3L－4

# RISC向けコンパイラにおけるループ構造変換*

本川 敬子†　久島 伊知郎‡

(株) 日立製作所 システム開発研究所§

## 1 はじめに

ループ交換に代表されるループ構造変換の技術は、従来では主としてベクトル計算機または並列計算機向けのコンパイラにおいて、ベクトル化や、粗・中粒度並列性の向上を目的として研究されてきた[1]。これらの技術はRISC向けコンパイラでは、メモリアクセスパターンの変更によるデータキャッシュの有効利用や、命令レベル並列性の向上のために重要になる。本稿では、RISCプロセッサ向けのループ構造変換の概略と、SPECベンチマークにおける効果について報告する。

## 2 RISC向けループ構造変換の特徴

### 2.1 最適化項目と目的

RISC向けループ構造変換の目的としては、

- キャッシュの有効利用（データ局所性の向上、競合の回避）
- レジスタの有効利用
- 命令レベル並列性の向上

などが挙げられる。実装した最適化の各項目について以下に簡単に説明する。

- ループ交換：イタレーション実行順序の変更により、キャッシュ向けにデータ局所性を高める[2]。また、配列参照のループ不変式化によりレジスタの有効利用を図る。
- ループ展開：ループ本体の拡大による命令レベル並列性の向上、配列参照の共通式化によるレジスタの有効利用、外側ループの展開によるキャッシュ向けデータ局所性の向上（図1参照）などを目的とする。
- アキュムレータ変数展開：スカラ変数への累算を行うループに対して、ループ展開をすると同時に累算用の変数（アキュムレータ変数）をリネームし、ループ終了後にリネームで生じた変数の和を計算するという変換により、命令レベル並列性を高める[3]（図2参照）。
- ループ解消（ループ完全展開）：ループ回数が定数で値が小さい場合には完全展開を行う。
- ループ巻き戻し：ユーザによってループ展開されたループを、巻き戻して展開前の状態にし、他の最適化を適用しやすくする。
- ループタイリング（ブロッキング）：ループネストのイタレーション空間をグループ化することでデータ局所性を高める[2]（図3参照）。
- ループ分配：キャッシュ競合の回避を目的とする。

```
      DO 10 I = 1, 1000
        ...
        DO 10 J = 1, 1000
          ... A(I, J) ...
   10 CONTINUE
==>
      DO 10 I = 1, 1000, 2
        ...
        DO 10 J = 1, 1000
          ... A(I, J) ...
          ... A(I+1, J) ...
   10 CONTINUE
```

図1 外側ループ展開の例

```
 DO 10 I = 1,1000          S1 = 0.0
   S = S+A(I)              DO 10 I = 1,1000,2
10 CONTINUE        ==>       S  = S+A(I)
                             S1 = S1+A(I+1)
                         10 CONTINUE
                          S = S + S1
```

図2 アキュムレータ変数展開の例

```
  DO 10 I = 1, 256
    DO 10 J = 1, 512
      DO 10 K = 1, 128
        C(I,J)=C(I,J)+A(I,K)*B(K,J)
10 CONTINUE
==>
  DO 10 JJ = 1, 512, 64
    DO 10 KK = 1, 128, 64
      DO 10 I = 1, 256
        DO 10 J = JJ, JJ+63
          DO 10 K = KK, KK+63
            C(I,J)=C(I,J)+A(I,K)*B(K,J)
10 CONTINUE
```

図3 ループタイリングの例

*Loop Restructuring Transformation in RISC Compiler
†Keiko MOTOKAWA
‡Ichiro KYUSHIMA
§Systems Development Labratory, Hitachi, Ltd.

### 2.2 最適化対象ループネスト

対象とするループネストは完全ネストに限定せず、不完全ネストも対象とする。不完全ネストを完全ネスト化するために、ループ分配やスカラ前方代入を適用する場合がある。またタイリングやループ展開などでは不完全ネストをそのまま変換することもある。

### 2.3 最適化の組み合わせ

各最適化を順次適用することにより、複数の最適化の組み合わせによるループ構造変換を実現している。最適化の適用順序は、各最適化の効果の大きさ、変換後のループの形態などを考慮して決める。例えば、他の最適化の適用を促進するためのループ巻き戻しは他に先だって実施する。また、何通りかの最適化が可能な場合、より大きな効果が見込まれるものを優先的に適用する。

## 3 SPECfp92 での評価

SPECfp92 の FORTRAN プログラム 12 本を対象として、ループ構造変換による効果を調べた。測定は日立 H9000/735 (2-scalar, 256K バイトデータキャッシュ) 上で行った。

最適化により 3% 以上の性能向上が得られたプログラムは 5 本で、性能が劣化するものはなかった。最適化の効果がある 5 本のプログラムについては各最適化ごとの効果を表 1 に、また nasa7 についてはモジュールごとの結果を表 2 に示す。効果は、ループ構造変換を全く行わない状態からある変換だけを行ったときの性能向上率で示している。ループ構造変換以外の最適化は全て行っている。また各最適化の適用ループネスト数を表 3 に示す。

調査の結果、以下のことが得られた。

- ループ展開は適用ループが多く、各プログラムに対して堅実に効果を上げている。キャッシュ向けに外側展開を行っている nasa7 では、特に効果が大きい。
- ループ展開以外の最適化は適用ループネスト数が少なく、ループ分配、ループタイリング、ループ巻き戻しについては nasa7 の特定モジュールにしか適用されなかった。しかし、対象ループネストに対する効果は極めて大きく、どれも欠くことのできない最適化である。
- ループ巻き戻しは nasa7 の mxm で大きな効果がある。これは巻き戻しをした結果、専用ライブラリ化が適用されることによる。
- 各最適化による向上率の合計よりも全最適化による向上率が低くなる場合があるのは、複数の最適化方法がある場合、各最適化による向上率では複数の項目に重複して効果が現れるからである。例えば gmtry では、全最適化適用時の性能向上はループタイリングによるものであるが、ループ交換の適用によっても大きな性能向上が得られる。

表 1 各最適化による性能向上率 (%)

| program | unr | per | dis | til | rer | all |
|---|---|---|---|---|---|---|
| doduc | 5.2 | 0.0 | 0.0 | 0.0 | 0.0 | 5.2 |
| wave5 | 3.1 | 3.4 | 0.0 | -0.3 | 0.3 | 6.3 |
| swm256 | 4.5 | 0.0 | 0.0 | 0.0 | 0.0 | 4.5 |
| hydro2d | 4.5 | 0.0 | 0.0 | 0.0 | 0.0 | 4.5 |
| nasa7 | 19.5 | 16.9 | 5.3 | 8.7 | 2.5 | 48.3 |

unr: ループ展開 (ループ解消、アキュムレータ変数展開を含む)
per: ループ交換　dis: ループ分配　til: ループタイリング
rer: ループ巻き戻し　all: すべて

表 2 各最適化による性能向上率 (%)(nasa7)

| program | unr | per | dis | til | rer | all |
|---|---|---|---|---|---|---|
| mxm | 0.1 | 0.2 | 0.0 | 0.0 | 96.1 | 96.1 |
| cfft2d | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.4 |
| cholsky | 73.2 | 0.0 | 0.0 | 0.0 | 0.0 | 72.1 |
| btrix | 16.7 | 17.2 | 1.5 | 0.0 | 1.3 | 30.8 |
| gmtry | -3.2 | 86.8 | 1.0 | 105.8 | 0.4 | 106.7 |
| emit | -0.3 | -1.0 | 0.0 | -0.1 | -0.1 | 3.4 |
| vpenta | 58.2 | 25.9 | 20.9 | -0.1 | 0.4 | 109.8 |

表 3 各最適化の適用ループネスト数

| program | unr | res | acc | per | dis | til | rer | nest | (sgl) |
|---|---|---|---|---|---|---|---|---|---|
| doduc | 142 | 16 | 8 | 0 | 0 | 0 | 0 | 249 | (224) |
| wave5 | 109 | 1 | 0 | 22 | 0 | 0 | 0 | 262 | (186) |
| swm256 | 13 | 0 | 1 | 0 | 0 | 0 | 0 | 16 | ( 8) |
| hydro2d | 71 | 3 | 1 | 0 | 0 | 0 | 0 | 105 | ( 49) |
| nasa7 | 38 | 7 | 1 | 6 | 2 | 1 | 1 | 69 | ( 34) |
| total | 373 | 27 | 11 | 28 | 2 | 1 | 1 | 701 | (501) |

unr: ループ展開　res: ループ解消 acc: アキュムレータ変数展開
nest: 全ループネスト数　sgl: 1 重ループネスト数

## 4 おわりに

RISC プロセッサ向けのループ構造変換を実装し、SPEC ベンチマークにおける効果について報告した。RISC では、キャッシュ・レジスタの有効利用や命令レベル並列性の向上などの効果がある。ループ構造変換は全てのベンチマークで効果があるわけではないが、nasa7 では 48% の性能向上が得られ、欠くことのできない最適化である。キャッシュ向けの各最適化は適用ループネストは少ないものの効果は大きく重要である。

## 参考文献

[1] M. J. Wolfe, *Optimizing Supercompilers for Supercomputers*, MIT Press, 1989.

[2] M. E. Wolf, "Improving Locality and Parallelism in Nested Loops," *Technical Report No. CSL-TR-92-538*, 1992.

[3] S. A. Mahlke, "Compiler Code Transformations for Superscalar-Based High-Performance Systems," *Proc. of Supercomputing*, 1992.