数理解析研究所講究録
463巻 1982年 72-89

並列数値計算のシミュレーション

筑波大　電子情報　　森　正武

1．はじめに

数値計算においていわゆるベクトル演算の意味での並列計算は既に広く採用されている．この意味での並列計算は、四則演算ないしはそれに近い基本的な演算を並列に実行するものである．それに対し、ここでシミュレートしようとしているものは、より複雑な演算の並列計算，すなわちタスクレベルでの並列計算である．

現在でも既に数多くの並列計算機が試作され、そこで数値計算が並列的に行われている．しかし、現在はまだ汎用の並列計算機構の設計思想やそのための言語が確立しているわけではなく、いわば試行錯誤の段階にあるといえる．したがって現段階では、われわれ数値解析の研究者は、もしも可能ならば既存のハードウエアに束縛されることなくソフトウエアによって自由に並列計算をシミュレートし、並列計算の経験を深め、その経験の上に立って並列計算の望ましいアーキテクチャあるいは言語を提言すべきであると考える．

われわれの所属する筑波大学電子情報工学系数値解析研究室には日本データゼネラル社のミニコン ECLIPSE S/140 があり、このシステムでは FORTRAN（FORTRAN 5）によってマルチタスクプログラミングが可能である。このマルチタスクの機能を利用することによって、複数個のタスク（サブルーチン）を見かけ上同時に並列に実行させ、タスク間の交信を用い，共有メモリとの間でデータのやりとりを行うことができる。このシステムを利用して、さまざまな形の並列計算のシミュレーションを実行することができるが、ここでは手始めとして数値積分を例として取上げ、マルチタスクを利用した並列計算のプログラムと、その実行結果を示すことにする。なお、研究会当日には、研究会のテーマに合わせて線型計算の例として行列とベクトルの積に対する並列計算プログラムを示し、その解説も行ったが、ここではとくにタスクレベルでの並列計算の有効性を示すために、別の例として当日発表した数値積分をとり上げた。

2. 数値積分の並列処理

積分

$$I = \int_a^b f(x)\,dx$$

きざみ幅一定の台形則

$$I_h = h\left[\frac{1}{2}f(a) + \sum_{j=1}^{n-1} f(a+jh) + \frac{1}{2}f(b)\right]$$

を適用することを考える。この計算において、各関数値 $f(a+jh)$ はそれぞれ独立に計算できる量であるから，この計算をN個のプロセッサを使って並列に実行すれば、全体の計算時間はシリアルな計算の場合に比較してほぼ $1/N$ に短縮される．

N個のプロセッサを使用して I_h を計算するための一つの方法として、次のようなやり方が考えられる．まず、区間の左端からN個の標本点をとってそれぞれの標本点における f の計算をN個のプロセッサに割当てて計算を行う．そして、いずれかのプロセッサでの関数計算が終了したならば直ちに未計算のいずれかの標本点における関数計算を次々に実行させてゆく．この方法を実行するには、各プロセッサにおける処理の終了を識別し、順次次の計算を割当てるためのプロセッサが別に1個必要になる．数値積分を自動積分の形で実行しようとすると、きざみ幅 h を細分しながら上の手順全体を何回かくり返すことになる。

ここで報告する計算方法は上述のものとは異なり、N個のプロセッサに計算すべき標本点をあらかじめ定めた規則に従

って割当ててゆく方法である．ここでは、はじめから自動積分の形でプログラムを作成することを考える．使用するプロセッサの個数をNとする．台形則による数値積分の手順は原理的には通常のシリアルな計算機による積分の場合と同じである．すなわち、まず全区間 $b-a$ をN個の小区間に等分して台形則による第0ステップの計算を行う．次に各小区間を2等分して第1ステップの計算を行って前段での計算値と比較する．収束しているとみなせるならばその値を返し、収束していなければこの手順を収束するまでくり返す．これをN個のプロセッサをもつ並列計算機（別にプログラムの流れ全体を制御するプロセッサももちろんもつ）で実行するには、いずれのステップでも同時にN個の標本点で関数計算が行われるように計算の手順を構成すればよい．ただし、通常そうするように、数値積分では計算時間のほとんどが被積分関数の関数計算のために費やされると仮定している．

計算の具体的手順は次の通りである．関数計算を並列に実行するためのプロセッサの個数は2のべき乗、すなわち $N=2^{N_p}$ を仮定する．

(1) $v_{-1}=\frac{1}{2}(f(b)-f(a))$ を計算する．

(2) $h_0=(b-a)/N$

とおいて

$$f_j^{(0)} = f(a + jh_0), \quad j = 1, 2, \cdots, N$$

をN個のプロセッサで並列計算し，N個のプロセッサすべての計算が終了したならば

$$v_0 = h_0 \left[v_{-1} + \sum_{j=1}^{N} f_j^{(0)} \right]$$

を計算する．

(3) 次の手順を $M = 1, 2, \cdots$ についてくり返す．

$$h_M = \frac{b-a}{2^{M-1} \times N}$$

とおいて、j 番目のプロセッサで

$$f_j^{(M)} = f\left((j-1) \times h_0 + (k - \tfrac{1}{2}) h_M\right), \quad k = 1, 2, \cdots, 2^{M-1}$$

の関数計算を行うという操作を $j = 1, 2, \cdots, N$ において並列に行い、N個のプロセッサすべての計算が完了したならば

$$v_M = \frac{1}{2} v_{M-1} + \sum_{j=1}^{N} f_j^{(M)}$$

を計算する．そして

$$|v_M - v_{M-1}| \leq \text{許容誤差限界}$$

が満たされたならば v_M を積分値として採用してプログ

ラムの実行を終了し、満たされなければ次のMへ進む.

次に示すプログラムの中では、プロセッサ（関数を並列に計算するための）の数は $N=2^{N_p}=8$ （$N_p=3$）に設定し、被積分関数を計算する標本点数の上限を $256=2^8$ に限定している.

3. 並列計算による数値積分のプログラム

ここで使用するマルチタスク用の命令は、TASK, WAIT, WAKE UP, ANTICIPATE の4種類である. TASKは

TASK サブルーチン名（引数1, 引数2, ···）, ID＝タスクのID番号, PR＝プライオリティ

の形で使用する. サブルーチン名が同一のタスクを複数個生成することができるが, それらはタスクのID番号によって区別される. この命令が実行されると, 直ちにサブルーチン名とID番号に対応するタスクが生成され, そのタスクの実行が開始される. それと同時に、TASK命令の次の文も並列に実行される. TASK命令によって生成され実行が開始されたタスクは, その最後の文が実行されると消滅してしまう. すなわち, タスクの終りにRETURN文は存在しない. つまり, TASK命令では、生成時に引数の値をタスクに引渡すことはできるが, 計算結果を引数を通じて戻すこと

は原理的に不可能である．したがって，計算結果はタスクの中で結果が得られた時点で共有メモリにストアしておくようにしなければならない．

WAITはプログラムの実行を中断するための命令で

WAIT　イベント番号

の形で使用する．また，WAKE UPは中断されたプログラムの実行を再開するための命令で

WAKE UP　イベント番号

の形で使用する．要するに，これらはプログラムの同期をとるためのものである．たとえば、WAIT 1（イベント番号1）が実行されると直ちにそこでプログラムの実行は中断される．そして、他の場所でWAKE UP 1が実行されると、中断していた（イベント番号1によって）上のプログラムは上述のWAIT 1の次の文から実行が再開される．

並列処理を行っているプログラムでは，たとえばイベント番号1をもつWAIT 1の方が同じイベント番号1をもつWAKE UP 1よりも時間的に後に実行されてしまうこともありうる．この場合，WAIT 1以後の文は永久に実行が再開されなくなる．この事態を避ける命令がANTICIPATEで，これは

ANTICIPATE　イベント番号

の形で使用される．この命令を先に実行しておくと，WAIT

がそれと同じイベント番号をもつ WAKE UP よりも後に実行されても，プログラムは中断されずに実行が続行される．

次に示す主プログラムの中で，台形則による数値積分のサブルーチン TRAP が1つのタスクとして生成され，実行される。積分の下限Aと上限B，被積分関数の関数サブプログラム名 F1，および誤差の許容限界 EPS は TRAP の引数としてタスク TRAP に引渡しているが、積分の結果 T は上述の理由によって共有メモリ

```
COMMON / COMVAL / T
```

に書き込んで主プログラムに戻している．

台形則のサブルーチン TRAP で行っている操作は 2. で述べた通りである。各ΔMステップでプロセッサの数 N＝8 に等しい NTASK＝8個のタスク TSUB を生成している。そして、ΔMステップで各タスク（プロセッサ）に、関数値を計算すべき標本点を割当てる操作をサブルーチン PART(M) が行っている。各タスク TSUB のプログラムとしての内容はすべて同一であるが，異なる入力引数に対して異なる計算が行われるので，それを区別するために IB 番号 1から8までが付けられている．

また，各ΔMステップで，8個のタスク TSUB の各々が終了したか否かを TRAP の中でサブルーチン ANDCMP（AND 型

completion）が判定している．ANDCMPが実行されると，8個のタスクすべてが完了するまでそこでTRAPの実行は中断され，すべてのタスクが完了するとANDCMPの次の文から実行が再開される．以上述べた方法においても，次々にタスクを生成し，また結果を集計するサブルーチンのためのプロセッサが別に必要になる．いずれにしても，数値計算の立場からは，並列計算機のプロセッサの数は 2^m 個でなく，2^m+1 あるいは 2^m+2 個等とすべきであろう．

ここに示したANDCMPは，すべてのタスクが完了するまで待つというAND型の判定を行うものであるが，目的によってはいずれかのタスクが一つでも完了したならば次へ進むためのOR型の判定が必要になることもあろう．プログラムの最後に示したORCMP（OR型 completion）はそのような場合に役立つものである．

最後に，このプログラムによって，$a=A=-1$，$b=B=1$，$EPS=10^{-14}$，そして被積分関数F1を

$$f(x)=\frac{1}{2+\sin \pi x}$$

としたときの実行結果を示した．$f(x)$ は2を周期とする解析的周期関数であるから，台形則によってM=3回の反復（実際にはM=2回の反復）で目的の精度の結果が得られている．

4. ソースプログラム

次に全ソースプログラムとその実行結果を示す．主プログラムで定義しているTRAPはマルチタスクのうちの一つのタスクとしてではなく通常のサブルーチンの形で定義することも可能であるが、ここではタスクを多重に定義している例を示す目的で一つのタスクとして定義した．

```
C*************************************
C*                                   *
C*       MAIN PROGRAM OF TRAP        *
C*                                   *
C*        --- MULTI-TASK ---         *
C*                                   *
C*************************************
      COMMON/COMVAL/V
C     ****
      DOUBLE PRECISION A,B,F1,EPS
      DOUBLE PRECISION V
C     ****
      EXTERNAL F1
C     ****
      OPEN 5,"@CONSOLE"
      OPEN 6,"@LIST"
C     ****
      A=-1.0D0
      B= 1.0D0
      EPS=1.0D-14
C     ****
      ANTICIPATE 1
C     ****
      TASK TRAP(A,B,F1,EPS),ID=33,PRI=10
C     ****
      WAIT 1
C     ****
      WRITE(6,2001) V
 2001 FORMAT(1X/5X,'V =',1PD23.15)
C     ****
      CLOSE 5
      CLOSE 6
C     ****
      STOP
      END
```

```
      SUBROUTINE TRAP(A,B,FUNC,EPS)
C*****************************************
C*                                       *
C*      NUMERICAL INTEGRATION            *
C*       BY TRAPEZOIDAL RULE             *
C*                                       *
C*       --- MULTI-TASK ---              *
C*                                       *
C*****************************************
      COMMON/COMVAL/V
      COMMON/COMPAR/A1,B1,W,H
      COMMON/COMFVL/VAL(32),FV(256)
      COMMON/COMNUM/NTASK,MARR,MINT
C     ****
      DIMENSION LA(32)
C     ****
      DOUBLE PRECISION A,B,FUNC,EPS
      DOUBLE PRECISION V
      DOUBLE PRECISION VAL,FV,A1,B1,W,H
      DOUBLE PRECISION HV,V1
C     ****
      EXTERNAL FUNC
C     ****
      NTASK2=3
      NTASK=2**NTASK2
C     ****
      MARR2=8
      MARR=2**MARR2
C     ****
      MINT=MARR/NTASK
C     ****
      W=B-A
      H=W/DFLOAT(MARR)
      A1=A
      B1=B
      V=0.5D0*(FUNC(A)-FUNC(B))
C********* INITIAL STEP (M=0) *********
      M=0
      CALL PART(M)
C     ****
      DO 110 L=1,NTASK
      LA(L)=L
  110 CONTINUE
C     ****
      CALL CLEAR
C     ****
      ANTICIPATE 2
C     ****
      DO 10 L=1,NTASK
      TASK TSUB(LA(L),FUNC),ID=L,PRI=1
   10 CONTINUE
C     ****
      WRITE(6,2011) M
 2011 FORMAT(1X/1X,'** ',I1,'-TH STEP **'/
     1 1X,'TASKS WAITING (FROM TRAP)')
```

```
C      ****
       WAKE UP 2
C      ****
       WRITE(6,2012)
 2012 FORMAT(1X,'TASKS STARTED (FROM TRAP)')
C      ****
       CALL ANDCMP
C      ****
       WRITE(6,2013)
 2013 FORMAT(1X,'ALL TASKS FINISHED',
     1 ' (FROM TRAP)')
C      ****
       HV=W/DFLOAT(NTASK)
       DO 210 L=1,NTASK
       V=V+VAL(L)
  210 CONTINUE
C      ****
       V1=HV*V
       NV=NTASK+1
       WRITE(6,2021) V1,NV
 2021 FORMAT(1X/1X,'VALUE = ',1PD22.15,
     1 '  (NO. OF POINTS = ',I3,')')
C      ****
       MEND=MARR2-NTASK2
       DO 30 M=1,MEND
C********** M-TH STEP **********
       CALL PART(M)
C      ****
       CALL CLEAR
C      ****
       ANTICIPATE 2
C      ****
       DO 20 L=1,NTASK
       TASK TSUB(LA(L),FUNC),ID=L,PRI=1
   20 CONTINUE
C      ****
       WRITE(6,2011) M
C      ****
       WAKE UP 2
C      ****
       WRITE(6,2012)
C      ****
       CALL ANDCMP
C      ****
       WRITE(6,2013)
C      ****
       V=0.0
       DO 220 L=1,NTASK
       V=V+VAL(L)
  220 CONTINUE
```

```
C      ****
       HV=0.5D0*HV
       V=0.5D0*V1+HV*V
       NV=2*NV-1
       WRITE(6,2021) V,NV
C      ****
       IF(DABS(V-V1).LE.EPS) GO TO 31
       V1=V
   30  CONTINUE
C      ****
   31  CONTINUE
C      ****
       WAKE UP 1
C      ****
       END


       SUBROUTINE TSUB(L,FUNC)
C*************************************
C*                                   *
C*    L-TH TASK GENERATED BY TRAP    *
C*                                   *
C*************************************
       COMMON/COMPAR/A,B,W,H
       COMMON/COMFVL/VAL(32),FV(256)
       COMMON/COMKSL/KS(32),KL(32),KM(32)
       COMMON/COMFIN/LFIN(32)
C      ****
       DOUBLE PRECISION FUNC,A,B,W,H,VAL,FV
C      ****
       WAIT 2
C      ****
       WRITE(6,2031) L
 2031  FORMAT(1X,'  TASK ',I2,' STARTED')
C      ****
       KSV=KS(L)
       KLV=KL(L)
       KMV=KM(L)
C      ****
       VAL(L)=0.0D0
       DO 10 K=KSV,KLV,KMV
       FV(K)=FUNC(A+H*DFLOAT(K))
       VAL(L)=VAL(L)+FV(K)
   10  CONTINUE
C      ****
       LFIN(L)=1
C      ****
       WRITE(6,2035) L
 2035  FORMAT(1X,'  TASK ',I2,' FINISHED')
C      ****
       END
```

```
      SUBROUTINE PART(M)
C*************************************
C*                                   *
C*    THIS ROUTINE ASSIGNS SAMPLING  *
C*    POINTS TO TASKS IN THE M-TH    *
C*    STEP.                          *
C*                                   *
C*************************************
      COMMON/COMNUM/NTASK,MARR,MINT
      COMMON/COMKSL/KS(32),KL(32),KM(32)
C     ****
      LFF=12*256
C     ****
      IF(M.GE.1) GO TO 1
C     ****
      DO 10 L=1,NTASK
      KS(L)=MINT*L
      KL(L)=KS(L)
      KM(L)=1
   10 CONTINUE
C     ****
      GO TO 2
C     ****
    1 CONTINUE
      DO 20 L=1,NTASK
      KS(L)=MINT*(L-1)+MINT/2**M
      KL(L)=MINT*(L-1)+MINT/2**M*(2**M-1)
      KM(L)=MINT/2**(M-1)
   20 CONTINUE
C     ****
    2 CONTINUE
      WRITE(6,2101) LFF,
     1 M,(L,KS(L),KL(L),KM(L),L=1,NTASK)
 2101 FORMAT(A1/1X,'**** M=',I2/4(3X,'L',
     1 '  KS  KL KM')/
     2 (4(2X,I2,I4,I4,I3)))
C     ****
      RETURN
      END


      DOUBLE PRECISION FUNCTION F1(X)
C*************************************************
C*                                               *
C*          INTEGRAND FOR TRAP                   *
C*                                               *
C*************************************************
      DOUBLE PRECISION X
      F1=1.0D0/(DSIN(3.1415 92653 58979 32D0*X)
     1 +2.0D0)
      RETURN
      END
```

```
      SUBROUTINE CLEAR
C*************************************
C*                                   *
C*    THIS ROUTINE IS TO CLEAR THE   *
C*    SIGN OF COMPLETION OF THE      *
C*    TASKS.                         *
C*                                   *
C*************************************
      COMMON/COMFIN/LFIN(32)
      COMMON/COMNUM/NTASK
C     ****
      LOGICAL LFIN
C     ****
      DO 130 L=1,NTASK
      LFIN(L)=.FALSE.
  130 CONTINUE
C     ****
      RETURN
      END


      SUBROUTINE ANDCMP
C*****************************************
C*                                       *
C*    THIS ROUTINE IS TO KNOW WHETHER    *
C*    ALL THE TASKS ARE COMPLETED OR     *
C*    NOT.                               *
C*                                       *
C*****************************************
      COMMON/COMFIN/LFIN(32)
      COMMON/COMNUM/NTASK
C     ****
      LOGICAL LFIN,LGO
C     ****
  101 CONTINUE
      LGO=.TRUE.
      DO 120 L=1,NTASK
      LGO=LGO.AND.LFIN(L)
  120 CONTINUE
      IF(.NOT.LGO) GO TO 101
C     ****
      RETURN
      END
```

```
**** M= 0
  L  KS  KL KM   L  KS  KL KM   L  KS  KL KM   L  KS  KL KM
  1  32  32  1   2  64  64  1   3  96  96  1   4 128 128  1
  5 160 160  1   6 192 192  1   7 224 224  1   8 256 256  1

** 0-TH STEP **
TASKS WAITING (FROM TRAP)
TASKS STARTED (FROM TRAP)
  TASK  1 STARTED
  TASK  2 STARTED
  TASK  3 STARTED
  TASK  4 STARTED
  TASK  5 STARTED
  TASK  6 STARTED
  TASK  7 STARTED
  TASK  8 STARTED
  TASK  4 FINISHED
  TASK  5 FINISHED
  TASK  6 FINISHED
  TASK  7 FINISHED
  TASK  8 FINISHED
  TASK  1 FINISHED
  TASK  2 FINISHED
  TASK  3 FINISHED
ALL TASKS FINISHED (FROM TRAP)

VALUE =  1.154761904761904D 00  (NO. OF POINTS =   9)


**** M= 1
  L  KS  KL KM   L  KS  KL KM   L  KS  KL KM   L  KS  KL KM
  1  16  16 32   2  48  48 32   3  80  80 32   4 112 112 32
  5 144 144 32   6 176 176 32   7 208 208 32   8 240 240 32

** 1-TH STEP **
TASKS WAITING (FROM TRAP)
TASKS STARTED (FROM TRAP)
  TASK  1 STARTED
  TASK  2 STARTED
  TASK  4 STARTED
  TASK  5 STARTED
  TASK  6 STARTED
  TASK  7 STARTED
  TASK  8 STARTED
  TASK  1 FINISHED
  TASK  2 FINISHED
  TASK  3 STARTED
  TASK  4 FINISHED
  TASK  5 FINISHED
  TASK  6 FINISHED
  TASK  7 FINISHED
  TASK  8 FINISHED
  TASK  3 FINISHED
ALL TASKS FINISHED (FROM TRAP)

VALUE =  1.154700540009818D 00  (NO. OF POINTS =  17)
```

```
**** M= 2
  L  KS  KL KM   L  KS  KL KM   L  KS  KL KM   L  KS  KL KM
  1   8  24 16   2  40  56 16   3  72  88 16   4 104 120 16
  5 136 152 16   6 168 184 16   7 200 216 16   8 232 248 16

** 2-TH STEP **
TASKS WAITING (FROM TRAP)
TASKS STARTED (FROM TRAP)
  TASK  1 STARTED
  TASK  2 STARTED
  TASK  3 STARTED
  TASK  4 STARTED
  TASK  5 STARTED
  TASK  7 STARTED
  TASK  8 STARTED
  TASK  1 FINISHED
  TASK  2 FINISHED
  TASK  3 FINISHED
  TASK  4 FINISHED
  TASK  5 FINISHED
  TASK  8 FINISHED
  TASK  6 STARTED
  TASK  7 FINISHED
  TASK  6 FINISHED
ALL TASKS FINISHED (FROM TRAP)

VALUE =  1.1547005383792510 00  (NO. OF POINTS =  33)


**** M= 3
  L  KS  KL KM   L  KS  KL KM   L  KS  KL KM   L  KS  KL KM
  1   4  28  8   2  36  60  8   3  68  92  8   4 100 124  8
  5 132 156  8   6 164 188  8   7 196 220  8   8 228 252  8

** 3-TH STEP **
TASKS WAITING (FROM TRAP)
TASKS STARTED (FROM TRAP)
  TASK  1 STARTED
  TASK  2 STARTED
  TASK  3 STARTED
  TASK  4 STARTED
  TASK  5 STARTED
  TASK  6 STARTED
  TASK  7 STARTED
  TASK  8 STARTED
  TASK  8 FINISHED
  TASK  1 FINISHED
  TASK  3 FINISHED
  TASK  4 FINISHED
  TASK  5 FINISHED
  TASK  7 FINISHED
  TASK  2 FINISHED
  TASK  6 FINISHED
ALL TASKS FINISHED (FROM TRAP)

VALUE =  1.1547005383792510 00  (NO. OF POINTS =  65)

    V =  1.1547005383792510 00
```

```
      SUBROUTINE ORCMP
C*****************************************
C*                                       *
C*    THIS ROUTINE IS TO KNOW WHETHER    *
C*    ONE THE TASKS IS COMPLETED OR      *
C*    NOT.                               *
C*                                       *
C*****************************************
      COMMON/COMFIN/LFIN(32)
      COMMON/COMNUM/NTASK
C     ****
      LOGICAL LFIN,LGO
C     ****
      LGO=.FALSE.
  101 CONTINUE
      DO 120 L=1,NTASK
      LGO=LGO.OR.LFIN(L)
  120 CONTINUE
      IF(.NOT.LGO) GO TO 101
C     ****
      RETURN
      END
```