計算機アーキテクチャ 136－8
（2000．1．7）

データ依存マクロタスクグラフに対するデータローカライゼーション手法

成清暁博[†] 松崎秀則[††] 小幡元樹[†]
吉田明正[†††] 笠原博徳[†]

本論文では，階層型粗粒度タスク並列処理における，データ依存エッジのみをもつマクロタスクグラフ全体または部分グラフを対象とした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，粗粒度並列処理される各階層において，ループ整合分割手法を用いて処理とデータを分割する．次に，分割されたデータ転送の多い粗粒度タスク集合をパーシャルスタティックタスク割当を用いたダイナミックスケジューリング方式により同一プロセッサに割り当て，粗粒度タスク間データ転送にローカルメモリ（LM）を利用してデータ転送オーバヘッドを軽減する．本手法は，任意形状のマクロタスクグラフ中のデータ依存のみをもつ部分グラフに適用でき，ループインデクス上下限値が変数として与えられるループをも対象とすることにより，多くの配列変数をLMに分散し，データ転送及びメモリアクセスオーバヘッドを軽減することを可能とする．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CG法の階層型粗粒度タスク並列処理では，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20%短縮されることが確認された．

A Data-Localization Scheme for Macrotask-Graphs with Data Dependencies

Akihiro Narikiyo,[†] Hidenori Matsuzaki,[††] Motoki Obata,[†]
Akimasa Yoshida[†††] and Hironori Kasahara [†]

This paper proposes a data-localization scheme for a part with data dependence edges in any kinds of macrotask-graphs in hierarchical coarse grain parallel processing. First, multiple loops having data dependence are decomposed into data-localization-groups in each macrotask-graph layer. Next, the compiler generates a hierarchical dynamic scheduling routine with partial static task assignment, which assigns macrotasks inside data-localization-group to the same processor or processor-cluster in each layer, so that shared data can be transferred via local memory. This data localization scheme can be applied to a part or the whole macrotask graph which only has data dependence edges. This data localization scheme also handles loops with the lower and upper limit given by variables. As a result, most of array data is transferred via local memory. Finally, this paper describes the performance evaluation on a multi-processor system OSCAR. The evaluation shows that hierarchical coarse grain parallel processing with data-localization can reduce execution time about 20% compared with hierarchical coarse grain parallel 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[1]が使用されている．例えば，イリノイ大学のPolaris[2]やスタンフォード大学のSUIF[3]などのような先端の並列化コンパイラでは，強力なデータ依存解析と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さまざまな形状のループが並列化可能になっている．しかしながら，これらのループ並列化手法では，単一ループのイタレーション間の並列性しか利用することができず，コストの大きいシーケンシャルループあるいは回転数の小さい並列ループを多く持つプログラムでは効果的な並列処理が行えないという問題があった．この問題を解決するために，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），及び階層的に粗粒度

† 早稲田大学 理工学部 電気電子情報工学科, Dept. of Electrical, Electronics and Computer Engineering, Waseda Univ.
†† （株）東芝, Toshiba Corporation
††† 東邦大学 理学部 情報科学科
Dept. of Information Science, Toho University

並列処理を行う階層型粗粒度タスク並列処理が有効と考えられる[4].

この階層型粗粒度タスク並列処理のように，粗粒度タスクをプロセッサクラスタ（PC）に実行時に割り当てる方式では，粗粒度タスク間で共有されるデータを集中共有メモリ（CSM）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なうのが一般的である．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（LM）を介してデータ授受を実現することが必要となる．

LMへのデータの分割・配置に関しては多くの研究が行われており，ユーザ指定によるデータ分割・配置のための High Performance Fortran（HPF）[5]，ループ内の作業配列をローカル化する Array Privatization 法[6]，自動データ分割・配置法[7][8][9]などが提案されている．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タスク並列処理においては，ループ分割後タスクの垂直実行によるローカリティの抽出[10]，複数の粗粒度タスク間での共有データをLM経由で授受するデータローカライゼーション手法[11][12][13]が提案されている．また，処理に時間がかかるループ集合が単一のPCに割り当てられるのを防ぐため，それぞれのループを分割し，分割されたデータ依存のあるマクロタスク間で，LMを用いたデータ転送を実現するための整合分割手法[11][12][13]も提案されている．

しかし，従来のデータローカライゼーション手法[11][12][13]は，お互いが唯一の直接後続マクロタスク及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をもつ（直列依存型の）ループ集合のみを対象としていた．本論文では，データ依存のみを持つ任意形状のマクロタスクグラフに適用可能とすることで，より多くのマクロタスク間データ授受をLM経由で行えるようにし，プロセッサ間データ転送オーバーヘッドを軽減して速度向上を図る．

また，従来の整合分割手法は，ループインデクス上下限値が定数であるか，または，定数伝搬によりコンパイル時に定数値となっていることが仮定されており，上下限値が変数の場合は対象としていなかった．しかし，実アプリケーションではループインデクス上下限値が変数で与えられることが多く，整合分割手法及びデータローカライゼーション手法の拡張が望まれていた．本論文では，このループインデクス上下限値が変数で与えられる場合の整合分割手法についても述べる．

本論文第2章では，階層型粗粒度タスク並列処理について概説する．第3章では，任意形状のマクロタスクグラフにおけるデータローカライゼーションについて述べる．第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. 階層型粗粒度タスク並列処理

本章では，階層型粗粒度タスク並列処理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

階層型粗粒度タスク並列処理では，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共有メモリ型マルチプロセッサシステムを対象とする．

階層型粗粒度タスク並列処理を適用する場合，プロセッサをグループ化して，階層的にプロセッサクラスタを定義する．具体的には，まず，マルチプロセッサシステム全体を第0階層プロセッサクラスタ（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n階層PC内のPEをグループ化して第$(n+1)$階層PCを定義する．

2.2 階層型粗粒度タスク並列処理のコンパイル手法

階層型粗粒度タスク並列処理[4]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）に割り当てられて並列処理される．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，ループ並列化，近細粒度並列処理，あるいは，階層的に粗粒度並列処理が適用される．以下では，この階層型粗粒度タスク並列処理のコンパイル手法について述べる．

2.2.1 階層型マクロタスク生成

階層型粗粒度タスク並列処理手法では，まず，プログラム（全体を第0階層マクロタスクとする）を第1階層マクロタスクに分割する．マクロタスクは，擬似代入文ブロック（BPA），繰り返しブロック（RB），あるいは，サブルーチンブロック（SB）の3種類から構成される[4]．

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．

2.2.2 階層型マクロフローグラフ・マクロタスクグラフ生成

マクロタスク生成後，マクロタスク間あるいは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，階層型マクロフローグラフ[4]を生成する．

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，各マ

クロタスクの最早実行可能条件を解析する．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1のような階層型マクロタスクグラフ（MTG）[4]で表すことができる．

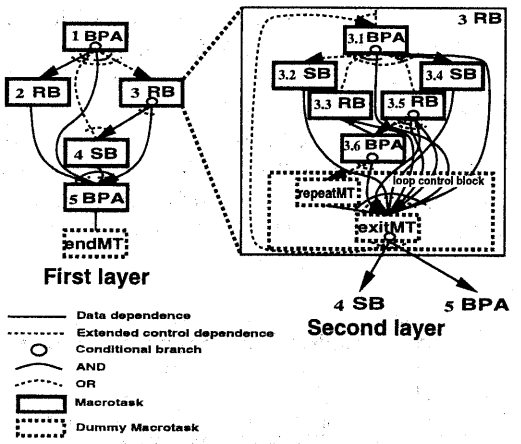

図1　階層型マクロタスクグラフ．

2.2.3 階層型ダイナミックスケジューリングルーチン生成

階層型粗粒度タスク並列処理では，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，Dynamic-CP法を採用している[4]．

3. 任意形状データ依存マクロタスクグラフにおけるデータローカライゼーション手法

各階層で粗粒度並列処理を行なう際，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を行なうことが必要である．そこで，本手法では各階層において，以下の手順でデータローカライゼーションを行う．

(1) 拡張されたループ整合分割法を適用し，変数インデクス上下限値を含むループの処理とデータをLM経由のデータ授受が行なえるように分割する．

(2) パーシャルスタティックタスク割当を用いたダイナミックスケジューリング[12][13]により，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する．

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，

(3) ローカルメモリ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 変数インデックス上下限値をもつループの整合分割

ループ整合分割法は，データ依存があり，ループインデクス上下限値が定数または変数である複数ループ（RB）を，分割後に複数PC上での並列処理を可能とし，かつ，LM経由データ授受を実現できるように配列の参照添字などを考慮して最適な分割を行なうものである．以下の節では，階層型粗粒度タスク並列処理用のループ整合分割法について述べる．

3.1.1 任意形状データ依存マクロタスクグラフを対象とするターゲットループグループ（TLG）生成

ループ整合分割では，まず，各階層のマクロタスクグラフ（MTG）から，ループ整合分割の対象となる部分MTGを選ぶ．この部分MTGのことを，ターゲットループグループ（TLG）と呼ぶ．

TLGは図2のように，1）直列依存型，2）merge依存型，3）diverge依存型の各データ依存形状をもつマクロタスク集合を対象とする．また，これらの形状を組み合わせた形状も対象とする．対象とするマクロタスクはDOALLループとループインデクス変数で配列を操作するシーケンシャルループである．収束ループは，その内部階層でTLGが生成されるため，当該階層ではTLGの対象としない．また，途中に擬似代入文ブロック（BPA）すなわち基本ブロックを含んでいても構わない．

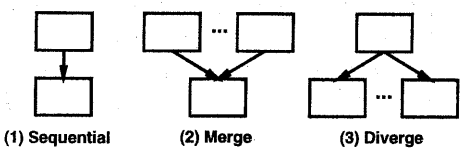

図2　TLGの対象とするマクロタスク間データ依存型．

例えば，図3（a）は，ループインデクス上下限値が変数で与えられる3つのループ（$RB1$, $RB2$, $RB3$）からなるmerge依存型のターゲットループグループ（TLG）である．

TLGとこれからTLGに加えようとする対象マクロタスク間のデータ転送において，後述するCommonly-Accessed-RegionのLocalizable Regionの実行時間の$1/N_{max}$より大である場合には，ローカライゼーション効果を発揮できないと判断し，対象マクロタスクはTLGに含めない．各ループの配列参照範囲が異なる場合は，これらのループの最大範囲をグループ変換インデクス範囲$GCIR$として分割を行う．

TLGの生成は，データ依存グラフにおいてデータ転送量が最大のエッジで接続された2つのマクロタス

ク集合から始め，そのTLGから伸びるエッジのうち条件を満足するマクロタスクをTLGに繰り返し追加していくことで行なわれる．ひとつのTLGが生成されると，同一階層中でTLGに含まれなかった残りのマクロタスク集合について同様の処理を，グループが生成できなくなるまで繰り返す．したがって，TLGは同一階層中に複数存在することもあり得る．

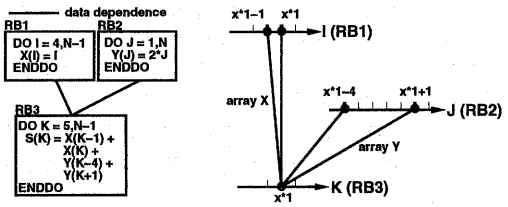

図3　変数インデクス範囲をもつループ集合から構成されるmerge依存型ターゲットループグループ（TLG）

3.1.2　TLG内ループ間データ依存解析

次に，各階層で生成された各TLG（m個のRBで構成されているものとする）内部において，ループ間データ依存を解析する．このループ間データ依存は，TLGの出口ノード（出口ノードが複数存在する場合は，そのうちの1つのノード）RB_m（TLG内でm番目のRB，標準ループと呼ぶ）のイタレーションから，TLG内の各RB_i（TLG内でi番目のRB，$1 \leq i \leq m-1$）のイタレーションへの直接的あるいは間接的（他RBを経由した）データ依存を意味する．

例えば，図3（a）のTLGにおいてループ間データ依存を求めると，図3（b）に示すように，RB_3のx番目のイタレーションが，RB_2のイタレーション（$x*1-4$番目，$x*1+1$番目）と，RB_1のイタレーション（$x*1-1$番目，$x*1$番目）にデータ依存していることがわかる．

3.1.3　TLG内変数インデックス範囲ループの整合分割

コンパイラは，各TLG内で使用されるデータの範囲を，標準ループRB_mのインデクス範囲で表し，これをグループ変換インデクス範囲（$GCIR$）と呼ぶ．その後，$GCIR$をメモリサイズも考慮し，PC数の整数倍（n）に均等に分割し，この各分割範囲を$DGCIR^p$（$1 \leq p \leq n$）とする．例えば，図3（a）のTLGの場合，$GCIR$と$DGCIR^p$（この例は3PC用）は図4のようになる．

次に，$DGCIR^p$（$1 \leq p \leq n$）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（$1 \leq i \leq m$）を分割する．具体的には，部分標準ループ（ループインデクス上下限値が$DGCIR^p$のもの）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（ループインデクス上下限値が$DGCIR^p$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．

例えば，図3（a）のTLGを，3PC用にループ整合分割法で分割すると，図4のように分割される．この場合，RB_1は，3つのLocalizable-Region（RB_1^1，RB_1^2，RB_1^3）と，2つのCommonly-Accessed-Region（$RB_1^{<1,2>}$，$RB_1^{<2,3>}$）に分割される．

図3（a）のように，ループインデクス上下限値が変数の場合，図4のように，分割後のループインデクス上下限値は線形式で表される．コンパイラは，3.3.1節で述べるLM経由データ授受を適用する配列の検出のための解析情報としてこれらの線形式を保存し，ループの実行開始の直前で初期値と終値を求めるためのオブジェクトコードを埋め込む．

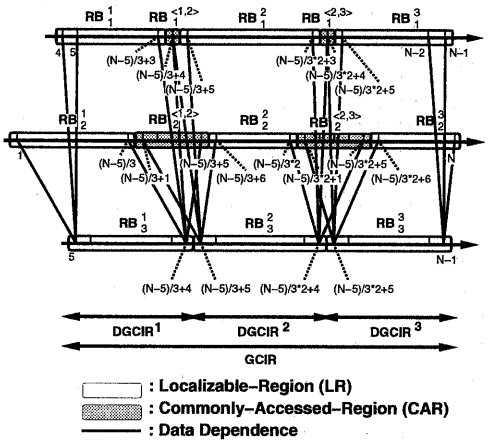

図4　ループ整合分割によるループインデクスの分割．

3.2　パーシャルスタティックタスク割当を用いたダイナミックスケジューリング

階層型粗粒度タスク並列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（データローカライゼーション）を実現するためには，それらのマクロタスク集合（以後，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PC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[12)13)]を用いたダイナミックスケジューリン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生成

各TLG内でループ整合分割が行なわれた後，データ転送量の多い部分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）とする．なお，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．

3.2.2 実行時のダイナミックスケジューリングルーチンの動作

パーシャルスタティックタスク割当を用いたダイナミックスケジューリングでは，DLG内のあるマクロタスクが最初に割り当てられる際に，DLG内の他のマクロタスクが割り当てられるべきPC番号を指定する．DLGの入口マクロタスクが複数存在する場合は，DLG内で最初に割り当てられるマクロタスクは常に一定とは限らない．DLG内のマクロタスクの最早実行可能条件が満たされた時には，各DLG用の実行待ちMTキューへ投入後，指定されたPCへ割り当てられる．

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で生成された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，配列の定義・参照範囲を解析し，PC内の各PE上の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 変数範囲に対応したLM経由データ授受を適用する配列の検出

ここでは，DLG内でLM経由データ授受を適用する配列変数を検出する．まず，DLG内部で，各配列変数ごとに，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する．配列の定義・参照範囲解析は，線形式で行われる．定義・参照範囲が複数存在する場合は，線形式同士を比較し，範囲の融合を試みる．融合できない場合は実行時に転送範囲の重複が生じる可能性があるが，複数の範囲のまま解析を進める．

次に，解析された転送範囲をもとに，配列Xについて，LM経由で転送を行なう（データローカライゼーションを適用する）か，従来の集中共有メモリ経由で転送を行なうかを決定する．データ転送にそれぞれ要する時間$t^X_{localize}$，t^X_{csm}を求めて比較する．ループインデクス上下限値が変数の場合，コンパイル時には転送範囲が不明だが，基本的に，配列1要素の転送時間に転送範囲の要素数を掛けたものとなるので，$t^X_{localize}$とt^X_{csm}の具体的な値は不明であっても，比較することは可能である．

$t^X_{localize} < t^X_{csm}$を満たす配列変数$X$に対しては，LM経由データ転送コードを生成する．

3.3.2 LMへのデータ配置

データローカライゼーションが適用されることが決定した配列については，DLG内における当該配列への定義・参照範囲を考慮して，必要最小限のメモリ容量をLM上に確保する．ただし，ループインデクス上下限値が変数の場合は，実行時に配列のあらゆる場所がアクセスされる可能性があるため，配列の定義範囲全容量をLMに確保する．

3.3.3 LM経由データ転送コード生成

DLG内では，データローカライゼーションの適用される配列変数に対して，各PE上のLMへのロード・ストア命令を生成する．

また，データローカライゼーションの適用される配列要素が，データローカライゼーショングループ外のマクロタスクと共有される場合には，CSM経由でデータ授受を行なう転送コードを生成する．

ロード・ストア命令やCSM経由転送命令は，配列の添字に線形式の指定が可能であり，その場合は転送されるべき要素範囲は実行時に算出される．

4. 性能評価

本章では，データローカライゼーションを伴う階層型粗粒度タスク並列処理を，アプリケーションプログラムを用いて，OSCAR[14]のシミュレータ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[14]は，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分散・集中共有メモリ型マルチプロセッサシステムである．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及びDSMへのロードやストアに1クロックを要し，CSM及び他PEのDSMへのロードやストアには4クロックを要する．

4.2 CG法プログラムを用いた性能評価

連立1次方程式反復求解法である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法プログラムは，第1階層が初期化部分と収束計算ループからなり，処理時間のほとんどは収束計算ループに費やされる．本手法は，この収束計算ループのボディ部（第2階層）に適用される．

OSCARシミュレータ上でこのプログラムを階層型粗粒度タスク並列処理で実行した結果を，表1に示す．4PEでの速度向上率について述べると，シーケンシャル処理（1PE）を1.0としたとき階層型粗粒度タスク並列処理による速度向上率は，3.06である．階層型粗粒度タスク並列処理に加えて，直列依存型のみを対象とした従来のデータローカライゼーション手法を適用した場合は，3.36となる．さらに本論文で提案するデータローカライゼーション手法を適用すると，3.83となり，データローカライゼーションを行わない場合に比べ20%，従来のデータローカライゼーションと比べ12%という顕著な速度向上を得ることができる．これは，より広範囲のマクロタスクをTLGに含んだことにより，ほとんどのマクロタスク間データ転送が

ローカルメモリ経由で実現されたためである.

表1 CG法プログラムによるOSCAR上の評価

処理方式	実行時間[s]			
	1PE	2PE	3PE	4PE
シーケンシャル処理	357.8	–	–	–
HMDF without DL	357.8	184.0	143.8	117.1
HMDF with old DL	357.8	181.9	132.3	106.6
HMDF with new DL	357.8	151.6	110.7	93.5

(注)HMDF：階層型粗粒度タスク並列処理
DL：データローカライゼーション
old DL：直列依存型マクロタスク集合のみを対象とするデータローカライゼーション
new DL：本論文で提案する，データ依存のみをもつ任意形状のマクロタスク集合を対象としたデータローカライゼーション

5. おわりに

本論文では，階層型粗粒度タスク並列処理におけるデータローカライゼーション手法において，変数インデックス範囲ループを含む，データ依存のみを持つ任意形状のマクロタスクグラフを対象とし，より多くのデータ転送を，LM経由で実現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，階層的に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，集中共有メモリ経由データ授受を行う階層型粗粒度タスク並列処理に比べて，例えばCG法では20%処理時間を短縮するなど，顕著な性能向上を得ることができた.

参考文献

1) Wolfe, M.: High Performance Compilers for Parallel Computing, *Addison-Wesley Publishing Company* (1996).
2) Blume, W., Doallo, R., Eigenmann, R., Grout, J., Hoeflinger, J., Lawrence, T., Lee, J., Padua, D., Paek, Y., Pottenger, B. and L.Rauchwerger, P. T.: Advanced Program Restructuring for High Performance Computers with Polaris, *Technical Report 1473, CSRD, University of Illinois, Urbana-Champaign* (1996).
3) Amarasinghe, S., Anderson, J., Lam, M. and Tseng, C.-W.: The SUIF Compiler for Scalable Parallel Machines, *Proc. of the seventh SIAM conference on parallel processing for scientific computing* (1995).
4) 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).
5) High Performance Fortran Forum: *High Performance Fortran Language Specification Version 2.0*, High Performance Fortran Forum (1997).
6) Tu, P. and Padua, D.: Automatic Array Privatization, *6th Annual Workshop on Languages and Compilers for Parallel Computing* (1993).
7) Agarwal, A., Kranz, D. A. and Natarajan, V.: Automatic Partitioning of Parallel Loops and Data Arrays for Distributed Shared-Memory Multiprocessors, *IEEE Trans. on Parallel and Distributed System*, Vol. 6, No. 9, pp. 943–962 (1995).
8) Gupta, M. and Banerjee, P.: Demonstration of Automatic Data Partitioning Techniques for Parallelizing Compilers on Multicomputers, *IEEE Trans. on Parallel and Distributed System*, Vol. 3, No. 2, pp. 179–193 (1992).
9) Anderson, J. and Lam, M.: Global Optimizations for Parallelism and Locality on Scalable Parallel Machines, *Proc. of the SIGPLAN '93 Conference on Programming Language Design and Implementation*, pp. 112–125 (1993).
10) Suvas Vajracharya, Steve Karmesin, Peter Beckman, James Crotinger, Allen Malony, Sameer Shende, Rod Oldehoeft, and Stephen Smith: SMARTS: Exploiting Temporal Locality and Parallelism through Vertical Execution, *International Conference on Supercomputing*, pp. 302–310 (1999).
11) 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).
12) Kasahara, H., Yoshida, A.: A Data-Localization Compilation Scheme Using Partial-Static Task Assignment, Journal of Parallel Computing, Vol. 24, No. 3, pp. 579–596, (1998).
13) 吉田明正, 越塚健一, 岡本雅巳, 笠原博徳: 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, 情報処理学会論文誌, Vol. 40, No. 5, pp. 2054–2063, (1999).
14) 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).