計算機アーキテクチャ　141－5
（2001．1．26）

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

廣岡孝志*,**　太田寛*,***　飯塚孝好*,**　菊池純男*,**
*アドバンスト並列化コンパイラ研究体
**日立製作所システム開発研究所
***日立製作所情報コンピュータグループ

分散共有メモリ向けコンパイラにおける手続き間自動データ分散方法の実装を行った．データ分散方法としては，「ファーストタッチ制御(FTC)方法」とデータ分散指示文を併用する．FTC方法の特徴は，コンパイラがOSのファーストタッチ方式データ分散を制御することで，複雑なデータ分散に適確に対応できることである．これらの併用により，従来のデータ分散方法が不得手とするプログラムパターンに対し，最適なデータ分散が実現可能となる．これに手続き間解析機能を搭載し，プログラム全体の解析結果に基づくデータローカリティ最適化を実現した．SGI/Origin2000 を用いた評価の結果，ベンチマークプログラム NPB2.3serial/FT，SP，CG，SPECfp95/tomcatv の 4 題について，本分散方法を行わない場合に比べて 16 プロセッサ時で平均 2.1 倍に性能が向上することを確認した．さらに，CG 中の間接参照配列に対して人手でファーストタッチ制御方法を適用することにより，本分散方法を行わない場合に比べて 6.0 倍，MPI プログラムに比べて 1.2 倍に性能が向上することを確認した．

Automatic Data Distribution Method for Distributed Shared Memory Compiler

Takashi HIROOKA*,**, Hiroshi OHTA*,***, Takayoshi IITSUKA*,**
and Sumio KIKUCHI*,**
*Advanced Parallelizing Compiler Project
**Systems Development Laboratory, Hitachi,Ltd.
***Information & Compuer Systems, Hitachi,Ltd.

We implemented an interprocedural automatic data distribution method for our distributed shared memory compiler. This method combines the "First Touch Control(FTC)" with data distribution directives. The characteristics of FTC is that our compiler controls first touch data distribution of the operating system and accurately determines complex data distributions. By this combined method, we can achieve appropriate data distributions for program patterns which conventional data distribution methods can't treat properly. In addition we implemented interprocedural analyses which improves data locality of the whole program. We evaluated NPB2.3serial/FT,SP,CG and SPECfp95/tomcatv on SGI/Origin2000. These benchmarks ran faster by 2.1 times(average) than those without our method in the case of 16 processors. Furthermore, we applied FTC to indirect array references of CG by hand and it ran faster by 6.0 times than that without our method and by 1.2 times than MPI program.

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．並列計算機を効率良く利用するためには，並列化率，ロードバランス，データローカリティが重要となる．本研究では，データローカリティの最適化に着目した．共有メモリモデルをサポートするが物理的に分散されたメモリを有するDSMでは，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と考えられる．そこで，前報告[7]では最適なデータ分散形状をコンパイラで自動的に決定する方法を提案した．本報告では，それを実装，評価した内容について述べる．

従来，物理分散メモリを有する並列計算機に対するデータ分散方法として，データ分散指示文によるユーザ指示[1]，OS が行うファーストタッチ方式データ分散[1]などが実用化されてきた．ところが，従来のデータ分散方法では，実プログラム中に比較的良く表れる部分配列参照（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を指す）などのプログラムパターンに対して，容易に最適なデータ分散を実現できない場合があった．この問題を解決するため，前報告[7]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で制御するファーストタッチ制御方法を提案した．本報告では，このファーストタッチ制御方法を適用した自動データ分散方法の実装，およびベンチマー

クプログラム NPB2.3serial[8]/FT,SP,CG,SPECfp95[9]/tomcatvを用いた性能評価について述べる．本研究で提案，実装した自動データ分散方法は，ファーストタッチ制御方法とデータ分散指示文を併用し，従来のデータ分散指示文では適切な指示が困難な部分配列参照などに対しても最適なデータ分散を実現し，幅広いプログラムで良好な並列性能を得られる点に特徴がある．加えて，手続き間解析機能の搭載によりプログラム全体の解析結果に基づくデータローカリティ最適化を実現した．

本稿の構成は以下の通りである．2章では，分散共有メモリ機構について説明した後，実装したDSM自動並列化コンパイラの構成を述べる．3章では，前報告[7]で提案したDSM向け手続き間自動データ分散方法の概要を述べ，4章で評価結果を分析する．5章で関連研究を紹介し，6章で結論を述べる．

2．システム構成

DSMを実現する手段の主流として，仮想メモリ空間をページ単位で切り分けて各ノードの物理メモリに割り付ける方式がある．今回，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もこの方式を採用している．本DSM方式において，ページをどのノードの物理メモリに割り付けるかを決める方法をデータ分散方法という．Origin2000には，自ノードに割り付けられたデータの参照の割合，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]．

(1)データ分散指示文によるデータ分散

プログラム中にユーザが挿入したデータ分散指示文に従って，コンパイラが各ノードにデータを割り付ける．

(2)ファーストタッチ方式データ分散

OSが，各ページを最初にアクセスしたノードの物理メモリに割り付ける．

上記従来方法では，手続き毎に引数配列の宣言形状が異なる場合や，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に，最適なデータ分散が実現できないという問題点があった．前報告[7]では，この問題を解決するファーストタッチ制御方法を提案した．これは，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である．

図1を用いて説明する．図1(a)に示した例題プログラム1の場合，従来方法では図1(b)に示すようなデータ分散指示文をプログラムの先頭に挿入していた．提案方法では，コンパイラが図1(c)に示すような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．

次に，図2に提案するDSM向け手続き間自動データ分散方法を実装したコンパイラの構成を示す．本方法は，既存のSMP向け自動並列化コンパイラWPP[6]にデータ分散部を挿入することで実現した．以後，本コンパイラをDSM自動並列化コンパイラと呼ぶ．DSM自動並列化コンパイラは，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（OpenMP指示文），およびデータ分散実施コードを挿入したソースプログラムを出力

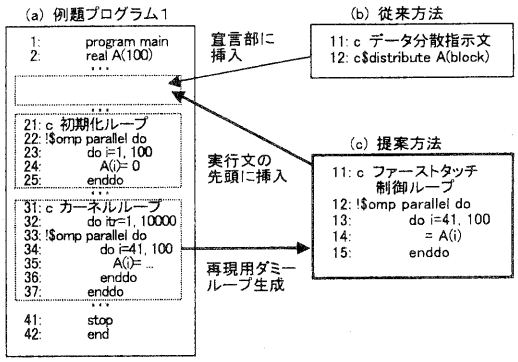

図1 ファーストタッチ制御方法

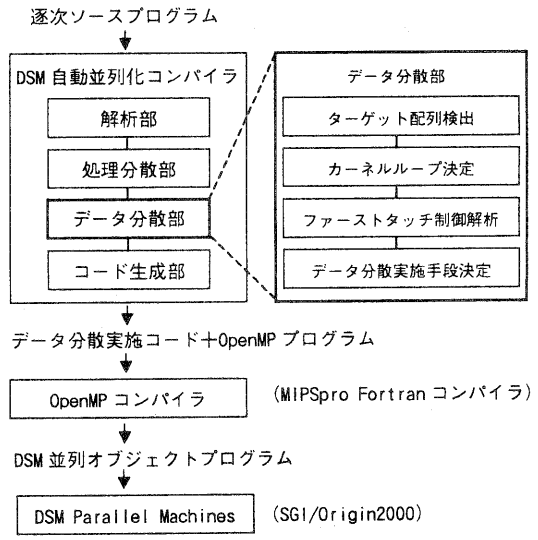

図2 DSM自動並列化コンパイラの構成

する．さらに，SGIがDSM向けに独自にデータ分散指示文をサポートしたOpenMPコンパイラが，このソースプログラムを入力としてDSM並列オブジェクトプログラムを出力し，並列実行を実現する．DSM型並列計算機としてはOrigin2000を用い，OpenMPコンパイラとしてはOrigin2000に搭載されたMIPSpro Fortran90コンパイラを用いた．

データ分散部では，まず処理分散部[6]で決定した並列化ループを検出し，これらのループ本体に参照を有する配列（以後，**ターゲット配列**と呼ぶ）を検出する．次にターゲット配列毎に各並列化ループ向けデータ分散形状を決定し，プログラム中で最も実行頻度が高いカーネルループ，およびターゲット配列のデータ分散形状を決定する．また，このときデータ再分散解析を行い，ターゲット配列のデータ再分散形状を決定する．次に提案方法の特徴技術の一つであるファーストタッチ制御データ分散を適用するために必要な解析を行い，最後にデータ分散実施手段の決定を行なってコード生成を行う．

3．DSM向け手続き間自動データ分散方法

分散メモリ機構を有する並列計算機の場合，各データは，そのデータが初めて参照されるまでに，何らかの分散形状で各ノードの物理メモリに配置されている必要がある．したがって，手続きに跨って参照される配列のデータ分散について考えると，その配列が初めて参照される手続きで決定したデータ分散形状をプログラム全体で利用し続けるか，必要な場面でデータ再分散を行うしかない．その結果，マシンの特性によっては，最初に行ったデータ分散の形状，およびデータ再分散コストが，性能に大きく影響を及ぼすことになる．

そこで，本自動データ分散方法では，手続き間解析によりプログラム全体でデータ分散形状を固定させた場合に最適となるデータ分散形状を決定して初期分散させ，データ再分散解析の結果を基にコストに見合う場合にはデータ再分散を実施してプログラム全体のデータローカリティをより最適化させる手法をとる．

4．評価

4.1 測定環境

性能測定は，SGI/Origin2000 上で行った．測定条件を表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	IRIX6.5.4
コンパイラ	MIPSpro Fortran90(Version7.3)
コンパイルオプション	-mp -64 -Ofast=IP27 -OPT:IEEE_arith=3

4.2 DSM自動並列化コンパイラの評価

評価には，SPEC Benchmark の SPECfp95/tomcatv，および NASA提供の標準的ベンチマーク NPB2.3serial/FT, SP, CG の4題を用いた．各ベンチマークの性能測定結果を図3から図6に示す．グラフは，横軸にプロセッサ数，縦軸に性能向上比を示し，コンパイラによる自動データ分散を行った場合(dist)と行わない場合(no dist)の性能を比較したものである．なお，no dist では，OSが行うファーストタッチ方式データ分散に従う．

(1) FT

FTは，カーネルループと主要配列が比較的限定されている．本自動データ分散方法により，主要配列の多くが自動データ分散のターゲットとして検出され，各配列に対しループ分散に合わせたデータ分散形状が求められた．データ分散実施手段としては，主要配列の宣言形状が手続き毎に異なることから，4題のベンチマークで唯一ファーストタッチ制御方法(FTC)が選択された．

性能向上比を図3に示す．各クラス(データサイズ)共，コンパイラによる自動データ分散を行わない場合，プロセッサ数の増大と共にデータローカリティが低下するため，十分なスケーラビリティを得ることができなかった．一方，自動データ分散機能を適用した場合，最適なデータ分散が実施され，データローカリティが改善されて大幅な性能向上を実現した．classB, 16プロセッサ時で89%性能が向上した．なお，このグラフにはファーストタッチ制御コードによるオーバヘッドも含まれている．

FTの場合，データ再分散により大きくデータローカリティが改善される部分がある．しかし，本方法ではコスト計算の結果，データ再分散は実施されなかった．これは，Origin2000 のデータ再分散コストが大きいためである．データ再分散のコストが小さくなり，データ再分散を実施してデータローカリティの改善が実現できれば，さらにスケーラビリティが向上すると考えられる．4.4節でデータ再分散の効果を評価する

(2) SP

性能向上比を図4に示す．SPは，コンパイラによる自動データ分散を実施しない場合でも比較的良好な並列性能を示していた．しかし，本自動データ分散方法により，さらに性能を向上させることができ，classB, 16 プロセッサ時で12%の性能向上を実現した．

(3) CG

CGは，カーネルループと主要配列が限定されている．また，ロードバランスが良いためデータローカリティに問題がなければ良好なスケーラビリティを得ることができると予想される．本自動データ分散方法により，主要配列3つ全てがターゲットとして検出された．

性能向上比を図5に示す．CGは，classAでは主要配列のデータが各プロセッサのキャッシュに載り，リモート参照が少ないため，コンパイラによる自動データ分散機能の有無に関係なく良好な並列性能を示した．しかし，classB以降の大きなデータサイズでは，コンパイラによる自動データ分散機能を適用しない場合，リモート参照の割合が増大して大きくスケーラビリティを悪化させている．コンパイラによる自動データ分散を適用した場合，大幅にスケーラビリティが改善し，classB, 16プロセッサ時で4.1倍に性能が向上した．さらに，間接参照配列にファーストタッチ制御方法を適用した場合の効果を予備評価した．結果を4.5節で詳細に報告する．

(4) tomcatv

性能向上比を図6に示す．tomcatvでも，自動データ分散機能により，データローカリティが改善され，16 プロセッサ時で20%性能が向上した．

4.3 ファーストタッチ制御コードのオーバーヘッド

データ分散実施手段としてファーストタッチ制御方法が選択されたFT，classAを用いてファーストタッチ制御コードのオーバーヘッドを測定した．表2に，プロセッサ数ごとの測定結果を示す．noopt_OH に，現状コンパイラのファーストタッチ制御コードの実行時間(msec)を示し，noopt_OH/KER に，プログラム中のカーネル部分に対する比率(%)を示す．各プロセッサ数において，概ね数%で推移している．これはデータローカリティ改善による性能向上の効果に比べて十分小さい．

ここで，さらにオーバーヘッドを削減する最適化を考える．ファーストタッチ制御ループでは，各ページを最適なノードに割り付けるため，ターゲット配列の全要素を参照させている．理論的には，1ページにつき1要素参照させ

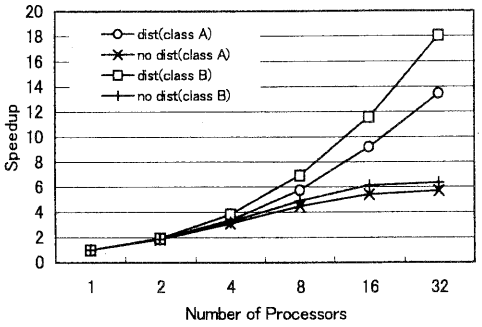

図 3 FT/ NPB2.3serial

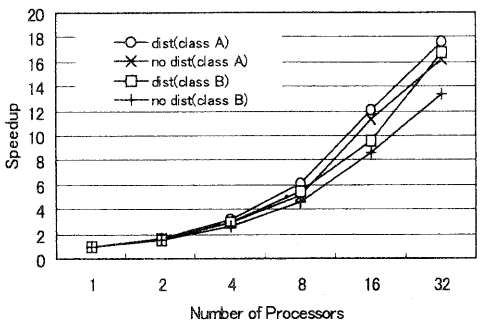

図 4 SP/ NPB2.3serial

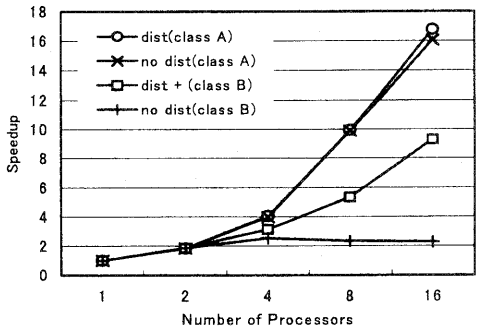

図 5 CG/ NPB2.3serial

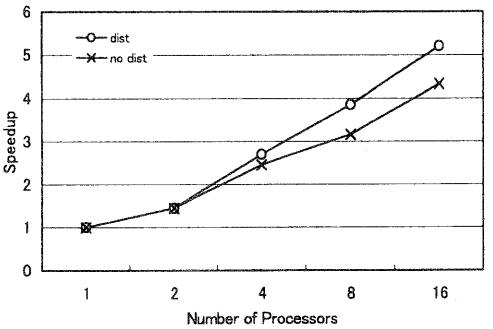

図 6 tomcatv/SPECfp95

れば良いはずなので，ファーストタッチ制御ループの繰り返し範囲を 1 ページにつき 1 回に変更する．Origin2000 の場合，デフォルトのページサイズは 16 キロバイトなので，単純に計算しても 1 要素のデータ長が 8 バイト場合，参照回数が二千分の一に削減できる．手動でこの最適化を施したファーストタッチ制御コードの実行時間を表 2 の opt_OH(msec)に示し，カーネル部分に対する比率(%)を opt_OH/KER に示す．元々ループ本体の演算量が少なく，スレッド起動オーバーヘッドが目立つ構造であったので二千分の一にはならないが，本最適化により，数十分の一から百分の一程度にオーバーヘッドを削減することができた．これで，オーバーヘッドの問題は無視できる程十分小さくなる．

4.4 データ再分散の評価

FT は，データ再分散を行うことによりデータローカリティを更に向上させることができる．そこで，FT を用いてデータ再分散の効果を評価する．評価のため，ソースプログラムに手動で以下の修正を施した．

- 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に最適となる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，実行文の先頭に挿入する．
- データ再分散区間の直前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．
- データ再分散区間内のターゲット配列をクローン配列にリネイムする．

表 2 FT(classA)の FTC オーバーヘッド

プロセッサ数	1	2	4	8	16	32
noopt_OH(msec)	2522	1758	1256	757	249	209
noopt_OH/KER(%)	2.11	2.71	3.50	3.62	1.93	2.35
opt_OH(msec)	26	19	9	5	5	16
opt_OH/KER(%)	0.02	0.03	0.03	0.02	0.04	0.18

OH：ファーストタッチ制御コードの実行時間
KER：カーネル部分の実行時間

- データ再分散区間の直後でクローン配列の全要素の値をターゲット配列にコピーする．

図 7 にデータ再分散を実施した場合の実行時間の比較を示す．縦軸に経過時間，横軸にプロセッサ数を示し，no redist にデータ再分散を行わない場合，redist にデータ再分散を行った場合の性能を示す．

グラフに示すように，全てのプロセッサ数の範囲でデータ再分散により性能が劣化した．データ再分散のオーバーヘッドがデータローカリティ改善の効果を上回ったためと考えられる．

4.5 間接参照配列へのファーストタッチ制御方法の適用

CG の場合，ソースプログラムの分析の結果，主要な配列の中に間接参照を有する配列を含むことが分かった．しかし，現在の実装では，間接参照配列に対するファーストタッチ制御によるデータ分散機能が未サポートである．ここでは，間接参照のように複雑なアクセスパターンになる

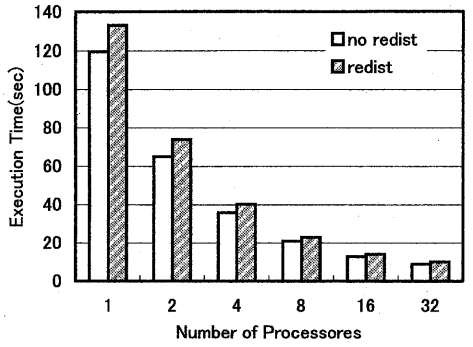

図7 FT(classA)の実行時間

能性がある場合，最適なデータ分散形状自体が複雑になる可能性があると考え，手動でファーストタッチ制御方法を適用して評価を行った．具体的には，ソースプログラムに対し，以下の人手変換を施した．

・ターゲット配列のクローン配列を生成し，パラメータ（ループ繰り返し範囲等）確定位置より前のターゲット配列をクローン配列にリネイムする．
・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．
・ファーストタッチ制御ループの直後でクローン配列の全要素の値をターゲット配列にコピーする．

CG のソースプログラム中の一部を切り出し，簡素化した変換例を図 8 に示す．本例は，配列 a がターゲット配列である場合の変換例である．CG では，ターゲット配列 a のカーネルループでループ繰り返し範囲に配列 rowstr が用いられる．したがって，配列 rowstr の各要素の値が決定するまでファーストタッチ制御ループを実行することができない．配列 rowstr の値は，手続き sparse の 14 行目までに確定する．よって，まず 3 行目で配列 a と同じ宣言形状のクローン配列 a_ftc を宣言する．次に，14 行目以前の配列 a の参照をすべてクローン配列に置換する（9 行目）．次に，14 行目の直後にカーネルループの参照パターンを再現するファーストタッチ制御ループを挿入する（16 行目から 23 行目まで）．続いて，その直後にクローン配列の全要素の値をターゲット配列にコピーするループを挿入する（24 行目から 29 行目）．このプログラムを OS の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並列実行し，性能を測定した．結果を図 9 の dist + indirect(FTC)に示す．

グラフに示すように，データローカリティの改善により，大幅にスケーラビリティが向上し，MPI プログラムを上回る性能が得られた．CG, classB, 16 プロセッサ時で，コンパイラによる自動データ分散を行わない場合に比べて 6.0 倍，指示文版に比べて 1.5 倍，MPI プログラムに比べて 1.2 倍の性能向上を示した．

特に，MPI プログラムを上回るスケーラビリティを得たことは興味深い．MPI プログラムにおいて，ある程度のプログラム開発工数で実現可能なデータ分散形状は，比較的

```
 1:       subroutine sparse( a, ..... )
                ...
 2: c FTC double precision   a(1)
 3:       double precision   a(1),a_ftc(15825000)
                ...
 4:         do k = 1, nzrow
                ...
 5:           xi = x(i)
 6:           if (xi .ne. 0.D0) then
 7:             nza = nza + 1
 8: c FTC       a(nza) = xi
 9:             a_ftc(nza) = xi
                ...
10:           endif
11:         enddo
12:         jajp1 = rowstr(j+1)
13:         rowstr(j+1) = nza + rowstr(1)
14:       enddo
15: c FTC code start
16: !$omp parallel
17: !$omp do schedule(static)
18:       do j=1,75000
19:         do k=rowstr(j),rowstr(j+1)-1
20:           a(k)=0
21:         enddo
22:       enddo
23: !$omp end do nowait
24: !$omp do schedule(static)
25:       do j=1,15825000
26:         a(j)= a_ftc(j)
27:       enddo
28: !$omp end do nowait
29: !$omp end parallel
30: c FTC code end
                ...
31:       return
```

図8 CG のファーストタッチ制御コード

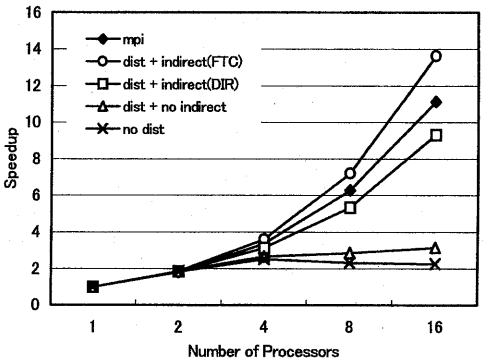

図9 CG(classB)の性能向上比

単純な block 分散，cyclic 分散，block-cyclic 分散ぐらいまでである．最適データ分散が上記のような場合は，MPI プログラムで良いスケーラビリティが得られてきた．また，データ分散指示文を用いた DSM 並列プログラムでも同様の傾向が見られた．ところが，カーネルループ中にターゲット配列の間接参照が存在する場合など，最適データ分散が複雑な形状になる可能性がある場合は，ファーストタッチ制御方法を用いた DSM 向け並列プログラムの方が，遥かに少ないプログラム開発工数で正確なデータ分散を実現でき，今まで最もスケーラビリティが良いと思われてきた MPI プログラムを上回る性能が得られることを示した．これにより，ファーストタッチ制御方法の有効性の一つを示すことができたと考える．また，分散メモリ型並列計算機では得ることが困難であったスケーラビリティを DSM

の特徴を生かして得たことは，DSM自体の優位性の一つを示したことになる．

表3にファーストタッチ制御コードのオーバーヘッドを示す．カーネル部分の実行時間に対して平均して0.5%前後であり，比率として十分小さい．今後は，適用条件に関する詳しい分析を行い，コンパイラへの実装を検討していく予定である．

表3 CG(classB)のFTCオーバーヘッド

プロセッサ数	1	2	4	8	16	32
OH(msec)	2704	2080	1116	718	536	436
OH/KER(%)	0.22	0.32	0.33	0.43	0.60	0.94

OH：ファーストタッチ制御コードの実行時間
KER：カーネル部分の実行時間

5．関連研究

データ分散形状を自動的に決定する方法は，これまでにも多数行われている．最適なデータ分散形状を決定する問題はNP完全であるため，Kennedyら[2]，Guptaら[3]，辰巳ら[4]，松浦ら[5]などが，近似解を求める様々なヒューリスティクを提案した．辰巳ら[4]は，単独のループを対象として，配列間の相対的な配置関係から分散次元とブロックサイズを決定する方法を示した．松浦ら[5]は，手続きに跨る複数ループを対象として，手続き間の配列アクセス情報を基に通信を最小化したデータ分散を決定する方法を示した．Guptaら[3]は，CC-NUMA向けの手続き間を対象としたループ，およびデータの分散方法を示した．いずれの方法も指示文を用いたデータ分散方法を想定しており，

・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
・手続きごとに引数配列の宣言形状が異なる場合
・カーネルループに間接参照が存在する場合

などに最適なデータ分散を実現する方法については論じられていない．提案方法では，これらが正確に実現可能となり，複数ループ，複数手続きに参照の跨る配列のデータ分散を自動決定することができる．

6．まとめ

本稿では，DSM向け手続き間自動データ分散方法の実装と評価について述べた．この中で，OSによるファーストタッチ方式データ分散とそのコンパイラ制御の組み合わせにより，従来より適切なデータ分散を実現できる可能性を示し，その有効性を検証した．また，上記の方法と従来利用されてきたデータ分散指示文を併用し，手続き間解析機能を搭載した自動データ分散方法を実装したことにより，従来のデータ分散方法では困難であった最適データ分散，およびプログラム全体で最適となるデータ分散が実現可能となった．

Origin2000を用いた評価の結果，今回開発したコンパイラにより，ベンチマークプログラムNPB2.3serial/FT，SP，CG，SPECfp95/tomcatvの4題で，コンパイラによる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，平均2.1倍に性能が向上することを確認した．また，FTを用いてファーストタッチ制御コードのオーバーヘッドを評価し，平均数%以下であることを確認し，データローカリティ改善の効果に比べて十分小さいことを示した．さらに，ファーストタッチ制御コードの最適化方法を示し，オーバーヘッドを数十分の一から百分の一に削減できる見込みを得た．次に，FTを用いてデータ再分散の効果を評価し，効果がないことを確認した．最後に，間接参照を有する配列に対し，ファーストタッチ制御方法を適用した場合，データローカリティの改善により，大幅にスケーラビリティが向上し，MPIプログラムを上回る性能が得られた．CG，classB，16プロセッサ時で，コンパイラによる自動データ分散を行わない場合に比べて6.0倍，指示文版に比べて1.5倍，MPIプログラムに比べて1.2倍に性能が向上した．

前報告[7]と今回の報告で，本方法の提案から実装，評価までをまとめた．今後は，自動データ分散方法全体としては，

・間接参照配列対応機能の実装．
・より多くのベンチマークによる評価を実施して有効性の検証を行い，性能向上，機能拡張へ反映させること．

ファーストタッチ制御方法については，

・有効なプログラムパターンの摘出．
・プログラムパターンごとの適用条件の検討．
・コンパイラへの実装範囲の拡大．

を行う予定である．また，アドバンスト並列化コンパイラ研究体において，本技術のソフトDSMを用いたPCクラスタへの適用を進める予定である．

謝辞 本研究は，新情報処理開発機構(RWCP)における成果を基に行ったものである．

参考文献

1) Chandra, R., Chen, D., Cox, R., Maydan, D.E., Nedeljkovic, N., Anderson, J.: Data Distribution Support on Distributesd Shared Memory Multiprocessors, Proc. PLDI' 97, pp.334-345 (1997).
2) Kennedy, K., Kremer, U.: Automatic Data Layout for High Performance Fortran, Proc. Supercomputing'95, (1995).
3) Gupta, M., Banerjee, P.: PARADIGM: A Compiler for Automatic Data Distribution on Multicomputers, Proc. ICS'93, pp.87-96 (1993).
4) 辰巳尚吾，窪田昌史，五島正裕，森眞一郎，中島浩，富田眞治：並列化コンパイラTINPARにおける自動データ分割部の実現，情報処理学会研究報告，96-PRO-8, pp.25-30 (1996).
5) 松浦健一郎，村井均，末廣謙二，妹尾義樹：データ並列プログラムに対する高速な自動データ分割手法，情報処理学会論文誌，Vol.41, No.5, pp.1420-1429 (2000).
6) 青木雄一郎，佐藤真琴，飯塚孝好，佐藤茂久，菊池純男：手続き間自動並列化コンパイラWPPの試作 － 実機性能評価 －，情報処理学会研究報告，98-ARC-130, pp.43-48 (1998).
7) 廣岡孝志，太田寛，菊池純男：ファーストタッチ制御による分散共有メモリ向け自動データ分散方法，情報処理学会論文誌，Vol.41, No.5, pp.1430-1438 (2000).
8) The NAS Parallel Benchmarks, http://www.nas.nasa.gov/Software/NPB/
9) SPEC Benchmarks, http://www.specbench.org/