# Fortranマルチグレイン並列処理におけるデータローカライゼーション手法

吉田明正[†] 前田誠司[††]
尾形航[†] 笠原博徳[†]

本論文では，粗粒度並列処理（マクロデータフロー処理）・中粒度並列処理（ループ並列化）・近細粒度並列処理を階層的に組み合わせたマルチグレイン並列処理におけるデータローカライゼーション手法を提案する．本手法では，各粒度の並列性を最大限に利用し，さらに，各粒度で並列処理されるタスク間でのデータ授受を，集中共有メモリ経由ではなく，プロセッサ上のローカルメモリ経由で行うことにより，データ転送オーバヘッドの軽減をはかっている．本手法のコンパイル手順は，まず，ループ，サブルーチン等の粗粒度タスク間並列性を引き出し，かつ，多量のデータ転送を必要とする粗粒度タスク集合が実行時に同一のプロセッサクラスタ（PC）に割り当てられるように，粗粒度タスクの分割・融合を行う．次に，粗粒度タスク内部を中粒度タスクあるいは近細粒度タスクに分割し，それらのタスクをPC内プロセッサ(PE)に，PE間データ転送量が小さくなるように割り当てる．この後，粗粒度タスク（融合されたものを含む）に対して，PE上のローカルメモリを介したデータ授受コードを生成する．本手法はマルチプロセッサシステムOSCAR上でインプリメントされており，OSCARシミュレータ上で行った性能評価の結果から処理時間が21.5%短縮されることが確認された．

# A Data-Localization Scheme for Fortran Multi-Grain Parallel Processing

AKIMASA YOSHIDA,[†] SEIJI MAEDA,[††] WATARU OGATA [†]
and HIRONORI KASAHARA [†]

This paper proposes a data-localization scheme for multi-grain parallel processing, which exploits parallelism effectively by macro-dataflow computation, loop concurrentization and near fine grain parallelization. In this scheme, data are transferred via local memory among tasks which are assigned to the same processor cluster and data transfer overhead due to centralized common memory accesses is reduced. This compilation scheme is composed of three phases. Firstly, coarse grain tasks like loops in a Fortran program are restructured considering coarse grain parallelism and data locality. Secondly, the compiler decomposes coarse grain tasks into medium grain tasks like loop iterations and near fine grain tasks like statements, and schedules them to processors inside the processor cluster considering data transfer. Thirdly, it generates data transfer code via local memory for coarse grain tasks which are composed of medium grain tasks and near fine grain tasks. The proposed scheme has been implemented on OSCAR and the performance evaluation on an OSCAR simulater is also discussed.

## 1. はじめに

従来，マルチプロセッサシステム上におけるFortran並列化コンパイラを用いた並列処理では，Doall, Doacross等のループ並列化（中粒度並列処理）[13),17)]のみが自動的に行われていたが，このような並列化手法では，複雑なループキャリィドディペンデンス*を持つループ，あるいはループ以外の部分の並列性を利用することができないという問題があった．この問題点を解決するために最近では，ループやサブルーチン等の粗粒度タスク間の並列性を利用する粗粒度並列処理（マクロデータフロー処理）[6)~8)]，あるいはステートメント程度の近細粒度タスク間の並列性を利用する近細粒度並列処理[9)]，さらにそれらを従来の中粒度並列処理（ループ並列化）と階層的に組み合わせたマルチグレイン並列処理手法[12)]が提案されている．

† 早稲田大学理工学部
School of Science and Engineering, Waseda University
†† (株) 東芝 研究開発センター
Research & Development Center, Toshiba Corporation

* イタレーション間にまたがるデータ依存．

このマルチグレイン並列処理では，粗粒度タスク（マクロタスク）をダイナミックスケジューリングによってプロセッサクラスタ（PC）に実行時に割り当てて複数PC上で粗粒度並列処理を行い，各PCに割り当てられたマクロタスクに対してはPC内部のプロセッサ（PE）上で階層的に中粒度並列処理あるいは近細粒度並列処理を行うというものである．このようにマクロタスクを処理すべきPCを実行時に決定する方式では，コンパイル時にデータを各PE上のローカルメモリ（LM）に効果的に分割・配置することが困難である．このため，従来のマルチグレイン並列処理では，粗粒度タスク間で共有されるデータを集中共有メモリ（CM）上に配置し，粗粒度タスク間のデータ授受を集中共有メモリを介して行わざるを得なかった．しかし，このような方式では，集中共有メモリを介したデータ転送オーバヘッドが大きくなってしまうという問題が生じる．このような問題点を解決するためには，データ分割・配置を適切に行い各プロセッサ上のローカルメモリを介したデータ授受を可能とする手法の開発が重要となる．

このようなローカルメモリへのデータの分割・配置に関する研究は現在活発に行われており，TuとPadua[1]，Li[2]は，作業配列変数をローカルメモリにとることにより単一ループの処理の高速化を行うためのArray Privatization法を提案している．しかし，この方法は，Doallループの各イタレーション内部だけでしかローカルメモリを利用しておらず，マルチグレイン並列処理のような方式で同一PC（複数PEで構成）上で実行される複数ループ間でのローカルメモリ経由データ授受には利用することはできない．また，LiとChen[3]，GuptaとBanerjee[4]，AndersonとLam[5]は，分散メモリマシン上での自動データ分割・配置法を提案している．しかし，これらの自動データ分割・配置法では，コンパイラがスタティックにデータ割り付けを行える場合にしか適用できず，中粒度並列性（ループのイタレーション間並列性）しか利用できないという制約がある．一方，ダイナミックスケジューリングを用いた粗粒度並列処理においては，複数の粗粒度タスク間での共有データをローカルメモリ経由で授受する方法[10]が提案されている．しかし，この方法では，粗粒度並列処理のみが適用対象であり，粗粒度タスク内部が階層的に並列処理（中粒度並列処理または近細粒度並列処理）される場合は考慮されていない．

そこで，本論文では，粗粒度・中粒度・近細粒度並列処理を階層的に組み合わせたマルチグレイン並列処理において，粗粒度タスク（内部はPC内PE上で階層的に中粒度・近細粒度並列処理される）間，および，シーケンシャルループのイタレーション（内部はPC内PE上で階層的に中粒度・近細粒度並列処理される）間で，PC内の各PE上のローカルメモリを介したデータ授受を行い，また，粗粒度タスク内部の近細粒度タスク間ではPE上のレジスタまたはローカルメモリを介したデータ授受を行うことにより，集中共有メモリアクセスを軽減するデータローカライゼーション手法を提案する．

本論文2章では，マルチグレイン並列処理について概説する．3章では，データローカライゼーションを行うためのタスクの再構成とスケジューリングについて述べる．4章では，データローカライゼーション用データ転送コードの生成方法について述べる．5章では，本手法をインプリメントしたコンパイラを用いて，マルチプロセッサシステムOSCAR[14]のシミュレータ上で行った性能評価について述べる．

## 2. マルチグレイン並列処理

マルチグレイン並列化コンパイレーションは，粗粒度並列処理[6)~8)]，中粒度並列処理，近細粒度並列処理[9]の3つの技術から構成される．本章では，これらのコンパイレーション手法と，対象マルチプロセッサアーキテクチャについて述べる．

### 2.1 対象マルチプロセッサアーキテクチャ

マルチグレイン並列処理では，5.1節で述べるOSCAR[14]のように，プロセッサ上にローカルメモリ（他PEからリード・ライト可能な分散共有メモリが望ましい）を持ち，各プロセッサがインターコネクションネットワークを介して集中共有メモリに接続されているマルチプロセッサシステムを対象とする．その際，インターコネクションネットワークに関しては，ソフトウェア的にプロセッサのクラスタリング（グループ化）を容易に実現できるバス結合やクロスバ結合が望ましい．

### 2.2 粗粒度並列処理（マクロデータフロー処理）

マクロデータフローコンパイレーション手法では，まず，Fortranプログラムを，擬似代入文ブロック（BPA），繰り返しブロック（RB），サブルーチンブロック（SB）の3種類の粗粒度タスク（マクロタスク（MT））に分割する[7]．

BPAは本質的には基本ブロックであるが，並列性抽出のために単一の基本ブロックを複数に分割し複数のBPAを生成したり，逆に単一の基本ブロックの処理時間が1回のダイナミックスケジューリングに要する時間に比べ短い場合には，複数の基本ブロックを融合し

1つのBPAを生成する[7]．RBは最外側ナチュラルループ[19]である．また，コード長を考慮し効果的にインライン展開できないサブルーチンブロックはSBとして定義する．

ただし，上記のRBの定義では，Doallループが1つのRBとして1つのプロセッサクラスタ（PC）に割り当てられてしまうため，PC内のPE数分の並列性しか利用できない．この問題を解決するために，Doallループは，イタレーション範囲を分割することにより，複数の部分Doallループに分割する[10]．また，データローカライゼーションを適用するマクロタスク集合は，実行時に同一PCに割り当てるために融合を行う[7),10)]．

以上のようにマクロタスクを生成した後，コンパイラは，BPA, RB, SB等のマクロタスク間のコントロールフロー，データフローを解析し，それらを表したマクロフローグラフ（MFG）を生成する．さらに，コントロール依存とデータ依存を考慮したマクロタスク間並列性を最大限に引き出すために，各マクロタスクの最早実行可能条件[6]を解析する．この各マクロタスクの最早実行可能条件は，マクロタスクグラフ（MTG）で表すことができる．

次に，コンパイラは，マクロタスク間の条件分岐等の実行時不確定性に対処するために，マクロタスクを実行時にプロセッサクラスタ（PC）に割り当てるダイナミックスケジューリングルーチンを生成する．

マクロデータフロー処理では，このようなダイナミックスケジューリングルーチンを用いて，マクロタスク（融合により生成された融合マクロタスクを含む）を，実行時にPCに割り当てる方式をとっているため，マクロタスク（融合マクロタスクを含む）間で共有されるデータあるいは初期データは，集中共有メモリに配置する．また，PCに割り当てられたマクロタスクは，さらにマクロタスク内部で，中粒度タスク，近細粒度タスク，または階層的にサブマクロタスクに分割され，PC内部のPE（あるいはサブPC）上で並列処理される．ここで，RBやSB内部でのマクロデータフロー処理を階層型マクロデータフロー処理[11]と呼ぶ．

### 2.3 中粒度並列処理（ループ並列化）

前述のように，PCに割り当てられたマクロタスクがDoall可能RBである場合，そのRBはPC内部のPEによって中粒度すなわちイタレーションレベルで並列処理される．Doallループのイタレーションの PEへのスケジューリングに関しては，セルフスケジューリング，チャンクスケジューリング，ガイデッドセルフスケジューリング等といったいくつかのダイナミックスケジューリング手法[13),16)]が提案されている．しかし，本手法ではデータのローカリティを高めるため，スタティックスケジューリングを用いて，ほぼ同数のイタレーションをPC内の各PEに割り当てる方式をとっている．

### 2.4 近細粒度並列処理

ループ並列化不可のRBのボディ部に存在する基本ブロック，あるいは，ループ外部のBPAでは，PC内のPEを用いて近細粒度並列処理[9]が行われる．

近細粒度並列処理を効率よく行うためには，並列性が十分得られ，なおかつデータ転送や同期によるオーバヘッドができるだけ小さくなるように，BPA（基本ブロック）を近細粒度タスクに分割することが必要である．本手法では，性能評価に使用するOSCARの処理能力やデータ転送能力を考慮して，近細粒度タスクとしてステートメントレベルの粒度を用いている．

このような近細粒度タスク集合のPC内PEへの割り当てには，データ転送時間を考慮したヒューリスティックアルゴリズムであるCP/DT/MISF法[13),15)]あるいはDT/CP法[18)]を採用している．

## 3. データローカライゼーションのためのタスクの再構成とスケジューリング

本章では，マルチグレイン並列処理におけるデータローカライゼーション手法を提案する．マルチグレイン並列処理において，各粒度のタスク間でローカルメモリを介してデータ転送を行うためには，タスクの再構成とスケジューリングの技術が重要となる．以下の節ではこれらの技術について述べる．

### 3.1 粗粒度タスクの再構成とPCへのスケジューリング

マルチグレイン並列処理においてデータローカライゼーションを効率的に行うためには，(1)粗粒度並列性が十分に得られ，かつ，多量のデータ転送を必要とするマクロタスク間で配列データの定義・参照範囲が等しくなるように，マクロタスクを生成し，(2)多量のデータ転送を必要とするマクロタスク集合を同一プロセッサクラスタ（PC）にダイナミックスケジューリングして実行することが必要となる．そこで，本手法では，RB整合分割法[10]とマクロタスク融合法[7),10)]を用いて，マクロタスクを再構成する．

#### 3.1.1 RB整合分割

本手法では，図1(a)のようなデータ依存のある複数RB（ループ）をイタレーション範囲で分割する際に，分割された部分RB間(例えば，図1(c)の$RB_{1a}$と$RB_{2a}$の組，または，$RB_{1c}$と$RB_{2b}$の組）でローカルメモリ経由データ授受を可能とするため，RB整合

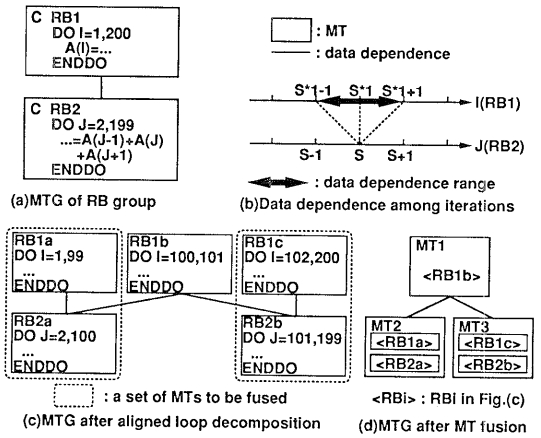



図1 RB整合分割例
Fig. 1 An example of aligned loop decomposition.

分割[10]を行う．

RB整合分割法では，まず，RB整合分割の対象とするRBグループを生成する．RBグループは，マクロタスクグラフ（MTG）上で，お互いが唯一の直接後続マクロタスクおよび唯一の直接先行マクロタスクであるような配列変数に関するデータ依存が存在するDoall可能RB集合（例えば，図1(a)の$RB_1$と$RB_2$）である．ただし，RBグループを生成する時に限り，RBで使用される変数の初期化等を行う小規模BPAからのデータ依存エッジは無視する．これは，小規模BPAの処理時間は小さいので，小規模BPAと他のマクロタスクの間の並列性は利用しても効果的でないからである．

次に，各RBグループ（$RB_i(1\leq i\leq N_{RB})$により構成されるものとする）内において，あるRBのイタレーションと他のRBのイタレーションとの間のデータ依存(インターRBイタレーション間データ依存と呼ぶ)を解析する．このインターRBイタレーション間データ依存の解析結果は，$RB_{N_{RB}}$のS番目のイタレーションが，直接的あるいは間接的（他RBを経由して）にデータ依存している各$RB_i(1\leq i\leq N_{RB}-1)$のイタレーション範囲（上・下限値はSの1次式で表現）で表す．例えば，マクロタスクグラフ(MTG)中に図1(a)のRBグループ（$RB_1$と$RB_2$で構成，$N_{RB}=2$）が存在する場合，インターRBイタレーション間データ依存を解析すると，$RB_2$の$J=S$のイタレーションが，$RB_1$の$I=S\times1-1$から$I=S\times1+1$の範囲のイタレーション集合にデータ依存していることがわかる．この結果を図示すると図1(b)のようになる．

この後，以下のように各$RB_i(1\leq i\leq N_{RB})$を分割する．なお，ここでは，図1(a)，(c)を用いて説明する．まず，図1(a)の$RB_2(RB_{N_{RB}})$を，図1(c)の$RB_{2a}, RB_{2b}$に分割する．ただし，分割数は通常PC数（図1(c)では2）あるいはPC数の整数倍とする．次に，図1(a)の$RB_1$（各$RB_i(1\leq i\leq N_{RB}-1)$）の分割では，インターRBイタレーション間データ依存解析結果（図1(b)）を用いて，$RB_{2a}$のイタレーション集合だけがデータ依存している$RB_1$のイタレーション集合を求め，これらのイタレーション集合を$RB_{1a}$として生成する．同様に，$RB_{2b}$だけがデータ依存している$RB_{1c}$を生成する．また，$RB_1$には，$RB_{2b}$と$RB_{2b}$が共通にデータ依存しているイタレーション集合が存在するため，それらのイタレーション集合を新たに$RB_{1b}$として生成する．このような分割を行うことにより，多量のデータ転送を必要とする部分RB集合（$RB_{1a}$と$RB_{2a}$の組，または，$RB_{1c}$と$RB_{2b}$の組）は，3.1.2項で述べるように，融合することが可能である．

上述のRB整合分割の適用されなかったDoall可能RBは，イタレーション数の等しい$m$個（$m$は通常PC数あるいはPC数の整数倍）の部分RBに分割される．

### 3.1.2 マクロタスク融合

本手法では，多量のデータ転送を必要とするマクロタスク（RB整合分割されたものを含む）集合をダイナミックスケジューリング環境下で同一のPCに割り当てることを保証するため，これらのマクロタスク集合をコンパイル時に融合し，実行時には融合されたマクロタスク集合を，ダイナミックスケジューリングにより1PCに割り当てる方式をとる．

本マクロタスク融合では，まず，RB整合分割の対象となった旧RBグループ内で，多量のデータを必要とする部分RB（分割されたRB）集合を融合する[10]．例えば，RB整合分割された図1(c)の場合，点線で囲まれた部分RB集合（$RB_{1a}$と$RB_{2a}$の組，または，$RB_{1c}$と$RB_{2b}$の組）は融合され，融合後のマクロタスクグラフ（MTG）は図1(d)のようになる．

次に，MTG全体に対し，データ転送とダイナミックスケジューリングオーバヘッドを考慮したヒューリスティックマクロタスク融合法[7]を適用する．

なお，融合により生成されたマクロタスク内部のPC内PE上での処理については，3.2節および3.3節で述べる．

## 3.2 中粒度タスクのPC内PEへのスケジューリング

マルチグレイン並列処理では，前述のように，プロセッサクラスタ（PC）に割り当てられたマクロタスク

(RB整合分割・マクロタスク融合適用後のマクロタスク)内部に，Doallループが存在する場合，PC内部のPE上で中粒度並列処理（ループ並列化）が行われる．

Doallループの中粒度タスク（イタレーション）は，イタレーション間の並列性を有効に利用するために，PC内の各PEに均等にスケジューリングされる．

なお，RB整合分割後に融合された融合マクロタスクの場合，融合マクロタスク内のループ間では配列データの定義・参照範囲がほとんど等しい．このため，それらのループを中粒度並列処理する場合には，各ループごとにイタレーション集合を均等にPEにスケジューリングしても，大部分のデータがPE上のローカルメモリを介して授受される．

### 3.3 近細粒度タスクのPC内PEへのスケジューリング

本節では，PC内のPEを用いて近細粒度並列処理が行われるBPA(シーケンシャルループ内部の基本ブロックあるいはループ外のBPA)において，効率的にデータローカライゼーションを行うための近細粒度タスク融合法と近細粒度タスクのスケジューリングについて述べる．

近細粒度タスク間で，PE上のレジスタまたはローカルメモリを介してデータ授受を行うには，データ転送の必要となる近細粒度タスク集合を同一のPEに割り当てなければならない．

そこで，本手法では，近細粒度タスクをヒューリスティック融合法[15]により融合した後，データ転送を考慮したスタティックスケジューリング法によって，PE間データ転送量が小さくなるように，近細粒度タスクをPEにスケジューリングする方式をとる．

なお，近細粒度タスクのスケジューリングアルゴリズムとしては，使用するマルチプロセッサシステムの各PE上での近細粒度タスクの処理時間とPE間データ転送時間の割合に応じて，CP/DT/MISF法[13),15)]あるいはDT/CP法[18)]を適用する．

## 4. データローカライゼーションのためのデータ転送コード生成

本章では，3章で述べた粗粒度タスクの再構成および中粒度タスクと近細粒度タスクのPC内PEへのスケジューリングが行われたマクロタスク（融合マクロタスク)に対して，PC内の各PE上のローカルメモリ(LM)経由でデータ授受を行う並列マシンコードを生成する．

具体的には，まず，融合マクロタスク内部で，各配列変数$i$ごとに，データローカライゼーション適用時のデータ転送時間$t^i_{localize}$と，集中共有メモリ(CM)経由データ授受を行う場合のデータ転送時間$t^i_{CM}$を求める．なお，$t^i_{localize}$は，配列変数$i$に関する，CMとLM間のデータ転送時間，LMへのロード・ストア時間，分散共有メモリ（DSM）を用いたPC内PE間の直接データ転送時間の合計である．$t^i_{CM}$は，配列変数$i$に関するCMへのロード・ストア時間である．

次に，$t^i_{localize}$と$t^i_{CM}$を比較し，$t^i_{localize} < t^i_{CM}$を満たす配列変数$i$に対しては，データローカライゼーション用コード，すなわち，後述する各種データ転送コードと各PE上のローカルメモリへのロード・ストア命令を生成する．一方，$t^i_{localize} < t^i_{CM}$を満たさない配列変数$i$に対しては，従来のCM経由データ授受コードを生成する．

### 4.1 融合MT内で参照前に定義されないデータのCMからLMへの転送

データローカライゼーションが適用される配列変数への定義・参照を行うステートメントは，そのステートメントを実行するPE上のローカルメモリのデータをアクセスするようにコード生成される．このため，データローカライゼーションの適用される配列変数への参照が発生する前に，参照する範囲のデータはローカルメモリ上に置かれていなければならない．そこで，融合マクロタスク内において，参照前に定義されない範囲のデータに対しては，融合マクロタスク実行開始時に集中共有メモリ（CM）から当該PE上のローカルメモリ（LM）に転送するコードを挿入する．

例えば，図2(a)のようなRB整合分割・マクロタスク融合により生成された融合マクロタスク（図1(d)の$MT_2$と同じ）の場合，融合マクロタスク内部の各Doallループのイタレーション集合は，図2(b)に示すようにPC内の各PE（図中のPCは2PE構成）に

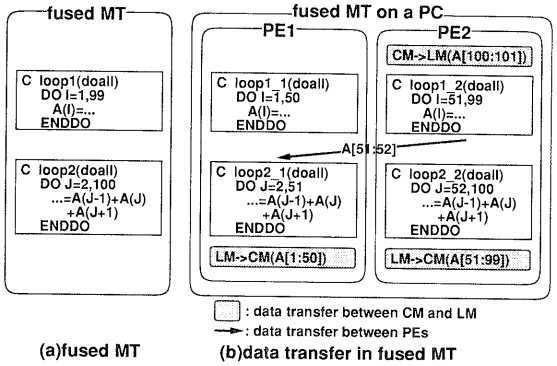



図2 複数ループ面でのデータ転送の例
Fig. 2 An example of data transfer between loops.

均等に割り当てられる．

ここで，配列変数Aにデータローカライゼーションが適用される場合，図2(b)のPE1では，ループ2_1で参照されるデータA[1：52]（A(1)からA(52)の配列要素を意味する）のうち，A[1：50]はループ1_1からLMを介して授受され，A[51：52]は4.2節で述べるようにPE2のループ1_2で定義後にPE1のLMに転送される．このため，集中共有メモリ（CM）からPE1のLMにデータ転送する必要はない．

一方，図2(b)のPE2では，ループ2_2で参照されるデータA[51：101]のうち，A[51：99]はループ1_2からLMを介して授受され，それ以外のデータA[100：101]はループ1_1，ループ1_2のいずれでも定義されていない．そこで，図2(b)のPE2の上部の網掛け長方形内に示すように，CM上のデータA[100：101]（他のマクロタスクにより定義されたデータ）をPE2のLMに転送するコードを挿入する．

### 4.2 複数ループ間でのPE間データ転送

同一PCに割り当てられる融合マクロタスク内の各ループは，PC内部のPE上で中粒度あるいは近細粒度タスクレベルで並列処理されるため，ループ間では必ずしも同一PE上のローカルメモリを介してデータ授受を行えるとは限らない．そこで，そのような場合には，ループ間にPE間データ転送命令を挿入する．

例えば，前述の図2(a)はRB整合分割・マクロタスク融合により生成された融合マクロタスクなので，図2(b)のようにPC内の各PEにおいて，割り当てられた各Doallループのイタレーション集合によって定義または参照される配列データ範囲はほぼ等しい．しかし，PE2上のループ1_2で定義されるデータの一部（A[51：52]）は，PE2上のループ2_2だけでなくPE1上のループ2_1においても参照されるため，図2(b)中の矢印で示されるようにPE間データ転送を行う命令を挿入する．

### 4.3 シーケンシャルループのイタレーション間でのPE間データ転送

本手法では，融合されたマクロタスク間だけに限らず，**図3**(a)のような複数のループをボディとして持つシーケンシャルループに対しても，データローカライゼーションを適用する．この場合，内部ループ1がDoallループであるとすると，ループ1の各イタレーションは，図3(b)に示すように，PC内の各PE（図中のPCは2PE構成）に均等に割り当てられて中粒度並列処理が行われる．一方，内部ループ2はシーケンシャルループであるため，図3(b)に示すように，ループ2のボディ部（基本ブロック）の'$A(J)=A(J-1)$+…'のステートメントがPE1に，'$B(J)=B(J-1)$+…'のステートメントがPE2に割り当てられて近細粒度並列処理が行われる．

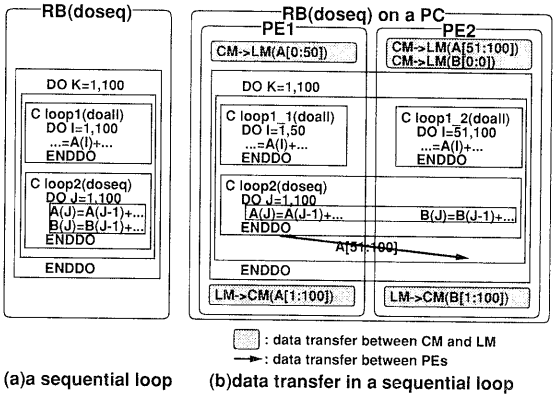



図3 シーケンシャルループのイタレーション間でのデータ転送の例

Fig. 3 An example of data transfer between iterations in a sequential loop.

ここで，配列変数A，Bにデータローカライゼーションが適用される場合，外側ループのあるイタレーション中のループ2でPE1上のローカルメモリ（LM）に定義されたデータA[1：100]が，その後続イタレーション中のループ1_1（PE1上）およびループ1_2（PE2上）で参照される．この場合，PE1上のループ1_1で参照されるデータA[1：50]に関しては，PE1上のLMを介して授受される．一方，PE2上のループ1_2で参照されるデータA[51：100]に関しては，図3(b)中の矢印で示すようなPE1からPE2へのデータ転送が必要となり，コンパイラはこの転送のための命令を挿入する．

### 4.4 融合MTの後続MTで参照されるデータのLMからCMへの転送

データローカライゼーションの適用により，融合マクロタスク内でPC内PE上のローカルメモリにストアされた定義データ(例えば，図2(b)において，PE1上のA[1：50]，PE2上のA[51：99]）が，融合マクロタスク外部の後続マクロタスクで参照される場合には，PE上のローカルメモリ（LM）のデータを集中共有メモリ(CM)に転送するコードを挿入する．例えば，図2(b)の融合マクロタスクの場合，図2(b)のPC内各PEの下部の網掛け長方形内に示されるように，PE上のLMからCMへのデータ転送コードを挿入する．

本手法では，このように，融合マクロタスク内でLM上に定義された有効なデータを，融合マクロタスク実

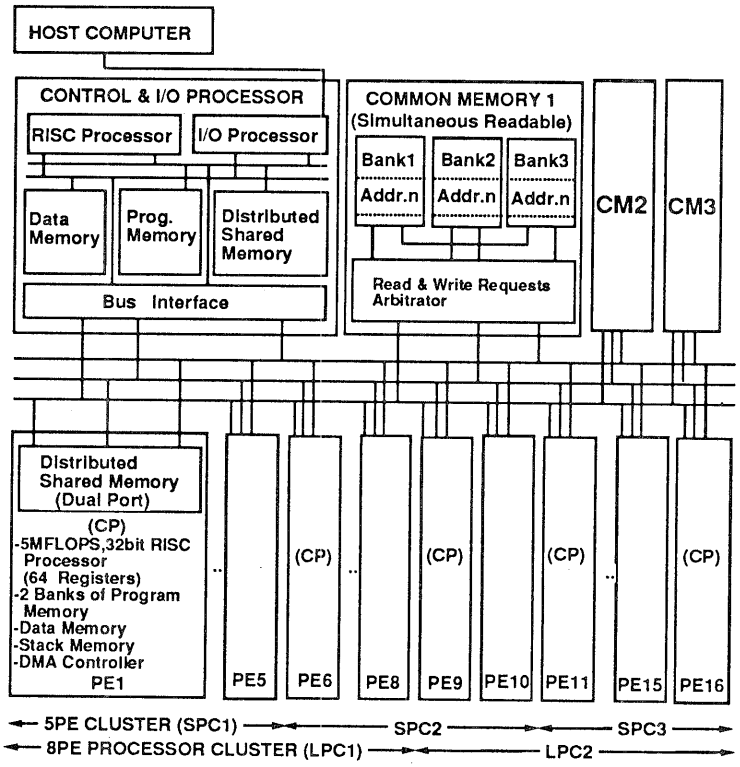



図4 OSCARのアーキテクチャ
Fig. 4 OSCAR's architecture.

行終了時にCM上に書き戻しているため，融合マクロタスク外部の後続マクロタスクは，どのPCに割り当てられても，CM上の有効なデータを参照することができる．

## 5. OSCAR上での性能評価

本章では，提案するマルチグレイン並列処理におけるデータローカライゼーション手法を，OSCAR[14]シミュレータ上で性能評価した結果について述べる．

### 5.1 OSCARのアーキテクチャ

性能評価に用いるOSCAR[14]（図4）は，ローカルメモリ（LM）を持つ32ビットRISCプロセッサ（PE）を，集中共有メモリ（CM）に3本のバスで接続した共有メモリ型マルチプロセッサシステムである．ローカルメモリ（LM）は，ローカルプログラムメモリ（LPM），ローカルデータメモリ（LDM），他PEからリード・ライト可能な分散共有メモリ（DSM）の領域からなる．なお，OSCARでは，PE上のLDMおよびDSMへのロードやストアに1クロックを要し，CMおよび他PEのDSMへのロードやストアには4クロックを要する．

また，OSCARは各PEを集中共有メモリ（CM）に平等結合したアーキテクチャとなっているが，複数のPEをソフトウェア的にクラスタリング（グループ化）することにより，3プロセッサクラスタ（PC）までのマルチプロセッサクラスタシステムとして利用することができる．その際，各プロセッサクラスタ（PC）では，PC内部のPEあるいはPC内で定義されるサブPCを用いて，各マクロタスクを中粒度レベル，近細粒度レベル，または粗粒度レベル（サブマクロタスクレベル）で階層的に並列処理することができる．ただし，PCのクラスタリングはソフトウェアで行われているため，各PCはPC内の集中共有メモリを持っていない．

### 5.2 CGS法プログラムにおける性能評価

本節では，非対称係数行列を持つ連立1次方程式の求解用CGS法プログラムの主収束ループを用いて提案手法の性能評価を行う．このループは，図5のマクロタスクグラフ（MTG）に示すように，15個のマクロタスクで構成されている．

このプログラムをOSCARシミュレータ上で1PEを用いて実行するとその処理時間は107.2[ms]であ

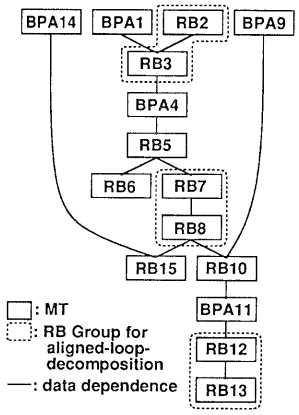



図5 CGS法プログラムのMTG
Fig. 5 MTG of a program for CGS method.

った．次に，2 PC（各PCは1 PEで構成，計2 PE）上で，CM経由データ転送を用いたマクロデータフロー処理を行うと処理時間は54.4[ms]に短縮された．また，2 PC（各PCは3 PEで構成，計6 PE）上で，CM経由データ転送のマルチグレイン並列処理を用いると，マクロデータフロー処理の粗粒度並列性に加えて，中粒度並列性と近細粒度並列性が引き出され，処理時間は22.3[ms]に短縮された．

さらに，提案するデータローカライゼーション手法を伴うマルチグレイン並列処理では，図5のMTG中の点線で囲まれた各RBグループに対してRB整合分割とマクロタスク融合が行われ，2 PC（各PCは3 PEで構成，計6 PE）上での処理時間は17.5[ms]に短縮された．この場合，CM経由データ転送のマルチグレイン並列処理に比べて21.5%処理時間が短縮されている．これにより，提案するデータローカライゼーション手法は，CMアクセスに伴うデータ転送オーバヘッドを顕著に軽減することが確かめられた．

## 6. おわりに

本論文では，マルチグレイン並列処理において，各粒度の並列性を最大限に利用しつつ，粗粒度タスク（内部は階層的に並列処理される）間，および，シーケンシャルループのイタレーション（内部は階層的に並列処理される）間では，プロセッサクラスタ（PC）内の各PE上のローカルメモリを介してデータ授受を行い，また，粗粒度タスク内部の近細粒度タスク間ではPE上のレジスタまたはローカルメモリを介してデータ授受を行うことにより，データ転送オーバヘッドを軽減するデータローカライゼーション手法を提案した．

また，OSCARシミュレータ上での性能評価の結果より，提案手法は従来の集中共有メモリのみを介してデータ授受を行うマルチグレイン並列処理に比べ，処理時間を顕著に短縮できることが確かめられた．

今後の課題としては，他のループとの間のデータローカライゼーション効果を高めるための，シーケンシャルループ内部の近細粒度タスクスケジューリング手法の開発，および，データ転送オーバヘッドのさらなる軽減を目指したデータローカライゼーションとデータプレローディング・ポストストアリング技術との統合等があげられる．

**謝辞** 本研究の一部は，文部省科学研究費（特別研究員奨励費053760，一般研究(b)05452354，一般研究(c)05680284）による．

## 参考文献

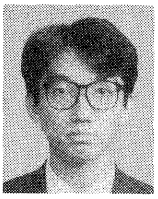
1) Tu, P. and Padua, D.: Automatic Array Privatization, *6th Annual Workshop on Languages and Compilers for Parallel Computing* (1993).
2) Li, Z.: Array Privatization for Parallel Execution of Loops, *Proc. of the 1992 ACM Int'l Conf. on Supercomputing*, pp. 313-322 (1992).
3) Li, J. and Chen, M.: Compiling Communication-Efficient Programs for Massively Parallel Machines, *IEEE Trans. on Parallel and Distributed System*, Vol. 2, No. 3, pp. 361-376 (1991).
4) Gupta, M. and Banerjee, P.: Demonstration of Automatic Data Partitioning Techniques for Parallelizing Compilers on Multicomputers, *IEEE Trans. Parallel and Distributed System*, Vol. 3, No. 2, pp. 179-193 (1992).
5) Anderson, J. M. and Lam, M. S.: Global Optimizations for Parallelism and Locality on Scalable Parallel Machines, *Proc. of the SIGPLAN '93 Conf. Programming Language Design and Implementation*, pp. 112-125 (1993).
6) 本多，岩田，笠原：Fortranプログラム粗粒度タスク間の並列性検出手法，信学論（D-I），Vol. J 73-D-I, No. 12, pp. 951-960 (1990).
7) 笠原，合田，吉田，岡本，本多：Fortranマクロデータフロー処理のマクロタスク生成手法，信学論（D-I），Vol. J 75-D-I, No. 8, pp. 511-525 (1992).
8) 本多，合田，岡本，笠原：Fortranプログラム粗粒度タスクのOSCARにおける並列実行方式，信学論（D-I），Vol. J 75-D-I, No. 8, pp. 526-535 (1992).
9) 本多，水野，笠原，成田：OSCAR上でのFortranプログラム基本ブロックの並列処理手法，信学論（D-I），Vol. J 73-D-I, No. 9, pp. 756-766 (1990).
10) 吉田，前田，尾形，笠原：Fortranマクロデータフロー処理におけるデータローカライゼーション手

法, 情報処理学会論文誌, Vol. 35, No. 9, pp. 1848-1860(1994).
11) 岡本, 合田, 宮沢, 本多, 笠原：OSCARマルチグレインコンパイラにおける階層型マクロデータフロー処理手法, 情報処理学会論文誌, Vol. 35, No. 4, pp. 513-521(1994).
12) Kasahara, H., Honda, H., Mogi, A., Ogura, A., Fujiwara, K. and Narita, S.: Multi-Grain Parallelizing Compilation Scheme for OSCAR, *4th Workshop on Language and Compilers for Parallel Computing* (1991).
13) 笠原：並列処理技術, コロナ社 (1991).
14) 笠原, 成田, 橋本：OSCARのアーキテクチャ, 信学論 (D), Vol. J 71-D, No. 8, pp. 1440-1445 (1988).
15) Kasahara, H., Honda, H. and Narita, S.: Parallel Processing of Near Fine Grain Tasks Using Static Scheduling on OSCAR, *IEEE ACM Supercomputing'90*, pp. 856-864(1990).
16) Polychronopoulos, C. D.: *Parallel Programming and Compilers*, Kluwer Academic Pub. (1988).
17) Padua, D. A. and Wolfe, M. J.: Advanced Compiler Optimizations for Super Computers, *C. ACM*, Vol. 29, No. 12, pp. 1184-1201(1986).
18) 藤原, 白鳥, 鈴木, 笠原：データプレロードおよびポストストアを考慮したマルチプロセッサスケジューリング, 信学論(D-I), Vol. J 75-D-I, No. 8, pp. 495-503(1992).
19) Aho, A. V., Sethi, R. and Ullman, J. D.: *Compilers (Principles, Techniques, and Tools)*, Addison Wesley (1988).

(平成6年9月16日受付)
(平成7年5月12日採録)

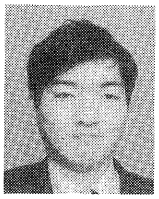
**吉田 明正**（正会員）

1968年生. 1991年早稲田大学理工学部電気工学科卒業. 1993年同大大学院修士課程修了. 現在, 同大大学院博士後期課程在学中. 1993年～1995年日本学術振興会特別研究員. 1995年より早稲田大学理工学部電気工学科助手. 並列化コンパイラ, 並列処理方式, マルチプロセッサアーキテクチャ等の研究に従事. 電子情報通信学会会員.



**前田 誠司**（正会員）

1969年生. 1992年早稲田大学理工学部電気工学科卒業. 1994年同大大学院修士課程修了. 同年（株）東芝入社. 在学中, 並列化コンパイラの研究に従事. 現在, 同社研究開発センターに所属.



**尾形 航**（正会員）

1967年生. 1991年早稲田大学理工学部電気工学科卒業. 1993年同大大学院修士課程修了. 現在, 同大大学院博士後期課程在学中. 並列実行方式, 近細粒度並列処理手法, 計算機アーキテクチャの研究に従事.

**笠原 博徳**（正会員）

1957年生. 1980年早稲田大学理工学部電気工学科卒業. 1985年同大大学院博士課程修了. 工学博士. 1983年～1985年早稲田大学電気工学科助手. 1985年カリフォルニア大バークレー短期客員研究員, 日本学術振興会特別研究員. 1986年早稲田大学電気工学科専任講師, 1988年同助教授, 現在に至る. 1989年～1990年イリノイ大Center for Supercomputing R&D客員研究員. 1987年IFAC World Congress第1回Young Author Prize受賞. 主な著書「並列処理技術」(コロナ社). マルチプロセッサスケジューリング, スーパーコンピューティング, 並列化コンパイラ等の研究に従事. 電子情報通信学会, 電気学会, シミュレーション学会, ロボット学会, IEEE, ACM等の会員.