ハイパフォーマンス
コンピューティング 77－18
（１９９９．８．３）

分散共有メモリ向け自動データ分散方法の提案

廣岡孝志† 太田寛† 菊池純男†
†新情報日立研究室

分散共有メモリ向けコンパイラにおける自動データ分散方法を提案する。まず、ファーストタッチ方式と呼ばれるOSのデータ分散方式をコンパイラで制御する「ファーストタッチ制御方法」を提案する。本方法により、カーネルループにおいて一部のみが参照される配列や、手続き間で宣言形状が異なる配列に対して、最適なデータ分散を実現することができる。その結果として、このような配列のデータローカリティを向上させ、またメモリアクセス集中を削減することができる。次に、ファーストタッチ制御方法、データ再分散解析、手続き間解析を用いた分散共有メモリ向け自動データ分散方法を提案する。ベンチマークプログラムNPB2.3serialのFTを用いた事前評価により、自動データ分散を行わない場合に比べて１６プロセッサ時で約７６%性能が向上することを確認した。

Automatic Data Distribution Method for Distributed Shared Memory

Takashi HIROOKA† Hiroshi OHTA† Sumio KIKUCHI†
†RWCP Hitachi Laboratory

We propose an automatic data distribution method for distributed shared memory compilers. First, We propose the *first touch control method* by which the compiler controls the first touch data distribution of the operating system. This method can achieve appropriate data distribution when there are partial array accesses in the kernel loop and when the declarations of array shape differ interprocedurally. As a result, it improves the data locality and reduce the memory hot-spot in such cases. Next, we propose an automatic data distribution method for distributed shared memory using the first touch control method, data redistribution analysis, and interprocedural analysis. By preliminary evaluation by the benchmark FT of NPB2.3serial, we show performance improvement of 76% in the case of 16 processors.

１．はじめに

分散共有メモリ(DSM)型並列計算機は、共有メモリの容易な並列プログラミング環境を提供しながら、分散メモリのスケーラビリティを確保できるアーキテクチャとして注目を集めている。しかし、メモリが物理的に分散している以上は、十分な性能を引き出すためには各ノードへの最適なデータ分散が必要不可欠であり、このためコンパイラの果たすべき役割は大きい。そこで、本研究では、DSM向け自動並列化コンパイラ開発の一環として、各ノードへの最適なデータ分散を自動化する方法を提案する。

DSM向けに最適なデータ分散を実現する方法については、リシェイプ分散等の研究が行われてきた[1][6]。また、自動データ分散方法に関する研究も数多く報告されている[2][3]。しかし、これらの方法では実プログラムで比較的良く出現するある種のパターンにおいて最適なデータ分散を実現することができない。それは、カーネルループにおいて配列の一部のみが参照される場合や、手続き間で配列の宣言形状が異なる場合などである。本稿では、これを解決するためにOSのファーストタッチ方式[1]をコンパイラで制御するデータ分散方法を示す。以後、このデータ分散方法を**ファーストタッチ制御方法**と呼ぶ。また、これを用いたDSM向け自動データ分散方法を提案する。

2．ファーストタッチ制御方法

本章では、従来のデータ分散方法では上手くいかない幾つかの場合を示し、これを解決する新しいデータ分散方法を示す。

2．1　従来のデータ分散方法

DSM を実現する手段の一つとして、仮想メモリ空間をページ単位で切り分け、各ノードの物理メモリに割り付ける方法がある。このとき、各ノードに分散された物理メモリへデータを割り付ける方法には、幾つかの方法がある。今回、検討のプラットフォームとして用いた代表的 DSM 型並列計算機 SGI/Origin2000*) には、ローカル参照の割合、すなわちデータローカリティを向上させる目的で用意されたデータ分散方法が２つある。

(1)データ分散指示文によるデータ分散[4]

プログラム中のユーザが挿入したデータ分散指示文に従って、コンパイラがデータを各ノードに割り付ける。

(2)ファーストタッチ方式データ分散[1]

OS が、各ページを最初にページフォールトを起こしたノードに割り付ける。

これら従来のデータ分散方法では、最適なデータ分散が実現できない場合がある。次節でその例を示し、これを解決するファーストタッチ制御方法を提案する。

2．2　ファーストタッチ制御方法

2．2．1　基本アイディア

本節では、OS が行うファーストタッチ方式データ分散をコンパイラが制御することによるデータ分散方法を提案する。初めは、簡単のため再分散については考えない。この場合、最適な静的データ分散は、プログラム中の実行比率の高い部分、すなわちカーネルループでデータローカリティの高いデータ分散となる。

ファーストタッチ制御方法の具体的な内容を説明する。まず、コンパイラがカーネルループ中の参照パターンを再現するダミーループを生成してプログラムの先頭に挿入し、OS が行うファーストタッチ方式データ分散に従ってデータ分散させることにより、カーネルループ向けのデータ分散を実現させる。なお、以下ではデータ分散の対象となる配列を**ターゲット配列**と呼ぶ。

図１(a)に例題プログラム１を示す。本プログラムは、初期化ループにおいて配列 A の全要素を参照するが、カーネルループでは部分配列 A(41:100)のみを参照する。このようなタイプのプログラムは Fortran プログラムでは比較的良く現れる。実プログラム上では手続き呼び出しの際に部分配列を引き渡すことにより、意味的に本例のようなプログラムとなることが多い。例では簡単のため手続き内のソースイメージで説明する。

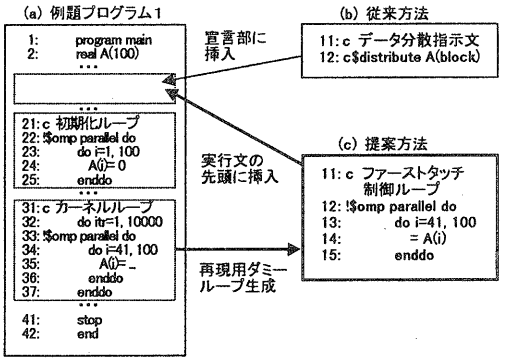

図１　ファーストタッチ制御データ分散方法

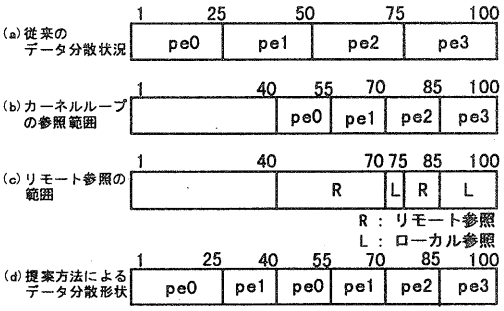

図２　データ分散状況

従来、図１(b)に示すようなデータ分散指示文を宣言部に挿入してデータ分散を行っていた[4]。この方法によると、データは全要素が図２(a)に示すように各プロセッサに均等に割り付けられる。ところが、ループ分散方法を block 分散と仮定すると、カーネルループにおける参照範囲は図２(b)に示すように A(41:100)を均等に各プロセッサに割り付けた状態となる。したがって、図２(a)と図２(b)の差である図２(c)に示す範囲、実にカーネルループの全参照における６７％がリモート参照となる。これではデータローカリティの低さから十分な並列性能を得ることができない。もう１つの従来方法として、プログラム変更を行わず、OS の行うファーストタッチ方式データ分散に従う方法があるが、初期化ループが全要素を参照するため、先の

方法と同様に図２(a)に示すようなデータ分散となり、データローカリティに関して同じ結果となる。

そこで、カーネルループ中におけるターゲット配列 A の参照パターンを再現する図１(c)に示すようなダミーループを生成する。以後、このループを**ファーストタッチ制御ループ**と呼ぶ。次に、ファーストタッチ制御ループを実行文の先頭に挿入し、OS が行うファーストタッチ方式データ分散に従って各要素を各プロセッサに割り付ける。その結果、図２(d)に示すようなデータ分散が実現でき、カーネルループにおける参照は１００%ローカル参照となる。

また、本方法は以下のような場合にも有効である。実プログラムにおいては、配列を引数として手続き間で渡していくことが多い。引数配列に指示文で静的なデータ分散を指示する場合、引数配列が出現する最親手続きで指示する必要がある。ところが、引数配列の宣言形状が引き渡されていく各手続きで異なる場合、最適データ分散はカーネルループを含む手続きにおける宣言形状をベースに表現されるため、最親手続きでそれを指示することができない。このような場合にも、ファーストタッチ制御方法によれば、手続き呼び出し関係ごとカーネルループの参照パターンを再現するコードを生成し、プログラムの先頭で実行させ、OS の行うファーストタッチ方式に従ってデータ分散させることにより、最適データ分散が実現できる。

２．２．２　本方法の拡張

以下に示すようなパラメータが変数である場合、パラメータの値が確定する位置によってファーストタッチ制御ループの挿入位置を変える必要がある。

・ターゲット配列の宣言寸法
・カーネルループのループ制御変数の上限、下限、増分
・ターゲット配列の添字式に含まれるループ制御変数以外の変数

パラメータが定数の場合は、単純に実行文の先頭にファーストタッチ制御ループを挿入すれば良いが、パラメータが変数の場合は、パラメータが確定した後にファーストタッチ制御ループを挿入しなければならない。さらに、パラメータの確定位置が初期化ループの前か後ろかで対処の方法が異なる。パラメータの確定位置が初期化ループの前の場合、パラメータ確定位置の直後にファーストタッチ制御ループを挿入する。パラメータの確定位置が初期化ループの後の場合、図３に示すような変換を行う。

まず、ターゲット配列と同じ宣言形状の配列（以下、**クローン配列**）を生成し、初期化ループ中のターゲット配列の参照をクローン配列にリネイムする。次に、パラメータ確定の直後にターゲット配列に対するファーストタッチ制御ループ、及びクローン配列からターゲット配列への全要素コピーを挿入する。以上のような方法により、パラメータが変数であった場合にも本ファーストタッチ制御データ分散方法の適用が可能となる。

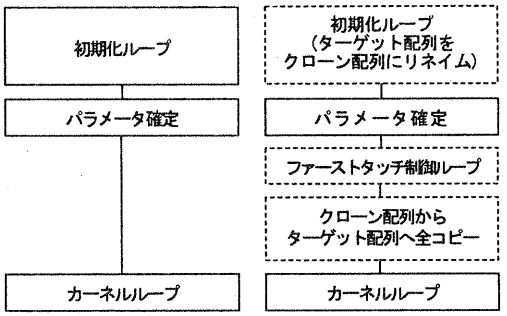

図３　パラメータ確定位置が初期化ループより後の場合の変換方法

さらに、データ再分散への適用を考える。配列参照パターンの異なる複数のカーネルループが存在して、途中で動的にデータ再分散を行いたい場合、以下の変換を行う。

・ターゲット配列のクローン配列を生成し、クローン配列を再分散後のデータ分散形状にデータ分散させるファーストタッチ制御ループを生成して実行文の先頭に挿入する。
・データ再分散位置でターゲット配列の全要素の値をクローン配列にコピーする。
・データ再分散位置以降のターゲット配列をクローン配列にリネイムする。

このように変換したプログラムを、OS が行うファーストタッチ方式データ分散に従ってデータ分散させることにより、データ再分散と同等の効果を得ることができる。

３．ＤＳＭ向け自動データ分散方法

本章では、２章で示したファーストタッチ制御方法、データ再分散解析、手続き間解析を用いた DSM 向け自動データ分散方法について述べる。

３．１　システム構成

図４にシステム構成を示す。本方法は、DSM 並列化トランスレータのデータ分散部として実装中である。本トランスレータは、Fortran77 で記述された逐次ソースプログラムを入力し、これに処理分散指示文、およびデータ分散指示文を挿入したソースプログラ

ムを出力する。さらにDSM並列化コンパイラが、このソースプログラムを入力し、DSM並列オブジェクトプログラムを出力してDSM並列実行を実現する。現在、DSM並列化コンパイラとしてはSGI/Origin2000に搭載されたMIPSPro Fortran77コンパイラを用いている。これに伴い、トランスレータが挿入する処理分散指示文としてはOpenMP指示文を用い、データ分散指示文としてはOrigin2000向けのSGI指示文を用いている。

データ分散部では、まず並列化ループ内に参照を有するターゲット配列を検出し、ターゲット配列毎のカーネルループを決定する。次にデータ再分散解析を行い、データ分散実施方法の判定を行う。このとき、ファーストタッチ制御データ分散を選択したターゲット配列に対し、ファーストタッチ制御コード生成用情報の解析を行う。最後に、ターゲット配列毎のデータ分散実施コード生成を行う。

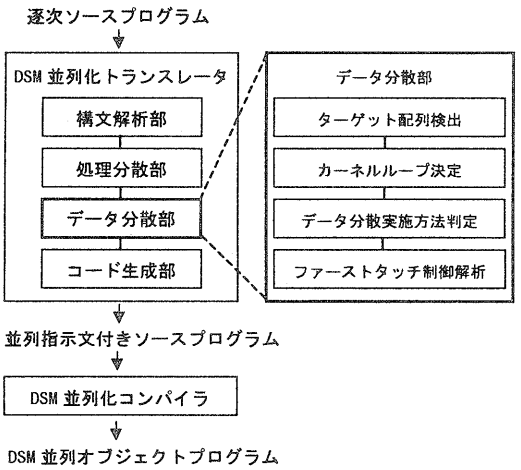

図4　システム構成

3．2　方式説明

一般にプログラム中の部分毎に最適なデータ分散は異なる。本研究で提案する自動データ分散方法では、以下の方針で設計した。まず、プログラム全体でデータ分散を固定した場合に最適となるデータ分散（静的データ分散）を決定し、初期データ分散とする。次に、データ再分散解析を行い、プログラム中からデータ再分散の必要があり、再分散コストに見合う部分を検出し、その部分に対してデータ再分散（動的データ分散）を実施する。

図5にDSM向け自動データ分散方法のアルゴリズムを示す。

本方法では、主に以下に示す6つの処理を行う。

・ターゲット配列の検出
・各並列化実施ループ向けデータ分散の決定
・カーネルループの決定
・再分散解析
・データ分散実施方法の判定
・ファーストタッチ制御向け解析

```
1:for（各ループ） do
2:   ・手続き間並列化実施ループの検出
3:   ・ターゲット配列の検出
4:endfor
5:for（各ターゲット配列） do
6:   ・並列化実施ループ向けデータ分散形状の決定
7:   ・手続き内カーネルループの決定
8:   if（ターゲット配列が引数配列） then
9:      ・手続き間カーネルループの決定
10:  endif
11:  ・データ再分散解析
12:  ・データ分散実施方法の判定
13:  if（ファーストタッチ制御データ分散） then
14:     ・カーネルループ向け参照パターンの検出
15:     ・カーネルループ向け代表参照パターンの決定
16:     ・ファーストタッチ制御コード再生用ループ構造
        情報の検出
17:  endif
18:endfor
```

図5　DSM向け自動データ分散方法のアルゴリズム

各処理の詳細を以下に示す。

(1)ターゲット配列の検出

まず、プログラム中の全ループの中から並列化実施ループ決定方法[5]に従って決定した手続き間並列化実施ループを検出する。次に、並列化実施ループのループ本体に参照を有する配列の中から、以下の条件を満たすものをターゲット配列として検出する。

・参照における添字式に並列化実施ループのループ制御変数を含む。

・ループ制御変数は、参照において1つの次元のみ、かつ1回のみ現れる。

・参照における添字式がA*i+Bである（A,Bは定数、iはループ制御変数）。

(2)各並列化実施ループ向けデータ分散の決定

ターゲット配列毎に当配列の参照をループ本体に含む並列化実施ループを検出し、各ループ向けデータ分散を決定する。ループ向けデータ分散は、参照において並列化実施ループのループ制御変数を添字式に含む次元をBLOCK分散した形状とする。

(3)カーネルループの決定

ターゲット配列毎のカーネルループを決定する。まず、データ分散毎に並列化実施ループをグループ化し、各並列化実施ループのコストを見積ってコストの合計が最大となるグループを検出する。なお、ループ本体の演算数とループ長の積をループのコストと呼ぶ。このグループの中でコスト最大の並列化実施ループを当ターゲット配列の手続き内カーネルループとす

る。次に、ターゲット配列が引数の場合、ターゲット配列が渡る各手続きにおける手続き内カーネルループを検出する。データ分散毎に手続き内カーネルループをグループ分けし、コストの合計が最大となるグループを検出する。このグループの中でコスト最大の手続き内カーネルループを手続き間カーネルループとする。

(4)再分散解析

まず、手続きローカル配列であるターゲット配列に対して以下の処理を行う。手続き内カーネルループ以外の並列化ループの内、(2)で求めたデータ分散が手続き内カーネルループと異なり、コストがデータ再分散しきい値以上のループを検出し、データ再分散対象ループとする。次に、引数であるターゲット配列に対して以下の処理を行う。手続き内カーネルループの内、手続き間カーネルループとデータ分散が異なり、コストがデータ再分散しきい値以上のループを検出し、このループを含む手続きをデータ再分散対象手続きとする。

(5)データ分散実施方法の判定

各ターゲット配列のデータ分散実施方法を判定する。まず、ターゲット配列が以下の条件を満たす場合ファーストタッチ制御による方法を選択する。

・カーネルループにおいて部分配列参照が存在する。
・引数配列であり、手続き毎に宣言形状が異なる。

その他の場合、データ分散指示文による方法を選択する。

(6)ファーストタッチ制御向け解析

ターゲット配列がファーストタッチ制御によるデータ分散方法を選択した場合、以下の解析を行う。

・配列の参照における全次元の添字式の集合を参照パターンと定義し、ターゲット配列のカーネルループにおける全参照パターンを検出する。
・全参照パターンの内、最も出現回数の多い参照パターンを検出し、代表参照パターンとする。
・代表参照パターンの添字式に含まれる全てのループ制御変数の上限、下限、増分をループ再現用解析情報として保持する。ターゲット配列が引数の場合、ターゲット配列が現れる最親手続きからカーネルループを含む手続きまでの手続き呼び出し関係を同じく保持する。保持された情報は、ファーストタッチ制御コード生成において、ループを再現するために利用される。

4．評価

本研究で提案するDSM向け自動データ分散方法の効果を事前評価するため、NPB2.3serialのFTに人手で本方法を適用してプログラム変換し、SGI/Origin2000を用いて測定した。本章では、その結果を述べる。

4．1　評価プログラム

FTは、3次元FFTに関するプログラムである。NPB2.3serialは、様々な並列機向けの並列プログラムのベースとして用いるための逐次プログラムである。本評価では、このプログラムを入力として用いる。まず、図6にFTのカーネルループを示す。このループは、ft(main) - fft - cffts1の順で呼び出された手続きcffts1の中にある。ターゲット配列は下線部で示したx、xoutの3次元配列であるが、上位の手続きにおいては、各々

・ft(main)手続きではu1、u2
・fft手続きではx1、x2

という別名で1次元配列として宣言されている。

```
 1:  complex*16 x(nx, ny, nz), xout(nx, ny, nz)
 2:  complex*16 y(nblock, nz, 2)
 3:  do k= 1, nz
 4:    do jj= 0, ny - nblock, nblock
 5:      do j= 1, nblock
 6:        do i= 1, nx
 7:          y(j, i, 1)= x(i, j+jj, k)
 8:        enddo
 9:      enddo
10:      call cfftz(y)
11:      do j= 1, nblock
12:        do i=1, nx
13:          xout(i, j+jj, k)= y(j, i, 1)
14:        enddo
15:      enddo
16:    enddo
17:  enddo
```

図6　NPB2.3serial/FTのカーネルループ

```
 1:  subroutine ftc_fft(x1, x2)
 2:  complex*16 x1(256*256*128), x2(256*256*128)
 3:  call ftc_cffts1(x1, x2)
 4:  return
 5:  end
 6:  subroutine ftc_cffts1(x, xout)
 7:  complex*16 x(256,256,128), xout(256,256,128)
 8:  !$omp parallel do
 9:  do k= 1, 128
10:    do jj= 0, 240, 16
11:      do j= 1, 16
12:        do i= 1, 256
13:          x(i, j+jj, k)= 0
14:          xout(j, j+jj, k)= 0
15:        enddo
16:      enddo
17:    enddo
18:  enddo
19:  return
20:  end
```

図7　ファーストタッチ制御コード

並列化実施ループ決定方法[5]に従い、このカーネルループを最外ループ(do k)で並列化する。したがって、ターゲット配列x、xoutを3次元目でblock分散させるようなデータ分散できればデータローカリティが最も高くなるが、上位手続きでの宣言形状が異なるので、前述の通りデータ分散指示文でこれを実現する手段はない。そこで、本方法では、図7に示すようなファーストタッチ制御コードを生成し、手続き呼び出し文"call ftc_fft(u1, u2)"をft(main)手続きの実行文の先頭に挿入し、OSの行うファーストタッチ方式データ分散に従ってデータ分散させる。これにより、ターゲット配列x、xoutを3次元目でblock分散させた形状

のデータ分散が実現でき、データローカリティを向上させることができる。なお、配列宣言寸法、ループ制御変数の上限、下限、増分は手続き間定数伝播により定数に置換されたものを利用している。

4．2　測定結果

FT に本自動データ分散方法を適用した場合と、適用しない場合の性能向上比を図８に示す。なお、測定条件は表１に示す。

表１　測定条件

マシン	SGI/Origin2000
ノード	16ノード(2cpu/ノード)
CPU	MIPS RISC R10000(195MHz)
キャッシュ	L1:32KB, L2:4MB
メモリ	11GB(11264MB)
OS	IRIX6.5
コンパイラ	MIPSPro Fortran77(Version7.2.1)
コンパイルオプション	-mp -Ofast=IP27 -OPT:IEEE_arith=3

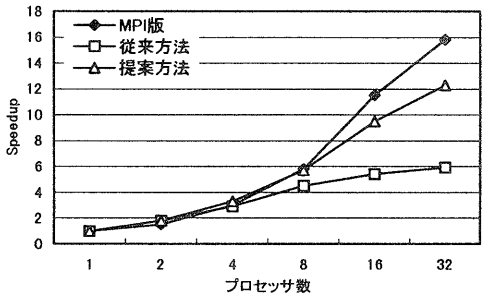

図８　NPB2.3serial/FT(classA)の性能向上比

評価では、以下に示す３つのバージョンの測定を行った。

(1)MPI 版プログラム

(2)従来方法：自動データ分散を行わないプログラム (OS ファーストタッチ方式データ分散)

(3)提案方法：自動データ分散を行うプログラム (コンパイラファーストタッチ制御データ分散)

自動データ分散を行わない場合は、プロセッサ数の増大と共にデータローカリティ低下の影響が大きくなり性能の伸びを妨げた。一方、自動データ分散を行った場合は、高いデータローカリティによって全てのプロセッサ数の範囲で良好な並列性能を示し、１６プロセッサ時では従来方法に比べて７６％の性能向上を示した。

なお、１６プロセッサ以上の範囲における MPI 版との性能差は、データ再分散によるデータローカリティの差の影響が大きいと考えられる。FT はデータ再分散を実施するとデータローカリティが向上することが分かっているが、SGI/Origin2000 では OS が行うデータ再分散が低速であるため、提案方法によるプログラムでは再分散を行っていない。実プログラムにおいて、再分散によりデータローカリティが大幅に改善されるプログラムは多い。したがって、DSM にとってデータ再分散機能の性能向上は重要であると考えられる。

また、以上の結果から自動データ分散機能を抜きにしては、DSM も SMP で良い性能を得ることができたプロセッサ数の範囲でしか通用しないことが明らかである。よって、DSM が SMP より大きなプロセッサ数で十分な並列性能を得るためには、高性能な自動データ分散機能が必須であると考えられる。

5．おわりに

本稿では、DSM 向け自動データ分散方法について述べた。この中で、従来のデータ分散方法では最適なデータ分散が実現できない幾つかの場合にも適用可能なファーストタッチ制御方法を提案した。また、ファーストタッチ制御方法、手続き間解析、データ再分散解析等により動的、静的に DSM 向けの最適データ分散を実現する自動データ分散方法を提案した。事前評価として、ベンチマークプログラム NPB2.3serial の FT を人手変換して本自動データ分散方法によるデータ分散実施コードを挿入し、SGI/Origin2000 上で性能測定を行った。その結果、１６プロセッサ時で自動データ分散を行わない場合に比べて約７６％性能が向上することを確認した。

今後は、本自動データ分散方法の早期実装、及び評価を行い、多くのベンチマークへ適用してその有効性を確認したい。また、そこから得た情報を基に機能拡張、及び性能向上に取り組んで行きたい。

参考文献

1) R.Chandra, D.Chen, R.Cox, D.E.Maydan, N.Nedeljkovic, J.Anderson, "Data Distribution Support on Distributesd Shared Memory Multiprocessors", Silicon Graphics Computer Systems, Digital Western Research Lab,PLDI'97 (1997).

2) T.N.Nguyan, Z.Li, "Interprocedural Analysis for Loop Scheduling and Data Allocation", Parallel Computing, Vol.24, No3-4, pp.477-504, (1998).

3) K.Kennedy, U.Kremer, "Automatic Data Layout for High Performance Fortran", Proc. of Supercomputing'95 (1995).

4) SGI MIPSpro Fortran77 Programmer's Guide, Silicon Graphics Inc

5) 飯塚, 佐藤, 蓮見, 菊池, "手続き間並列化コンパイラ WPP の試作 － 現状と今後の課題 －", 情報処理学会第５６回全国大会, (1998).

6) 廣岡, 太田, 菊池, "配列リシェイプを用いた分散共有メモリ向けデータ再分散の最適化", 情報処理学会第５７回全国大会, (1998).

*) Origin2000 は、Silicon Graphics, Inc.の商標である。