7F−7　多重ループの最適なベクトル化手法の提案

新開正史　安江俊明　金子正教　山名早人　村岡洋一
e-mail:harray-ml@muraoka.info.waseda.ac.jp
(早稲田大学　理工学部)

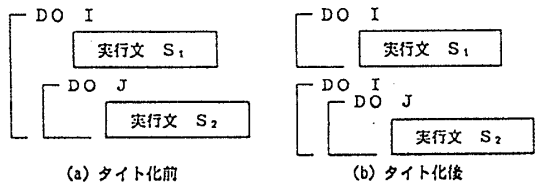
1　はじめに

本稿では、多重ループの最適なベクトル化を実現するために、①内側ループからのタイト化、②積極的なループ分割、という2つの解析方針に基づくベクトル化手法を提案する。

従来の多重ループのベクトル化手法では、①外側ループからタイト化するためループ分割が十分できない、②ループ分割による損得の評価が不完全である、という問題があり、最適なベクトル化ができない。そこで、本稿ではこれらの問題を解決するための解析手法を提案するとともに、実機(富士通のVP2200)において本手法を定量的に評価する。

2　タイト化[1]

図1(a)に示すループIのようにボディ部に実行文とループを含む多重ループを分割することにより、(b)のように多重ループを最内ループのみが実行文を含む形に変換する手法をタイト化と呼ぶ。



(a) タイト化前　(b) タイト化後

図1　タイト化

タイト化による利点には次の2つがある。

(1) ベクトル化の対象部分の拡張

(2) 外側ループでのベクトル化が可能

(1)については、図1のS_1の部分が最内ループとなり、ループIでのベクトル化が可能となる。(2)については、同図のS_2の部分がタイトな多重ループとなり、一重化またはループ交換による外側ループでのベクトル化が可能となる。

一方、タイト化の問題点は、分割されたループがベクトル化されずにスカラ実行として残った場合、ループ終了判定部のオーバヘッドが分割した分だけ増加してしまうことである。

3　多重ループの最適なベクトル化の方針

多重ループの最適なベクトル化を行うには次の2つの方針が重要である。

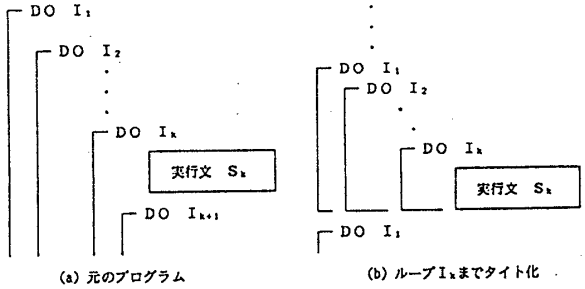
(I) 内側ループからのタイト化

(II) 積極的なループ分割

以下に、例題を示しながら詳しく説明する。

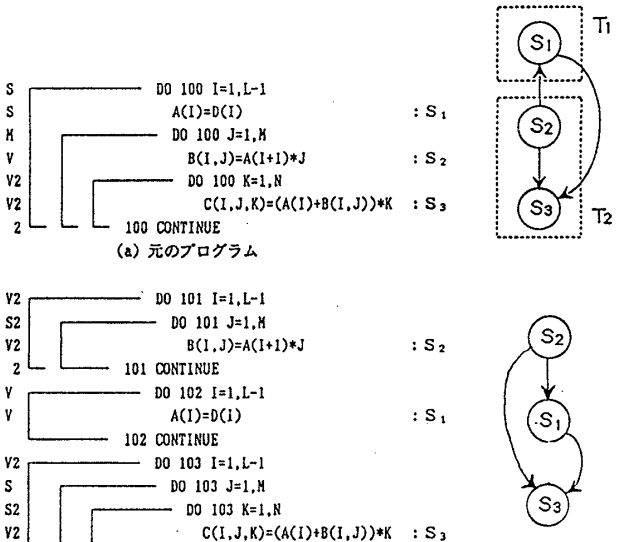
3.1　内側ループからのタイト化の重要性

既に、外側ループからタイト化する手法は報告されている(文献[2])。この方法では、図2に示すようにループI_kまでのタイト化が済んでいるとすると、そのループに含まれる実行文S_kはループ$I_1, I_2, \cdots, I_k$についてループ交換、一重化を考慮してベクトル化できる。つまり、外側ループからタイト化していくと同時に、コンパイラは実行文をベクトル中間言語に変換できる。ただし、外側ループからタイト化する手法にはループ分割が十分にできないという問題点がある。例として図3(a)に示すプログラムを考える。このプログラムを外側ループからタイト化する場合、ループIについて、まず$T_1=\{S_1\}$と$T_2=\{S_2, S_3\}$の分割の可能性を調べる。この場合、分割線をまたぐ依存関係が$T_1\rightarrow T_2$ ($S_1\rightarrow S_3$)と$T_2\rightarrow T_1$ ($S_2\rightarrow S_1$)の両方向が存在するため、ループ分割できない。



(a) 元のプログラム　(b) ループI_kまでタイト化

図2　外側ループからのタイト化



(a) 元のプログラム

(b) 内側からタイト化した結果

図3　プログラムA

これに対して、内側ループからタイト化していく場合、まずループJについてS_2とS_3に分割する。次にループIについてS_1とS_2とS_3の分割が可能かどうか調べる。これは、S_1とS_2を入れ換えることにより可能である。最終的な結果を(b)に示す。

この例のように、内側ループからタイト化しないとループ分割できない場合がある。

3.2　積極的なループ分割の重要性

分割されたループがスカラ実行として残った場合、ループ終了判定部のオーバヘッドが生じ、実行時間は増加してしまうので(図4(a))、ループ分割するべきではない。一方、(b)に示すようにタイト化されたループのいくつかがベクトル実行可能な場合には、次の2つの要求のトレードオフが問題となる[3]。

(A) 分割によりベクトル化可能部分を最大限広げる

(B) 分割によるオーバヘッドを最小限にする

既存のベクトル化コンパイラではこれらの問題に加えて、ベクトル実行の際の立ち上がり時間などアーキテクチャの特性を考慮して、ループ分割の損得を評価している。例えば、図5(a)に

An optimized vectorization scheme for multiply nested loops
Masashi SHINKAI,Toshiaki YASUE,Masanori KANEKO,
Hayato YAMANA and Yoichi MURAOKA
School of science and engineering,WASEDA University

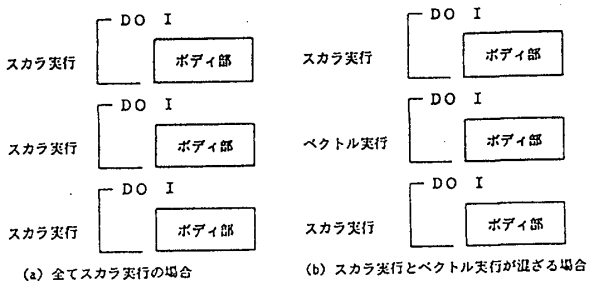


図4 ループIが3つに分割された例

示すプログラムはループIについて分割することにより(b)のようになる。この(a)のプログラムに対して既存のベクトル化ではループIについて「部分ベクトル化によるオーバヘッドが大」と評価して、ループ分割しない。しかし、実際にはループ分割した場合の(b)の方が(a)よりも実行時間が短い。これは、S_2の部分が(a)ではループJで内積のマクロ命令によりベクトル化されるのに対して、(b)ではループ交換によりループIでベクトル化されるためである。図6(a)に示すプログラムもループKについて分割することにより(b)のようになる。この(a)のプログラムに対して既存のベクトル化では一重化を検出できず、ループ分割しない。しかし、(b)ではS_1, S_2の部分がループJ, Kの一重化によりベクトル化されるため、実行時間は短縮される。

これらの例のように実際のプログラムでは、ループ分割した後にループ交換や一重化が適用できる場合が存在するので、ループ分割しないと、より効果のあるベクトル化を検出できなくなる。

4 多重ループの最適なベクトル化の実現方法

ここでは、3で提案した方針(I)(II)の実現方法を述べる。(I)(II)の方針は従来の手法に比べて多重ループの最適なベクトル化を実現できるが、コンパイラにインプリメントする場合には次の問題が生じる。

(I)の方針では、各ループをタイト化する段階でその外側ループがまだタイト化されていないので、ループ交換・一重化を考慮したベクトル化はできない。そのため、最外側ループのタイト化が終わった段階でベクトル中間言語に変換することになる。(II)の方針では、ループ分割できる部分は全て分割してタイト化するので、各ループのループ分割による損得も、最外側ループのタイト化が終わった段階で評価することになる。

以上述べたように(I)(II)の方針を用いた場合、最外側ループのタイト化が終わった段階で、ベクトル中間言語への変換及びループ分割による損得の評価をするので、コンパイル時間が増大する。そのため、コンパイラとしての実現上問題となる。そこで次に示す解決策をとる。

(III) タイト化する多重ループの選択可

タイト化する多重ループを選択することで、プログラムの中でCPU時間を大量に消費する部分のみに解析を適用することができるのでコンパイル時間の問題を解決することができる。

また、ソースtoソースで変換することができれば、アーキテクチャに依存しないのでより有効な実現方法となる。

5 評価

例として示したプログラムA, B, C（図3, 図5, 図6）についてループ繰り返し回数L=M=N=100として、VP2200上で実行時間を計測したところ、(b)は(a)に比べて実行時間がそれぞれ約53, 34, 19％短縮された。

6 おわりに

本稿では、多重ループの最適なベクトル化のための方針とその重要性及び実現方法を示し、その有効性を定量的に評価した。

本稿の評価では、ループ繰り返し回数L,M,Nを100に固定したが、厳密に評価するにはL,M,Nを変化させて実行時間を測定する必要がある。

最外側ループタイト化終了後に行う各ループの分割採用・不採用の評価方法、及び上記のループ繰り返し回数と実行時間の関係に関する調査は今後の課題である。

現在、本稿で提案した手法をソースtoソースで実現するツールを作成中である。

```
          REAL A(M,L),B(M,N),C(L,N)
S         DO 300 K=1,N
S2          DO 200 I=1,L
V2            SUM=0                          : S1
V2            DO 100 J=1,M
V2              SUM=SUM+A(J,I)*B(J,K)        : S2
2     100     CONTINUE
V2            C(I,K)=SUM                     : S3
2     200   CONTINUE
      300 CONTINUE
```

(a) 元のプログラム

```
          REAL A(M,L),B(M,N),C(L,N),SUMQ(L)
S         DO 300 K=1,N
V           DO 201 I=1,L
V             SUMQ(I)=0                      : S1
      201   CONTINUE
V           DO 202 I=1,L
S             DO 100 J=1,M
V               SUMQ(I)=SUMQ(I)+A(J,I)*B(J,K) : S2
      100     CONTINUE
      202   CONTINUE
V           DO 200 I=1,L
V             C(I,K)=SUMQ(I)                 : S3
      200   CONTINUE
      300 CONTINUE
          SUM=SUMQ(L)                        : S4
```

(b) ループ分割後

図5 プログラムB

```
          REAL DW(L,M,N)
S         DO 40 K=1,N
V           DO 10 J=1,M
V             DW(1,J,K)=0                    : S1
       10   CONTINUE
S2          DO 20 I=2,L
V2            DO 20 J=1,M
V2              DW(I,J,K)=DW(I,J,K)-R*(DW(I,J,K)   : S2
                          -DW(I-1,J,K))
2      20   CONTINUE
       40 CONTINUE
```

(a) 元のプログラム

```
          REAL DW(L,M,N)
V         DO 41 K=1,N
V           DO 10 J=1,M
V             DW(1,J,K)=0                    : S1
       10   CONTINUE
       41 CONTINUE
V2        DO 42 K=1,N
S2          DO 20 I=2,L
V2            DO 20 J=1,M
V2              DW(I,J,K)=DW(I,J,K)-R*(DW(I,J,K)   : S2
                          -DW(I-1,J,K))
2      20   CONTINUE
       42 CONTINUE
```

(b) ループ分割後

図6 プログラムC

謝辞

本研究の遂行にあたり、数々の御助言を頂いた本研究室の神舘淳氏をはじめとする一晴一グループの皆様に感謝致します。

参考文献

[1]猪野他：”ループ分割による多重DOループのベクトル化方式”，第30回情処全大，4B-2(1985)

[2]田中他：”Fortran最適化の強化 − 多重ループのベクトル化 −”，第32回情処全大，4F-5(1986)

[3]安村他：”自動ベクトルコンパイラにおける部分ベクトル化の方式”，情処論，Vol.24，No.1(1983)