4R-01

分散共有メモリ向け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

1. はじめに

分散共有メモリ(DSM)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,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ことにより,複雑な形状の最適データ分散にも少ない解析量で正確に対応できる「ファーストタッチ制御方法」を用いる.これとデータ分散指示文を併用し,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,最適なデータ分散を実現する点を特徴とする.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.本稿では,その実装と評価について報告する.

2. 分散共有メモリ向け自動データ分散方法

2.1 概要

システム構成を図1に示す.本自動データ分散技術は,既存のSMP向け自動並列化コンパイラ WPP[1]にデータ分散部を挿入することで実現した.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する.さらに,DSM 向けにデータ分散指示文を追加した拡張 OpenMP 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並列実行を実現する.

データ分散部では,まず処理分散部[1]で決定した並列化ループを検出し,こ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ループ(以後,**カーネルループ**と呼ぶ),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う.

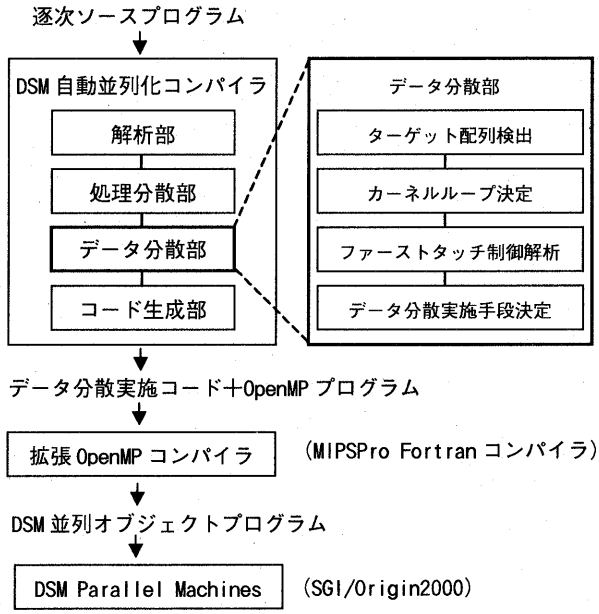

図1 システム構成

2.2 ファーストタッチ制御データ分散方法

従来のデータ分散方法としては,データ分散指示文による方法や 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ファーストタッチ方式データ分散などがあった.ところが,これら従来方法では,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間接参照が存在する場合,あるいは手続き毎に引数配列の宣言形状が異なる場合など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.

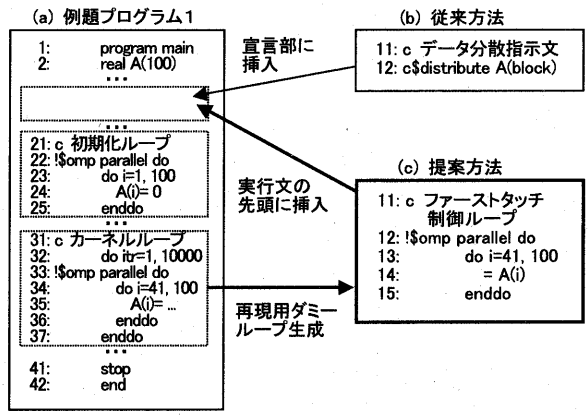

図2 ファーストタッチ制御方法

文献2)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.本方法では,図2(a)に示した例題プログラム 1 が入力された場合,コンパイラが図

Automatic Data Distribution Method for Distributed Shared Memory
Takashi HIROOKA*,**, Hiroshi OHTA*,***, Takayoshi IITSUKA*,** and Sumio KIKUCHI*,**
*Advanced Parallelizing Compiler Project
**Systems Development Laboratory, Hitachi,Ltd.
***Information & Compuer Systems, Hitachi,Ltd.

2(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．これにより，データローカリティが向上し，プログラムの処理速度を高速化させることが可能となる．

3. 評価

評価には，NASA 提供の標準的ベンチマーク NPB2.3serial/CG(classB) を用い，測定は，SGI/Origin2000 上で行った．

CGは，カーネルループと主要配列が限定されている．また，処理分散の並列化効率が高く，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3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きるため，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すが，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(no dist)，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(dist + indirect(DIR))，大幅にスケーラビリティが改善し，16プロセッサ時で4.1倍に性能が向上した．さらに，間接参照のように複雑なアクセスパターンになる可能性がある場合，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．

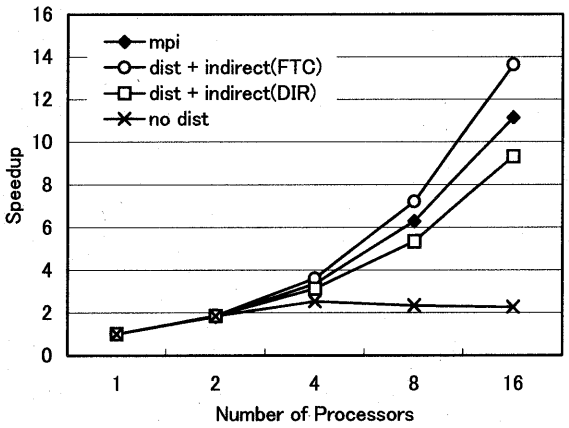

図3 CG(classB)の性能向上比

まず，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4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．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（3,9行目）．
・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（17～23行目）．
・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（24～28行目）．

```
 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4 CGのファーストタッチ制御コード

評価の結果(dist + indirect(FTC))，図3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を向上させ，MPIプログラムを上回る性能が得られることを確認した．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．

4. おわりに

分散共有メモリ向けの自動データ分散方法を実装し、その評価結果を報告した。今後はアドバンスト並列化コンパイラ研究体の活動として，本技術のPCクラスタ向けへの適用を進めていく予定である。

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) 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）．
2) 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）．