2P－4

HPF処理系における最適化機能
― 実行時PE判定のブロック化 ―

和田　清美 †，佐藤　真琴 †，山本　富士男 ‡
† 日立製作所 システム開発研究所，　‡ 神奈川工科大学

1．はじめに

超並列機上で一つのプログラムを多数のPE(Processing Element)に分配して実行させるために、HPF(High Perfirmance Fortran)では、ユーザはFortranプログラムにデータを各PEに分配する指示を付加するだけで済み、プログラム(解法手順)の分割や通信の生成はコンパイラが行う。本稿では、DOループに対する従来のプログラム分割方式の課題と、その解決法の提案および評価を報告する。

2．プログラム分割処理の概要

プログラム分割処理は、各PEに計算を分配するための方法を決定し、各PEが分担して実行できる形にプログラムを変換する。分担するPEの決定は、「計算結果の格納先であるデータを持つPEが、その計算を実行する」"owner computes rule"に基づいて行う。プログラム分割方法には、単純法と添字関数法[1]が知られている。単純法によるプログラム分割では、ループは元の範囲を繰り返し、ループ繰り返し毎に、各PEが計算を実行すべきか否かを判定する。添字関数法によるプログラム分割では、ループ制御変数からなる代入文の左辺配列添字式から、添字関数(ループ制御変数を引数として配列添字を返す関数)とその逆関数を求め、添字逆関数に各PEが保持する配列添字範囲をあてはめることにより、各PEは自PEが保持するデータを書き換えるループ範囲のみを実行する。

Optimization in HPF compiler system - Vectorization of guards selecting execution processors
Kiyomi Wada †, Makoto Sato †,
Fujio Yamamoto ‡
† Systems Development Laboratory, Hitachi, Ltd.
1099 Ohzenji, Asao, Kawasaki, 215 Japan
‡ Kanagawa Institute of Technology
1030 Shimoogino, Atsugi, 243-02 Japan

3．課題

プログラム分割処理において、図1のようなループ内代入文の左辺配列添字の複数次元に同一ループ制御変数が現れる場合、図2のように1次元めと2次元めでブロック分割すると、対角線上のPEでは■で示す配列要素を、その右隣のPEでは▨で示す配列要素を書き換え、それ以外のPEでは配列要素を書き換えない。これまで、各PEに割り当てられた配列添字範囲の設定が困難であるため、図3のような単純法を適用していた。単純法は、条件判定のみの無駄な処理に時間を費やすため、実行性能が悪くなる。

```
DO k=1, 16
  A(k, k+1)= A(k, k+1) * w
```

図1．ソースプログラム

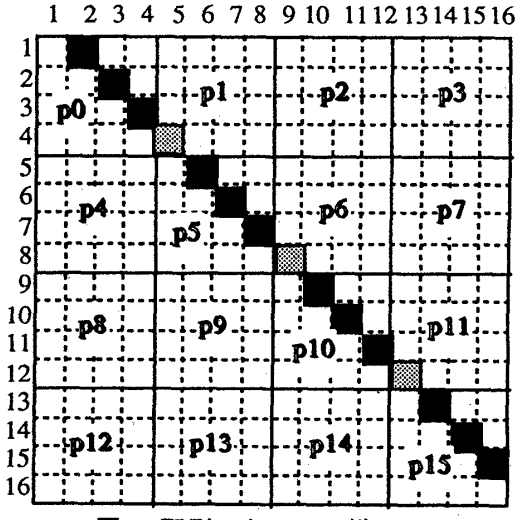

図2．配列アクセスの様子

```
DO k=1, 16
  IF (owner(A(k, k+1)) )THEN
    A(local_index (k), local_index (k+1) )= ···
```

図3．単純法によるノードプログラム

4．実行 PE判定のブロック化による解決法

そこで、PE番号をデータ分割した各次元別に表す次元別PE番号を導入し、ループ内代入文を実行するPEの判定をループの外で行う(実行PE判定のブロック化)添字関数法を考案した。処理方法を示す。

(1) 1次元PE番号を多次元PE番号に変換する。

(2) 代入文左辺配列添字の各次元毎に、添字関数法によりループ範囲Global ITeration Set(GITS)を求める。

(3) 各次元毎のGITSの積集合が空集合にならないような、次元別PE番号が満たす条件式を求める。

(4) (3)の条件式を任意の次元のGITSにあてはめる。

(5) 添字関数法により、ループ範囲Local ITeration Set(LITS)を求める。

例として、上記(1)〜(5)を図1に適用する。

(1) 図4に示すように、1次元めと2次元めに対する次元別PE番号をそれぞれp_1, p_2とする。

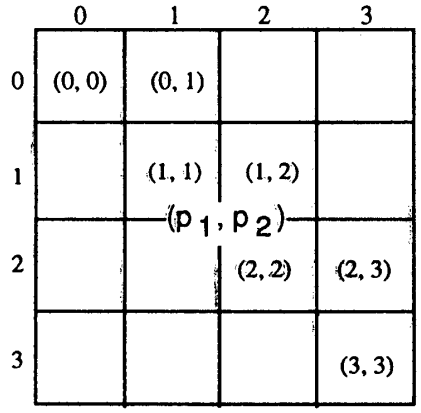

図4. PEの割り当て

(2) 代入文左辺配列の1次元めと2次元めの配列添字範囲 Global IndeX Set(GIXS)を$GIXS_1$, $GIXS_2$とすると、$GIXS_1$=[4*p_1+1:4*p_1+4],$GIXS_2$=[4*p_2+1:4*p_2+4]より、$GITS_1$=[4*p_1+1:4*p_1+4],$GITS_2$=[4*p_2:4*p_2+3]になる。

(3) $GITS_1$と$GITS_2$の積集合が空集合にならないための条件式は、4*p_1+1 ≦ 4*p_2+3 かつ 4*p_2 ≦ 4*p_1+4で、p_1, p_2は共に整数より、p_1=p_2またはp_1+1=p_2となる。

(4) p_1 = p_2のとき、GITS=[4*p_1+1 : 4*p_1+3]、p_1+1 = p_2のとき、GITS=[4p_1+4 : 4p_1+4]になる。

(5) p_1 = p_2のとき、LITS=[1 : 3]、p_1+1 = p_2のとき、LITS=[4 : 4]となる。

以上の結果、ノードプログラムは図5になり、実行するPEの判定がループの外に出たことにより、ループ範囲が分割された。

```
IF (p1 .EQ. p2) THEN
  DO k=1, 3
    A(k, k+1)= A(k, k+1) * w
IF (p1+1 .EQ. p2) THEN
    A(4, 1)= A(4, 1) * w
```

図5. 実行 PE 判定 ブロック化によるノードプログラム

5．評価結果

図6は境界要素法に対して、人手でプログラム分割を行ったときの実測結果である。測定マシンはnCUBE2で、配列は1次元めと2次元めをブロック分割した。図6より、実行PE判定をブロック化した添字関数法ではPE台数効果が得られた。

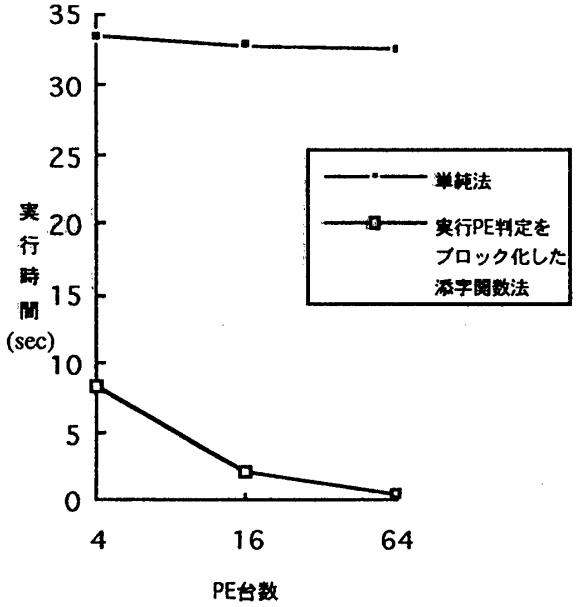

図6. 境界要素法 (512 ×512) の実行性能比較

6．結論

次元別PE番号を導入し、ループ内代入文を実行するPEの判定をループの外で行う添字関数法を提案、評価し、有効性を確認した。

参考文献

[1] S.Hiranandani,K.Kennedy,C.-W.Tseng:Compiling Fortran D for MIMD Distributed-Memory Machines, COMMUNICATION OF THE ACM, August 1992, Vol.35, No.8, p66 - 80