階層型粗粒度並列処理における同一階層内ループ間データローカライゼーション手法

吉田明正[†1] 越塚健一[†2]
岡本雅巳[†3] 笠原博徳[†4]

本論文では，階層的に粗粒度並列処理を行う階層型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を提案する．本手法では，階層型ループ整合分割法を用いて各階層の処理とデータを分割し，パーシャルスタティックタスク割当てを用いた階層型ダイナミックスケジューリング方式により，各階層において相互間に多量のデータ転送が生じる可能性がある粗粒度タスクの集合を当該階層の同一プロセッサクラスタに割り当て，さらに各プロセッサ上のローカルメモリを有効利用してデータ転送オーバヘッドを軽減する．マルチプロセッサシステムOSCAR上で行った性能評価の結果，本データローカライゼーション手法を用いた階層型マクロデータフロー処理では，データローカライゼーションを用いない場合に比べて処理時間が10～20%短縮されることが確かめられた．

A Data-localization Scheme among Loops for Each Layer in Hierarchical Coarse Grain Parallel Processing

AKIMASA YOSHIDA,[†1] KEN'ICHI KOSHIZUKA,[†2] MASAMI OKAMOTO[†3] and HIRONORI KASAHARA[†4]

This paper proposes a data-localization scheme for hierarchical macro-dataflow processing, which hierarchically exploits coarse-grain parallelism. The proposed data-localization scheme consists of three parts: (1) hierarchical loop aligned decomposition, which decomposes multiple loops having data dependences into data-localization-groups in each layer; (2) generation of hierarchical dynamic scheduling routine with partial static task assignment, which assigns macrotasks inside data-localization-group to the same processor-cluster in each layer; (3) generation of data transfer code via local memory inside data-localization-group. Performance evaluation on a multiprocessor system OSCAR shows that hierarchical macro-dataflow processing with data-localization can reduce execution time by 10–20% compared with hierarchical macro-dataflow processing without data-localization.

1. はじめに

マルチプロセッサシステム上での自動並列化コンパイラを用いた並列処理では，従来よりループ並列化手法[1),2)]が広く用いられている．最近の並列化コンパイラ，たとえば，イリノイ大学のPolaris[3)]やスタンフォード大学のSUIF[4)]等では，レンジテスト，シンボリック解析，ランタイム解析，インタープロシージャ解析等のデータ依存解析手法と，アレイプライベタイゼーション，ユニモジュラ変換，タイリング等のループリストラクチャリング手法を組み合わせることで，種々のループの効果的な並列処理が可能になっている．しかしながら，このループ並列化手法は，成熟期に入っており，今後大幅な技術向上は望めない．今後のプロセッサ数の増加とともに，十分なループ回転数のあるプログラム以外は，スケーラブルな性能向上は困難である等の問題がある．この問題点を解決するために，最近では，ループ並列性の利用に加え，ループやサブルーチン等の粗粒度タスクレベルの並列性を利用するマクロデータフロー処理（粗粒度並列処理）[5)]，および，階層的に粗粒度並列処理を行う階層型マクロデータフロー処理[6)]が提案されている．

†1 東邦大学理学部情報科学科
Department of Information Science, Toho University
†2 NTTコミュニケーションウェア株式会社
NTT Communicationware Corporation
†3 株式会社東芝
Toshiba Corporation
†4 早稲田大学理工学部電気電子情報工学科
Department of Electrical, Electronics and Computer Engineering, Waseda University

この階層型マクロデータフロー処理のように，制御依存およびデータ依存を満たしながら粗粒度タスクをプロセッサクラスタ（PC）に実行時にダイナミックスケジューリングする方式では，通常，粗粒度タスク間で共有されるデータを集中共有メモリ（CSM）上に配置し，粗粒度タスク間のデータ授受は集中共有メモリを介して行われる．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．この問題点を解決するためには，処理とデータを適切に分割・配置し，各プロセッサ上のローカルメモリを介してデータ授受を実現することが必要となる．

ローカルメモリへのデータの分割・配置に関する研究は，従来より活発に行われており，ユーザ指定によるデータ分割・配置のための High Performance Fortran (HPF1.0, HPF2.0)[7]，ループ内の作業配列をローカル化するアレイプライベタイゼーション法[8]，自動データ分割・配置法[9)~12)]等が提案されている．しかしながら，これらの方式は適用対象が単一ループに限られている，あるいは，ユーザやコンパイラがデータをスタティックに割り当てできる場合にしか適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（マクロデータフロー処理）においては，Doall ループ間あるいは Doall/シーケンシャルループ間での共有データをローカルメモリ経由で授受するデータローカライゼーション手法が提案されている[13),14)]．しかし，これらの手法は，単階層の粗粒度並列処理のみを対象としており，階層的な粗粒度並列処理手法である階層型マクロデータフロー処理には適用できなかった．そこで，本論文では，階層型マクロデータフロー処理における各階層の粗粒度タスク（ループ）間において，データローカライゼーションを実現する方法を提案する．

本論文 2 章では，階層型マクロデータフロー処理について概説する．3 章では，階層型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について述べる．4 章では，本手法をインプリメントした自動並列化コンパイラを用いて性能評価した結果について述べる．

2. 階層型マクロデータフロー処理

本章では，階層型マクロデータフロー処理の対象アーキテクチャとコンパイレーション手法について述べる．

2.1 対象マルチプロセッサアーキテクチャ

階層型マクロデータフロー処理では，プロセッサ（PE）上にローカルメモリ（LM）または分散共有メモリ（DSM）を持ち，各プロセッサがインタコネクションネットワークを介して集中共有メモリ（CSM）に平等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．

階層型マクロデータフロー処理を適用する場合，プログラムの各階層の並列性に応じて，PE を複数のグループに分け，階層的にプロセッサクラスタ（PC）を定義する．具体的には，まず，マルチプロセッサシステム全体を第 0 階層 PC と定義し，第 0 階層 PC 内の PE をグループ化して第 1 階層 PC を定義する．同様に，第 i 階層 PC 内の PE をグループ化して第 $(i+1)$ 階層 PC を定義する．

2.2 階層型マクロデータフロー処理のコンパイレーション手法

階層型マクロデータフロー処理[6]では，ループやサブルーチン等の粗粒度タスク（マクロタスク）が，プロセッサクラスタ（PC）に割り当てられて粗粒度並列処理が行われる．このとき，プロセッサクラスタに割り当てられたマクロタスク内部では，階層的にマクロデータフロー処理，あるいは，ループ並列化や近細粒度並列処理[15]が適用される．以下の項では，階層型マクロデータフロー処理のコンパイレーション手法について概説する．

2.2.1 階層型マクロタスク生成

階層型マクロデータフロー処理を行うには，対象プログラムを階層的に粗粒度タスク（マクロタスク）に分割する必要がある．具体的には，まず，対象プログラム全体（第 0 階層マクロタスクとする）を，第 1 階層マクロタスク（粗粒度タスク）に分割する．ここで，マクロタスクは，擬似代入文ブロック（BPA），繰返しブロック（RB），あるいは，サブルーチンブロック（SB）のいずれかである[6]．BPA は本質的には基本ブロックであるが，並列性向上のために単一の基本ブロックを分割して複数の BPA を生成したり，逆にスケジューリングおよびデータ転送のオーバヘッド軽減のため，複数の基本ブロックを融合して 1 つの BPA を生成する．RB は最外側ナチュラルループ[16]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は SB として定義する．

次に，第 1 階層マクロタスク（RB または SB）の内部に複数のサブマクロタスク（サブ BPA，サブ RB，サブ SB）を含んでいる場合には，それらのサブマクロタスクを第 2 階層マクロタスクとして定義する．以

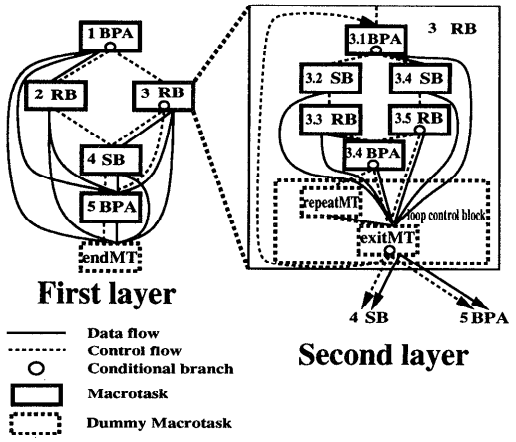


図1 階層型マクロフローグラフ
Fig. 1 Hierarchical macro-flow-graph.

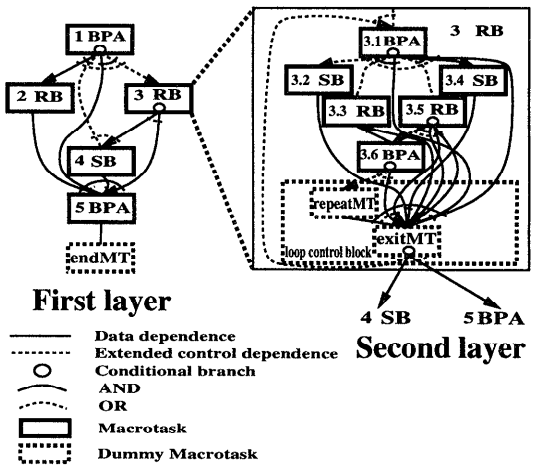


図2 階層型マクロタスクグラフ
Fig. 2 Hierarchical macrotask-graph.

後，同様に第 i 階層において，第 $(i+1)$ 階層マクロタスクを定義する．

ただし，上記の RB（あるいはサブ RB）の定義では，Doall ループが 1 つの RB として扱われ，1 つのプロセッサクラスタ（PC）上で実行されてしまうため，PC 内の PE 数分の並列性しか利用できない．このため，RB（ループ）は，3.1 節で述べる階層型ループ整合分割法により，ループインデクス範囲を分割して複数の部分ループを生成し，分割後の部分ループを RB として定義する．

2.2.2 階層型マクロフローグラフ生成

マクロタスクを階層的に生成した後，マクロタスク間あるいはある階層のマクロタスク内部で生成されるサブマクロタスク間のコントロールフローとデータフローを解析し，**図1** のような階層型マクロフローグラフ[6]を生成する．階層型マクロフローグラフにおいて，各ノードはマクロタスク，実線エッジはデータフロー，点線エッジはコントロールフローを表している．またノード内の小円はノードが条件分岐ノードであることを表している．図 1 の右側は，第 1 階層（図 1 の左側）の MT_3 内部で生成された第 2 階層のサブマクロフローグラフである．

2.2.3 階層型マクロタスクグラフ生成

次に，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，各階層において，各マクロタスクの最早実行可能条件[5),18)]を解析する．最早実行可能条件は，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の並列性を最大限に表している．この各マクロタスクの最早実行可能条件は，**図2** のような階層型マクロタスクグラフ（MTG）[6]で表すことができる．

2.2.4 階層型ダイナミックスケジューリングルーチン生成

階層型マクロデータフロー処理では，マクロタスク間の条件分岐等の実行時不確定性に対処し，さらにスケジューリングオーバヘッドを最小化するために，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，各階層ごとにコンパイラが生成する．ダイナミックスケジューリングアルゴリズムとしては，実行時に出口ノードからの最長パスの大きいマクロタスクを優先的にプロセッサクラスタに割り当てる Dynamic-CP 法を採用している[6]．

なお，階層型マクロデータフロー処理においてデータローカライゼーションを適用する場合には，3.2 節で述べるパーシャルスタティック割当てを用いた階層型ダイナミックスケジューリングルーチンを生成する．

3. 階層型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

階層型マクロデータフロー処理を用いて各階層で粗粒度並列処理を行う場合に，データ転送オーバヘッドを軽減し効率良い並列処理を実現するためには，各階層においてデータローカライゼーション（PE 上のローカルメモリ経由データ転送）を行うことが必要である．そこで，本論文では，階層型マクロデータフロー処理の各階層においてデータローカライゼーションを実現する手法を提案する．

本手法では，まず，対象プログラムに対して，(1) 階層型ループ整合分割法を適用して，ローカルメモリ経由データ授受が行えるように各階層の処理とデータ

を分割し，(2) パーシャルスタティックタスク割当てを用いた階層型ダイナミックスケジューリングにより，各階層において多量のデータ転送を必要とするマクロタスク集合を実行時に当該階層の同一プロセッサクラスタ（PC）に割り当てる．また，(2) により同一PCに割り当てられるマクロタスク集合に対しては，(3) ローカルメモリ経由データ授受を可能とするデータ転送コードを生成する．

3.1 階層型ループ整合分割

階層型ループ整合分割法は，各階層においてデータ依存のある複数ループ（RB）を，分割後に複数PC上での並列処理を可能とし，かつ，ローカルメモリ経由データ授受を実現できるように分割を行うものである．以下の項では，階層型ループ整合分割法について述べる．

3.1.1 ターゲットループグループ（TLG）の階層的生成

階層型ループ整合分割法では，まず，各階層のマクロタスクグラフ（MTG）から，階層型ループ整合分割の対象となるターゲットループグループ（TLG）を選ぶ．ここで，ターゲットループグループ（TLG）は，各階層の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ループ（RB）集合（Doallループ，リダクションループ，ループキャリイドデータ依存を持った完全ネストのシーケンシャルループ）である．なお，TLG生成の際には，前処理として，各RBにループインターチェンジと，各RBのストライドを1にするリストラクチャリングを必要に応じて適用しておく．

たとえば，**図3**(a)の部分MTGにおいては，第1階層において，RB_1（Doallループ）とRB_2（リダクションループ）からなるTLG_1が生成され，RB_3内部の第2階層においては，RB_{31}（シーケンシャルループ），RB_{32}（Doallループ），RB_{33}（リダクションループ）からなるTLG_2が生成される．

3.1.2 TLG内でのループ間データ依存解析

次に，各階層で生成された各TLGにおいて，TLG内部の複数ループ（RB）にまたがるイタレーション間データ依存を解析する．

たとえば，図3(a)のTLG_2の場合，図3(c)に示すように，RB_{33}の$I=k$のイタレーションは，RB_{32}の$I=k$のイタレーションに，配列Cによってデータ依存している．また，RB_{32}の$I=k$のイタレーションは，RB_{31}の$I=k$と$I=k+1$のイタレーションに，配列Bによってデータ依存している．

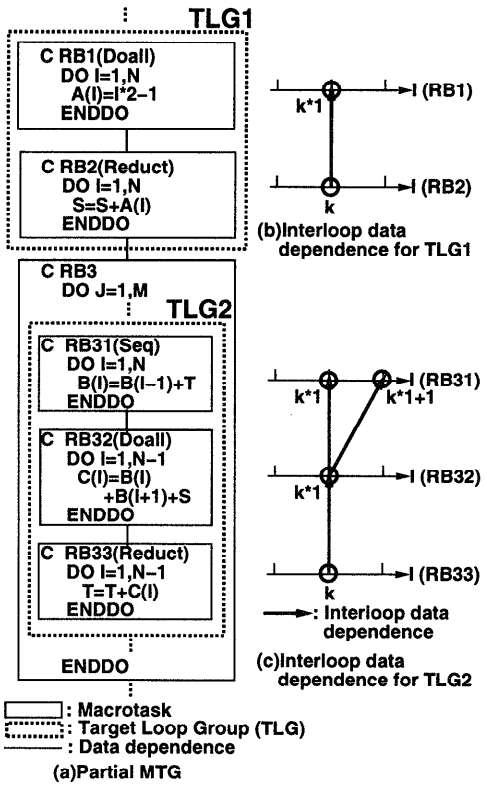


図3 ターゲットループグループ（TLG）
Fig. 3 Target loop group (TLG).

本論文では，このようなデータ依存を，直接的ループ間データ依存と呼び，$DirILD(RB_i, RB_j, k)$と表記する．ここで，$DirILD(RB_i, RB_j, k)$は，RB_jの$I=k$のイタレーションがデータ依存しているRB_iのイタレーション集合を表している．図3(a)のTLG_2の場合，図3(c)に示されるように$DirILD(RB_{32}, RB_{33}, k) = \{k\}$，$DirILD(RB_{31}, RB_{32}, k) = \{k, k+1\}$となる．

次に，本解析では，TLG（$RB_1, \cdots, RB_m$により構成されているとする）内の出口ノードRB_m（標準ループと呼ぶ）と各RB_i $(1 \le i \le m-1)$との間の，ループ間データ依存（$ILD(RB_i, RB_m, k)$と表記）を求める．この$ILD(RB_i, RB_m, k)$は，RB_mの$I=k$のイタレーションが，RB_i $(1 \le i \le m-1)$に，直接的あるいは間接的（他ループを経由して）にデータ依存しているイタレーション集合を表しており，RB_mに対しては式(1)，RB_i $(m-1 \ge i \ge 1)$に対しては式(2)により求められる．

$$ILD(RB_m, RB_m, k) = \{k\}. \quad (1)$$

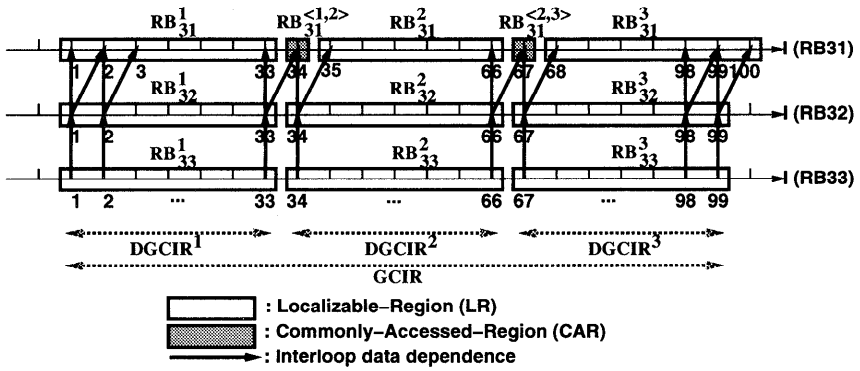


図4 階層型ループ整合分割により分割されたループインデクス（TLG_2 の場合）
Fig. 4 Loop indexes decomposed by hierarchical loop-aligned-decomposition (in TLG_2).

$$ILD(RB_i, RB_m, k) = \bigcup_{RB_j \in SucDepRB(RB_i)} \left(\bigcup_{t \in ILD(RB_j, RB_m, k)} DirILD(RB_i, RB_j, t) \right). \quad (2)$$

ここで，$SucDepRB(RB_i)$ は，RB_i にデータ依存している TLG 内のループ集合を表している．

この解析を図3(a) の TLG_2 に適用すると，$ILD(RB_{33}, RB_{33}, k) = \{k\}$，$ILD(RB_{32}, RB_{33}, k) = \bigcup_{t \in ILD(RB_{33}, RB_{33}, k)} DirILD(RB_{32}, RB_{33}, t) = \{k\}$，$ILD(RB_{31}, RB_{33}, k) = \bigcup_{t \in ILD(RB_{32}, RB_{33}, k)} DirILD(RB_{31}, RB_{32}, t) = \{k, k+1\}$ となる．この結果，図3(c) に示されるように，RB_{33} の $I = k$ のイタレーションは，RB_{32} の $I = k$ のイタレーション，および，RB_{31} の $I = k$ と $I = k+1$ のイタレーションにデータ依存していることが求められる．同様に，図3(a) の TLG_1 の場合，図3(b) のように，RB_2 の $I = k$ のイタレーションが，RB_1 の $I = k$ のイタレーションにデータ依存していることが求められる．

3.1.3 TLG 内ループの整合分割

コンパイラは，各TLGごとに，そのTLG（$RB_1, \cdots, RB_m$ により構成）内のループで使用されるデータの範囲を，標準ループ RB_m のループインデクス範囲で表す．これをグループ変換インデクス範囲（$GCIR$）と呼ぶ．その後，$GCIR$ を PC 数（TLG が処理される階層の PC 数）の整数倍（n）に均等に分割し，この各分割範囲を $DGCIR^p$ $(1 \le p \le n)$ とする．たとえば，図3(a) の TLG_2（ただし $N = 100$）の場合，$GCIR$ は**図4** に示されるように [1:99]（1 から 99 までの範囲）となり，その $GCIR$ を3分割（第2階層 PC 数 = 3 とする）すると $DGCIR^p$ $(1 \le p \le 3)$ はそれぞれ，[1:33]，[34:66]，[67:99] となる．

次に，$DGCIR^p$ $(1 \le p \le n)$ と TLG 内のループ間データ依存の解析結果を用いて，各 RB_i $(1 \le i \le m)$ を分割する．具体的には，部分標準ループ（RB_m のループインデクス範囲を $DGCIR^p$ に変更したループのイメージ）がデータ依存する RB_i のイタレーション集合を，Localizable-Region（RB_i^p）として生成し，複数の部分標準ループ（ループインデクス範囲が $DGCIR^p$ と $DGCIR^{p+1}$ のもの）が共通にデータ依存している RB_i のイタレーション集合を，Commonly-Accessed-Region（$RB_i^{<p,p+1>}$）として生成する．

たとえば，図3(a) の TLG_2（$N = 100$ の場合）を，階層型ループ整合分割法で3分割すると，図4 のように分割される．この場合，RB_{31} は，3つの Localizable-Region（RB_{31}^1，RB_{31}^2，RB_{31}^3）と，2つの Commonly-Accessed-Region（$RB_{31}^{<1,2>}$，$RB_{31}^{<2,3>}$）に分割される．また，階層型ループ整合分割法によりループインデクスが分割された後，TLG 内の部分RB（Localizable-Region，Commonly-Accessed-Rcgion）における配列データの定義・参照範囲は，**表1** のようになる．

3.2 パーシャルスタティックタスク割当てを用いた階層型ダイナミックスケジューリング

階層型マクロデータフロー処理により粗粒度並列処理される各階層において，多量のデータ転送を必要とするマクロタスク間で，プロセッサ上のローカルメモリを介したデータ授受を実現するためには，それらのマクロタスク集合（データローカライゼーショングループ（DLG）と呼ぶ）を，ダイナミックスケジューリング環境下で，同一階層の同一 PC（プロセッサクラスタ）に割り当てなければならない．これを実現するために，本手法では，パーシャルスタティックタスク割当てを用いた階層型ダイナミックスケジューリン

表1 階層型ループ整合分割後の配列データの定義・参照範囲（TLG_2 の場合）
Table 1 Array data ranges after hierarchical loop-aligned-decomposition (in TLG_2).

	LR^1	CAR^1	LR^2	CAR^2	LR^3
	RB^1_{31}	$RB^{<1,2>}_{31}$	RB^2_{31}	$RB^{<2,3>}_{31}$	RB^3_{31}
ループインデクス I	[1:33]	[34:34]	[35:66]	[67:67]	[68:100]
配列 B（参照）	[0:32]	[33:33]	[34:65]	[66:66]	[67:99]
配列 B（定義）	[1:33]	[34:34]	[35:66]	[67:67]	[68:100]
	RB^1_{32}		RB^2_{32}		RB^3_{32}
ループインデクス I	[1:33]		[34:66]		[67:99]
配列 B（参照）	[1:34]		[34:67]		[67:100]
配列 C（定義）	[1:33]		[34:66]		[67:99]
	RB^1_{33}		RB^2_{33}		RB^3_{33}
ループインデクス I	[1:33]		[34:66]		[67:99]
配列 C（参照）	[1:33]		[34:66]		[67:99]

[x:y]: x から y までの範囲

グルーチンを生成する．

3.2.1 データローカライゼーショングループの階層的生成

パーシャルスタティックタスク割当てを用いた階層型ダイナミックスケジューリングを行うには，まず，各階層において同一PCに割り当てるべきマクロタスク集合（データローカライゼーショングループ（DLG））を決定する必要がある．本手法では，各TLGに対して，階層型ループ整合分割が行われた後，多量のデータ転送を必要とする部分RB集合（Localizable-Region（LR））を，データローカライゼーショングループ（DLG）として定義している．この際，処理時間の短いCommonly-Accessed-Region（CAR）は，ダイナミックスケジューリングのオーバヘッドを軽減するため，隣接するLocalizable-Region（LR）に融合しておく．

たとえば，階層型ループ整合分割の適用された図4のTLGに対しては，**図5**に示すようなデータローカライゼーショングループ（DLG_1, DLG_2, DLG_3）が生成される．この例においても，前述のようにCARは隣接LRに融合されており，図5の DLG_1 内の $RB^{1'}_{31}$ は，図4の RB^1_{31}（LR）に $RB^{<1,2>}_{31}$（CAR）が融合されたものである．

3.2.2 パーシャルスタティックタスク割当てを用いたスケジューリングルーチン生成

階層型マクロデータフロー処理では，2.2.4項で述べたように，低オーバヘッドのダイナミックスケジューリングを実現するために，コンパイラがダイナミックスケジューリングルーチンを生成している．この際，本手法では，ダイナミックスケジューリング環境下で，各データローカライゼーショングループ（3.2.1項で生成）内のマクロタスク集合が同一PCに割り当てられるように，従来の階層型ダイナミックスケジューリングルーチンを以下のように拡張している．

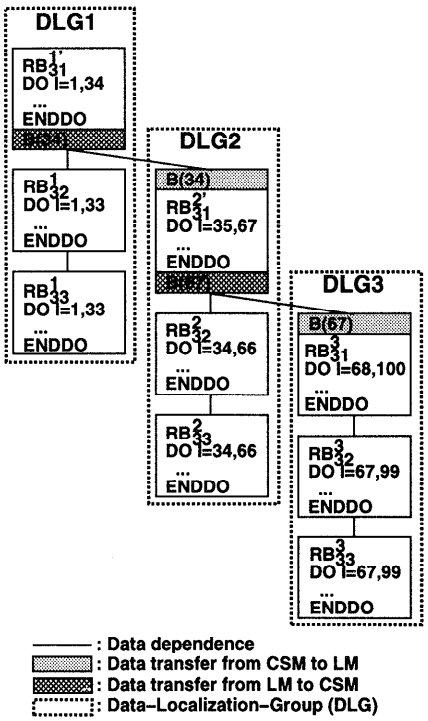


図5 データローカライゼーショングループ（DLG）
Fig. 5 Data-localization-group (DLG).

本方式では，ダイナミックスケジューラが，実行時に，最早実行可能条件を満たしたマクロタスク（レディマクロタスク）を，実行待ちマクロタスクキューへ投入し，実行待ちマクロタスクキューの中で，最長パス長の大きいマクロタスクから順に，PCに割当てを行う．

このとき，PCに割り当てるべきマクロタスクが，あるデータローカライゼーショングループ（DLG_d）の入口マクロタスク（$MT_{(d,entrance)}$）である場合，ダ

イナミックスケジューラは，$MT_{(d,entrance)}$ を割り当てたPCの番号（たとえば PC_p）を，DLG_d 内の後続マクロタスク（$MT_{(d,i\neq entrance)}$）の割当て指定PC番号として，パーシャルスタティックタスク割当て用テーブルに実行時に記録する．

一方，PCに割り当てるべきマクロタスクが，あるデータローカライゼーショングループ（DLG_d）の入口以外のマクロタスク（$MT_{(d,i\neq entrance)}$）である場合，ダイナミックスケジューラは，パーシャルスタティックタスク割当て用テーブルに記録されている $MT_{(d,i\neq entrance)}$ の割当て指定PC番号のPC（$MT_{(d,entrance)}$ の割り当てられたPC，すなわち PC_p）に割当てを行う．この結果，$MT_{(d,entrance)}$（例，図5の $RB^{1'}_{31}$）と $MT_{(d,i\neq entrance)}$（例，図5の RB^{1}_{32}，RB^{1}_{33}）は，同一PCに割り当てられることになる．

なお，本ダイナミックスケジューラでは，DLG内のマクロタスクとDLG以外のマクロタスクを同時に取り扱っているため，あるDLGの割り当てられたPCがアイドルになる場合は，当該DLG以外のレディ状態のマクロタスクがそのPCに割り当てられることになり，PC利用率の向上が図れる．

3.3 データローカライゼーション用データ転送コード生成

本節では，3.2.1項で生成された各階層のデータローカライゼーショングループ内のマクロタスク間で，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．

3.3.1 LM経由データ授受を適用する配列の検出

ここでは，各階層で生成された各データローカライゼーショングループ（DLG）内において，定義あるいは参照される配列変数の中から，ローカルメモリ経由データ授受を適用する配列変数を検出する．具体的には，まず，データローカライゼーショングループ内部で，定義あるいは参照される各配列変数（たとえば配列 X）ごとに，データローカライゼーション適用時に必要となるデータ転送を解析してそのデータ転送に要する時間 $t^X_{localize}$ を算出し，また，従来の集中共有メモリ（CSM）経由データ授受を行う場合のデータ転送時間 t^X_{CSM} も求める．次に，各配列変数 X ごとに，$t^X_{localize}$ と t^X_{CSM} を比較し，$t^X_{localize} < t^X_{CSM}$ を満たす配列変数 X に対しては，ローカルメモリ経由データ転送コードを生成する．

3.3.2 LM経由データ転送コード生成

各データローカライゼーショングループ（DLG）内では，データローカライゼーション（LM経由データ授受）の適用される配列変数に対して，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．これにより，DLG内ではLM経由でデータ授受を行うことが可能となる．

また，DLG内でデータローカライゼーションの適用される配列データが，DLG外のマクロタスクと共有される場合には，その共有される部分データのみをCSM経由で受け渡す転送コードを生成する．たとえば，図5において，異なるDLGに属するRB間（$RB^{1'}_{31}$ と $RB^{2'}_{31}$ の間）で共有されている配列データ（B(34)）は，CSM経由で転送される．

4. 性能評価

本章では，データローカライゼーションを用いた階層型マクロデータフロー処理を，複数のアプリケーションプログラムを用いて，OSCAR[17]のシミュレータ（OSCARのマシンコードをクロックレベルでシミュレート可能）上で性能評価した結果について述べる．

4.1 OSCARのアーキテクチャ

性能評価に用いるOSCAR[17]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（CSM）に3本のバスで接続した集中/分散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，CSM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．OSCARは各PEを集中共有メモリ（CSM）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，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．

4.2 Tomcatvプログラムを用いた性能評価

本節では，SPECfp95ベンチマークのTomcatvプログラムを用いて，階層型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の性能評価を行う．このプログラムは，Vectorized Mesh生成プログラムであり，初期化部分と収束計算ループから構成されている．このプログラムを階層型マクロデータフロー処理により実行する場合，収束計算ループは第1階層マクロタスクとして定義され，その収束計算ループの内部では，第2階層マクロタスクが定義される．なお，この収束計算ループの内部（第2階層）は，6個のDoallループ（このうち，3つのDoallループ

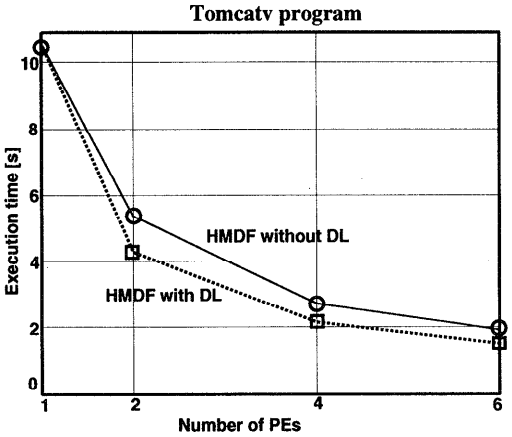


図6 Tomcatvプログラムによる OSCAR 上での性能評価
Fig. 6 Performance evaluation using Tomcatv program on OSCAR.

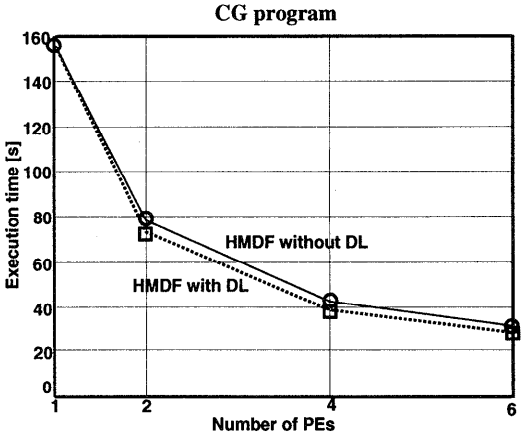


図7 CG 法プログラムによる OSCAR 上での性能評価
Fig. 7 Performance evaluation using CG program on OSCAR.

に対しては，Scalar Privatization とループインターチェンジを適用している）と，1つのリダクションループ，4つの基本ブロックから構成されている．本性能評価においてプロセッサクラスタの構成は，第1階層は1PC（プロセッサクラスタ）とし，第2階層は，2PC，4PC，または6PC（各PCは1PE構成）とする．

まず，このプログラムを OSCAR シミュレータ上で，データローカライゼーションを用いない階層型マクロデータフロー処理により実行した際の処理時間は，**図6** の実線（HMDF without DL）に示されるとおりである．この場合，1PE上での処理時間は，4PC（4PE）で1/3.87，6PC（6PE）で1/5.45に短縮されている．

次に，このプログラムをデータローカライゼーションを用いた階層型マクロデータフロー処理で実行すると，処理時間は図6の点線（HMDF with DL）に示されるように，データローカライゼーション手法を適用しない場合と比べて，4PC（4PE）で20.4%，6PC（6PE）で22.0%の速度向上が得られている．このことから，階層型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の有効性が確認された．

4.3 CG 法プログラムを用いた性能評価

次に，連立 1 次方程式反復解法である CG（Conjugate Gradient）法プログラムを用いて提案手法を性能評価する．CG 法プログラムを階層型マクロデータフロー処理により実行する場合，CG 法プログラムを構成する初期化部分と収束計算ループが第1階層マクロタスクとして定義され，第1階層マクロタスクである収束計算ループの内部において，第2階層マクロタスクが定義される．この収束計算ループ内部の第2階層マクロタスクは，それぞれ，4つの Doall ループ，2つのリダクションループ，5つの基本ブロックである．本性能評価においてプロセッサクラスタの構成は，第1階層は1PC（プロセッサクラスタ）とし，第2階層は，2PC，4PC，または6PC（各PCは1PE構成）とする．

OSCAR シミュレータ上でこのプログラムをデータローカライゼーションを用いない階層型マクロデータフロー処理で実行した結果は，**図7** の実線（HMDF without DL）に示されるように，1PE上での処理時間を4PC（4PE）で1/3.65，6PC（6PE）で1/4.89まで短縮している．

さらに，データローカライゼーションを用いた階層型マクロデータフロー処理では，図7の点線（HMDF with DL）に示すように，処理時間が4PC（4PE）で1/3.99，6PC（6PE）で1/5.39に短縮されており，本データローカライゼーション手法を用いた階層型マクロデータフロー処理の有効性が確認された．

5. おわりに

本論文では，階層型マクロデータフロー処理により粗粒度並列処理される各階層において，粗粒度並列性を最大限に利用しつつ，粗粒度タスク（ループ）間データ転送にプロセッサ上のローカルメモリを有効利用し，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．

OSCAR シミュレータ上での性能評価により，提案したデータローカライゼーション手法は，各階層で粗粒度並列処理される粗粒度タスク間データ転送を軽減

し，データローカライゼーションを用いない階層型マクロデータフロー処理に比べて，処理時間を顕著に短縮できることが確かめられた．

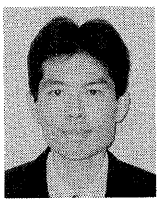
今後の課題としては，階層型マクロデータフロー処理により実行されるサブルーチン間でデータローカライゼーションを実現することがあげられる．

謝辞 本研究の一部は，文部省科学研究費補助金（奨励研究（A）10780205），および，通産省次世代情報処理基盤技術事業並列分散コンピューティング分野マルチプロセッサコンピューティング領域の一環として行われた．

参考文献

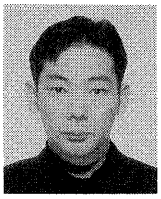
1) Banerjee, U.: *Loop Transformations for Restructuring Compilers*, Kluwer Academic Pub. (1993).
2) Wolfe, M.: *High Performance Compilers for Parallel Computing*, Addison-Wesley (1996).
3) Blume, W., Doallo, R., Eigenmann, R., Grout, J., Hoeflinger, J., Lawrence, T., Lee, J., Padua, D., Paek, Y., Pottenger, B., Rauchwerger, L. and Tu, P.: Advanced Program Restructuring for High Performance Computers with Polaris, Technical Report 1473, CSRD, University of Illinois, Urbana-Champaign (1996).
4) Amarasinghe, S., Anderson, J., Lam, M. and Tseng, C.-W.: The SUIF Compiler for Scalable Parallel Machines, *Proc., 7th SIAM conference on parallel processing for scientific computing* (1995).
5) 本多弘樹，岩田雅彦，笠原博徳：Fortran プログラム粗粒度タスク間の並列性検出手法，電子情報通信学会論文誌（D-I），Vol.J73-D-I, No.12, pp.951–960 (1990).
6) 岡本雅巳，合田憲人，宮沢 稔，本多弘樹，笠原博徳：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).
7) High Performance Fortran Forum: *High Performance Fortran Language Specification Version 2.0,* High Performance Fortran Forum (1997).
8) Tu, P. and Padua, D.: Automatic Array Privatization, *6th Annual Workshop on Languages and Compilers for Parallel Computing* (1993).
9) Chen, T.-S. and Sheu, J.-P.: Communication-Free Data Allocation Techniques for Parallelizing Compilers on MUlticomputers, *IEEE trans. parallel and distributed systems*, Vol.5, No.9 (1994).
10) Agarwal, A., Kranz, D.A. and Natarajan, V.: Automatic Partitioning of Parallel Loops and Data Arrays for Distributed Shared-Memory Multiprocessors, *IEEE Trans. Parallel and Distributed System*, Vol.6, No.9, pp.943–962 (1995).
11) 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).
12) Anderson, J. and Lam, M.: Global Optimizations for Parallelism and Locality on Scalable Parallel Machines, *Proc. SIGPLAN '93 Conference on Programming Language Design and Implementation*, pp.112–125 (1993).
13) 吉田明正，前田誠司，尾形 航，笠原博徳：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).
14) 吉田明正，前田誠司，尾形 航，笠原博徳：Fortran 粗粒度並列処理における Doall/シーケンシャルループ間データローカライゼーション手法，電子情報通信学会論文誌，Vol.J78-D-I, No.2, pp.162–169 (1995).
15) Kasahara, H., Honda, H. and Narita, S.: Parallel Processing of Near Fine Grain Tasks Using Static Scheduling on OSCAR, *IEEE ACM Supercomputing'90* (1990).
16) Aho, A., Sethi, R. and Ullman, J.: *Compilers (Principles, Techniques, and Tools)*, Addison-Wesley (1988).
17) 笠原博徳，成田誠之助，橋本 親：OSCAR のアーキテクチャ，電子情報通信学会論文誌（D），Vol.J71-D, No.8 (1988).
18) 笠原博徳：並列処理技術，コロナ社 (1991).

(平成 10 年 8 月 31 日受付)
(平成 11 年 1 月 8 日採録)



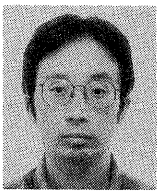
吉田 明正（正会員）

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．1996 年同大大学院博士課程修了．博士（工学）．1993～1995 年日本学術振興会特別研究員．1995～1997 年早稲田大学理工学部電気電子情報工学科助手．1997 年東邦大学理学部情報科学科専任講師，現在に至る．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，電気学会，IEEE，ACM 各会員．



越塚 健一

1972年生．1995年早稲田大学理工学部電気工学科卒業．1997年同大大学院修士課程修了．同年日本電信電話（株）入社．現在，NTTコミュニケーションウェア（株）所属．在学中，並列化コンパイラの研究に従事．



岡本 雅巳（正会員）

1966年生．1990年早稲田大学理工学部電気工学科卒業．1992年同大大学院修士課程修了．1997年同大大学院博士課程修了．1993～1996年早稲田大学情報科学研究教育センター助手．1997年（株）東芝入社，現在に至る．発電監視制御システムの開発に従事．電子情報通信学会会員．

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．1983～1985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，1997年電気電子情報工学科教授，現在に至る．1989～1990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．1997年本会坂井記念特別賞受賞．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各会員．