資　料

FORTRAN プログラムの動特性を把握する一手法について*

梅谷征雄** 高橋　栄** 渡辺　坦**

Abstract

We introduce here a measurement technique which enables us to get static and dynamic profile of a FORTRAN program. We can get these data after next steps.

(1) Insert new source statements for measurement (counting, formatting of outputs etc.) into adequate places of an original FORTRAN program.

(2) Compile the modified FORTRAN program.

(3) Execute the modified program.

The step (1) is performed by the computer program named FORMAP (FORTRAN Metrize and Analyse Program). We can get next statistics after execution.

- Execution count of every executable FORTRAN statement appeared in the program
- Static count and dynamic execution count of each statement type
- DO loop's profile
- Distribution of I/O intervals

The emphasis of this paper is on the technique of adding measurement facility to the original source program, and on the explanation of outputs. These outputs are expected to be used by hardware designers, compiler designers, and users who try to tune up their programs, etc.

The measurement technique described here is an extension of D. E. Knuth's idea shown in reference[1].

1. はじめに

計算機システムが利用者の諸要求を満足し，利用者に十分な便宜を与えうるためには，ハードウェアシステムの上で働くソフトウェアの動作が十分に把握されていなければならない．このような動作把握は，一方ではシステム設計者がソフトウェアの動作を考慮した効率の良いハードウェアシステムを設計するために，他方では利用者が与えられたハードウェア資源を有効に使いこなすために必要である．

システム設計者の立場に立つ場合，銀行などの専用オンラインシステムでは，ソフトウェアとして，いわゆる制御プログラムの動作が重要であるが，技術計算用のシステムではユーザプログラムの走行比率が高いことから，その部分の動特性も十分に把握する必要がある．

従来，制御プログラムの動作解析の手段として用いられてきた命令トレースないしイベントトレース手法はユーザプログラムの解析には用いにくい．

命令トレース手法は，データ量が多くなるのでユーザプログラムの解析には不適当である．またイベントトレースについては，ユーザプログラムは制御プログラムと異なり多様であるため，イベントの設定が煩わしい．

一方，ユーザプログラムは FORTRAN などの高水準言語で記述される場合が多いこと，それがコンパイラにより機械命令に落とされることから，制御プログラムとは異なる解析手法を考案することができる．

* On a Method to Obtain Dynamic Profile of FORTRAN Program by Yukio UMETANI, Sakae TAKAHASHI and Tan WATANABE (central Research Laboratory, Hitachi Ltd.)

** (株)日立製作所中央研究所

ここに紹介する FORMAP (FORTRAN Metrize and Analyse Program) は，FORTRAN で書かれたユーザプログラムの動特性を把握する一手法を提供する．

FORMAP は，FORTRAN 変換プログラムの一種であり，原プログラムに計測用の各種ステートメントと，計測結果を編集出力するためのサブルーチンを付加する．

FORMAP で処理した FORTRAN プログラムをコンパイル，実行すると，そのプログラムに関する次の情報が得られる．

（1） 各ステートメントごとの実行回数と，総和に対する比率

（2） ステートメントのタイプ別出現回数と，総和に対する比率（静的比率）

（3） ステートメントのタイプ別実行回数と，総和に対する比率（動的比率）

（4） DO ループの特性（全実行ステップに対する比率，ループ回数など）

（5） 入出力命令発行間隔

（1）は対象プログラムの動的プロフィルを与え，主に利用者により有効に利用されるものと予想する．

（2），（3）は FORTRAN 言語の利用状況に関する一般的な知見を与える一方，コンパイラの性能設計にも有効な情報を提供するであろう．

（4）は最近注目されつつある，いわゆるベクトル・プロセッサ[1),2)]の効果を算定するために利用できる．

（5）は，制御プログラムとの交信の頻度を与え，マルチプログラミング率を推定する根拠となる．

以下，本報告では FORMAP の仕様と方式に焦点を絞り，これを使った解析の結果については追って稿を改めて報告したい．

従来，FORTRAN プログラムの動特性を解析した例としては，Knuth[3)]がスタンフォード大学の33個の FORTRAN プログラムとロッキード社のプログラムを解析した結果が報告されている．FORMAP は，Knuth の解析の手法を自動化し，発展させたものである．

また文献4）に本報告に似た手法による FORTRAN プログラムの解析例が報告されており，参照させていただいた．

なお，最近，九大牛島氏から FORDAP[5)] の報告があり，参照させていただいた．FORDAP は主に利用者の立場から考案されたものであり，システム設計に役立てることを第1目的とする FORMAP とは計測項目，結果の編集方法などに若干の差があるが，基本的には両者は同じ手法で計測を行なっている．

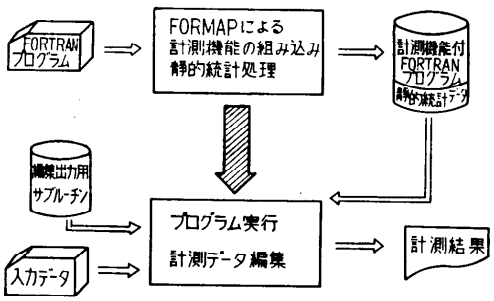


Fig. 1 Mesurement procedure using FORMAP

2. FORMAP による計測手続

FORMAP による FORTRAN プログラム特性の計測手続の大筋を **Fig. 1** に示す．

FORTRANにて書かれた原プログラムはFORMAPによって計測用のステートメントとサブルーチンコールを付加され，またステートメントのタイプ別出現回数などの静的統計データの収集が行われ，それらはまとめて外部記憶装置に書き出される．ついで，計測用ステートメントとサブルーチンコールを付加されたプログラムは通常の FORTRAN コンパイラによってコンパイルされ，編集出力用サブルーチンとリンクをとられて入力データと共に実行される．実行の結果として種々の動的統計データが計測され，それらは，実行の最後にコールされる編集出力サブルーチンの1つによって，先に収集された静的統計データと共に編集出力される．このように FORMAP はソースプログラムからソースプログラムへの変換を行うので，変換のプロセスと実行計測のプロセスは必ずしも同じ計算機のもとでランされる必要はない．このことは FORMAP 自身は特定の計算機システムでしか動作できなくても，計測の対象としては幾種類かの計算機システム（とそれに伴う言語仕様の幅）をカバーできる利点を生む．たとえば，大型計算機用の高水準 FORTRAN で書かれた原プログラムを，中小型計算機を使って計測可能なプログラムに変換することが可能である．

以下 Fig. 1 に示した計測手法の個々を順を追って説明する．

3. 計測機能の組み込み

3.1 ソースプログラムの変換

次の計測機能が原プログラムに付加される.

(1) 各ステートメントの実行回数をかぞえるカウンタの挿入.

すべての実行ステートメントの前に，その実行回数をカウントするステートメントを挿入する．実行ステートメントにステートメント番号があればそれをカウントステートメントにつけかえる．

```
 20  A=B+C
     GO TO 30
       ⇓
 20  I¥¥¥¥¥(13)=I¥¥¥¥¥(13)+1
     A=B+C
     I¥¥¥¥¥(14)=I¥¥¥¥¥(14)+1
     GO TO 30
```

(2) 入出力命令発行間隔ステートメント数をかぞえるカウンタの挿入.

入出力命令発行間隔を測定する指定がある場合に限り，入出力以外の実行ステートメントの前に全ルーチン共通のカウント・ステートメントを挿入し，入出力ステートメントの前にはそれを集計するサブルーチン T¥¥¥¥¥ のコールステートメントを挿入する.

```
      WRITE(6,100) DATA
 20   A=B+C
        ⋮
      REWIND 3
        ⇓
      I¥¥¥¥¥(12)=I¥¥¥¥¥(12)+1
      CALL T¥¥¥¥¥
      WRITE (6,100)  DATA
 20   I¥¥¥¥¥(13)=I¥¥¥¥¥(13)+1
      IO¥¥¥¥=IO¥¥¥¥+1
      A=B+C
        ⋮
      I¥¥¥¥¥(78)=I¥¥¥¥¥(78)+1
      CALL T¥¥¥¥¥
      REWIND 3
```

(3) 上記2種のカウンタの宣言文の挿入.

すべてのルーチンに，実行回数カウンタ I¥¥¥¥¥(*) と入出力命令発行間隔数カウンタ IO¥¥¥¥ の COMMON 宣言文を挿入する.

```
      PROGRAM EXAMPL
        ⋮
      END
      SUBROUTINE SUBSUB
        ⋮
      END
        ⇓
      PROGRAM EXAMPL
      COMMON/C¥¥¥¥¥/I¥¥¥¥¥(183)
      COMMON/D¥¥¥¥¥/IO¥¥¥¥
        ⋮
      END
      SUBROTINE SUBSUB
      COMMON/C¥¥¥¥¥/I¥¥¥¥¥(183)
      COMMON/D¥¥¥¥¥/IO¥¥¥¥
        ⋮
      END
```

(4) 統計データ編集用サブルーチンを呼ぶCALL文の挿入.

被測定プログラムの実行終了前に計測データを編集して出力するために，入出力発行間隔集計ルーチン T¥¥¥¥¥ の CALL 文に続いて，統計データ編集用サブルーチン S¥¥¥¥¥ の CALL 文を STOP 文の直前に挿入する.

```
 999  CALL
        ⇓
      I¥¥¥¥¥(124)=I¥¥¥¥¥(124)+1
      CALL T¥¥¥¥¥
      CALL S¥¥¥¥¥
      STOP
```

(5) READ ステートメントにエンド・ファイル処理の指定を挿入.

エンド・ファイル処理の指定がない READ ステートメントに対して，ファイル・エンド時に統計データ編集用サブルーチンに制御がわたるように，エンド・ファイル処理の指定を挿入する.

```
      READ(5,100) DATA
        ⇓
      I¥¥¥¥¥(37)=I¥¥¥¥¥(37)+1
      CALL T¥¥¥¥¥
      READ(5,100, END=30000) DATA
        ⋮
```

```
30000  CONTINUE
       CALL T¥¥¥¥¥
       CALL S¥¥¥¥¥
       STOP
```

(6) DO ループの端末文のステートメント番号を新しい番号に変える.

DO ループの端末文が CONTINUE ステートメント以外である場合，端末文の実行回数カウントステートメントに従来端末文の持っていたステートメント番号を付加すると，端末文が DO ループの外に出てしまうのでそれを防ぐためにステートメント番号を新設して，DO ステートメント中の端末指定ステートメント番号をそれで置きかえる．また従来の端末文の直後に新設したステートメント番号を有する CONTINUE ステートメントを挿入する.

```
       DO 50 J=1, N
       W=A(LL, J)
       A(LL, J)=A(I, J)
  50   A(I, J)=W
         ⇓
       I¥¥¥¥¥(101)=I¥¥¥¥¥(101)+1
       DO 30001 J=1, N
       I¥¥¥¥¥(102)=I¥¥¥¥¥(102)+1
       W=A(LL, J)
       I¥¥¥¥¥(103)=I¥¥¥¥¥(103)+1
       A(LL, J)=A(I, J)
  50   I¥¥¥¥¥(104)=I¥¥¥¥¥(104)+1
       A(I, J)=W
30001  CONTINUE
```

(7) ロジカル IF ステートメントを2つのステートメントに分ける.

ロジカルステートメントの場合，IF ステートメントの実行回数だけでなく，IF 条件が肯定された時に実行される後続文の実行回数も正しくかぞえるために，ロジカル IF ステートメントを2つのステートメントに分離する．分離時にステートメント番号を新設する.

```
       IF (A.EQ.B) D=C
         ⇓
       I¥¥¥¥¥(69)=I¥¥¥¥¥(69)+1
       IF (A.EQ.B)
      *GO TO 30008
       GO TO 30009
30008  I¥¥¥¥¥(70)=I¥¥¥¥¥(70)+1
       D=C
30009  CONTINUE
```

(8) カウンタの初期値などを与える Block Data を作成する.

個々の実行ステートメントの実行回数をかぞえるカウンタ・アレイの初期値，カウンタ・アレイの長さ，その他統計方法のオプションを指定したパラメータの設定を行う Block Data を組み込む.

2.2 編集出力用サブルーチン

FORMAP により計測機能を付加された FORTRAN プログラムが実行時にコールするサブルーチンのうち，代表的なものに次の2つがある.

(1) 計測結果集計編集サブルーチン(S¥¥¥¥¥)

実行の最後にコールされるサブルーチンで，原プログラムの変換時および実行時にそれぞれ収集された静的，動的統計データを編集し出力する.

(2) 入出力命令発行間隔集計ルーチン (T¥¥¥¥¥)

各入出力ステートメントが実行される直前に毎回呼ばれる．これは，前の入出力ステートメント実行から今回までに実行された（入出力以外の）ステートメント数をランク分けして，発行頻度表に加算するルーチンである.

4. 計測結果

FORMAP により計算機能を付加された FORTRAN プログラムを実行することにより得られた計測結果には次の5種類がある.

(1) 各ステートメントごとの実行回数と全実行ステートメント総数に対する比率.

(2) 実行文のタイプ別出現回数，実行回数と全体に対する比率.

(3) 非実行文のタイプ別出現回数と全体に対する比率.

(4) プログラム中にあらわれる DO ループの特性.

(5) 入出力命令発行間隔.

以下それらのおのおのを例に即して説明する．例題として取り上げたプログラムは，掃き出し法にて連立一次方程式を解く科学技術計算プログラムの一例である．これは，東京大学大型計算センタから提出されたベンチマークプログラム群中の一題である.

```
*****************************************************************
*****   EXECUTION COUNT FOR SOURCE STATEMENT   *****
*****************************************************************

EXECUTION      %      STATEMENT                 SOURCE STATEMENT                                    SEQ.NO.
  COUNT               TYPE CODE

                         77            PROGRAM JOB06
                         56            IMPLICIT REAL*8(A-H,O-Z)
                         55            IMPLICIT INTEGER (*)
                         60            DIMENSION A(50,52),AN(50)
     1       0.00780      7            CALL CLOCK (Y1,3)                                                     1
     1       0.00780      7            CALL CLOCK(ITIME1,3)                                                  2
     1       0.00780      1            N1=50                                                                 3
     1       0.00780      9            DO 10 M1=2,10                                                         4
     9       0.07019      1            NSTOP=0                                                               5
     9       0.07019      1            A(2,1)=1.0                                                            6
     9       0.07019      1            N111=N1+1                                                             7
     9       0.07019      9            DO 1 J=1,N111                                                         8
    63       0.49130      1               A(1,J)=1.0                                                         9
    63       0.49130      1            A(2,J+1)=A(2,J)+0.100                                                10
    63       0.49130     10          1 CONTINUE                                                             11
     9       0.07019      9            DO 2 J=1,N111                                                        12
    63       0.49130      9            DO 2 I=2,N1                                                          13
   375       2.92443      1            A(I,J)=A(2,J)*A(I-1,J)                                               14
   375       2.92443     10          2 CONTINUE                                                             15
     9       0.07019      9            DO 3 I=1,N1                                                          16
    54       0.42112      1            AN(I)=A(I,N1+1)                                                      17
    54       0.42112      1            A(I,1)=AN(I)                                                         18
    54       0.42112     10          3 CONTINUE                                                             19
     9       0.07019     14            WRITE(6,600) N1                                                      20
                         66        600 FORMAT(1H1///10X,9HORDER N1=,I3///10X,10HINPUT DATA,5X,8HA(N1,N1))
     9       0.07019      9            DO 4 I=1,N1                                                          21
    54       0.42112     14            WRITE(6,601) (A(I,J),J=1,N1)                                         22
    54       0.42112     10          4 CONTINUE                                                             23
                         66        601 FORMAT(1H ,8E15.7)
     9       0.07019     14            WRITE(6,602) (AN(I),I=1,N1)                                          24
                         66        602 FORMAT(///10X,10HINPUT DATA,5X,6HAN(N1),(/8E15.7))
     9       0.07019      7            CALL LINSW(A,AN,N1,NSTOP)                                            25
     9       0.07019     14            WRITE(6,603) (AN(I),I=1,N1)                                          26
                         66        603 FORMAT(///10X,8HSOLUTION,5X,6HAN(N1),(/6E15.7))
     9       0.07019     10         10 CONTINUE                                                             27
     1       0.00780      7            CALL CLOCK(ITIME2,3)                                                 28
     1       0.00780      1            ITIME2=ITIME2-ITIME1                                                 29
     1       0.00780     14            WRITE(6,99999) ITIME2                                                30
                         66      99999 FORMAT(1H1, 'COMPUTED TIME=' , I10)
     1       0.00780      7            CALL CLOCK (Y2,3)                                                    31
     1       0.00780      1            Y3=Y2-Y1                                                             32
     1       0.00780     14            WRITE (6,90909) Y3                                                   33
                         66      90909 FORMAT(1H1///// 35X,'COMPUTED TIME IS', I15,'   *100 MICRO SEC'
                                      * /// )
     1       0.00780     11            STOP                                                                 34
                         78            END
```

Fig. 2 Execution Count for source statement.

```
*************************************************************************
*****   STATIC(SOURCE) AND DYNAMIC(EXECUTION) COUNT FOR STATEMENT TYPE   *****
*************************************************************************

                    ******************************
                    ***   EXECUTABLE STATEMENT   ***
                    ******************************
```

TYPE CODE	STATEMENT TYPE	STATIC COUNT	%	DYNAMIC COUNT	%
1	ARITHMETIC ASSIGNMENT STATEMENT	68	41.21211	6031	47.03267
2	UNCONDITIONAL GO TO	7	4.24242	365	2.84645
3	ASSIGNED GO TO	0	0.00000	0	0.00000
4	COMPUTED GO TO	2	1.21212	62	0.48351
5	ARITHMETIC IF	16	9.69697	2995	23.35646
6	LOGICAL IF	0	0.00000	0	0.00000
7	CALL	5	3.03030	13	0.10138
8	RETURN	2	1.21212	13	0.10138
9	DO	14	8.48485	788	6.14521
10	CONTINUE	11	6.66667	2448	19.09068
11	STOP	1	0.60606	1	0.00780
12	PAUSE	0	0.00000	0	0.00000
13	SEQUENTIAL READ	0	0.00000	0	0.00000
14	SEQUENTIAL WRITE	15	9.09091	107	0.83444
15	REWIND	0	0.00000	0	0.00000
16	BACKSPACE	0	0.00000	0	0.00000
17	ENDFILE	0	0.00000	0	0.00000
18	DIRECT READ	0	0.00000	0	0.00000
19	DIRECT WRITE	0	0.00000	0	0.00000
20	FIND	0	0.00000	0	0.00000
21	ENCODE	0	0.00000	0	0.00000
22	DECODE	0	0.00000	0	0.00000
23	ASSIGN	0	0.00000	0	0.00000
24	PRINT	0	0.00000	0	0.00000
25	PUNCH	0	0.00000	0	0.00000
26	BUFFERIN AND BUFFEROUT	0	0.00000	0	0.00000
27	OTHER READ	0	0.00000	0	0.00000
28	OTHER WRITE	0	0.00000	0	0.00000
29	OTHER IF	0	0.00000	0	0.00000
30	OTHER EXECUTABLE STATEMENT	0	0.00000	0	0.00000
	TOTAL OF EXECUTED STATEMENTS	141		12823	

Fig. 3 Static and dy namic count for Executable statement type.

4.1 ステートメントごとの実行回数と全体に対する比率

Fig. 2(前頁参照)に出力例を示す．中央の欄には計測対象の FORTRAN プログラムをそのままの形で表示する．左端の Execurion Count 欄には各実行文の動的実行回数を表示する．その右側には，各ステートメントの実行回数の全実行回数に対する比率を % で表示する．さらにその右欄に各ステートメントの種別をコードで表示する．右端の欄にはプログラム中の実行文に一連番号をつけ表示する．

Fig. 2 は FORMAP により得られた計測結果の最も原始的かつ詳細な表示であり，以後に示す大部分の統計はこれを整理しなおして得られる．

4.2 実行文のタイプ別出現回数，実行回数と全体に対する比率

Fig. 3(前頁参照)に出力例を示す．左端の TYPE CODE は Fig. 2 の STATEMENT TYPE CODE に対応するものであり，その右欄に実行文の種別を，さらにその右欄に出現回数と実行回数を全体に対する比率と共にしるす．

この情報は，いわゆる FORTRAN のステートメントミックス[3]を求める上での基礎データとなろう．

4.3 非実行文のタイプ別出現回数と全体に対する比率

Fig. 4 に出力例を示す．Fig. 3 と同様である．ただし実行回数に関する統計はない．

4.3 プログラム中にあらわれる DO ループの特性

Fig. 5(次頁参照)に出力例を出す．これは技術計算用プログラムにおいて特に重要な DO ループの比率や形状などの特性を得るためのものである．

Fig. 5 にて左端の欄はネストのレベルを示す．すなわち最外側の DO ループのネストレベルは 1 で，内側に入るにしたがって 1 つずつ大きくなる．次の欄は DO ループ端末文のステートメント番号を示す．この 2 つで DO ループを指定する．次の Execution Count と % は，DO ループ内のステートメントの実行回数の総和と，プログラム内の全ステートメントの実行回数の総和に対する比率である．これは，ソース

```
**********************************************************************
*****  STATIC(SOURCE) AND DYNAMIC(EXECUTION) COUNT FOR STATEMENT TYPE  *****
**********************************************************************

                    ****************************
                    ***  NON-EXECUTABLE STATEMENT  ***
                    ****************************

TYPE      STATEMENT TYPE                          STATIC
CODE                                              COUNT

 51 INTEGER                                          0      0.00000
 52 REAL                                             0      0.00000
 53 COMPLEX                                          0      0.00000
 54 LOGICAL                                          0      0.00000
 55 IMPLICIT INTEGER                                 1      0.45000
 56 IMPLICIT REAL                                    2      1.21212
 57 IMPLICIT COMPLEX                                 0      0.00000
 58 IMPLICIT LOGICAL                                 0      0.00000
 59 DOUBLE AND QUADRUPLE PRECISION                   0      0.00000
 60 DIMENSION                                        2      1.21212
 61 COMMON                                           0      0.00000
 62 EQUIVALENCE                                      0      0.00000
 63 EXTERNAL                                         0      0.00000
 64 DATA                                             0      0.00000
 65 ENTRY                                            0      0.00000
 66 FORMAT                                          13      7.87879
 67 BLOCK DATA                                       0      0.00000
 68 DEFINE FILE                                      0      0.00000
 69 NAMELIST                                         0      0.00000
 70 FUNCTION                                         0      0.00000
 71 INTEGER FUNCTION                                 0      0.00000
 72 REAL FUNCTION                                    0      0.00000
 73 COMPLEX FUNCTION                                 0      0.00000
 74 LOGICAL FUNCTION                                 0      0.00000
 75 OTHER FUNCTION                                   0      0.00000
 76 SUBROUTINE                                       2      1.21212
 77 PROGRAM                                          1      0.60606
 78 END                                              3      1.81818
 79 DEBUG                                            0      0.00000
 80 STATEMENT FUNCTION DEFINING STATEMENT            0      0.00000
 81 OTHER SPECIFICATION STATEMENT                    0      0.00000
 82 OTHER NON-EXECUTABLE STATEMENT                   0      0.00000

TOTAL OF NON-EXECUTABLE STATEMENTS                  24

TOTAL OF EXECUTED AND NON-EXECUTABLE STATEMENTS     165
```

Fig. 4 Staticcount for non-executable statement type.

```
                              *****************************************
                              *****   DO LOOP PROFILE FOR ROUTINE  *****
                              *****************************************

PROGRAM JC806

NEST   LABEL    EXECUTION    %              WIDTH    DO RANGE    MEAN       DO LOOP NAME
LEVEL  OF DO     COUNT                      (C/L)    LENGTH(L)   COUNT(C)

  1      10        1381  10.76971           60.000     23        1380.000   1.
  2       1         196   1.54410            7.000      3          21.000   1.  1.
  2       2         822   6.41036           30.111      3          90.333   1.  2.
  3       2         813   6.34017            5.952      2          11.905   1.  2.  1.
  2       3         171   1.33354            6.000      3          18.000   1.  3.
  2       4         117   0.91242            6.000      2          12.000   1.  4.


SUBROUTINE LINSW(A,AM,M1,NSTOP)

NEST   LABEL    EXECUTION    %              WIDTH    DO RANGE    MEAN       DO LOOP NAME
LEVEL  OF DO     COUNT                      (C/L)    LENGTH(L)   COUNT(C)

  1      10          63   0.49130            6.000      1           6.000   2.
  1      30        3630  28.30850            9.460      7          66.222   3.
  2      20        3357  26.17952            2.866      5          14.329   3.  1.
  1      40         625   4.87405            8.000      3          24.000   4.
  1      50         555   4.32816            5.273      3          15.818   5.
  1      60         273   2.12899            4.056      1           4.056   6.
  1      70        1440  11.22952            4.053      1           4.053   7.
  1      80         369   2.87764            4.000     10          40.000   8.


SUBROUTINE CLOCK(I,J)

     NO DO LOOPS APPEAR
```

Fig. 5 Do loop's profile.

ステートメントを単位とした，プログラム内におけるその DO ループの動的比率を与える．次の Width は，DO ループの平均繰り返し数で，後に述べる Mean Count を DO Range Length で割って算出する．DO Range Length は，DO ループ内の見かけ上のステートメント数，Mean Count は DO 文1回あたりのループ内ステートメントの実行回数の総和である．DO ループの大きさの目安を与える．当然外側のループ程大きな値を与える．最後の DO Loop Name は，ネストのレベルを考慮して DO ループに名前を付け直したものである．

この例に示されるように，一般に技術計算プログラムにおいては，少数の DO ループの動的比率がきわめて高くなる性質がある．

4.4 入出力命令発行間隔

Fig. 6(次頁参照)に出力例を示す，入出力ステートメントの発行（間隔ソースステートメント単位）の頻度分布を対数尺にて求めたものである．これは，マルチプログラム効果の考察やバッチプログラムのモデル化を行う場合の基礎データとして利用できる．

5. オーバヘッドの評価

大小二種類のプログラムについて，原プログラムと計測機構を付加されたプログラムの必要メモリサイズと実行時間の例を **Table 1** に示す．

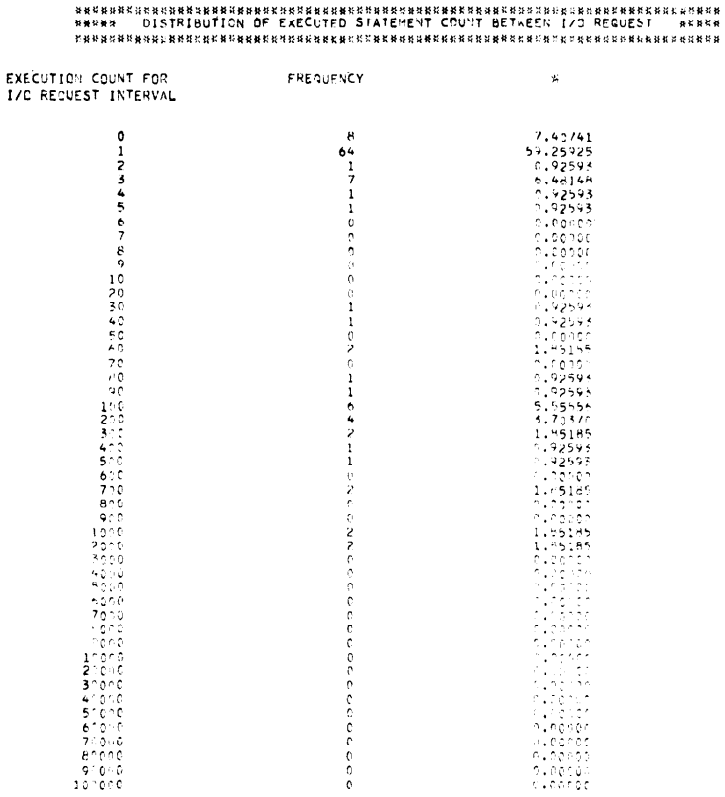
メモリオーバヘッドは，大小プログラム共に2倍(100%) 程度，時間オーバヘッドは小さなプログラムで 2〜3 倍，大きなプログラムで 20% 前後である．

6. む す び

以上，FOPTRAN プログラムの動特性を計測する

Table 1 Estimation of space and time overhead by the use of FORMAP

		原プログラム	計測機構を付加されたプログラム
小さなプログラム	ステップ数	350	950
	メモリ量	37 kB	70 kB
	CPU TIME	13 sec	33 sec
大きなプログラム	ステップ数	2500	5500
	メモリ量	256 kB	500 kB
	CPU TIME	18 min. 28 sec.	21 min. 31 sec.



DISTRIBUTION OF EXECUTED STATEMENT COUNT BETWEEN I/O REQUEST

EXECUTION COUNT FOR I/O REQUEST INTERVAL	FREQUENCY	%
0	8	7.407741
1	64	59.259259
2	1	0.925925
3	7	6.481481
4	1	0.925925
5	1	0.925925
6	0	0.000000
7	0	0.000000
8	0	0.000000
9	0	0.000000
10	0	0.000000
20	0	0.000000
30	1	0.925925
40	1	0.925925
50	0	0.000000
60	2	1.851851
70	0	0.000000
80	1	0.925925
90	1	0.925925
100	6	5.555555
200	4	3.703703
300	2	1.851851
400	1	0.925925
500	1	0.925925
600	0	0.000000
700	2	1.851851
800	0	0.000000
900	0	0.000000
1000	2	1.851851
2000	2	1.851851
3000	0	0.000000
4000	0	0.000000
5000	0	0.000000
6000	0	0.000000
7000	0	0.000000
8000	0	0.000000
9000	0	0.000000
10000	0	0.000000
11000	0	0.000000
12000	0	0.000000
13000	0	0.000000
14000	0	0.000000
15000	0	0.000000
16000	0	0.000000
17000	0	0.000000
18000	0	0.000000
19000	0	0.000000
20000	0	0.000000

Fig. 6 Distribution of executed statement count between I/O Request.

--手段を与える FORMAP について，計測の方式と計測結果を中心に紹介した．

FORMAP は，ソースステートメントを単位とした計測を行うので，個々のプログラムの具体的な処理との関係がつけやすく，システム解析者のみならずユーザにも有益な情報を提供する．その計測データは，ハードウェア機構の設計，FORTRAN コンパイラの設計，バッチシステムのモデル化，あるいはユーザによるプログラムの性能上の Tune Up など多目的に利用可能と思われる．

FORMAP の欠点としては，ソースプログラムレベルで計測機構を付加しておいて，それをコンパイル・実行するという二段の手続きをとるために，ジョブコントロール・カードの用意など使用手続きがやや煩雑であること，計測のためのメモリオーバヘッド，時間オーバヘッドが大きいことが挙げられる．これらの克服は将来の課題である．使用手続きの煩雑さに対しては，FORMAP と同様の機能をコンパイラのオプションとして持つ方式も考えられる．また計測のためのオーバヘッドに対しては，文献4）で行われたように，制御の流れが一本のステートメント群に対しては，先頭のステートメントにのみカウントステートメントを削入する方式が考えられる．

最後に，本研究の遂行にあたって，熱意と適確な技量を持ってプログラムの開発にあたられた NBC 日立製作所中央研究所分室の古屋典子嬢，開発の過程で適切な助言をいただいた日立製作所中央研究所堀越主任研究員，開発の機会を与えていただいた同研究所堤第7部長，および計算機の使用その他の点で開発に御尽力いただいた同研究所新井元計算センタ長に厚く感謝の意を表します．

参 考 文 献

1) CDC STAR-100 Computer, Hardware Reference Manual, Control Data Corporation, Publicatior No. 60256000
2) A Description of the Advanced Scientific Computer System, Texas Instruments Incorporated
3) D. E. Knuth: An Empirical Study of FORTRAN programs. Software-Practice & Experience, Vol. 1, pp. 105～133 (1971)
4) 恒川: FORTRAN Analyser について，昭和46年度情報処理学会大会誌 p. 261～262
5) 藤村，牛島: FORTRAN プログラムの動的解析プログラム，昭和 50 年度情報処理学会講演論文集 No. 29

(昭和50年9月18日受付)
(昭和51年6月11日再受付)