並列化支援環境 PCASE における分散メモリ対応機能

蒲池恒彦† 妹尾義樹†

本論文では，現在われわれが開発している並列化支援環境 PCASE (Parallelizing CASE) について主に分散メモリマシンを対象とした機能を述べる．PCASE は Fortran プログラムの並列化作業を支援するためのシステムであり，グラフィック・ユーザインタフェースを備えた対話型並列化支援ツール Xpallas と，SPMD (Single-Program Multiple-Data) モデルに基づく並列プログラムを生成するトランスレータ DCM (Data distribution & Communication Manager) とからなる．Xpallas はデータ依存関係解析に基づいて DO ループの並列性を抽出すると共に，階層メモリマシンを仮定してデータ転送の解析を行う．一方，DCM では Xpallas が抽出したデータ転送とユーザが指示するデータの分散配置情報から，分散メモリマシン上でイタレーションマッピング，および実行時制御に基づくデータ転送コード／同期コードの生成を行い，対象マシン上で動作可能な並列プログラムを自動生成する．われわれは，PCASE を当社で開発した分散共有メモリマシン Cenju 2 にインプリメントするため，分散メモリマシン対応の DCM のプロトタイプとして Cenju 2-DCM を開発し，Cenju 2 プロトタイプ (16 台構成) 上で評価を行った．その結果，SCG (Scaled Conjugate Gradient) 法を用いた Poisson solver のプログラム (問題サイズ：60×60×60) において，ユーザが配列の分散配置を指定した場合に，16 台で約 11 倍の性能向上を得た．

Data Distribution Management for Distributed Memory Machines in Parallel Programming Environment PCASE

Tsunehiko Kamachi† and Yoshiki Seo†

This paper presents data distribution management and communications generation techniques for distributed memory machines in parallel programming environment PCASE. PCASE consists of two parts. One is a machine independent interactive parallelizer Xpallas which generates IPF (Intermediate Parallel Fortran) programs, and the other is DCM (Data distribution & Communication Manager) which accepts IPF programs and generates target parallel programs based on SPMD (Single-Program Multiple-Data) model. Xpallas extracts parallelism from DO loops and finds data transfer assuming that the target machine is a hierarchical memory model. DCM realizes iteration mapping of parallelized loops based on owner compute rule, and inserts communication and synchronization codes referring the information about data transfer and data distribution described in IPF program. We carried out performance measurement on a 16 node Cenju 2, with two VR 3000 RISC microprocessors and 64 MB memory per node, using 3 D poisson solver program (60×60×60) based on SCG (Sealed Conjugate Gradient) method. In this experiment, we achieved 11 times speedup as compared with executing on one node when data distribution is specified.

1. はじめに

近年，並列計算機の研究が国内外の研究機関で活発に行われており，数百台〜数千台規模の商用マシンが開発されるようになってきている．しかしながら，これら並列計算機の能力を十分に引き出す並列プログラムを開発するためには，プログラムの構造や並列化手法に関する知識はもちろんのこと，対象となる並列マシンのアーキテクチャやその特性についての知識も必要となってくる．さらに，膨大な既存ソフトウェアを抱えているユーザにとっては，これらを並列プログラムに書き換えるという莫大な作業が課せられることになり，並列計算機の実用化を阻害する最大の要因となっている．

このような背景からわれわれは，Fortran プログラムの並列化作業を支援するための並列化支援環境 PCASE (Parallelizing CASE) を開発中である[1]．PCASE は，逐次 Fortran プログラムをユーザとの共同作業で並列プログラムに変換する対話型のプリコンパイラであり，マシン非依存の対話型並列化支援ツール *Xpallas* とマシン依存部分の解析を行い，対象マシンの並列プログラムを生成するトランスレータ

† 日本電気(株) C & C 研究所
C & C Research Laboratories, NEC Corporation

DCM (Data distribution & Communication Manager) からなる.

最近では並列化支援システムの研究が Rice 大を中心に，特に分散メモリマシンを対象として盛んに行われるようになってきており[2),3)]，Express[4)]，FORGE 90[5)]などの商用システムも登場しているが，PCASE はデータ転送解析手法に特長がある．つまり，Xpallas では，共有メモリとローカルメモリを持つ階層メモリマシンを想定したデータ転送解析を行い，ここで抽出したデータ転送から DCM で対象マシンに適合したデータ転送を抽出する．このとき，ユーザはデータ転送を直接削除することも可能であり，データの分散配置情報を付加することで，マシン依存部に最適化を委ねることができる．また，並列化の効果を DO ループ，サブルーチン，プログラム全体を単位として定量的に予測する静的性能予測機構を備えており，この予測結果を参照することにより，ユーザはループおよびデータ転送の最適化を効果的に行うことができる．

われわれは，PCASE を当社で開発した分散共有メモリマシン Cenju 2 にインプリメントするため，分散メモリマシン対応の DCM のプロトタイプとして Cenju 2-DCM を開発した．Cenju 2-DCM は Xpallas が生成した中間並列化形式プログラムから Cenju 2 の並列プログラムを生成するトランスレータであり，以下の特長を有する．

1. 分散指定されたデータに関しては，データの局所性を考慮した制御を行い，分散指定されていないデータに関しては，共有メモリイメージを分散メモリ上に実現する．
2. 分散配置するデータに関しては，基本的には owner compute rule[3)]に基づいたイタレーション割り付けを行うが，ステートメント単位に割り付ける方式とループボディ単位に割り付ける方式をサポートする．
3. Cenju 2 の特徴を活かすため，他 PE のローカルメモリの参照を含め，すべてのデータ転送は他 PE のローカルメモリへの書き込みで実現する．

本稿では，PCASE の概要を述べた後，Cenju 2-DCM の詳細および Cenju 2 上での評価結果について述べる．

2. PCASE の概要

2.1 PCASE の構成

図 1 に PCASE の構成を示す．PCASE は入力された Fortran プログラムからデータ依存関係解析に基づいて DO ループの並列性を抽出し，中間並列化形式プログラム (IPF : Intermediate Parallel Fortran) に変換する Xpallas と，IPF を対象マシン上で動作可能な並列プログラムに変換するトランスレータ DCM (Data distribution & Communication Manager) とで構成される．

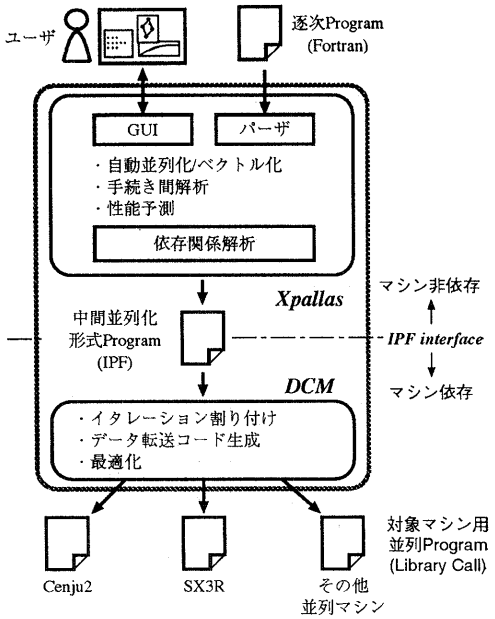


図 1 PCASE の構成
Fig. 1 Configuration of PCASE.

2.2 マシンモデル

PCASE が扱うマシンモデルを**図 2** に示す．このモデルでは「各プロセッサ固有の Local Area，各プロセッサからアクセス可能な Shared Area，各プロセッサからすべての空間をアクセス可能であるが，アクセス速度が不均一な Distributed Area の 3 種類の属性の空間がサポートされている」ことが想定されている．また，Xpallas はループの並列化判定，ベクトル化判定の両方の機能を有しており，各プロセッサにはベクトルプロセッサが付加されていてもよい．

2.3 Xpallas-DCM 間インタフェース

PCASE では，上記マシンモデルにおいて，Xpallas-DCM 間のインタフェース (IFP インタフェース) を並列マシンの汎用的なメモリ構造である，階層メモリマシン (共有メモリと各プロセッサに固有のローカルメモリを備える) に設定している．

これは，Xpallas をマシン独立な並列性抽出システムとして構築することを可能にするためである．

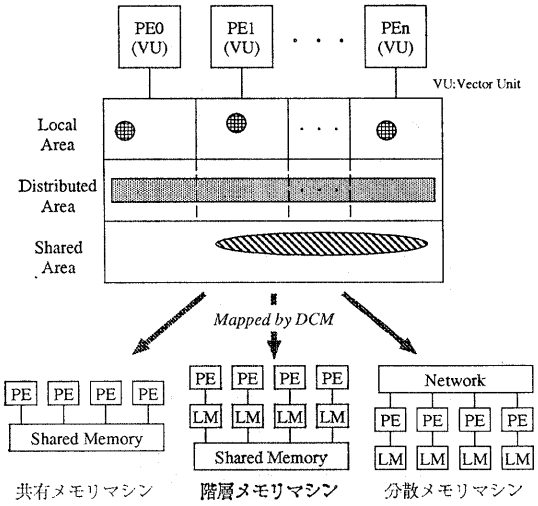


図 2 対象マシンモデル
Fig. 2 Target machine model.

特に，分散メモリ型並列マシンにおいて非常に重要となるデータ転送解析に関しては，Xpallas の解析対象マシンを階層メモリマシンに設定することで「潜在的なデータ転送」を抽出することができる．「潜在的なデータ転送」とは階層メモリマシンでの共有メモリ-ローカルメモリ間のデータ転送であり，ローカルメモリの有無，また，ローカルメモリ上でのデータの時間的あるいは空間的な局所性によっては不必要になるかもしれないデータ転送である．また，これを認識することは，対象マシンに依存したデータ転送の最適化を行うにあたって重要な意味を持つ．

ここで，ユーザは以下の手順により Xpallas が抽出した「潜在的なデータ転送」から自己の対象マシンにおいて必要なデータ転送のみを抽出することができる．また，これらの最適化は Xpallas が提供する性能予測機能の予測結果を参照することで効果的に行える．

1. Xpallas が提供する GUI を介して時間的局所性によって不要となるデータ転送を削除する．
2. 後述する DISTRIBUTE 指示行を用いてローカルメモリ上でのデータの分散配置手法を記述し，空間的局所性に起因する不要データ転送を削除する．

ただし，現在 Xpallas ではデータの分散配置を意識した解析は行っていないため，ユーザによって指示されたデータ分散配置情報をそのまま DCM に伝える機能のみを有し，データ分散配置に関する最適化は DCM に委ねられている．

以下，本章では Xpallas および DCM の機能について簡単に述べる．

2.4 Xpallas の機能

Xpallas はデータ転送機能のほかに以下の機能を有する．

1. データ依存関係解析に基づいて，DO ループの並列化（現在は DOALL ループのみ）/ベクトル化/ループ再構成（ループ入れ換え，ループ分割）の可能性を調べる．
2. 定数伝播，配列アクセス範囲解析，並列化解析を手続き間に渡って解析する[10]．
3. グラフィック・ユーザインタフェース（GUI）を介して CALL グラフ，依存関係情報等のプログラム解析情報を表示し，また，ユーザによる並列化制御の指示を受理する．
4. プログラムを仮実行して得たトレースデータ（各行の実行回数）と対象マシンの特性を記述したハードウェア記述ファイル（マシンサイクル，各命令の実行サイクル数，通信コスト，同期コスト等）からプログラムの最適化の効果を DO ループ，サブルーチン，プログラム全体を単位として定量的に予測し，最適化の指針にすると共に GUI を介してユーザへその結果を表示する．
5. データ転送解析結果，各 DO ループの実行形式（並列実行/ベクトル実行/スカラ実行）の結果，および並列実行のための制御に関する指示行を入力されたプログラム中に挿入して IPF を生成する．

2.5 DCM の機能

Xpallas が生成した IPF を受理し，対象マシン用の並列プログラムを SPMD (Single-Program Multiple-Data) モデルに基づいて生成する．この際，実行時制御に基づくデータ転送コード，同期コード，およびその他の並列実行制御コードを挿入する．また，対象マシンが分散メモリマシンである場合には後述するデータ分散配置に関する指示行の情報をもとに，並列ループの各 PE への割り付け，データ転送コードの生成を行う．

3. Cenju 2-DCM

3.1 概 要

Cenju 2-DCM は Xpallas が生成した IPF から Cenju 2 用の並列プログラムを生成するトランスレー

タである．

これまで，さまざまな研究機関で分散メモリマシンをターゲットとした並列言語/コンパイラが研究されているが[3),8)]，分散メモリマシンにおいては，データを如何に分割し，分散されたメモリに配置するかが並列プログラムの性能を大きく左右する．ただし，現在 Xpallas では，分散メモリを意識したデータ分割のための解析は行っておらず，データの分散配置方法については，ユーザが指示行で指定する方式を採っている．また，Cenju 2-DCM は，Xpallas が生成した IPF を入力として，

1. メモリ管理
2. イタレーションの割り付け
3. データ転送，および同期コードの生成/挿入
4. 総和計算などのマクロ演算の並列化
5. 最適化

を行い，Cenju 2 がサポートしている初期化，通信，同期などの並列実行ライブラリを挿入して Cenju 2 の並列プログラムを生成する．また Cenju 2 は，

1. 他の PE のローカルメモリへシステムでユニークなグローバルアドレスによってアクセスすることができる．
2. 各 PE は送受信用の 2 つの DMAC (Direct Memory Access Controller) を備え，ローカルメモリ間でデータのバーストコピーを高速に行うことができる．
3. 各 PE 間はマルチキャスト機能を組み込んだ多段接続網で接続される．

という特徴を有する[9)]．これらの特徴を考慮し，Cenju 2 の性能を最大限に引き出すように DCM を実装した．

3.2 指示行インタフェース

Cenju 2-DCM が受理する指示行を**表 1** に，Xpallas の出力例 (IPF) を**図 3** に示す．

LOCAL 指示行は，並列ループ内の共有変数に対するアクセス属性，アクセス範囲，並列化されるループ変数によってアクセスされる次元を解析した結果，Xpallas が並列ループの直前に挿入する指示行であり，階層メモリマシンにおいて共有メモリ-ローカルメモリ間で生じる「潜在的なデータ転送」を表している．つまり，LOCAL 指示行に記述された変数のうち，参照データは並列ループの実行に先だって共有メモリからローカルメモリへコピーし，更新データは実行後にローカルメモリから共有メモリへ書き戻す必要があるデータである．この LOCAL 指示行インタフェースにより，ユーザは Xpallas を用いて，対象マシンを意識することなく逐次 Foratran レベルでデータ転送の抽出/最適化を行うことができる．一方，DCM では LOCAL 指示行を参照するだけで，デー

表 1 IPF 指示行一覧
Table 1 IPF directives.

指示行	意味
PARDO	直後の DO ループを並列実行する
SCALAR	直後の DO ループをスカラ実行する
VECTOR	直後の DO ループをベクトル実行する
LOCAL symbol[(pattern)], type	直後の DO ループ内でアクセスされる共有変数のアクセス範囲 (pattern). アクセス属性 (type: 参照，更新，参照／更新)
RESERVE[=*n*]	*n* 台のプロセッサを確保する
BARRIER	全プロセッサで同期をとる
MACRO type symbol	直後のループは MACRO 演算である (type: SUM, MAX, MIN)
DISTRIBUTE symbol[dist-format]	配列 (symbol) の分散配置の指定

```
      program test
      parameter(n=32)
      real*8 A(n,n),B(n,n),C(n,n)
*PDIR DISTRIBUTE A(BLOCK,*)
*PDIR DISTRIBUTE B(BLOCK,*)
*PDIR DISTRIBUTE C(BLOCK,*)
*PDIR RESERVE[=16]
        .
        .
*PDIR PARDO
*PDIR LOCAL A(i,1:31),MOD
*PDIR LOCAL B(i+1,2:32),REF
*PDIR LOCAL C(i,1:31),REF
      do i = 1 , n-1
        do j = 1 , n-1
          A(i,j)=B(i+1,j+1)+C(i,j)
        enddo
      enddo
        .
        .
```

図 3 Xpallas の出力例 (IPF)
Fig. 3 Example of IPF.

タ転送コードを生成することが可能となり，また，データ転送のブロック化が容易に行える．

LOCAL 指示行において，ループ変数の1次式で表されている次元が並列ループによってアクセスされる並列化次元であり，その他の次元のアクセス範囲は初期値，終値，増分値の3つ組で表される．また，アクセスの属性は参照 (REF)，更新 (MOD)，参照/更新 (BOT) の3種である．図3の例では，並列ループでアクセスされる配列Bの次元は1次元目であり，ループ変数の1次式 "i+1" で参照されることを表している．

また，DISTRIBUTE 指示行は，Fortran-D[6]，Vienna Fortran[7]，HPF[11]における分散配置指定用の指示行と同様の指示行であり，各次元に対して以下の分散配置方法が指定可能である．

BLOCK(M) サイズMで分割し，おのおのを各 PE へ割り付ける．Mを省略した場合はプロセッサ数で均等分割する．

CYCLIC(M) サイズMを1単位とし，ラウンドロビン方式で各 PE へ割り付ける．Mを省略した場合は M=1 と解釈する．

＊ 分散を行わない．

図3の例では配列A，B，Cをすべて1次元目で BLOCK 分割することを指示している．ただし，現在は DCM による解析を簡略化するため，DISTRIBUTE 指示行の記述には次の制限を設けている．

1. 分割指定が可能なのは1つの次元のみとする．
2. 手続き間に渡った分散配置解析は行っていないため，DISTRIBUTE 指示は各プログラム単位で各配列ごとに記述する．
3. 動的再配置はサポートしない．

また，DISTRIBUTE 指示行を挿入したプログラムを Xpallas の入力とした場合は，Xpallas は本指示行で指示されたデータの分散配置を考慮した並列化解析を行う．すなわち，複数ループが並列化可能である場合は，データ転送量が最小になるように配列の分割次元をアクセスするループを優先的に並列化する．

3.3 メモリ管理

DISTRIBUTE 指示行で指定された配列は，分散されたおのおのの領域を，各 PE で1つのコピーしか存在しない single copy データとして各 PE のメモリ上へ配置する．ここで，各 PE へ割り付けられた領域をそれぞれ各 PE の owner 領域と呼ぶ．一方，DISTRIBUTE 指示行で指定されなかった配列は，全 PE でデータの一貫性を保証する multiple copy データとして全 PE のメモリ上に配置する．ただし，現在では分散配置する配列も全 PE で全領域を確保している（**図4**参照）．

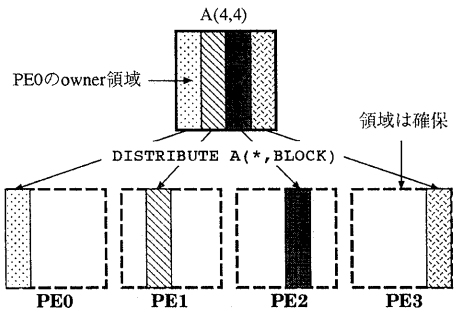


図4 分散配置データのメモリ管理
Fig. 4 Memory management of distributed data.

3.4 イタレーションの割り付け

イタレーションの割り付けはデータの局所性を考慮し，基本的には *owner compute rule* に基づいて行う．すなわち，各 PE は自己の owner 領域への書き込みが生じるイタレーションを実行する．具体的には，自己の owner 領域へのアクセスか否かを判定する IF 文でステートメントをガードする（このガードを実行ガードと呼ぶ）．owner 領域か否かの判定にはガード基準配列を用いて行う．ここでガード基準配列とは，LOCAL 指示行で指示されている配列のうち，MOD または BOT の属性を持つ配列である（図3の例ではガード基準配列はAである）．図3のループのイタレーションの割り付け結果の例を**図5**に示す．

前述のように基本的には owner compute rule に基

```
      .
      .
do i = 1 , n-1
  if(A(i,*) は owner 領域である)then
    do j = 1 , n-1
      A(i,j)=B(i+1,j+1)+C(i,j)
    enddo
  endif
enddo
      .
      .
```

図5 イタレーションの割り付け
Fig. 5 Iteration mapping.

```
*PDIR PARDO
*PDIR LOCAL A(i),BOT
*PDIR LOCAL C(i),REF
*PDIR LOCAL B(i+1),MOD
      do i = 1 , n-1
S1:     A(i) = C(i)*2
S2:     B(i+1) = A(i)
      end do
```

図 6 フロー依存がある場合
Fig. 6 Example of flow dependency.

づいてイタレーションの割り付けを行うが，並列ループ中に MOD あるいは BOT 属性を持つ配列が複数存在する場合は，以下のように，おのおののステートメントに対して単純に owner compute rule を適用することができない場合がある．

- ループ中にフロー依存がある場合
  図 6 の例で考える．このとき，S1，S2 の間にフロー依存が存在する．このとき，S1，S2 に対して単純に owner compute rule を適用し，それぞれ異なる実行ガードを生成すると，S1 で A(i)を更新する PE と S2 で A(i) を参照する PE が異なる場合が生じる．したがって，この場合は S1 と S2 の間でデータ転送，同期を行う必要があり，実際に並列ループ内でデータ転送を行うか，可能であればループ分割を行って，ループ間でデータ転送を行うことになる．いずれにしても効率が悪くなる．
- 手続き呼び出しがある場合
  並列ループ内に手続き呼び出しがある場合，手続き内のステートメントをガードできない．
- IF 文中で分散配置された配列の参照がある場合
  並列ループ内に IF ブロックが存在する場合は，IF 文に対するガードが複雑になり，実行時のオーバヘッドが大きくなる．

この場合はガード基準配列を 1 つ選択し，1 種類の実行ガードで並列ループのボディをガードする．ただし，上記の問題がない場合には，MOD または BOT 属性を持つ配列の数，配列のサイズ，最適化の効果などによって実行時のオーバヘッドが異なり，これを定量的に判断して最適な方式を自動的に選択するのは困難なため，ステートメント単位にガードする方式とループボディをガードする 2 つの方式を選択できるようにしてある．

3.5 データ転送コードの生成

イタレーションの割り付けを行った後，LOCAL 指示行と DISTRIBUTE 指示行の情報をもとに各並列ループ単位で並列実行に伴うデータ転送の解析を行い，実行時制御に基づくデータ転送コードを生成する．データ転送コードの生成方法は，並列ループ内でアクセスする配列が single copy データであるか multiple copy データであるかによって以下のように異なる．

3.5.1 single copy データ

並列ループ内で自己の owner 領域以外の領域をアクセスする場合，その領域の owner である PE のメモリ上のデータをアクセスしなければならず，データ転送コードを生成する必要がある．single copy データでは，以下の条件のいずれかを満たす配列へのアクセスが他の PE の owner 領域へのアクセスを引き起こす．

1. LOCAL 指示行内で指示されている並列化次元のインデックスがガード基準配列のそれと異なる．
2. 分割次元の配列サイズがガード基準配列のそれと異なる．
3. 分割方法がガード基準配列のそれと異なる．
4. 分割次元と並列化次元が異なる．
5. 並列化次元を持たない

この配列をデータ転送対象列と呼び，おのおののアクセス属性ごとに，次のようにデータ転送コードを生成する．

参照 (REF)

本属性データは，参照前に参照する領域の owner である PE のメモリ上から読み出す必要があるデータである．しかしながら，Cenju 2 では他 PE のメモリからの読み出しに比較して，他 PE のメモリへの書き込みの方が圧倒的に高速であり，DMA 転送を行うことでさらに高速化が可能である．このように Cenju 2 の特徴を考慮し，すべてのデータ転送は他 PE のメモリへの書き込みで実現する．すなわち，自己の owner 領域以外の参照は，(1)他 PE が参照する領域の owner である PE が，参照する PE のメモリ上の領域を更新し，(2)その後に参照するという手順で実現する (図 7 参照)．

具体的なデータ転送コードの生成方法は，上記の条件によって異なるが，ここでは，LOCAL 指示行における並列化次元のインデックスがガード基準配列のそ

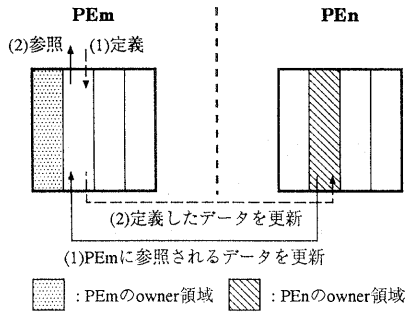


図 7 single copy データのデータ転送
Fig. 7 Data transfer for single copy data.

れと異なる場合のデータ転送コードの生成法について述べる．

本データ転送コードは，並列ループの直前に実行時制御に基づいてデータを転送する点に特徴がある．すなわち，転送すべき PE および配列要素に関する情報を実行時に取得し，データの転送を行う．これは，並列ループの直前にデータ転送を行うことによって，転送するデータをブロック化し，Cenju 2 の DMA 転送の効果を引き出すためであり，また，実行時制御を行うことで，静的にはわからない情報がある場合でも対応可能とするためである．

データ転送時の処理は次のように行う．まず，他の PE が並列ループ内で参照する要素のうち，自己のローカルメモリに割り付けられている要素があるか否か調べる．これは，他のすべての PE に対して，データ転送対象配列へアクセスするインデックスを，並列ループと同一の初期値，終値，増分値のループ，および同一の実行ガードを用いて調べることで行う．ここで，自己の owner 領域を参照する PE とその参照範囲が判明する．次に，参照される自己の owner 領域の要素を当該 PE に転送し，ローカルメモリ上の内容を更新する．**図 8** は図 3 の例で生成するデータ転送コードの処理を表したものである（データ転送対象配列は B）．

具体的には，データ転送コードは LOCAL 指示行，DISTRIBUTE 指示行の情報をもとに以下のステップで生成する．

1. 他の PE に対してデータを転送すべきか否かを調べるために，プロセッサの論理 ID をループ変数にしたループを生成する．
2. 調査対象のプロセッサから自分を除くための条件文を生成する．
3. 並列ループと初期値，終値，増分値が同一のループを生成する．
4. 並列ループの実行ガードと同一の実行ガードを生成する．
5. 調査する PE が参照する領域が自己ローカルメモリ上に割り付けられているか否かを判定する条件文を生成する．
6. 上の条件を満たすデータを当該 PE のローカルメモリへ書き込むための転送コードを生成する．並列化次元以外の転送範囲は LOCAL 指示行より取得する（図 3 の例では配列 B の 2 次元目の転送範囲は [2: 32] である）．

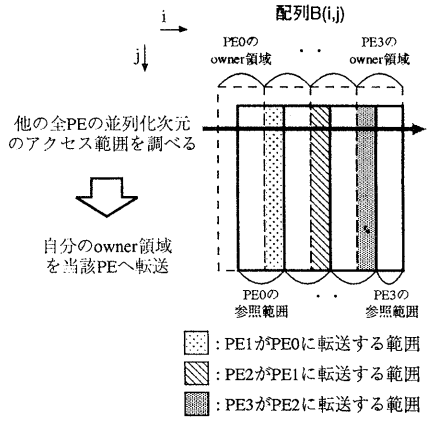


図 8 参照データのためのデータ転送生成
Fig. 8 Data transfer generation for read data.

配列Bのデータ転送コード

```
do tolid = 0 , number of PEs - 1
  do i = 1 , n-1
    if(A(i,*)はtolidのowner領域) then
      B(i+1,*)がowner領域ならばB(i+1,2:n)をtolidの当該領域に書き込む
    endif
  enddo
enddo
do i = 1 , n-1
  if(A(i,*)はowner領域) then
    B(i+1,*)がowner領域でなければownerPEからB(i+1,2:n)をread
    do j = 1 , n-1
      A(i,j) = B(i+1 , j+1) + C(i , j)
    enddo
  endif
enddo
```

lid : my PE number
tolid : target PE number

図 9 データ転送コードの生成
Fig. 9 Data transfer code generation.

図 9 は図 3 の例から上記のステップに基づいて生成したデータ転送コードであり，並列ループ中でのデータ転送をループの外へ移動し，さらに読み出しを書き込みへと変更している．

更新（MOD）

並列ループの直後にアクセスした領域のうち，owner

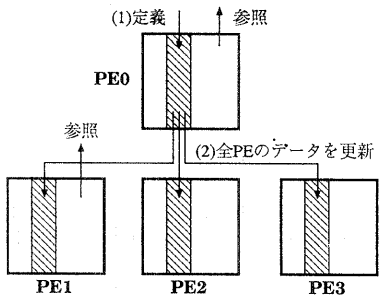


図 10 multiple copy データのデータ転送
Fig. 10 Data transfer for multiple copy data.

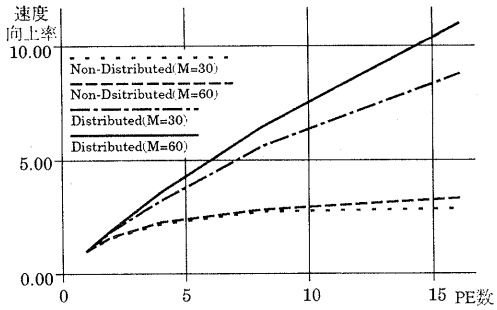


図 11 評価結果
Fig. 11 Performance measurement.

領域以外のデータを転送する（図7参照）．更新データに関しては，更新した直後にデータ転送コードを生成する方法と，並列ループ実行直後に更新したデータをブロック化して転送する方法の2つの方法をサポートしている．ブロック化するデータ転送コードの生成方法は，参照データに対するデータ転送コードと同様の方法で生成することができる．

参照/更新（BOT）

データ転送コードの生成は参照と更新の属性の場合と同様に行う．

3.5.2 multiple copy データ

multiple copy データは全 PE で一貫性を保証するため，データを更新した場合は，他のすべての PE のローカルメモリ上の当該領域も更新する必要がある．ただし，参照時はデータ転送の必要はない．したがって，データ転送の対象となる配列は MOD, BOT の属性を持つ配列となる．single copy データの場合と異なり，アクセスした範囲はすべてブロードキャストする（**図 10** 参照）．

4. 評価結果

SCG 法を用いた3次元の Poisson solver のプログラム（約 400 ライン）をサンプルとして，PCASE の評価を Cenju 2 プロトタイプ上（16 台構成）で行った．評価はすべての配列に対して分散指定を行った場合（Distributed）と，全く行わなかった場合（Non-Distributed）の両方について行った．分散配置指定はあらかじめ DISTRIBUTE 指示行をソースプログラム中に挿入しておき，これを PCASE の入力とした．分散配置指定を行わない場合は，Xpallas の GUI を介して不要データ転送を削除した．ただし，並列化に関してはいずれの場合もユーザ指示は一切行っていない．**図 11** に示すように分散配置を行った場合は 16 台で M=60 のときに約 11 倍，行わなかった場合は約 3.5 倍の速度向上率が得られている（いずれの場合も並列化の初期化時間を含む）．また，分散指定を行わなかった場合は 1 PE あたりの転送量が 1,829 K バイトであったのに対し，分散指定を行った場合は約 32 K バイトであった．これらの結果より，分散指定を行うことでさらにデータ転送が削除できることがわかる．

5. おわりに

以上，並列化支援環境 PCASE について，主に分散メモリマシンを対象とした機能を述べた．PCASE では階層メモリアーキテクチャをベースとした解析を行うことで，データを分散配置するプログラミング方法と分散配置しない方法の両方を効率よくサポートしており，ユーザに対して柔軟なプログラミング環境を提供している．また，LOCAL 指示行を IFP インタフェースに導入しているため，データ転送解析はすべて Xpallas で行うことができ，DCM では分散を行うデータに対してのみ，分散配置情報を用いたデータ転送の最適化を行うだけでデータ転送コードの生成が可能である．評価結果より，PCASE では必ずしもすべての配列に対して分割指定を行わなくとも，許容範囲の性能を得ることができるといえる．また，一般に分散メモリマシンにおいては，他のプロセッサのローカルメモリへのアクセスは，読み出しに比較して書き込みの方が高速であり，本稿で述べたデータ転送コードの生成方法は，Cenju 2 のみならず他のマシンにおいても有効である．今後は，数多くのアプリケーションに対して評価を行うと共に，IPF を現在標準化が進められている HPF (High Performance Fortran)[11] へ置き換え，PCASE を HPF へ対応させる予定である．また，HPF 記述支援を行うためのデータ分散解

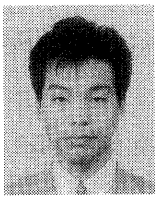
析機構の Xpallas への組み込み，分散配置される配列のメモリ管理の効率化，動的再配置のサポートなど DCM の機能強化を図る予定である．

謝辞 本研究の機会を与えていただき，また有益な示唆をいただいた日本電気(株)研究開発グループC&C研究所山本所長，同コンピュータ・システム研究部小池部長に深謝いたします．

参考文献

1) 蒲池，妹尾，松野：並列化支援環境 PCASE における分散メモリ対応機能，並列処理シンポジウム JSPP '93, pp. 31-38 (1993).
2) Kennedy, K., McKinley, K. S. and Tseng, C.-W.: Interactive Parallel Programming Using the ParaScope Editor, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 2, No. 3, pp. 329-341 (1991).
3) Hiranandani, S., Kennedy, K. and Tseng, C.-W.: Compiler Support for Machine Independent Parallel Programming in Fortran D, *Language, Compiler and Run-Time Environments for Distributed Memory Machines*, pp. 139-176, Elsevier Science Publishers B. V. (1992).
4) 磯留，Kolawa, A., Flower, J.：自動並列化システム ASPAR と並列処理実行環境 Express，信学技報，CPSY 91-31, pp. 213-220 (1991).
5) Applied Parallel Research: *FORGE* 90 *Distributed Memory Parallelizer User's Guide* (*Version 8.0*) (1992).
6) Fox, G., Hiranandani, S., Kennedy, K., Koelbel, C., Kremer, U., Tseng, C.-W. and Wu, M.-Y.: Fortran D Language Specification, Technical Report TR 90-141, Rice University (1990).
7) Chapman, B. M., Mehrotra, P. and Zima, H. P.: Vienna Fortran—A Fortran Language Extention for Distributed Memory Multiprocessors, *Language, Compiler and Run-Time Environments for Distributed Memory Machines*, pp. 39-62, Elsevier Science Publishers B. V. (1992).
8) Rühl, R. and Annaratone, M.: Parallelization of FORTRAN Code on Distributed-memory Parallel Processors, *Proceedings of the 1990 ACM International Conference on Supercomputing*, pp. 342-353 (1990).
9) 松下，山内，中田，小池：並列マシン Cenju 2 のアーキテクチャ，情報処理学会研究会報告，92-ARC-95, pp. 17-23 (1992).
10) 草野，妹尾，島村：手続き間定数伝播処理の検討，第 46 回情報処理学会全国大会論文集，5-67 (1993).
11) High Performance Fortran Forum: High Performance Fortran Language Specification, Version 1.0 (1993).

(平成5年9月24日受付)
(平成6年2月17日採録)



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年九州大学大学院総合理工学研究課情報システム学専攻修了．同年日本電気(株)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，並列処理アーキテクチャの研究に従事．

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．同年日本電気(株)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，特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持つ．現在 C&C 研究所主任．1988年本会論文賞受賞．