# WHILE型ループの自動ベクトル化，並列化

3Q−3　末廣謙二　津田孝夫
京都大学工学部情報工学科

## 1　はじめに

現在利用可能なスーパコンピュータの多くは，演算パイプライン方式によるベクトル計算機である．これらの計算機では，ループをベクトル処理することによってプログラムの高速実行を達成しているが，ループの繰り返し回数（ベクトル長）を実行前にプロセッサに与えなければならないという制約のために，ほとんどの場合コンパイラでベクトル化／並列化の対象とされるループはDO型ループに限られていた．

スーパコンピュータの応用範囲の拡大にともなって，繰り返し回数が事前に未知であるWHILE型のループについてもベクトル化／並列化が強く望まれるようになってきている．本稿では，WHILE型ループをベクトル処理するための手法を提案する．

## 2　基本的な方法

文献[4]には，WHILE型ループのベクトル化の方法として，次のような方法が提案されている．いずれもWHILE型ループをDO型ループに変換することが基本となっている．

**ループ分割**　ループの正確な繰り返し回数をベクトル処理に先だって調べる．すなわち，ループの脱出条件を計算するのに必要な部分だけをスカラ処理して繰り返し回数を計数し，その回数分のベクトル長でその他の部分のベクトル処理を行なう．もとのループは，繰り返し回数を計数するためにスカラ処理する前半ループとそれ以外のベクトル処理する後半ループとに分割されることになる．

**ストリップマイニング**　ベクトル長の定まらない演算に対して，あるベクトル長を仮定してとりあえずベクトル処理してみる．このとき変数の値の途中経過をすべて一時配列に保存しておく．ベクトル処理の結果ループの脱出条件がまだ成立せず，実行回数が不足している場合にはさらにベクトル処理を続け，脱出条件が成立した場合には，一時配列から成立したときの変数値を取り出して最終結果とする．

しかしこれらの方法には様々な制限があり，適用可能または有効である場合は極めて限られる．

## 3　よりすすんだ方法

### 3.1　ループ分割後のストリップマイニング

ループ分割では，ループの脱出条件の計算に必要な部分は必ず前半ループに含めてスカラ処理しなければならない．このため，全体としてベクトル化率があまり高くならない．

そこで，ループ分割により変換されたループの前半ループについて，さらにストリップマイニングを試みる．ループ分割とストリップマイニングを併用すると，ループ分割単独ではベクトル処理ができなかった部分をベクトル処理することができるようになり，より高いベクトル化率を達成できる．また，ストリップマイニング単独の場合と比べても，後半ループでベクトル処理される部分はストリップマイニングによるオーバーヘッドがかからないため，いくらか効率がよくなる．

### 3.2　ループの二重化

ループ分割では，前半ループにおける計算の途中経過を後半ループで引用するために，変数の値の履歴を一時配列に保存しなければならないことがある．ループの繰り返し回数の上限がわからないときは，一時配列のための領域を静的に確保できないために事実上ベクトル処理が不可能である．

この問題を解決するため，元のループにストリップマイニングを行なって内側ループをあらためてベクトル化対象ループとみなす．内側ループはループの繰り返し回数の上限がわかっているので，ループ分割による方法を適用することができるようになる．この方式を，ここではベクトル化方式としてのストリップマイニングと区別してループの二重化と呼ぶ．

### 3.3　ストリップマイニング長の動的変更

ベクトル処理にはオーバーヘッドが存在するため，ベクトル処理が可能であってもベクトル長が短いときにはスカラ処理した方が速いことがある．ストリップマイニングによる方法では，ループの繰り返し回数が実際には少ない場合にも必ず決まった単位でベクトル処理を行な

---

Automatic Vectorization/Parallelization of WHILE Loops
Kenji SUEHIRO, Takao TSUDA
Kyoto University

うため，スカラ処理した場合よりも遅くなってしまう可能性がある．

そこで，ループの繰り返し回数がまったく予想できない場合，ストリップマイニング長をはじめ短くしておき，徐々に長くすれば良いと考えられる．このようにすると，実際の繰り返し回数が多くなるほど効率の良いベクトル処理が期待できるとともに，繰り返し回数が少ない場合に著しく効率が低下するのを防ぐことができる．

## 4 実現

以上の方法は FORTRAN 前処理系として実現されている．この前処理系は，入力された FORTRAN プログラムの中の WHILE 型ループに，ループ分割・ストリップマイニング等を施して DO ループに変換する．出力されたプログラムを通常の自動ベクトル化／並列化コンパイラにかけることによって，WHILE 型ループがベクトル化／並列化される．また，京都大学情報工学教室（津田研究室）で研究・開発中の自動ベクトル化 Pascal コンパイラ **V-Pascal** にははじめからこの機能が採り入れられている．

WHILE 型ループを DO ループに変換する手順の概略は，以下の通りである．

**ループ分割による変換**

1. 制御フローグラフの作成…ベクトル化したいループの内部での制御の流れを抽出した制御フローグラフ（control-flow graph）を作成する[1].
2. データフロー解析による依存関係解析…さきに作成した制御フローグラフについてデータフロー解析を行ない，ループ内部の頂点相互のデータ依存関係を調べる[2]．またこのとき同時に文献 [3] の方法により，制御関係解析を行なう．
3. ループの分割…データ依存解析の結果を元に，制御フローグラフの頂点に対して前半・後半のいずれのループに含められるべきかを印づける．
4. 値の保存が必要な変数の検出…どの変数のどの定義の履歴を保存しなければならないかを知るために，さきに作成した制御フローグラフに再びデータフロー解析を行なう．
5. コードの生成…前半ループ部，後半ループ部，後処理部に分けてコードを生成する．

**ストリップマイニングによる変換**

ほぼ機械的に変換できる．

## 5 評価

以下のプログラムを前処理系にて処理し，日立 S-820/80 上の FORTRAN77/HAP コンパイラによりコンパイルして，実行時間を計測した．

```
10  CONTINUE
      IF ((N.LE.0).OR.(N.GT.10000)) GOTO 20
      I = I + 1
      REC1(N) = I
      REC2(N) = REC1(N) * REC1(N)
      REC3(N) = N * I
      N = NEXT(N)
    GOTO 10
20  CONTINUE
```

結果は図 1 に示す通りである．いずれの方式においても CPU 時間はループの繰り返し回数にほぼ比例している．効率の良さは「ループ分割後ストリップマイニング」「ストリップマイニングのみ」「ループ分割のみ」の順となっており，本稿で提案した手法が適用可能性だけでなく効率の点でも優れていることを示している．

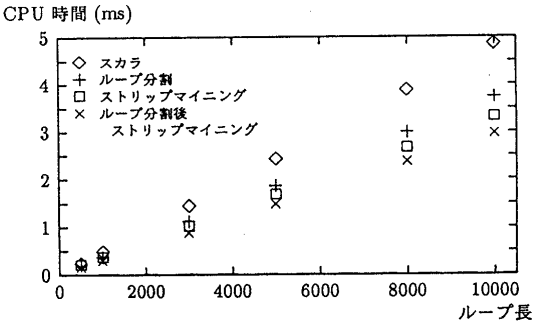


図 1: 実行時間の測定結果

## 6 おわりに

本稿では WHILE 型ループをベクトル化／並列化する手法について提案した．本稿で提案した手法は従来の手法に比べて適用可能性が高く，効率の点でも優れている．WHILE 型ループがベクトル化／並列化されることでスーパコンピュータの適用範囲がさらに広がることが期待される．

## 参考文献

[1] Aho, A. V., Sethi, R. and Ullman, J. D.: *Compilers—Principles, Techniques, and Tools*, Addison-Wesley, 1986.

[2] Banerjee, U.: *Dependence Analysis for Supercomputing*, Kluwer Academic, 1988.

[3] 國枝義敏, 津田孝夫: 自動ベクトル化コンパイラのための制御関係解析法, 情報処理学会論文誌, Vol. 30, pp. 1164–1174, 1989.

[4] Wolf, M.: *Optimizing Supercompilers for Supercomputers*, Pitman, 1989.