

H P F 処理系における最適化機能

2 P - 3

- 実行時判定の削除 -

佐藤 真琴†, 広岡 孝志†, 和田 清美†, 山本 富士男‡

† (株)日立製作所システム開発研究所, ‡ 神奈川工科大学

1. 緒言

分散メモリ型超並列機に対するプログラミングでは、他プロセッサ (P E) のメモリに配置されたデータを参照するための P E 間通信の記述などユーザに大きな負担がかかる。これの緩和を狙った H P F (High Performance Fortran) という Fortran 言語の並列向け拡張仕様が提案されており、高性能なコンパイラが期待されている。従来のコンパイル方法 [1] では、ループ中の複数の代入文について、それらの左辺配列の添字が異なる場合、ループ中に多くの実行時判定文が生じ、性能を悪化させた。本発表ではこれを削除して高い実行性能を得る最適化について述べる。

2. プログラム分割

プログラム分割では、各 P E が計算を分担できる形にプログラムを変換する。分担 P E の決定には「計算結果の格納先であるデータを持つ P E が、その計算を実行する」"owner computes rule"を、生成プログラム形式には、全 P E が同じプログラムを実行する SPMD を、変換アルゴリズムには、代入文の左辺配列の添字をループ制御変数の関数と見なす方法 [1] を採用した。ここで述べる実行時判定削除法は、上記方法（以下、従来法と呼ぶ）による結果にさらに変換を加えるプログラム分割方法である。

3. 実行時判定削除法

3. 1 各 P E に対する変換

図 1 (A) の逐次ループの並列化を考える。配列 A、B は、(A) 上部のように 3 分割する。

(B) は従来法による結果である（従来法 1）。2 つの文の添字のずれのため I F 文が必要になる。

(C) は (A) に対してループ分割した後に従来法を適用した結果である（従来法 2）。

従来法 1 はすべてのプログラムに適用できるが、ループ範囲は、各文の実行範囲（ある P E が文を実行するためのループ制御変数の範囲）よりも大きく、

全ループ繰り返しで多くの文について I F 文の実行が必要、などの要因で実行性能は悪い。

従来法 2 はループ分割可能な場合のみ適用できる。実行性能は従来法 1 より優れるが、ループ終了判定の増加等のループ分割オーバーヘッドがある。

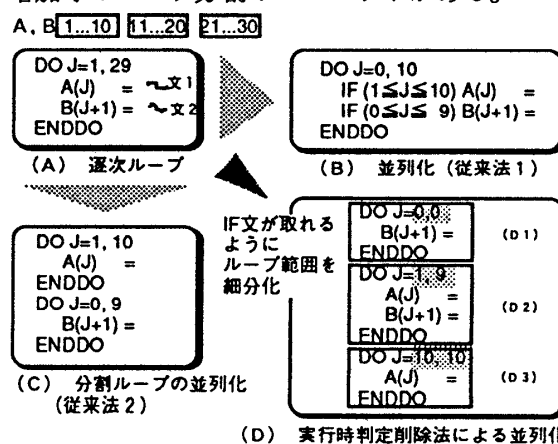


図 1 複数文に対する並列化 (PE 3 台、PE 1 の場合)

(D) は I F 文が取れるようにループ範囲を細分化する実行時判定削除法による結果である。これは以下のステップで実現される [2]。

(1) ループ実行範囲を、ループ内の各文の実行範囲で細分する。各細分範囲では、各文は常に実行されるか、実行されないかのいずれかになる。

(2) 常に実行される文の実行時判定文を削除し、常に実行されない文は文自体を削除する。

本方法は、ループの実行範囲を場合分けしただけなのでほとんどすべてのループに適用可能であり、ループ分割オーバーヘッドもない。即ち、適用プログラム、実行性能ともに従来法より優れる。

3. 2 全 P E に対する変換

本節では、前節の変換の結果、P E によってループの数が異なる場合の対応策を提案する。

図 2 に、図 1 の文 1 と文 2 の実行範囲、ループ実行範囲の細分結果を示す。

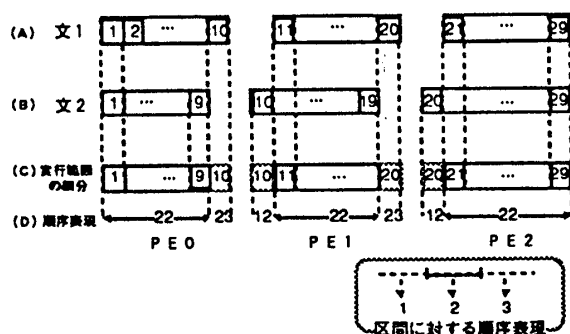


図2 文の実行範囲の細分と順序表現

PE 0では1～9と10の2つ、PE 1では3つ、PE 2では2つのループ実行範囲に細分される。生成プログラムはSPMDなのでPE毎の違いを吸収する必要がある。このため、複数区間を基準にした順序表現数を導入する。

〔定義1〕ある数に対するある区間を基準にした順序表現数とは、その数が基準区間に含まれるなら2、基準区間より値が小さい[大きい時は1[3]、である。

〔定義2〕ある数に対する複数の区間を基準にした順序表現数とは、各区間を基準にした順序表現数を並べてできる数である。

(例) 図2のPE 1では、10は文1の区間より小、かつ、文2の区間に含まれるので、両区間を基準にした順序表現数は、各々、1、2となり、上記複数区間を基準にした順序表現数は12となる。

以下にこれを用いたプログラム生成方法を示す。

(1) PE毎に、各文の実行範囲から成る複数の区間を基準とし、細分区間の下限値に対する順序表現数を求める。

(2) 全PEに対し、未選択の順序表現数の内、最

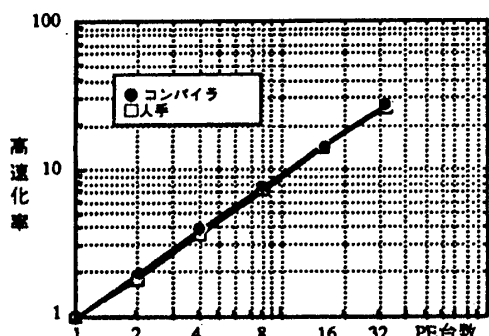


図5 Shallow-Water (512x512) の実行性能比較

小の数を選び、この数が存在するPEではこの数を含む細分区間の下[上]限値をループ下[上]限値とするループを生成し、この数が存在しないPEではループ下[上]限値を1[0]にする。

図3は上記ステップによる結果である。

C ループ上下限設定
 IF (PE.EQ.0) L1=1;U1=0; L2=1;U2=9; L3=10;U3=10
 IF (PE.EQ.1) L1=0;U1=0; L2=1;U2=9; L3=10;U3=10
 IF (PE.EQ.2) L1=0;U1=0; L2=1;U2=9; L3=10;U3=0
 DO J=L1, U1 ; 並列化ループ1
 B(1+J)=
 ENDDO
 DO J=L2, U2 ; 並列化ループ2
 A(J)=
 B(1+J)=
 ENDDO
 DO J=L3, U3 ; 並列化ループ3
 A(J)=
 ENDDO

□ 並列化ループ1の上下限値
 ■ ループ実行せず

図3 実行時判定削除法の変換結果

ループ1は順序表現数12に対する。図2のPE 1、PE 2では、12を持つ数は文2の実行範囲にのみ含まれるので、文2だけを実行する。PE 0には12がないので、ループは実行しない。

4. 評価結果

表1は図1に対する実測結果である。実行時判定削除法は、従来法1、2より実行性能が良い。

表1 実行時判定削除法と従来法の実行時間比較

比較基準	並列化法 (従来法1)	ループ分割+並列化 (従来法2)	実行時判定 削除法
通常の場合を100	100	68	53
ループ分割を100	—	100	78

図5はShallow-Waterに対するコンパイラによる並列化と人手並列化との実測結果である。測定マシンはnCUBE2で、配列は2次元目をBLOCK分割した。両者の性能はほぼ同じになった。

5. 結論

ループ内の実行時判定文を削除する実行時判定削除法を提案した。PE毎にループの数が異なる場合にも対応した。性能評価により有効性を確認できた。

参考文献

- [1] S.Hiranandani他2名：Compiling Fortran D for MIMD Distributed-Memory Machines, Comm. ACM, August 1992, Vol.35, No.8, p66 - 80
- [2] (三吉)他5名：メッセージ交換型並列計算機のための並列化コンパイラTINPARー最適化手法と性能評価ー, 情報処理学会研究報告, 94-HPC-54, 1994