPE上にマッピングされるので，ITもREDと同様にマッピングされる．この結果，ITに対応づけられたループもITのマッピング方法にしたがってPE上にマッピングされることになり，3重ループのイタレーションマッピングが実現される．なお，イタレーションテンプレートはIDISTRIBUTE指示行を用いて直接PE上にマッピングすることも可能である．

以上のように，本処理系では配列のマッピングと同じ方法でイタレーションマッピングを指示できるため，イタレーション空間と配列の整列関係を記述できるなど，配列と同様に柔軟なイタレーションマッピングを実現できるだけでなく，図-1の例のように，多重ループを多次元プロセッサ上に一括してマッピングすることも可能である．また，処理系にとっても，配列とイタレーションの

```
      SUBROUTINE RELAX(U,F,RED,NX,NY,NZ,H,ITER)
      ............
!HPF$ PROCESSORS PE(2,2,4)
!HPF$ ALIGN U(I,J,K) WITH *RED(I,J,K)
!HPF$ ALIGN F(I,J,K) WITH *RED(I,J,K)
!HPF$ DISTRIBUTE RED *(BLOCK,BLOCK,BLOCK) ONTO *PE
!DIR$ ITEMPLATE  IT(:,:,:)
!DIR$ IALIGN IT(I,J,K) WITH RED (I+1,J+1,K+1)
      ............
!HPF$ INDEPENDENT
!DIR$ ITERATIONS IT(I,J,K)
      DO 2 K=2,NZ-1
        DO 2 J=2,NY-1
          DO 2 I=2,NX-1
            IF (RED(I,J,K) .EQV. .TRUE.) THEN
              U(I,J,K) = FACTOR*(HSQ*F(I,J,K) +
     *                    (U(I-1,J,K) + U(I+1,J,K) +
     *                     U(I,J-1,K) + U(I,J+1,K) +
     *                     U(I,J,K-1) + U(I,J,K+1)))
            ENDIF
    2 CONTINUE
```

図-1 PDE 1 におけるイタレーションマッピング指示の適用例

```
      SUBROUTINE COMP
      ..........
!HPF$ TEMPLATE TWOD(302,152)
!HPF$ ALIGN WITH TWOD::PVORT,YGRADS,VTSP,UTSP
!HPF$ DISTRIBUTE TWOD(*,BLOCK)
      INTERFACE
        EXTRINSIC(NONE) SUBROUTINE CMSLOW(J,YGRADS,UTSP,VTSP)
        ..........
        COMMON/BLK/ ...,PVORT(302,152),...
        DIMENSION YGRADS(302,152),UTSP(302,152),VTSP(302,152)
        INTENT (IN)    :: YGRADS(1:IM,J:J)
        INTENT (INOUT) :: UTSP(1:IM+2,j+1:J+1)
        INTENT (INOUT) :: VTSP(1:IM+2,j+1:J+1)
        INTENT (IN)    :: PVORT(1:IM+2,j+2:J+2)
      END SUBROUTINE
      END INTERFACE

!HPF$ INDEPENDENT
      DO 100 J=1,JM
      ...
        CALL CMSLOW(J,YGRADS,UTSP,VTSP)
```

図-2 Baro における独自 EXTRINSIC インタフェースの適用例

マッピングを同じ枠組みで扱えるという利点がある．

一方，逐次ブロックマッピングにおいては，特定のプロセッサ1台で実行する SINGLE ブロック，全プロセッサで実行する ALL ブロック，特定の配列要素がマッピングされているプロセッサ群で実行する OWNER ブロックの3種の実行形態を定義し，これらのブロックの範囲を指示する指示行を用意した．ALL/SINGLE ブロックは，イタレーションマッピングを行った結果通信が頻発し，並列実行することによってかえって実行時間が増加する場合など，ループを全プロセッサあるいは特定のプロセッサで実行した方が有利な場合に有効である．また，OWNER ブロックを用いれば，配列の境界要素など特定の要素に対する処理を行う場合に不要な通信を抑制することができる．

2.2 独自 EXTRINSIC インタフェース

HPF では，副作用のない関数呼び出しを除いては，手続き呼び出しを含むループを並列化することができないが，我々は，独自の EXTRINSIC インタフェースを構築し，副作用のある手続きも並列ループから呼び出せるようにした．この機能を用いれば，たとえ呼び出される手続きのソースプログラムが存在しなくとも，当該手続きが内部で呼び出されるループを並列化することが可能となる．なお，EXTRINSIC インタフェースとは，HPF 手続きから他言語で記述された手続きを呼び出すために HPF で用意されているインタフェースであり，呼び出される手続きは Fortran 90 のインタフェースブロック内で宣言されなければならない．

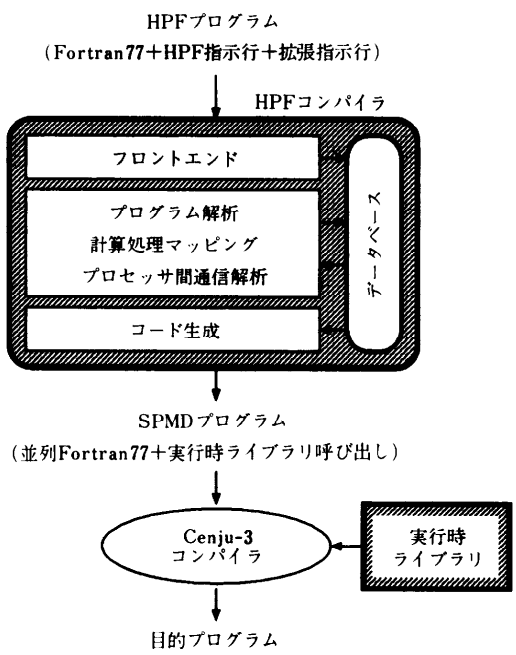

図-3 HPF 処理系の構成

手続き呼び出しを含むループを並列化する場合，手続き内でアクセスされる配列に対する通信コードは呼び出し側で生成されるため，副作用の情報，すなわち手続き内での配列のアクセス情報が必要となる．この情報はインタフェースブロック内で INTENT 文を用いて記述できるが，現在の INTENT 文の仕様では参照/更新情報 (IN/OUT/INOUT) のみしか記述できないため，正確なアクセス情報を記述することができない．このため，我々はアクセス範囲を任意の整数式を用いて記述できるように INTENT 文を拡張した．図-2 にベンチマークプログラム Baro に適用し

た例を示す．INTERFACE～ENDINTERFACEで囲まれた部分がインタフェースブロックである．拡張INTENT文で手続きCMSLOWの引数であるYGRADS，UTSP，VTSPおよびコモン変数のPVORTの参照/更新範囲を“:”で区切った上限/下限で記述している．この情報に従って，HPFコンパイラは適切な通信コードを生成することができる．

3. コンパイル処理の概要

HPFでは，データのマッピングはHPFで定義される抽象プロセッサ上で表現され，抽象プロセッサから物理プロセッサへのマッピングは処理系に委ねられている．我々は，移植性が高く，かつ実行時の処理が柔軟に行えるHPF処理系を構築するため，**図-3**に示すように，HPF処理系をHPFコンパイラと実行時ライブラリで構成した．本処理系では，HPFコンパイラがプログラム解析結果に基づいて並列化可能なループを抽出した後，計算処理マッピング[4]およびプロセッサ間通信[1),7)]を抽象プロセッサ上で実現するコードを生成する．そして，実行時ライブラリで抽象プロセッサから物理プロセッサへのマッピングが施され，物理プロセッサ上での並列実行およびプロセッサ通信が実現される．通信最適化に関しては，コンパイル時に十分な情報が得られる場合にはHPFコンパイラで，静的には確定されない情報がある場合には実行時ライブラリで行われる[7)]．

以上のように，本処理系ではHPFコンパイラはターゲットとなる並列システムの物理プロセッサ構造や，提供される低レベルな通信ライブラリのインタフェースを意識する必要がない．したがって，並列システムで利用可能な通信ライブラリを用いて実行時ライブラリを構築すれば，HPFコンパイラを変更することなしにCenju-3以外の他システムへ本処理系を移植することが可能である．さらに，現在通信用のライブラリは標準の通信ライブラリであるMPI (Message-Passing Interface)[8)]を用いて構築しているため，MPIをサポートしているシステムであれば実行時ライブラリを再コンパイルするだけで移植が可能である．

4. 評価結果

表-1に示したベンチマークプログラムを我々の拡張HPFで並列化しCenju-3上で評価した．表中ex 1～ex 3は**2**章で説明した独自EXTRINSICインタフェース，イタレーションマッピング，逐次ブロックマッピングの各拡張仕様をそれぞれ表しており，“x”は各拡張仕様を適用できたか否かを表している．Cenju-3は64台のPEを多段結合網で接続した分散メモリ型並列コンピュータで，各PEには75 MHzで

表-1 ベンチマークプログラム一覧

Program	Suite	計算モデル	Shape	分散	ex 1	ex 2	ex 3
Shallow	NCAR	Shallow Water	(513, 513)	(*, block)			x
Grid	APR	Simple Grid Relation	(502, 502, 2)	(*, block, *)			x
X 42	APR	Fourth Order Differencing	(512, 512)	(*, block)			x
Baro	APR	Atmospheric Model	(302, 152)	(*, block)	x		x
Scalgam	APR	Monte Carlo	スカラ	(*, block)	x		x
PDE 1	Genesis	3 D Red-Black SOR	(128, 128, 128)	(block, block, block)		x	x
SP	NAS	Simulated CFD	(64, 64, 64)	(*, *, block)		x	x
EP	NAS	Embarrassingly Parallel	スカラ	(*, block)	x	x	x

ex 1：独自EXTRINSICインタフェース，ex 2：イタレーションマッピング，ex 3：逐次ブロックマッピング

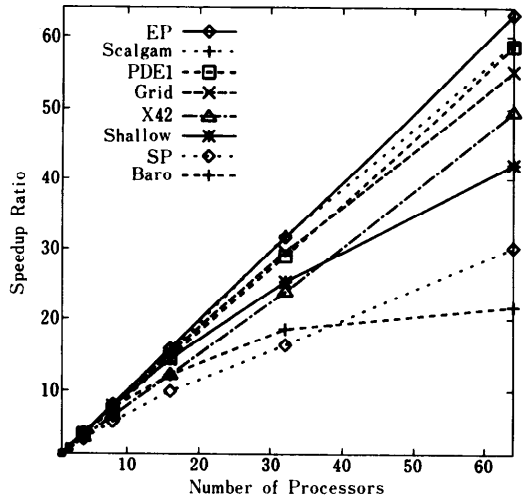

図-4 ベンチマークプログラムの実行結果（速度向上率）

表-2 Baroにおける拡張仕様適用の効果

適用/非適用	original	1 PE	2 PE	4 PE	8 PE	16 PE	32 PE	64 PE
適用(秒)	8.90	8.88	4.36	2.14	1.17	0.73	0.48	0.41
非適用(秒)	8.90	9.04	10.25	10.79	14.04	21.26	23.75	21.43

動作するRISCマイクロプロセッサVR 4400と64 MBのローカルメモリを備えている[2]．

図-4に実行結果として，オリジナルプログラムを1 PEで実行した時間に対する速度向上率を表したグラフを示す．逐次ブロックマッピングはすべてのプログラムに適用することで効果が得られている．Baro，Scalgam，EPでは独自EXTRINSICインタフェースを用いて手続き呼び出しを含むループを並列化した．**表-2**にBaroにおいて独自EXTRINSICインタフェースを適用しなかった場合との比較を示す．適用しなかった場合はループを並列化できなかったため，速度向上がまったくみられない結果となっている．また，PDE 1，SPではイタレーションマッピング指示を行ってループを並列化した．

5. おわりに

以上，我々が開発したHPF処理系，とくにHPFの拡張仕様について述べた．ベンチマークプログラムを用いたCenju-3上での評価ではいずれのプログラムにおいても拡張仕様の適用によってよい結果が得られており，その有効性を示すことができた．今後も引続きほかのベンチマークプログラムで評価を行っていくとともに，我々が拡張した指示行をHPF Forumへ提案していく予定である．

謝辞 本研究の機会を与えていただいた第一コンピュータソフトウェア事業部 片山事業部長代理，C＆C研究所コンピュータシステム研究部中崎部長に感謝いたします．

参考文献

1) 蒲池ほか：HPF処理系の実現と評価，情報処理学会論文誌，Vol. 37，No. 7，pp. 1255-1264 (July 1996)．
2) High Performance Fortran Forum: High Performance Fortran Language Specification Version 1.1 (Nov. 1994)．
3) 広瀬ほか：並列コンピュータCenju-3のプロセッサ間通信方式とその評価，並列処理シンポジウム，pp. 241-248 (1995)．
4) 末広ほか：HPF処理系における計算処理マッピング，並列処理シンポジウム，pp. 369-376 (1995)．
5) Tseng, Chau-Wen: An Optimizing Fortran D Compiler for MIMD Distributed-Memory Machines, Ph. D. Thesis, Rice University (1993)．
6) Koelbel, C. et al.: Supporting Shared Memory Data Structures on Distributed Memory Architectures, Proc. of 2nd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, pp. 177-186 (1990)．
7) Kamachi, T. et al.: Generating Realignment-Based Communication for HPF Programs, Proc. of the 10th International Parallel Processing Symposium (1996)．
8) Message Passing Interface Forum: MPI: A Message-Passing Interface Standard (1995)．

（平成8年10月23日受付）

蒲池 恒彦（正会員）

1964年生．1988年九州大学工学部情報工学科卒業．1990年同大学院総合理工学研究科情報システム学専攻修了．同年NEC入社．現在，C＆C研究所コンピュータシステム研究部主任．並列計算機アーキテクチャ，自動並列化コンパイラ，並列化支援ツールなどに興味をもつ．

末広　謙二（正会員）

1965年生．1988年京都大学工学部電気工学科卒業．1990年同大学院情報工学専攻修士課程修了．1993年同博士課程単位取得退学．同年NEC入社．現在，C＆C研究所コンピュータシステム研究部主任．並列計算機の基本ソフトウェアの研究開発に従事．

草野　和寛（正会員）

1965年生．1989年九州大学工学部情報工学科卒業．1991年同大学院総合理工学研究科情報システム学専攻修了．同年NEC入社．現在，C＆C研究所コンピュータシステム研究部勤務．分散メモリマシンのための並列化支援システムの研究に従事．

妹尾　義樹（正会員）

1961年生．1984年京都大学工学部情報工学科卒業．1986年同修士課程修了．同年NEC入社．以来C＆C研究所にてスーパーコンピュータの研究開発に従事．とくに並列処理アーキテクチャ，分散メモリマシンのための並列化支援システム，並列アルゴリズムに興味をもつ．現在C＆C研究所主任．工学博士．1988年本会論文賞受賞．