

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

4U-3

廣岡孝志† 太田寛† 菊池純男†

†新情報処理開発機構日立研究室

1. はじめに

分散共有メモリ(DSM)型マルチプロセッサ向けの並列化コンパイラにおけるデータ再分散最適化方法を提案する。本最適化は、再分散対象配列をリシェイプすることにより、データのローカリティを向上させ、スケーラブルな性能向上を得るものである。

DSM を実現する方法として、物理メモリへのデータ割付けをページ単位で行う方法がある。ところが、このとき指示文でデータ分散形状を指定しても、分散後の連続領域サイズがページサイズ以下の場合、指示通りのノードにデータが割付けられない。この問題を解決するためのコンパイル技法として、配列の形状を変更してから分散するリシェイプ分散が提案されており、これによりデータローカリティが改善され、性能が向上することが示されている¹⁾²⁾。ところが、実アプリケーションでしばしば現れるノード間データ移動が必要なデータ再分散対象配列は、従来、リシェイプ分散の対象外とされてきた³⁾。本研究では、データ再分散対象配列をリシェイプする方法を提案し、その有効性を実機評価により示す。

2. DSM 向けデータ再分散最適化方法

2.1 概要

配列リシェイプの目的は、分散後の連続領域サイズを最大化することにより、データローカリティを向上させることにある。具体的には、配列全体の要素数を変更することなしに配列の次元数を増やし、増設した外側次元をデータ分散することにより実現する。本提案では、再分散を行う前後の各々の分散次元につきリシェイプを行う。すなわち、図1のようにデータ構造とデータ分散指示文を変換する。これに伴いループ構造や配列添字を変換し、元のプロ

グラムとの整合性を保つ。なお、図1中のPはプロセッサ数を示す。

dimension A(N1,N2)	dimension A(N1/P,N2/P,P,P)
c\$distribute A(block,*)	c\$distribute A(*,*,block,*)
c\$redistribute A(*,block)	c\$redistribute A(*,*,*,block)

(a) 最適化前

(b) 最適化後

図1 データ構造の変換

2.2 アルゴリズム

本最適化の適用条件、及びアルゴリズムを以下に示す。本最適化実施の条件は、異なる次元を以下のパターンで再分散する場合とする。

(a) (block,*) $\rightarrow$ (*,block)(b) (cyclic,*) $\rightarrow$ (*,cyclic)(c) (block,*) $\rightarrow$ (*,cyclic)(d) (cyclic,*) $\rightarrow$ (*,block)

本最適化方法のアルゴリズムは、以下のステップから成る。

(1) 配列のリシェイプ

配列宣言を $A(\dots, N1, \dots, N2, \dots)$ から $A(\dots, \lceil N1/P \rceil, \dots, \lceil N2/P \rceil, \dots, P, P)$ に変更する。 $\lceil \rceil$ は、切り上げを表す。ただし、図1、2の例題では、割り切れるものとして切り上げ記号を省略した。

(2) 配列添字変換

前述の4つの場合につき、添字 $(\dots, i, \dots, j, \dots)$ をそれぞれ以下のように変換する。

(a) $(\dots, \text{mod}(i,b), \dots, \text{mod}(j,b), \dots, i/b, j/b)$ (b) $(\dots, i/p, \dots, j/p, \dots, \text{mod}(i,p), \text{mod}(j,p))$ (c) $(\dots, \text{mod}(i,b), \dots, j/p, \dots, i/b, \text{mod}(j,p))$ (d) $(\dots, i/p, \dots, \text{mod}(j,b), \dots, \text{mod}(i,p), j/b)$

ただし、bはブロックサイズとする。

(3) ループ変換

ループ交換可能である場合、ループ変換を実施する。まず、配列のリシェイプで増設した次元数分、元のループの外側にループ長Pのループを生成し、元のループ長をプロセッサ数で割った値とする。

図2に前述の(a)の場合のループ変換の例を示す。

Data Redistribution Optimization for Distributed Shared Memory by Array Reshape

Takashi Hirooka †, Hiroshi Ohta †, Sumio Kikuchi †

† RWCP Hitachi

ループ変換を実施するとデータのローカリティが向上し、また図2(b)の下線部に示すように配列Aの添字が簡単に表現できる。

```

do j=1, N2
  do i=2, N1
    A(i,j)=A(i-1,j)
  enddo
enddo
do j=1, N2
  i_low=1
  do j2=1, P
    do i2=1, P
      do j1=1, N2/P
        if(i2 .eq. 1) i_low=2
        do i1=i_low, N1/P
          i_org= i1+b*(i2-1)
          i1x= mod(i_org-1-1,b)+1
          i2x= ((i_org-1)-1)/b+1
          A(i1,j1,i2,j2)=A(i1x,j1,i2x,j2)
        enddo
      enddo
    enddo
  enddo
enddo

```

(a) 最適化前

(b) 最適化後

図2 ループ構造の変換

(4) ビーリング

元の添字が、'i + 定数' の場合、ループの最初または最後の幾つかの繰り返しを展開することにより、ループ本体内の除算や剰余計算を除去でき、さらに性能が向上する。

3. 評価

本最適化の効果を示すため、図3の評価用プログラムを作成し、SGI/Origin2000^{*)}上で性能測定を行った。

```

1: dimension A(4096,8192)
2: c$distribute A(block,*) ; 初期データ分散指示
3: do j=2,8192
4:   do i=1,4096
5:     A(i,j)=A(i,j-1) ; 計算ループ1
6:   enddo
7: enddo
8: c$redistribute A(*,block) ; データ再分散指示
9: do j=1,8192
10:   do i=2,4096
11:     A(i,j)=A(i-1,j) ; 計算ループ2
12:   enddo
13: enddo

```

図3 評価プログラム

本プログラムは、2次元配列Aに対して、計算ループ1で1次元目 block 分散向けの計算を行い、計算ループ2で2次元目 block 分散向けの計算を行うものである。このプログラムを用い、以下に示す3通りの測定を行った。

(1) 従来方法1

リシェイプを行わない図3に示したままの並列プログラム。

(2) 従来方法2

従来のリシェイプを行ったもの。ただし、従来のリシェイプは再分散配列を対象としないため、図3の8行目の再分散指示文をコメントアウトし、プログラム全体を通して(block,*)でデータ分散した。

(3) 提案方法

本提案の最適化を実施し、配列Aの1次元目と2次元目を両方リシェイプした並列プログラム。

なお、本評価では、提案方法の最適化はソースレベルで人手で行った。

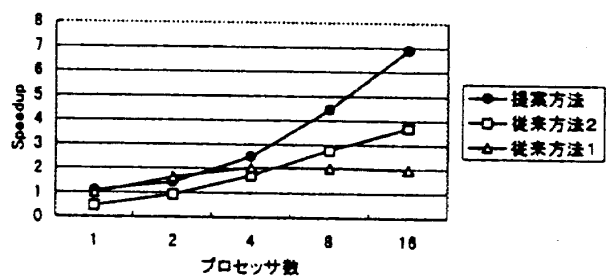


図4 性能向上比

結果を図4に示す。従来方法1、2の並列プログラムは、キャッシュ効果、データローカリティ、スレッド起動オーバーヘッドの何れかが障害となり、スケラブルな並列性能が得られなかった。提案方法は、全てのプロセッサ数の範囲で良好な並列性能を示し、従来方法に比して、数10%から数100%性能が向上した。

4. おわりに

分散共有メモリ型マルチプロセッサ向けのデータ再分散最適化方法を提案し、その評価結果を報告した。今後、適用範囲の拡大等を引き続き行い、データローカリティ最適化の一環として研究を進めていく予定である。

参考文献

- 1) J.M.Anderson 他2名 "Data and Computation Transformations for Multiprocessors", Stanford University, PPOPP'95
- 2) R.Chandra 他5名 "Data Distribution Support on Distributed Shared Memory Multiprocessors", Silicon Graphics Computer Systems, Digital Western Research Lab, PLDI'97
- 3) SGI MIPSpro Fortran77 Programmer's Guide, Silicon Graphics Inc

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