2E－4

マルチグレイン並列処理における インタープロシージャ解析*

松井 巌徹†, 岡本 雅巳‡, 松崎 秀則†, 小幡 元樹†, 吉井 謙一郎‡, 笠原 博徳†

†早稲田大学理工学部電気電子情報工学科,‡(株) 東芝

1 はじめに

マルチプロセッサシステム上における Fortran 自動並列化コンパイラにおいて、筆者等は基本ブロック、ループ、およびサブルーチン間の粗粒度並列処理、ループのイタレーション間の中粒度並列処理、ステートメント間の近細粒度並列処理を階層的に組み合わせ、プログラム全域の並列性を利用するマルチグレイン並列処理 [1, 2] を提案している。また、粗粒度並列処理の実行方式として階層型マクロデータフロー処理 [3, 4] を採用している。この粗粒度並列処理の並列性を引き出すためには、インタープロシージャ解析が必須となる。イリノイ大学では、配列の添字の詳細とループ範囲を伝搬するアトムイメージ法によるインタープロシージャ解析 [5] を行なっている。しかし、各々の配列の次元が異なったり、宣言されている配列が整合配列や擬寸法配列などを含む場合に解析ができないという問題があった。また、スタンフォード大学では、選択的にクローニングする手法 [6] を用いて解析を行なっているが、プログラムサイズが大きくなるなどの欠点があった。これら問題を解決するために本稿では、新しいインタープロシージャ解析手法を提案することにする。

2 マルチグレイン並列処理

OSCAR マルチグレインコンパイラ [2, 7] ではプログラムを以下に示す三種類のマクロタスク (MT) に階層的に分割する。

- BPA(Block of Pseudo Assignment statements)
- RB(Repetition Block)
- SB(Subroutine Block)

各階層の MT は階層的に定義されたプロセッサクラスタ (PC) 間で並列処理される。また、各階層で PC に割り当てられた MT は、さらに PC 内のプロセッサエレメント (PE) により階層的に並列処理される。例えば、MT が RB ならば Do-all や Do-across などの中粒度並列処理、あるいはループボディの近細粒度並列処理、また RB が大規模であり内部でサブ MT が階層的に定義できる場合には階層的にマクロデータフロー処理を適用する。

3 データ依存解析手法

自動並列化コンパイラにおいて、ソースコードの並列性を最大限に引き出すためには、強力なデータ依存解析が必要である。データ依存解析でも特に難しいとされるのが、SB を含む引数間のデータ依存解析、すなわちインタープロシージャ解析である。本章では、粗粒度並列性を引き出すためのプロシージャ間データ依存解析手法について述べる。

*An Interprocedural Analysis for Multi-Grain Parallelizing Compilation Scheme

Gantetsu MATSUI†, Hidenori MATSUZAKI†,
Motoki OBATA†, Ken-ichiro YOSHII†,
Hironori KASAHARA†
†Waseda University. 3-4-1 Ohkubo Shinjuku-Ku, Tokyo 169
Masami OKAMOTO‡
‡TOSHIBA Corporation

3.1 インライン展開した中間コードを用いたインタープロシージャ解析

インライン展開は、サブルーチンを呼び出し側プログラム中に展開し、引数などで渡されている隠された情報がすべて明らかになるため、厳密なデータ依存解析を可能とする。しかし、全てのサブルーチンをインライン展開するとコード長、コンパイル時間が増大するため、必ずしも実用的でない。一方、従来のアトムイメージ法等によるインタープロシージャ解析は、コードサイズが増大することなく、インライン展開と同等の解析を可能とするために研究されているが、現在のところインライン展開をほど、正確な情報が得られない。

これらを解決するために、本稿では従来のインタープロシージャ解析法では解析できないサブルーチンに対し、厳密なインタープロシージャ解析の目的のために通常のインライン展開ではなく、インライン展開した中間コードを用いてインタープロシージャ解析を行ない、解析後はコードを元に戻すという方式を提案する。こうすることにより、解析後のサイズは変わらず、解析自体もインライン展開をしたときと同じように厳密に行なえると共に、高度なクローニングも可能とする。

3.2 階層にまたがるデータ依存解析

ここでは、マルチグレイン並列処理においてマクロタスクの階層を越えて、並列性を引き出すためのインタープロシージャ解析について述べる。例えば、図 1 のようなサンプルプログラムについて考える。このプログラムの MAIN 関数は 5 つの MT から構成されており、その中の MT3 から SUB1、MT4 から SUB2 が呼ばれ、さらに SUB2 の中から SUB3 が呼ばれる形となっている。

```
      PROGRAM SAMPLE
      INTEGER IDIM
      INTEGER X(1000), Y(1000)
C     - MT1 -
      READ (*,*) IDIM
C     - MT2 (LOOP1) -
      DO 10 I = 1, 1000
        X(I) = I
        Y(I) = I + 1
 10   CONTINUE
C     - MT3 (LOOP2) -
      DO 20 I = 1, IDIM
        - SB1 -
        CALL SUB1 (IDIM,X,Y,I)
 20   CONTINUE
C     - MT4 (SB2) -
      CALL SUB2 (IDIM,X,Y)
C     - MT5 -
      WRITE (*,*) X
      STOP
      END

      SUBROUTINE SUB1 (IMAX ,A ,C ,I)
      INTEGER S
      INTEGER A(IMAX,IMAX,10), C (10,*)
      S = 0
C     - LOOP3 -
      DO 100 J = 1, I
        S = S + 1
        DO 200 K = 2, 10
          A(I, J, K) = I * C (K, S)
200     CONTINUE
300   CONTINUE
      END

      SUBROUTINE SUB2 (IMAX,B,C)
      INTEGER B(IMAX*IMAX,10), C(10,*)
C     - LOOP4 -
      DO 300 L = 1, IMAX * IMAX
        B (L+1, 1) = B (L, 1) + C (1, L)
300   CONTINUE
C     - SB3 -
      CALL SUB3 (IMAX,C)
      END

      SUBROUTINE SUB3 (IMAX ,C)
      INTEGER C (10 ,*)
C     - LOOP5 -
      DO 400 II = 1, 10
        DO 500 JK = 1, IMAX * IMAX
          C (II, JK) = 0
500     CONTINUE
400   CONTINUE
      END
```

図 1: サンプルプログラム

まず MT3 と MT4 のデータ依存について調べる。MT3 と MT4 はともに SB を含むため、インタープロシージャ解析が必要である。この例のように、各々のサブルーチンで呼ばれている配列の次元が異なり、その配列が整合配列や擬寸法配列である場合、従来のアトムーメージを用いたインタープロシージャ解析 [5] では、解析できず、SUB1 内の配列 A と SUB2 内の配列 B の間に依存があると判定されてしまう。しかし、前節で述べた解析用の中間語レベルインライン展開手法を用いると、各々の配列の添字を一次元化してデータ依存解析を行なうため、より詳しいデータ依存解析を行なうことができる。

One-dimensioned Array Subscript :

A(I,J,K) ⟶ X(I + IDIM * (J - 1) + IDIM * IDIM *(K-1))

B(L,1) ⟶ X(L)

Range of Array Subscript :

(IDIM * IDIM + 1) ≦ A(I,J,K) ≦ 10 * IDIM * IDIM

2 ≦ B(L-1,1) ≦ IDIM * IDIM -1

1 ≦ B(L,1) ≦ IDIM * IDIM

図 2: インライン展開を用いたインタープロシージャ解析の例

図 2のように、SUB1 内の 配列 A と SUB2 内の配列 B は、それぞれ MAIN 関数 の SB1 と SB2 の配列 X によって呼ばれているが、その配列 X の一次元化した配列添字によるデータ依存解析を行なうと、それぞれの 配列 X の範囲に重なりがないため、 配列 A と配列 B の間には依存がないことがわかる。さらに配列 C 同士にも 1 次元目の要素に注目すると依存がないので、SUB1 内の LOOP3 と SUB2 内の LOOP4 の間にはデータ依存がないことがわかる。しかしこの例では、SUB1 内の LOOP3 と SUB2 内の SUB3 を解析してみると、配列 C においてデータ依存があるため、MT3 と MT4 に関して見れば並列に実行することはできない。

そこで、本手法でその一つ内側階層の MT まで見て依存解析を行なう。本手法の概要は以下の通りである。

1. そのブロックがサブルーチンブロックであれば、インライン展開した中間コードを用いて以下に述べるようなデータ依存解析を行なう。
2. 解析によりサブルーチンを含むループの並列化が可能であると分かれば元のサブルーチンを呼び出す形のループ並列化を行なう。あるいは SB 間の並列性があることがわかれば、SB 間の並列性を利用する。
3. 内側階層同士の MT のデータ依存解析で並列効果の出ると思われる SB に対し、拡張したクローニング及びサブルーチン分割を行なう。

図 1の例では、SUB2 内の LOOP4 と SB3 に対し、図 3のようにサブルーチンのクローニング、分割により MT4-1(LOOP4) と MT4-2(SB3) に分け、SB3 を一つ上の階層にもってくる。そうすることにより、MT3 と MT4-1 の並列性を利用することができる。

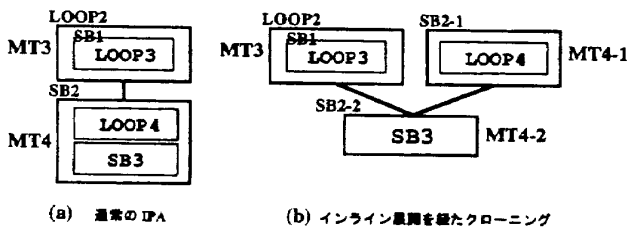

図 3: 階層にまたがるデータ依存解析

3.3 CALL 文を含むループの DOALL 解析

前節までは、粗粒度レベルでのインタープロシージャ解析を述べたが、ループ並列化である中粒度並列処理においてもインタープロシージャ解析は必須である。ここでは、CALL 文を含むループの DOALL 解析について述べる。

前章で述べた図 1を例にすると、MT3 のループブロックの中で SUB1 が呼ばれているが、アトムイメージ等のインタープロシージャ解析手法ではループインデックス J に I が使用されていることと、配列 C にスカラー S が使用されているため、この SUB1 を含む LOOP2 を DOALL 判定することができない。しかし、今回の手法と GIV などのシンボリック解析 [8] を用いることにより、SUB1 内の LOOP3 は、図 4 のようなループ依存解析となり、MT3 のインデックス I のループにおいても DOALL 判定することができる。

```
DOALL 20 I = 1, IDIM
  DOALL 100 J = 1, I
    DOALL 200 K = 2, 10
      A(I,J,K) = I * C(K, (-I + I * I + 2J) / 2) )
    ENDDOALL
  ENDDOALL
ENDDOALL
```

図 4: 図 1 の LOOP3 における DOALL 解析イメージ

4 まとめ

本稿では、マルチグレイン自動並列化コンパイラにおける粗粒度タスク間のデータ依存解析手法として、インライン展開した中間コードを用いたインタープロシージャ解析、階層にまたがるデータ依存解析及びリストラクチャリング手法、さらに CALL 文を含むループの DOALL 解析について述べた。

今後の課題としては、インライン展開した時の解析をより詳しく行なうためのシンボリック解析、またコードサイズが大きくなった場合を考慮して分割コンパイルを行なった時のインタープロシージャ解析などがあげられる。

本研究の一部は、通産省次世代情報処理基盤技術開発事業並列分散分野マルチプロセッサコンピューティング領域研究の一環として行なわれた。

参考文献

[1] 笠原: "並列処理技術", コロナ社 (1991-06).

[2] H.Kasahara, H.Honda, S.Narita: "A Multi-Grain Parallelizing Compilation Scheme for OSCAR", Proc. 4th Workshop on Languages and Compilers for Parallel Computing (Aug. 1991).

[3] 本多, 岩田, 笠原, Fortran プログラム粗粒度タスク間の並列性検出手法, 信学論, J73-D-I(12)"(1990-12).

[4] H.Kasahara, H.Honda, M.Iwata, M.Hirota: "A Compilation Scheme for Macro-dataflow Computation on Hierarchical Multiprocessor Systems", Inter. Conf. on Parallel Processing (Aug. 1990).

[5] Zyiyuan Li, Pen-Chung Yew: "Interprocedural Analysis for Parallel Computing", CSRD, (1988).

[6] M.W. Hall, S.P. Amarasinghe, B.R. Murphy, S. Liao, and M.S. Lam: "Detecting Coarse-Grain Parallelism Using an Interprocedural
Parallelizing Compiler", Proc. of Supercomputing'95, (Dec 1995).

[7] 岡本, 合田, 宮沢, 本多, 笠原: "OSCAR マルチグレインコンパイラにおける階層型マクロデータフロー処理", 情処学論, Vol.35(4) (1994-4).

[8] M.R. Haghighat, C.D. Polychronopoulos: "Symbolic Analysis for Parallelizing Compilers", Kluwer Academic Publishers, 1995.

[9] Andrew Ayers, Robert Gottlieb, Richard Schooler: "Aggressive Inlining", Proc. of 1997 ACM SIGPLAN conf. on PLDI, (May 1997).