4H−8

ＯＳＣＡＲ上でのＦｏｒｔｒａｎサブルーチンの並列処理

茂木章善、本多弘樹、笠原博徳
早稲田大学理工学部電気工学科

１．はじめに

筆者らは従来より複数プロセッサクラスタを持つマルチプロセッサシステム上でのFortran プログラムの粗粒度レベルでの並列処理（マクロデータフロー）手法について提案している。本稿では、このマクロデータフロー処理におけるサブルーチン並列処理について述べる。Fortranにはサブルーチンと関数の二種類の副プログラムが存在するが、その処理手法に対しては同様の手法が適用できるので本稿ではサブルーチンについてのみ述べる。

２．ＯＳＣＡＲのアーキテクチャ[1]

ＯＳＣＡＲのプロセッサ・クラスタ（PC）は１６台のプロセッサ（PE）、３つのローカル共有メモリ（LCM）、及びローカルコントロールプロセッサ（LCP）を３本のバスで接続した平等型のメモリ共有型マルチプロセッサシステムである。

このＯＳＣＡＲ上では、１６台のPEを８台ずつ２クラスタ、あるいは５台ずつ３クラスタというように分割し、コントロールＩ／Ｏプロセッサあるいは通常のPEにダイナミックスケジューラの働きをさせることにより、疑似マルチクラスタシステムを構成することが出来る。

３．Ｆｏｒｔｒａｎサブルーチンの並列処理手法

３．１　概略

筆者らは従来よりDOループ、基本ブロック等をタスクとした粗粒度タスクレベルでのFortran 並列処理（マクロデータフロー）手法を提案している[2]。このマクロデータフロー法においてサブルーチンの処理を行う場合、以下の３つの処理方法が考えられる。

Ⅰ.インライン展開
Ⅱ.１つのサブルーチンを１つのマクロタスクとして定義する
Ⅲ.サブルーチン呼び出しの前後でバリア同期を取り、そのサブルーチン内部にマクロデータフロー概念を適用することによりマルチクラスタで処理する。[3]

現在筆者らはこの３つの手法をインプリメントしているが、本稿では特にⅡの手法を中心に述べる。

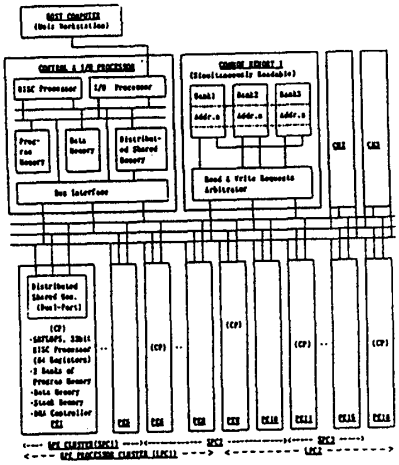


図１　ＯＳＣＡＲのアーキテクチャ

Parallel Processing of Fortran Subroutines on OSCAR
Akiyoshi MOGI, Hiroki HONDA, Hironori KASAHARA
Dept. of Electrical Eng.,WASEDA Univ.

３．２　データフロー解析

PC間でマクロタスクの並列処理を行うためにはマクロタスクレベルでのデータフロー解析が必要となる。すなわちサブルーチンをマクロタスクとして処理するためには、手続き間データ依存解析[4]-[6]が必要となる。以下にその方法を示す。

①コール・グラフの作成

サブルーチンの呼び出し関係を表すコール・グラフを作成する。図２に見られるように、コール・グラフはノードがサブルーチンを、エッジが呼び出し関係をそれぞれ表す。このコール・グラフを基に、各サブルーチンの呼び出しの深さを決定する。図２の例では、サブルーチンS3は深さ２で、残りのサブルーチンは深さが１となる。サブルーチンはネストされて呼び出されることがあり、その場合にはより深いサブルーチンから②に述べるデータ依存解析を行っていく。

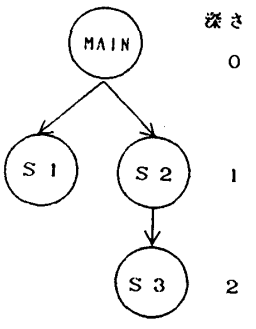
②データ依存解析

①で求められた呼び出しの深さの深い順にそのサブルーチンの内部から外へ階層的にデータフロー解析を行う（中間語→ステートメント→基本ブロック→…→サブルーチン全体）。呼び出し側から見たサブルーチンで定義・参照される変数というのは呼び出しの際の実引数、あるいは呼ぶサブルーチンと共通に持っているCOMMON変数である。よって、サブルーチンの始まりから終了までに定義・参照されたもののうちで引数（あるいはCOMMON変数）になっているものを見つけ出し、それらと実引数（あるいは呼び出し側のCOMMON変数）との対応をつけてやれば、呼び出し側での（call文の）定義・参照変数が求められることになる。

図２（ａ）の例では、まずS3について解析を行い、その後他のサブルーチンについて解析を行い、最後に主プログラムについて解析を行うことになる。S3から始めると、S3で定義される変数がX,L,Z の３つであるが、X,Z のみがS3の引数である。よってS3の呼び出し文CALL S3(A,B,C)によって定義される変数はA,C である事がわかり、同様に、参照される変数はB であることが分かる。次に、S2を見てみると、今度はZ がCOMMON変数であるので、Z 及び引数であるX に着目すればよい。S2の呼び出しCALL S2(B)によって定義される変数はD であり、参照される変数はBである。S1についても同様にして考えればよくCALL S1(A,B,C)によって定義される変数はC であり参照される変数はA,B である。これらの結果から２つの文（各々がマクロタスクとなる）CALL S1 とCALL S2 は並列実行が可能であることが分かる。

```
PROGRAM MAIN
COMMON /COM1/D
CALL S1(A,B,C)
CALL S2(B)
END
SUBROUTINE S1(X,Y,Z)
Z = ・ X ・ Y ・
RETURN
END
SUBROUTINE S2(X)
COMMON /COM1/Z
CALL S3(A,B,C)
Z = ・ X ・ B ・
RETURN
END
SUBROUTINE S3(X,Y,Z)
REAL X,Y,Z,L
X = ・・・・・・・・・
L = ・・・ Y ・・・
Z = ・・ L ・・・・
RETURN
END
```

（ａ）プログラム例



（ｂ）（ａ）のコール・グラフ

図２　解析例

3．3　サブルーチン内部の並列処理

1つのサブルーチンの内部は、その並列性により、スタティックスケジューリングを用いた細粒度並列処理、DOALL 等を用いた中粒度並列処理、階層的なマクロデータフローにより並列処理される。

3．4　サブルーチンと他のマクロタスクとの並列処理

プログラムのメインルーチンにおいて、サブルーチン呼び出しはサブルーチンブロック（SB）と認識され、他のDOブロックなどと同様に1つのマクロタスクとなり、ダイナミックスケジューリングを用いて1つのPCに割り当てられ、粗粒度のレベルで並列処理される。

4　ＯＳＣＡＲ上での並列処理

本手法をＯＳＣＡＲの疑似プロセッサクラスタ上にインプリメントし、その効果を確かめてみた。ソースプログラムを図3に、マクロフローグラフ、マクロタスクグラフを図4、5に、実行結果を表1に示す。表1には、同じプログラムをインライン展開したものをＯＳＣＡＲ上で実行した結果も同時に示す。表1において上段の値がサブルーチンによるもの、下段の値がインライン展開によるものである。両者ともその実行時間に大差はなく、本手法が効率よく実現されていることが分かる。また処理時間を見ても1PEで9.63秒かかった処理が2プロセッサクラスタ（各3PE）で1.95秒、3プロセッサクラスタ（各2PE）で1.83秒というように短縮され、サブルーチン間の並列性が自動的に抽出され効率よい並列実行が行われていることが確かめられた。

5　むすび

本稿では、ＯＳＣＡＲ疑似マルチクラスタ上でのFortran サブルーチン並列処理手法のインプリメントについて述べた。今後は分割コンパイルへの対応及び手続き間データ依存解析能力の向上に関する研究を行っていく予定である。

6　参考文献

[1] 笠原、成田、橋本、" ＯＳＣＡＲのアーキテクチャ"
信学論 J71-D(8) pp.1440-1445 8 1988

[2] 本多、広田、笠原 " 階層型マルチプロセッサシステム
ＯＳＣＡＲ上でのFortran並列処理手法"
並列処理シンポジウムJSPP'89論文集 pp.251-258 2 1989

[3] 小椋、合田、本多、笠原、成田" ＯＳＣＡＲ上でのFortran
プログラムの階層的マクロデータフロー処理手法"
第42回情処全国大会 3 1991

[4] Michael Burke,Ron Cytron "Interprocedural Dependence Analysis and Parallelization",Proceedings of the SIGPLAN '86 Symposium on Compiler Construction

[5] Remi Triolet "INTERPROCEDUARAL ANALYSIS FOR PROGRAM RESTRUCTURING WITH PARAFRASE",CSRD-538

[6] Zhiyuan Li,"INTRAPROCEDURAL AND INTERPROCEDURAL DATA DEPENDENCE ANALYSIS FOR PARALLEL PROCESSING",CSRD-910

表1　サンプルプログラムの実行結果［秒］
（上段：サブルーチン、下段：インライン展開）

	PE 1	PE 2	PE 3	PE 4	PE 5	PE 6
PC 1	9.6326	4.9770	3.3962	2.5856	2.1152	1.8022
	9.6326	4.9770	3.3968	2.5853	2.1136	1.8144
PC 2	5.3277	2.8115	1.9459	——	——	——
	5.3309	2.8282	1.9466	——	——	——
PC 3	3.3222	1.8266	——	——	——	——
	3.3219	1.8275	——	——	——	——

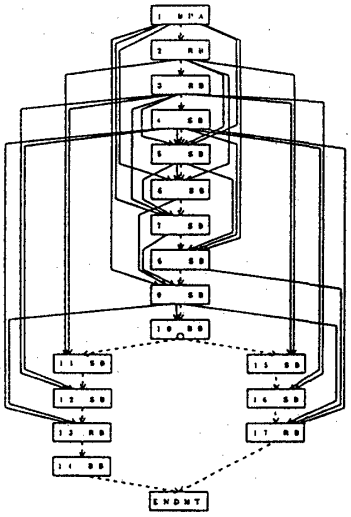


図4　サンプルプログラムのマクロフローグラフ

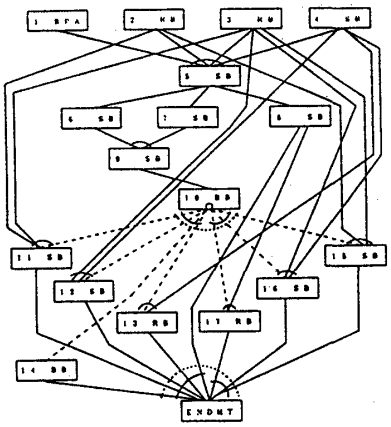


図5　サンプルプログラムのマクロタスクグラフ

```
C     PROGRAM SAMPLE FOR OSCAR
      REAL A(300,300),B(300,300),C(300,300)
      REAL X(300,300),Y(300,300),Z(300,300)
      REAL V1(300),V2(300),V3(300),V4(300)
      REAL D1
      COMMON /COM1/V1
C     ---------------------- MT 1
      D1=0.0
      V1(1)=1.0
C     ---------------------- MT 2
      DO 110 I1=1,300
        DO 100 J1=1,300
          A(I1,J1)=I1+J1-300
  100   CONTINUE
  110 CONTINUE
C     ---------------------- MT 3
      DO 130 I2=1,300
        DO 120 J2=1,300
          B(I2,J2)=I2-J2+1
  120   CONTINUE
  130 CONTINUE
C     ---------------------- MT 4
      CALL MTGEN(C)
C     ---------------------- MT 5
      CALL MLVEC(A,B,C)
C     ---------------------- MT 6
      CALL MTVEC(A,V2)
C     ---------------------- MT 7
      CALL MTVEC(B,V3)
C     ---------------------- MT 8
      CALL MTVEC(C,V4)
C     ---------------------- MT 9
      CALL VECSUB(V2,V3,D1)
C     ---------------------- MT10
      IF(D1.GT.0.0) THEN
C     ---------------------- MT11
        CALL MATADD(A,B,X)
C     ---------------------- MT12
        CALL MATSUB(B,C,Y)
C     ---------------------- MT13
        DO 170 I3=1,300
          DO 160 J3=1,300
            Z(I3,J3)=C(I3,J3)/D1
  160     CONTINUE
  170   CONTINUE
C     ---------------------- MT14
      ELSE
C     ---------------------- MT15
        CALL MATSUB(B,A,Y)
C     ---------------------- MT16
        CALL MATADD(C,B,X)
C     ---------------------- MT17
        DO 190 I4=1,300
          DO 180 J4=1,300
            Z(I4,J4)=C(I4,J4)+D1*V4(I4)
  180     CONTINUE
  190   CONTINUE
C
      ENDIF
C
      END
C
      SUBROUTINE MTGEN(X)
      REAL X(300,300)
      DO 110 I=1,300
        DO 100 J=1,300
          X(I,J)=0.0
  100   CONTINUE
        X(I,I)=1.0
  110 CONTINUE
      RETURN
      END
C
      SUBROUTINE MATADD(X,Y,Z)
      REAL X(300,300),Y(300,300),Z(300,300)
      DO 120 I=1,300
        DO 110 J=1,300
          Z(I,J)=X(I,J)+Y(I,J)
  110   CONTINUE
  120 CONTINUE
      RETURN
      END
C
      SUBROUTINE MATSUB(X,Y,Z)
      REAL X(300,300),Y(300,300),Z(300,300)
      DO 120 I=1,300
        DO 110 J=1,300
          Z(I,J)=X(I,J)-Y(I,J)
  110   CONTINUE
  120 CONTINUE
      RETURN
      END
C
      SUBROUTINE MLVEC(A,B,C)
      REAL A(300,300),B(300,300),C(300,300)
      REAL DY,R1,TA,TB,TC,SA,SB,SC,D1,D2,D3
      COMMON /COM1/V1(300)
      DO 100 I=2,300
        DY=V1(I-1)
        R1=0.314*DY
        TA=A(I,I)*DY
        TB=B(I,I)*DY
        TC=C(I,I)*DY
        SA=TA*TA
        SB=TB*TB
        SC=TC*TC
        D1=TA-TB
        D2=TB-TC
        D3=TC-TA
        V1(I)=(SA+SB+SC)*R1*(D1*D2
  100 CONTINUE
      RETURN
      END
C
      SUBROUTINE MTVEC(A,V)
      REAL A(300,300),V(300)
      COMMON /COM1/V1(300)
      DO 110 I=1,300
        V(I)=0.0
        DO 100 J=1,300
          V(I)=V(I)+A(I,J)*V1(J)
  100   CONTINUE
  110 CONTINUE
      RETURN
      END
C
      SUBROUTINE VECSUB(S,V,R)
      REAL S(300),V(300),R
      DO 100 I=1,300
        R=R+S(I)-V(I)
  100 CONTINUE
      RETURN
      END
```

図3　サンプルプログラム