手続き間並列化コンパイラ WPP の試作 - 変数プライベート化技術 - *

1 E − 3

佐藤 真琴† 青木 雄一郎‡ 菊池 純男‡

†（株）日立製作所 システム開発研究所 ‡新情報処理開発機構

1 はじめに

プログラムの並列化ではプログラムからより多くの並列性を抽出する技術が重要である。そのため、我々は、手続き呼び出しを含むdoループを並列化する共有メモリマルチプロセッサ向け手続き間並列化コンパイラ WPP(Whole Program Parallelizer)[2] を開発中である。本発表では、コモン変数のプライベート化、手続き間初期値・条件付き終値保証について発表する。

2 構成

WPP は、手続き間解析部と手続き間並列化部から成る。図 1 は手続き間並列化部の構成（左）と手続きごとの中間語群（右）を示す。手続き間並列化部は、手続き間解析部がクローニング等を行なった中間語を入力し、下位から上位の手続きに向かって並列性解析を、上位から下位の手続きに向かって並列化ループと変数属性の決定を行ない、最後に任意の順序で並列化コードを生成する。

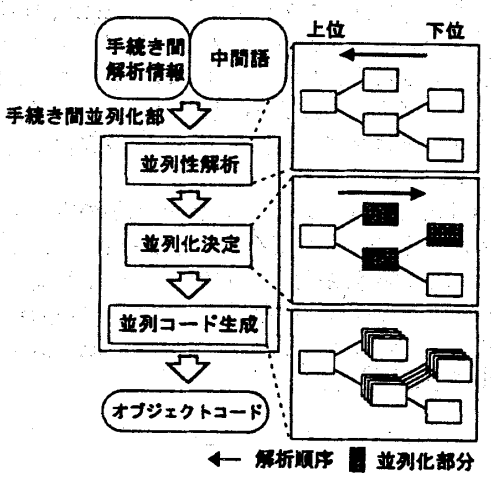

図 1 手続き間並列化部の構成と中間語群

3 変数プライベート化

プログラムでは、高速化のため、同じ計算は最初の1回だけ行なって一時変数に保存し、後はその値を利用するのが一般的である。ところが、一時変数が現れるループをそのまま並列化して実行すると、各プロセッサが各自の計算結果をその1つの変数に上書きするため、後で利用する際には必ずしも自プロセッサが書いた値とは等しくなく、結果が不正になり得る。そこで、一時変数に対し各プロセッサごとに別の領域を割り当てる（プライベート化）ことで変数への上書きを防ぐ。

3.1 コモン変数のプライベート化

プライベート変数の実現方法の一つは、プロセッサごとのローカルメモリやスタック等に、その領域を実行時に割り当てることである [3, 4]。コモン変数のプライベート化には以下の点を考慮する必要がある。

(1) 並列ループ内に手続き呼び出しがある場合、プライベートコモンの生存区間は手続きをまたがるため、プライベートコモンのまま他手続きに渡さねばならない。

(2) 異なる手続きではコモンブロック中の変数の並びが異なっていても良いので、あるコモン変数に対してプライベート化等を行なうと他手続きの複数のコモン変数に影響がでる場合がある。

したがって、プライベートコモンはコモンブロック全体を上記の方法でプライベート化するのが一般的であろう。ところが、これは、プライベート化不要な部分にも領域を割り当てるため多くのメモリが必要、実行時に領域割り当てオーバヘッドがかかる等の問題がある。プライベート変数の他の実現方法は元の変数にスレッド番号を表わす次元を追加した配列を静的に確保すること（高次元化、スカラ展開 [4] の類似）である。本コンパイラではプライベート化が必要なコモン変数のみ静的に高次元化する。さらにプライベートコモンを (1) が必要なものと不要なものに分け、前者のみ (2) をチェックすることで適用範囲を拡大する。

定義 手続き呼び出しを持つ並列ループ内や、手続き間並列化ループから呼び出される手続き内のプライベートコモンをグローバルプライベートコモン (GPC)、手続き呼び出しのない並列ループ内のプライベートコモンをローカルプライベートコモン (LPC) と呼ぶ。

アルゴリズム

- プログラム全体を解析し、コモンブロック／オフセットごとにコモン情報を作成する。
- コモン変数が出現するループごとに属性 (GPC/LPC) を決定する。
- 同オフセットの全GPC の全体長が一致するならGPC を高次元化し、同オフセットの全LPC の全体長が一致してそれがGPC の全体長と等しいならLPC も高次元化する。上記以外の LPC はスタック領域に確保する。

*Prototyping of Interprocedural Parallelizing Compiler WPP - Variable Privatization Techniques -

†Makoto Satoh, ‡Yuichiro Aoki, ‡Sumio Kikuchi.
†Systems Development Labratory, Hitachi, Ltd.,
‡Real World Computing Partnership.

3.2 手続き間初期値・条件付き終値保証

ループ並列化の際、プライベート化される変数が、定義の前に使用がある、または、ループ直後で使用がある場合は、各々、プライベート変数に対する初期値設定（初期値保証）とプライベート変数から元の変数への値の書き戻し（終値保証）が必要である。これらを手続き呼び出しを含むループに対して効率良く実現するため、手続き間情報を用いて初期値・終値保証すべき変数名および配列添字範囲（変数情報）を絞る。

アルゴリズム
(1) 初期値保証　定義より前に使用のある露出使用情報から、初期値保証すべき変数情報を求め、それらに対して初期値保証コードを生成する。
(2) 終値保証　定義情報とライブ情報の共通部分から終値保証すべき変数情報を求める。この内、確定定義である部分はループの各繰り返しで必ず値が定義されるのでループ最終回での値を終値とする確定終値保証コードを生成する。確定定義でない部分は最後に定義される回数が不明なので、これを実行時に検査する条件付き終値保証コードを生成する。条件付き終値保証は、各変数の定義の有無を表す変数にスレッド番号を表す次元を付加したマスク配列を変数ごとに用意し、変数の定義点でそれを定義するスレッドに対するマスク配列の値を真にし、並列ループ終了後、マスク配列の値が真になる最大のスレッドのプライベート変数の値を最終値とする。

4 例

図 2 はGPC/LPC 化と終値保証の例である。L1 は手続き間、L2 は手続き内並列化される。コモン変数 A の全体長が異なるので A はsub1 でGPC、sub2 でLPC になる。S1 で $A(1:10)$ が参照され、S2 は、条件文成立時のみ実行されるので確定定義でなく、条件付き終値保証コード L3, S3 が生成される。

5 評価

表 1 はSPEC92fp の内の 4 本による評価結果である。評価は、疑似的にスレッドを実現するツールと日立 SR2201 のハードウェアカウンタにより、各スレッドの実行サイクル数を計測して行なった。並列化率は、逐次実行において、プログラム全体に対する並列部分の実行時間の割合を示す。括弧内はSUIF[1] の結果である。並列化率ではSUIF と同等であることがわかる。

6 おわりに

手続きにまたがる、コモン変数のプライベート化と手続き間条件付き初期値・終値保証の実装方法について述べ、SPEC92 による評価結果を示した。

表 1 SPEC92fp による手続き間並列化結果

プログラム名	並列化率 [%] 手続き間無	並列化率 [%] 手続き間有	GPC/LPC	手続き間条件終値保証
doduc	33.7(20)	46.0(48)	GPC	○
mdljdp2	8.3(10)	86.2(85)	GPC	-
mdljsp2	6.5(25)	79.8(80)	GPC	-
nasa7	93.1(85)	97.8(85)	LPC	-

括弧内はSUIF[1] の結果

```
      common /com/N,A(100),B(100)
L1:   do j = 1, 100
        call sub1
      enddo
S1:   B(1:10)=A(1:10)
      call sub2
      ------------  sub1  ------------
      common /com/N,A(100),B(100)
      do i = 1, 100
        if(i<N) then
S2:       A(i) = ...
            ... = A(i)
        endif
      enddo
      ------------  sub2  ------------
      common /com/N,A(200)
L2:   do j = 1, 100
      do i = 1, 100
        A(i) = ...
          ... = A(i)
      enddo; enddo
```

(a) 変換前

```
      common /com/N,A(100),B(100)
      common /p_com/p_A(100,npe)
      common /p_final/p_mask(100,npe)
L1:   do j = 1, 100/npe   ! 手続き間並列
        call sub1
      enddo
L3:   do k = 1,10
      do kk = npe,1,-1
        if(p_mask(k,kk)=.true.)then
          A(i)=p_A(i,kk)
          exit
        endif
      enddo; enddo
S1:   B(1:10)=A(1:10)
      call sub2
      ------------  sub1  ------------
      common /com/N,A(100),B(100)
      common /p_com/p_A(100,npe)
      common /p_final/p_mask(100,npe)
      do i = 1, 100
        if(i<N) then
S2:       p_A(i,mype) = ...
S3:       p_mask(i,mype)=.true.
            ... = p_A(i,mype)
        endif
      enddo
      ------------  sub2  ------------
      common /com/N,A(200)
      real local_a(200)
L2:   do j = 1, 100/npe   ! 手続き内並列
      do i = 1, 100
        local_a(i) = ...
          ... = local_a(i)
      enddo; enddo
```

(b) 変換後

図 2 プライベートコモン化と終値保証の例

参考文献

[1] S. Amarasinghe 他 6 名. "Multiprocessors from a Software Perspective", *IEEE Micro*, pp.52-61, 1996.

[2] 飯塚他 3 名. "手続き間並列化コンパイラ WPP の試作 - 現状と今後の課題 -", 情報処理第 56 回全国大会, 1998.

[3] P. Tu, D. Padua. "Automatic Array Privatization", *Proc. of LCPC '93*, pp.500-521, LNCS Vol.768, Springer, 1993.

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