SR8000 性能モニターとログメッセージを用いたチューニング

（株）日立製作所

１．はじめに

これまで、本「スーパーコンピューティングニュース」の以下の記事において、SR8000 の診断ログメッセージ出力機能、および、性能モニター機能についてご紹介させて頂きました。

「SR8000 の有効利用法」　Vol.1, No.3 1999

「SR8000 性能モニター機能の利用法」　Vol.1, No.4 1999

この 2 つの機能を用いてプログラムをチューニングすることで、より効果的に SR8000 を使用することが可能になります。

ここでは、診断ログメッセージ（以下、ログメッセージと呼びます）と性能モニターを使用して、実際にプログラムをチューニングする過程を紹介しながら、SR8000 の有効的な使用方法を説明します。また、性能モニター機能を使用する上での注意点について説明します。

２．性能モニターとログメッセージを用いたプログラムチューニング

プログラムの実行時間を短縮し性能を向上させるには、SR8000 の性能モニター機能、及びコンパイラーがソースプログラムと融合して出力するログメッセージを活用してプログラムチューニングを行うことが有効です。

本章では、これら 2 つのツールの使い方を紹介しながら、次に示す例題プログラムをチューニングしてみます。

```
      PROGRAM MAIN
      PARAMETER(NN=10000000)
      PARAMETER(NN2=3)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NN),B(NN),C(NN),L(NN)
      DIMENSION D(NN2)
C
      WRITE(6,*) 'Test start'
      DO 5 I=1,60
         CALL SUB1(NN,A,B,L,I)
         CALL SUB2(NN,A,B,C,L)
         CALL SUB3(NN,A,B,C,L)
         CALL SUB4(NN,A,B,C,D,NN2)
         CALL SUB5(D,NN2,S)
         WRITE(6,*) I,'Sum=',S
    5 CONTINUE
      WRITE(6,*) 'Test end'
      STOP
      END
```

図 2.1　例題プログラム sample.f

図 2.1 に示す例題プログラム sample.f は 5 つのサブルーチン SUB1、SUB2、SUB3、SUB4、

SUB5 から構成されています。それぞれのサブルーチンはソースファイル sub1.f、sub2.f、sub3.f、sub4.f、sub5.f に記述されているとします。5 つのサブルーチンの処理内容については、チューニングの際に、ログメッセージと合わせた形で示します。

2.1 プログラム全体の把握

まず、プログラム全体の実行時間の内訳を調べます。

処理の重いサブルーチンを特定するため、関数／サブルーチン単位の実行時間を採取する性能モニターのオプション -Xfuncmonitor を指定し、さらに、DO ループが要素並列化されているか、擬似ベクトル化されているかを判断するために、ログメッセージを出力するオプション-optlog も指定してコンパイルをします（図 2.2）。

```
prompt> f77 -0ss -procnum=8 -optlog -c sample.f -Xfuncmonitor
prompt> f77 -0ss -procnum=8 -optlog -c sub1.f -Xfuncmonitor
prompt> f77 -0ss -procnum=8 -optlog -c sub2.f -Xfuncmonitor
prompt> f77 -0ss -procnum=8 -optlog -c sub3.f -Xfuncmonitor
prompt> f77 -0ss -procnum=8 -optlog -c sub4.f -Xfuncmonitor
prompt> f77 -0ss -procnum=8 -optlog -c sub5.f -Xfuncmonitor
```

図 2.2　例題プログラムのコンパイル

コンパイル終了後、ログツール（loggen コマンド）によりログメッセージを作成します（図 2.3）。

```
prompt> loggen sample.f  →  sample.log が作成されます。
prompt> loggen sub1.f  →  sub1.log が作成されます。
prompt> loggen sub2.f  →  sub2.log が作成されます。
prompt> loggen sub3.f  →  sub3.log が作成されます。
prompt> loggen sub4.f  →  sub4.log が作成されます。
prompt> loggen sub5.f  →  sub5.log が作成されます。
```

図 2.3　ログメッセージの作成

この結果、ログメッセージファイル（*.log）がソースファイルと同じディレクトリー（コンパイルを実行したディレクトリー）に作成されます。

性能モニターを使用するのでリンク時にはオプション-lpl -parallel を追加指定します（図 2.4）。

```
prompt> f77 sample.o sub1.o sub2.o sub3.o sub4.o sub5.o -lpl -parallel -o sample
```

図 2.4　例題プログラムのリンク

最後に、作成されたロードモジュールを実行します。本例題では実行性能情報ファイルは

テキスト形式で出力することにします。そこで、環境変数 APDEV_OUTPUT_TYPE に TEXT を設定して、プログラムを実行します（図 2.5）。

```
prompt> setenv APDEV_OUTPUT_TYPE TEXT   （注意：C シェルの場合の記述です。）
prompt> ./sample
```

図 2.5　例題プログラムの実行

プログラム実行後、実行性能情報ファイル pl_sample_24678_0.txt が出力されます（ファイル名の 24678 は実行時のプロセス番号です）。図 2.6 にその内容を示します。

```
(Exe.R %)   :Execution ratio
(Par.E %)   :Parallel efficiency
(Max. total):Total of the Max User Time for each info (=CPU elapse time).
< Process Information >
  Node no.                        :                  0
  Process no.                     :              24678
  Load file name                  : sample
  User Time [sec]    (Max. total) :           53.403911 (Par.E  12.50 %)  … (a)

～  途中省略  ～

< Function Information >
  Function name                   : MAIN
  Source file name                : sample.f
  Line no.                        :                  8
  Execution count                 :                  1
                           (0-2) :                  1                  0                  0
                           (3-5) :                  0                  0                  0
                           (6-7) :                  0                  0
  User Time [sec]          (Max) :           0.023781 (Exe.R   0.04 %) (Par.E  12.50 %)  … (b)

～  途中省略  ～

< Function Information >
  Function name                   : SUB1
  Source file name                : sub1.f
  Line no.                        :                  4
  Execution count                 :                 60
                           (0-2) :                 60                  0                  0
                           (3-5) :                  0                  0                  0
                           (6-7) :                  0                  0
  User Time [sec]          (Max) :          16.845119 (Exe.R  31.54 %) (Par.E  12.50 %)  … (c)

～  途中省略  ～

< Function Information >
  Function name                   : SUB2
  Source file name                : sub2.f
  Line no.                        :                  4
  Execution count                 :                 60
                           (0-2) :                 60                  0                  0
                           (3-5) :                  0                  0                  0
                           (6-7) :                  0                  0
  User Time [sec]          (Max) :          16.824714 (Exe.R  31.50 %) (Par.E  12.50 %)  … (d)
```

```
～ 途中省略 ～

< Function Information >
  Function name                 : SUB3
  Source file name              : sub3.f
  Line no.                      :          3
  Execution count               :         60
                        (0-2) :         60                0                0
                        (3-5) :          0                0                0
                        (6-7) :          0                0
  User Time [sec]       (Max) :   9.861240 (Exe.R  18.47 %) (Par.E  12.50 %)  … (e)

～ 途中省略 ～

< Function Information >
  Function name                 : SUB4
  Source file name              : sub4.f
  Line no.                      :          5
  Execution count               :         60
                        (0-2) :         60                0                0
                        (3-5) :          0                0                0
                        (6-7) :          0                0
  User Time [sec]       (Max) :   9.848676 (Exe.R  18.44 %) (Par.E  12.50 %)  … (f)

～ 途中省略 ～

< Function Information >
  Function name                 : SUB5
  Source file name              : sub5.f
  Line no.                      :          4
  Execution count               :         60
                        (0-2) :         60                0                0
                        (3-5) :          0                0                0
                        (6-7) :          0                0
  User Time [sec]       (Max) :   0.000381 (Exe.R   0.00 %) (Par.E  12.50 %)  … (g)

～ 以下省略 ～
```

図 2.6　**例題プログラムの実行性能情報ファイル**

図 2.6 **における、**< Function Information > **の項目** "User Time [sec]" **（図中の（**b**）から（**g**））に、各サブルーチンの実行時間が採取されています。これらをまとめて、例題プログラムの実行時間の内訳として表** 2.1 **に示します。**

表 2.1 **各サブルーチンの実行時間と全体に対する割合**

	実行時間 [sec]	**全体に対する割合** [%]
MAIN	0.023781	0.04
サブルーチン SUB1	16.845119	31.54
サブルーチン SUB2	16.824714	31.50
サブルーチン SUB3	9.861240	18.47
サブルーチン SUB4	9.848676	18.44
サブルーチン SUB5	0.000381	0.00
合計	53.403911	100.00

表 2.1 より、実行時間がプログラム全体に占める割合の高いサブルーチン SUB1、SUB2、SUB3、SUB4 をチューニング対象とします。MAIN とサブルーチン SUB5 は全体の実行時間に対する割合が 1%以下と低いため、全体の性能にほとんど影響を与えません。

2.2 サブルーチン SUB1 のチューニング

表 2.1 より、SUB1 は全体の実行時間に対する割合が 31.54[%]と最も高いので、まずこのサブルーチンのチューニングを行います。

始めに、ログツールを使用（図 2.3）して作成された、ログメッセージファイル sub1.log の内容（図 2.7）を調べ、DO ループが要素並列化されているか、擬似ベクトル化されているかを確認します。

```
      SUBROUTINE SUB1(NN,A,B,L,NC)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NN),B(NN),L(NN)

** ループ入口で並列処理 開始
** 並列手続き名 : _parallel_func_1_SUB1
** 並列ループ
** ループ出口で並列処理 終了
**
**    [DO 10]
XX    対象となる配列参照が無いので擬似ベクトル化を適用しなかった。
**
      DO 10 I=1,NN
         L(I)=I
         A(I)=DBLE(NC)*DBLE(I)*1.0D-4
         B(I)=EXP(A(I)*1.0D-5)
   10 CONTINUE
      RETURN
      END
```

図 2.7　ログメッセージファイル sub1.log の内容

ログメッセージファイルをみると、サブルーチン SUB1 のループ（DO 10）は要素並列化されていることが分かります。また、このループでは配列参照をしていない（配列 A はループ中で定義されている）ので擬似ベクトル化は適用されませんがこれは問題ありません。

ここでは、倍精度の数学関数 EXP に着目します。べき乗、EXP、LOG、SIN、COS などの数学関数は擬似ベクトル化数学関数の対象ですが、ログメッセージファイルを見ると DO 10 に対しては適用されていないことが分かります（擬似ベクトル化数学関数が適用された場合には、「ベクトルライブラリ化を行った。」というメッセージが出ます）。これは、EXP の引数が式（A(I)*1.0D-5）になっているためです。このような場合は、ディレクティブ *voption pvfunc(2) を指定します（コンパイルオプション-pvfunc=2 を指定して sub1.f をコンパイルしても同様な動作をします）。

ディレクティブ挿入後、sub1.f を再コンパイルしログメッセージファイルを作成すると、図 2.8 に示すように、擬似ベクトル化数学関数（_hf_pvdexp）が適用されたことが確認できます。

```
      SUBROUTINE SUB1(NN,A,B,L,NC)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NN),B(NN),L(NN)
*voption pvfunc(2)

** ループ入口で並列処理 開始
** 並列手続き名 : _parallel_func_1_SUB1
** 並列ループ
** ループ出口で並列処理 終了
**
**    [DO 10]
**    ベクトルライブラリ化を行った(_hf_pvdexp)。
**    最内側ループ展開(2 倍)を行った。
XX    対象となる配列参照が無いので擬似ベクトル化を適用しなかった。
**
      DO 10 I=1,NN
         L(I)=I
         A(I)=DBLE(NC)*DBLE(I)*1.0D-4
         B(I)=EXP(A(I)*1.0D-5)
   10 CONTINUE
      RETURN
      END
```

図 2.8　ディレクティブ挿入後のログメッセージファイル sub1.log の内容

次に、この変更による効果を確認します。ディレクティブを挿入した sub1.f を使ってロードモジュールを作成、実行したときの実行性能情報ファイルの内容を図 2.9 に示します。

```
(Exe.R %)   :Execution ratio
(Par.E %)   :Parallel efficiency
(Max. total):Total of the Max User Time for each info (=CPU elapse time).
< Process Information >
  Node no.                        :                  0
  Process no.                     :              24902
  Load file name                  : sample
  User Time [sec]    (Max. total) :          41.455374 (Par.E  12.50 %)  … (h)

～  途中省略  ～

< Function Information >
  Function name                   : SUB1
  Source file name                : sub1.f
  Line no.                        :                  5
  Execution count                 :                 60
                           (0-2) :                 60                  0                  0
                           (3-5) :                  0                  0                  0
                           (6-7) :                  0                  0
  User Time [sec]          (Max) :           4.925110 (Exe.R  11.88 %) (Par.E  12.50 %)  … (i)

～  以下省略  ～
```

図 2.9　ディレクティブ挿入後の実行性能情報

図 2.6 の(a)、(c)と図 2.9 の(h)、(i)の比較を表 2.2 に示します。

表 2.2　サブルーチン SUB1 のチューニング効果

	チューニング前の実行時間 [sec]	チューニング後の実行時間 [sec]
全体	53.403911	41.455374
サブルーチン SUB1	16.845119	4.925110

表 2.2 よりサブルーチン SUB1 が 3 倍以上高速化され、全体実行時間が約 20%短縮したことが分かります。

次に、サブルーチン SUB1 は要素並列化されているので、負荷バランスが良いかを調べるため、性能モニター機能の-Xparmonitor オプションを用いてコンパイル、実行し、要素並列化単位の性能情報を採取します（図 2.10）。

```
prompt> f77 -Oss -procnum=8 -optlog -c sub1.f -Xparmonitor
prompt>f77 sample.o sub1.o sub2.o sub3.o sub4.o sub5.o -lpl -parallel -o sample
prompt> ./sample
```

図 2.10　要素並列化単位の性能情報採取方法

再コンパイルして実行した結果、出力される実行性能情報ファイルを図 2.11 に示します。

```
～　途中省略　～

< Parallel Information >
  Function name                 : SUB1
  Source file name              : sub1.f
  Line no.                      :                5
  Execution count               :              480
                       (0-2) :               60                60                60
                       (3-5) :               60                60                60
                       (6-7) :               60                60
  User Time [sec]      (Max) :            4.919381 (Exe.R  11.86 %) (Par.E  99.97 %)  … (j)
                     (Total) :           39.342258 (Exe.R  51.84 %)
                       (0-2) :            4.919381          4.917095          4.917553
                       (3-5) :            4.917294          4.918263          4.917259
                       (6-7) :            4.918280          4.917134
  LD/ST                        :         5351763360 (Exe.R  52.62 %) (Par.E 100.00 %)  … (k)
                       (Max) :          668970420
                       (0-2) :          668970420         668970420         668970420
                       (3-5) :          668970420         668970420         668970420
                       (6-7) :          668970420         668970420
  Instruction count            :        21801649093 (Exe.R  69.05 %) (Par.E 100.00 %)  … (l)
                       (Max) :         2725208184
                       (0-2) :         2725208184        2725205884        2725206166
                       (3-5) :         2725206139        2725205719        2725205634
                       (6-7) :         2725205686        2725205681
  Floating-point arithmetic count :    18001300371 (Exe.R  83.91 %) (Par.E 100.00 %)  … (m)
                       (Max) :         2250163485
                       (0-2) :         2250163240        2250162367        2250162539
                       (3-5) :         2250163485        2250162197        2250162161
                       (6-7) :         2250162289        2250162093
  MIPS                         :         4431.787262
                       (0-2) :          553.973824        554.230890        554.179323
```

```
                    (3-5) :      554.208557      554.099186      554.212353
                    (6-7) :      554.097305      554.226481
  MFLOPS                  :     3659.261432
                    (0-2) :      457.407820      457.620284      457.577693
                    (3-5) :      457.602029      457.511532      457.604979
                    (6-7) :      457.510003      457.616622
～　以下省略　～
```

図 2.11　要素並列化単位の実行性能情報

図 2.11 **の**(j)**から**(m)**の** Parallel efficiency**(並列化効率)がほぼ** 100%**であることより** SUB1 **の要素並列化箇所の** IP（Instruction Processor）**間の負荷バランスは均等であることが確認できます。**

以上のように、サブルーチン SUB1 **のチューニングは**

(1) ログメッセージファイルの確認

(2) 性能モニターによる性能情報ファイルの確認

により実施し、効果の確認も行いました。他のサブルーチンに関しても同様にチューニングしていきます。

2.3 サブルーチン SUB2 のチューニング

表 2.1 **において、サブルーチン** SUB2 **が全体実行時間に占める割合は** 31.50[%]**です。**

まず、ログメッセージファイル sub2.log **を見て、**DO **ループが要素並列化されているか、擬似ベクトル化されているかを確認します（図** 2.12）。

```
      SUBROUTINE SUB2(NN,A,B,C,L)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NN),B(NN),C(NN),L(NN)

** ループ入口で並列処理 開始
** 並列手続き名 : _parallel_func_1_SUB2
** 並列ループ
**    I : TLOCAL 変数
      DO 20 J=1,NN/500-1
         C(J)=0.0D0

**
**    [DO 30]
**    最内側アキュムレータ変数展開(8 倍)を行った。
**    擬似ベクトル化を適用した。
**
         DO 30 I=1,L(J)
            C(J)=C(J)+A(I)+B(I)
   30    CONTINUE
   20 CONTINUE
c

** 並列処理を継続
** 並列ループ
** --- ループ入口でバリア出力 ---
** ループ出口で並列処理 終了
**
**    [DO 40]
**    最内側ループ展開(8 倍)を行った。
**    擬似ベクトル化を適用した。
```

```
**
      DO 40 J=NN/500,NN
         C(J)=A(J)+B(J)
   40 CONTINUE
      RETURN
      END
```

図 2.12　ログメッセージファイル sub2.log の内容

ログメッセージファイルより、サブルーチン SUB2 の 2 つのループ（DO20 と DO40）は共に要素並列化されていることが分かります。また、擬似ベクトル化も適用されています。ログメッセージファイルには性能上の問題点を示すメッセージはありません。

次にサブルーチン SUB2 は要素並列化箇所があるので性能モニター機能の -Xparmonitor オプションを用いて要素並列化単位の性能情報を採取します。図 2.10 と同様の手順で取得した実行性能情報ファイルを図 2.13 に示します。

```
～　途中省略　～

< Parallel Information >
  Function name                  : SUB2
  Source file name               : sub2.f
  Line no.                       :                4
  Execution count                :              480
                          (0-2) :               60                60                60
                          (3-5) :               60                60                60
                          (6-7) :               60                60
  User Time [sec]           (Max) :        16.715202 (Exe.R  40.34 %) (Par.E  99.76 %)  … (n)
                          (Total) :       133.404793 (Exe.R  84.37 %)
                          (0-2) :        16.715202         16.696189         16.696424
                          (3-5) :        16.686933         16.676125         16.628834
                          (6-7) :        16.658848         16.646238
  LD/ST                          :     14515370640 (Exe.R  73.81 %) (Par.E  55.15 %)  … (o)
                          (Max) :      3289803360
                          (0-2) :       338008440         759905820        1181780460
                          (3-5) :      1603655820        2025530460        2447405820
                          (6-7) :      2869280460        3289803360
  Instruction count              :     42336068702 (Exe.R  78.45 %) (Par.E  54.46 %)  … (p)
                          (Max) :      9718106245
                          (0-2) :       862837067        2128491918        3394106080
                          (3-5) :      4659715533        5925324577        7190937140
                          (6-7) :      8456550142        9718106245
  Floating-point arithmetic count :     24615347091 (Exe.R  82.42 %) (Par.E  53.98 %)  … (q)
                          (Max) :      5699788956
                          (0-2) :       452245571        1202239525        1952233217
                          (3-5) :      2702223719        3452212981        4202205309
                          (6-7) :      4952197813        5699788956

～　以下省略　～
```

図 2.13　SUB2 の要素並列化単位の実行性能情報

図 2.13 の(o)から(q)の Parallel efficiency をみると約 50%であり、SUB2 の要素並列化箇所の IP 間の負荷バランスが悪いことが分かります。そこで、ソースファイル sub2.f（図 2.12）の DO 20 と DO 30 の２重ループについて調査します。DO 30 のループ長 L(J) は外側の DO

20 のインデックス J に依存しています。また、配列 L はサブルーチン SUB1(図 2.8)で L(I)=I と定義されています。

要素並列化は外側のループ DO 20 に対して行われており、DO 20 のループ長(=NN/500-1)は 19999(図 2.1 のプログラムの MAIN 部分で NN=10000000 と定義されている)なので、各 IP の担当する DO 20 のインデックスと DO30 のループ長は表 2.3 のようになります。

表 2.3　各 IP の担当する DO20 のインデックスと DO30 のループ長の関係

計算をする IP 番号	DO20 のインデックス J	DO30 のループ長 L(J)
0	1	L(1)=1
0	2	L(2)=2
0	3	L(3)=3
0	:	:
0	2500	L(2500)=2500
1	2501	L(2501)=2501
1	2502	L(2502)=2502
:	:	:
7	19998	L(19998)=19998
7	19999(=NN/500-1)	L(19999)=19999

表 2.3 より、IP0 は主に DO30 のループ長の短い部分を計算し、IP7 はループ長の長い部分を担当しているので、IP 間の負荷バランスが悪いことが分かります。

この場合、表 2.4 に示すように要素並列化すれば、IP 間の負荷バランスをほぼ均等にすることができます。

表 2.4　IP 間負荷バランスのよい分割

計算をする IP 番号	DO20 のインデックス J	DO30 のループ長 L(J)
0	1	L(1)=1
1	2	L(2)=2
2	3	L(3)=3
:	:	:
3	2500	L(2500)=2500
4	2501	L(2501)=2501
5	2502	L(2502)=2502
:	:	:
5	19998	L(19998)=19998
6	19999(=NN/500-1)	L(19999)=19999

表 2.4 のようにループを要素並列化する分割方法をサイクリック分割といい、該当ループ

にディレクティブ*poption cyclic を挿入することにより実現できます。

ディレクティブ挿入後 sub2.f を再コンパイルして作成したログメッセージファイルを図 2.14 に示します。

```
      SUBROUTINE SUB2(NN,A,B,C,L)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NN),B(NN),C(NN),L(NN)
*poption cyclic

** ループ入口で並列処理 開始
** 並列手続き名 : _parallel_func_1_SUB2
** 並列ループ（サイクリック分割）
**    I : TLOCAL 変数
      DO 20 J=1,NN/500-1
         C(J)=0.0D0

**
**    [DO 30]
**    最内側アキュムレータ変数展開(8 倍)を行った。
**    擬似ベクトル化を適用した。
**
         DO 30 I=1,L(J)
            C(J)=C(J)+A(I)+B(I)
   30    CONTINUE
   20 CONTINUE
c

** 並列処理を継続
** 並列ループ
** --- ループ入口でバリア出力 ---
** ループ出口で並列処理 終了
**
**    [DO 40]
**    最内側ループ展開(8 倍)を行った。
**    擬似ベクトル化を適用した。
**
      DO 40 J=NN/500,NN
         C(J)=A(J)+B(J)
   40 CONTINUE
      RETURN
      END
```

図 2.14　ディレクティブ挿入後のログメッセージファイル sub2.log の内容

ログメッセージファイルをみると DO20 はサイクリック分割されていることがわかります。

一般的に要素並列化は DO ループを順に等分割しますが、IP 間の負荷バランスが悪い場合には、ディレクティブ *poption cyclic が有効な場合があります。

次にもう一度 -Xparmonitor オプションを用いて要素並列化単位の性能情報を採取します。ディレクティブ挿入後の実行性能情報ファイルを図 2.15 に示します。

```
～　途中省略　～

< Function Information >
  Function name                 : SUB2
  Source file name              : sub2.f
```

```
  Line no.                        :                5
  Execution count                 :               60
                          (0-2) :               60                0                0
                          (3-5) :                0                0                0
                          (6-7) :                0                0
  User Time [sec]           (Max) :         0.004152 (Exe.R   0.01 %) (Par.E  12.50 %)  … (r)

～  途中省略  ～

< Parallel Information >
  Function name                   : SUB2
  Source file name                : sub2.f
  Line no.                        :                5
  Execution count                 :              480
                          (0-2) :               60               60               60
                          (3-5) :               60               60               60
                          (6-7) :               60               60
  User Time [sec]           (Max) :         8.597627 (Exe.R  25.89 %) (Par.E  99.91 %)  … (s)
                          (Total) :        68.717576 (Exe.R  73.63 %)
                          (0-2) :         8.593819         8.587522         8.589850
                          (3-5) :         8.597390         8.597627         8.569606
                          (6-7) :         8.592853         8.588908
  LD/ST                           :  14515370640 (Exe.R  73.81 %) (Par.E  99.94 %)  … (t)
                          (Max) :      1815508920
                          (0-2) :      1813708920       1814008980       1814308920
                          (3-5) :      1814608980       1814908920       1815208980
                          (6-7) :      1815508920       1813108020
  Instruction count               :  42335801840 (Exe.R  78.45 %) (Par.E  99.95 %)  … (u)
                          (Max) :      5294788341
                          (0-2) :      5290288392       5291037595       5291788227
                          (3-5) :      5292537618       5293288691       5294038040
                          (6-7) :      5294788341       5288034936
  Floating-point arithmetic count :  24615355996 (Exe.R  82.42 %) (Par.E  99.97 %)  … (v)
                          (Max) :      3077969654
                          (0-2) :      3076169456       3076469623       3076769736
                          (3-5) :      3077069458       3077369864       3077669582
                          (6-7) :      3077969654       3075868623

～  以下省略  ～
```

図 2.15　SUB2 **の要素並列化単位の実行性能情報（ディレクティブ挿入後）**

図 2.15 **の**(s)**から**(v)**の** Parallel efficiency **がほぼ** 100%**であることにより** SUB2 **の要素並列化箇所の** IP **間の負荷バランスが均等になったことが分かります。**

最後に、性能向上効果を確認します。サブルーチン SUB2 **の実行時間は図** 2.15 **の**(r)**と**(s)**の和（要素並列化されない箇所と要素並列化箇所の和）であることと表** 2.1 **より、次の表** 2.5 **が得られます。**

表 2.5　**サブルーチン** SUB2 **のチューニング効果**

	チューニング前の実行時間 [sec]	**チューニング後の実行時間** [sec]
サブルーチン SUB2	16.824714	8.601779

2.3 **サブルーチン** SUB3 **のチューニング**

表 2.1 **において、サブルーチン** SUB3 **が全体実行時間に占める割合は** 18.47[%]**です。**

まず、ログメッセージファイル sub3.log を見て、DO ループが要素並列化されているか、擬似ベクトル化されているかを確認します。

```
      SUBROUTINE SUB3(NN,A,B,C,L)
      DIMENSION A(NN),B(NN),C(NN),L(NN)

XX 逐次ループ
**    B : ループ内に不明依存がある
**
**    [DO 50]
**    最内側ループ展開(4 倍)を行った。
**    擬似ベクトル化を適用した。
**
      DO 50 I=1,NN
         B(L(I))=C(I)+A(I)
   50 CONTINUE
       RETURN
       END
```

図 2.16　ログメッセージファイル sub3.log の内容

ログメッセージファイルをみると、ループ DO 50 が要素並列化されていないことが分かります。原因はメッセージ「B: ループ内に不明依存がある」に示されているように、配列 B のループ内の依存性の有無をコンパイラーが判断できないためです。

配列 B のインデックスに使用される配列 L はサブルーチン SUB1（図 2.8）で L(I)=I と定義されているので、実は配列 B のループ内の依存性はありません。このような場合、ディレクティブ *poption indep(b) を挿入することによってコンパイラーに、依存性の無いこと（並列化が可能であること）を明示します。

一方、擬似ベクトル化に関しては適用されているので問題はありません。

上記ディレクティブ挿入後、sub3.f を再コンパイルして作成したログメッセージファイルを図 2.17 に示します。同時に要素並列化箇所の実行性能情報を採取するために再コンパイル時には、-Xparmonitor も指定しておきます。

```
      SUBROUTINE SUB3(NN,A,B,C,L)
      DIMENSION A(NN),B(NN),C(NN),L(NN)
*poption indep(b)

** ループ入口で並列処理 開始
** 並列手続き名 : _parallel_func_1_SUB3
** 並列ループ
** ループ出口で並列処理 終了
**
**    [DO 50]
**    最内側ループ展開(4 倍)を行った。
**    擬似ベクトル化を適用した。
**
      DO 50 I=1,NN
         B(L(I))=C(I)+A(I)
   50 CONTINUE
       RETURN
       END
```

図 2.17　ディレクティブ挿入後のログメッセージファイル sub3.log の内容

図 2.17 よりループ DO 50 が要素並列化されていることが確認できます。

次に、サブルーチン SUB3 を再コンパイルして作成したロードモジュールを実行して、実行性能情報ファイルを作成します。内容を図 2.18 に示します。

```
～  途中省略  ～

< Function Information >
  Function name                 : SUB3
  Source file name              : sub3.f
  Line no.                      :                4
  Execution count               :               60
                         (0-2) :               60                0                0
                         (3-5) :                0                0                0
                         (6-7) :                0                0
  User Time [sec]        (Max) :         0.003723 (Exe.R   0.01 %) (Par.E  12.50 %)  … (w)

～  途中省略  ～

< Parallel Information >
  Function name                 : SUB3
  Source file name              : sub3.f
  Line no.                      :                4
  Execution count               :              480
                         (0-2) :               60               60               60
                         (3-5) :               60               60               60
                         (6-7) :               60               60
  User Time [sec]        (Max) :         1.244504 (Exe.R   4.73 %) (Par.E  99.76 %)  … (x)
                       (Total) :         9.932619 (Exe.R  28.39 %)
                         (0-2) :         1.244418         1.238768         1.244396
                         (3-5) :         1.238705         1.244504         1.238741
                         (6-7) :         1.244450         1.238638
  LD/ST                         :      2456284320 (Exe.R  35.27 %) (Par.E 100.00 %)  … (y)
                         (Max) :       307035540
                         (0-2) :       307035540        307035540        307035540
                         (3-5) :       307035540        307035540        307035540
                         (6-7) :       307035540        307035540
  Instruction count             :      3881334762 (Exe.R  22.94 %) (Par.E 100.00 %)  … (z)
                         (Max) :       485166948
                         (0-2) :       485166791        485166768        485166849
                         (3-5) :       485166776        485166898        485166948
                         (6-7) :       485166904        485166828
  Floating-point arithmetic count :      599993855 (Exe.R   7.21 %) (Par.E 100.00 %)  … (aa)
                         (Max) :        74999292
                         (0-2) :        74999240         74999176         74999228
                         (3-5) :        74999241         74999241         74999292
                         (6-7) :        74999212         74999225

～  以下省略  ～
```

図 2.18　SUB3 の要素並列化単位の実行性能情報（ディレクティブ挿入後）

図 2.18 の(y)から(aa)の Parallel efficiency が 100.00 %であることにより SUB3 の要素並列化箇所の IP 間の負荷バランスは均等であることが分かります。

最後に効果を確認します。サブルーチン SUB3 の実行時間は図 2.18 の(w)と(x)の和（要素並列化されない箇所と要素並列化箇所の和）であることと表 2.1 より、次の表 2.6 が得られ

ます。

表 2.6　サブルーチン SUB3 のチューニング効果

	チューニング前の実行時間 [sec]	チューニング後の実行時間 [sec]
サブルーチン SUB3	9.861240	1.248227

2.4 サブルーチン SUB4 のチューニング

表 2.1 において、サブルーチン SUB4 が全体の実行時間に占める割合は 18.44[%]です。

まず、ログメッセージファイル sub4.log に注目し、DO ループが要素並列化されているか、擬似ベクトル化されているかを確認します。

```
      SUBROUTINE SUB4(NN,A,B,C,D,NN2)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NN),B(NN),C(NN)
      DIMENSION D(NN2)

** ループ入口で並列処理 開始
** 並列手続き名 : _parallel_func_1_SUB4
** 並列ループ
**    I : TLOCAL 変数
**
**    [DO 60]
**    外側ループ展開(8 倍)を行った。
**
      DO 60 J=1,NN2
         D(J)=0.0D0

** 並列処理を継続
** ループ出口で並列処理 終了
**
**    [DO 70]
**    最内側アキュムレータ変数展開(2 倍)を行った。
**    擬似ベクトル化を適用した。
**
         DO 70 I=1,NN
            D(J)=D(J)+J*(C(I)+A(I)+B(I))
   70    CONTINUE
   60 CONTINUE
      RETURN
      END
```

図 2.19　ログメッセージファイル sub4.log の内容

ログメッセージファイルを見る限り、要素並列化および擬似ベクトル化に関する問題はありません。

次に、サブルーチン SUB4 は要素並列化箇所があるので性能モニター機能の -Xparmonitor オプションを用いて要素並列化単位の性能情報を採取します。図 2.20 に実行性能情報ファイルを示します。

```
～　途中省略　～
```

```
< Parallel Information >
  Function name                  : SUB4
  Source file name               : sub4.f
  Line no.                       :                5
  Execution count                :              480
                          (0-2) :               60                60                60
                          (3-5) :               60                60                60
                          (6-7) :               60                60
  User Time [sec]           (Max) :         9.847847 (Exe.R  36.08 %) (Par.E  37.50 %)
                          (Total) :        29.542075 (Exe.R  62.87 %)
                          (0-2) :         9.847787          9.845204          9.847847
                          (3-5) :         0.000236          0.000250          0.000250
                          (6-7) :         0.000250          0.000251
  LD/ST                          :      6075030240 (Exe.R  68.53 %) (Par.E  37.50 %)  … (ab)
                          (Max) :       2025004980
                          (0-2) :       2025004980        2025004980        2025004980
                          (3-5) :             3060              3060              3060
                          (6-7) :             3060              3060
  Instruction count              :     15075044863 (Exe.R  63.94 %) (Par.E  37.50 %)  … (ac)
                          (Max) :       5025004736
                          (0-2) :       5025004736        5025004310        5025003686
                          (3-5) :             6430              6429              6423
                          (6-7) :             6423              6426
  Floating-point arithmetic count :     7199945473 (Exe.R  57.14 %) (Par.E  37.50 %)  … (ad)
                          (Max) :       2399981970
                          (0-2) :       2399981970        2399981711        2399981792
                          (3-5) :                0                 0                 0
                          (6-7) :                0                 0

～  以下省略  ～
```

図 2.20　SUB4 **の要素並列化単位の実行性能情報**

図 2.20 **の**(ab)**から**(ad)**の** Parallel efficiency **をみるとすべて** 37.5%**であり、**SUB4 **の要素並列化箇所の** IP **間の負荷バランスが悪いことが分かります。特に浮動小数点演算は、**IP0、IP1、IP2 **のみが実行しています。**

ログメッセージファイル sub4.log（**図** 2.19）**を見ると、要素並列化は外側のループ** DO 60 **に対して行われていますが、**DO60 **のループ長** NN2 **はプログラムの** MAIN **部分で** NN2=3 **と定義されています。このため、**DO60 **は** 3 **つの** IP **で実行されてしまいます。**

一般的には外側のループで要素並列化を適用した方が効率は良いのですが、このように、極端に外側のループ長が短い場合は、内側ループを要素並列化して IP **間の負荷バランスを均等化する方が効率が良くなります。**

そこで、外側のループ DO60 **で要素並列化が適用されないようにディレクティブ** *poption noparallel **を挿入します。挿入後、**sub4.f **を再コンパイルして作成されたログメッセージファイルを図** 2.21 **に示します**

```
      SUBROUTINE SUB4(NN,A,B,C,D,NN2)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NN),B(NN),C(NN)
      DIMENSION D(NN2)
*poption noparallel

** ループ入口で並列処理 開始
```

```
** 並列手続き名 : _parallel_func_1_SUB4
XX 逐次ループ
**
**    最内側ループ解消(8 倍)を行った。
**
      DO 60 J=1,NN2
         D(J)=0.0D0

** 並列処理を継続
** 並列ループ
**    D : リダクション配列(SUM)
** --- ループ入口でバリア出力 ---
** --- ループ出口でバリア出力 ---
** ループ出口で並列処理 終了
**
**    [DO 70]
**    最内側アキュムレータ変数展開(8 倍)を行った。
**    擬似ベクトル化を適用した。
**
         DO 70 I=1,NN
            D(J)=D(J)+J*(C(I)+A(I)+B(I))
   70    CONTINUE
   60 CONTINUE
      RETURN
      END
```

図 2.21　ディレクティブ挿入後のログメッセージファイル sub4.log の内容

ログメッセージファイルを見ると外側の DO60 では要素並列化は抑止され、内側の DO70 で要素並列化されていることが分かります。

最後に、もう一度 -Xparmonitor コンパイルオプションを用いて要素並列化単位の性能情報を採取します。実行性能情報ファイルを図 2.22 に示します。

```
～  途中省略  ～

< Function Information >
  Function name                 : SUB4
  Source file name              : sub4.f
  Line no.                      :                    6
  Execution count               :                   60
                         (0-2) :                   60                    0                    0
                         (3-5) :                    0                    0                    0
                         (6-7) :                    0                    0
  User Time [sec]        (Max) :             0.000689 (Exe.R   0.00 %) (Par.E  12.50 %)… (ae)

～  途中省略  ～

< Parallel Information >
  Function name                 : SUB4
  Source file name              : sub4.f
  Line no.                      :                    6
  Execution count               :                  480
                         (0-2) :                   60                   60                   60
                         (3-5) :                   60                   60                   60
                         (6-7) :                   60                   60
  User Time [sec]        (Max) :             2.478543 (Exe.R  12.56 %) (Par.E  99.97 %)… (af)
                       (Total) :            19.821631 (Exe.R  53.45 %)
                         (0-2) :             2.477562             2.478334             2.477469
```

```
                          (3-5) :          2.478446          2.477133          2.478543
                          (6-7) :          2.476626          2.477519
  LD/ST                         :    3037597500 (Exe.R  52.12 %) (Par.E 100.00 %)… (ag)
                          (Max) :     379702680
                          (0-2) :     379702680          379699260          379699260
                          (3-5) :     379699260          379699260          379699260
                          (6-7) :     379699260          379699260
  Instruction count             :    8887694731 (Exe.R  51.11 %) (Par.E 100.00 %)… (ah)
                          (Max) :    1110965942
                          (0-2) :    1110965942         1110961524         1110961211
                          (3-5) :    1110961401         1110960977         1110961386
                          (6-7) :    1110960989         1110961301
  Floating-point arithmetic count :  7199928622 (Exe.R  57.14 %) (Par.E 100.00 %)… (ai)
                          (Max) :     899992219
                          (0-2) :     899992219          899991078          899990737
                          (3-5) :     899991036          899990760          899991094
                          (6-7) :     899990769          899990929

～ 以下省略 ～
```

図 2.22　SUB4 **の要素並列化単位の実行性能情報（ディレクティブ挿入後）**

図 2.22 **の**(ag)**から**(ai)**の** Parallel efficiency **が** 100.00 %**であることにより** SUB4 **の要素並列化箇所の** IP **間の負荷バランスが均等になったことが分かります。**

最後に、この変更の効果を確認します。サブルーチン SUB4 **の実行時間は図** 2.22 **の**(ae)**と**(af)**の和（要素並列化されない箇所と要素並列化箇所の和）であることと表** 2.1 **より、次の表** 2.7 **を得ます。**

表 2.7　**サブルーチン** SUB4 **のチューニング効果**

	チューニング前の実行時間 [sec]	**チューニング後の実行時間** [sec]
サブルーチン SUB4	9.848676	2.479232

2.5 チューニングの効果まとめ

4 つのサブルーチンに関して、SR8000 **性能モニター機能とログメッセージを利用してプログラムチューニングを実施しました。最後に全ファイルを** -Xfuncmonitor **指定でコンパイル、実行し、実行性能情報ファイルを出力してチューニングの効果を再確認します。**

出力された実行性能情報ファイルを図 2.23 **に示します。**

```
(Exe.R %)   :Execution ratio
(Par.E %)   :Parallel efficiency
(Max. total):Total of the Max User Time for each info (=CPU elapse time).
< Process Information >
  Node no.                      :                  0
  Process no.                   :              25465
  Load file name                : sample
  User Time [sec]    (Max. total) :          18.549709 (Par.E  12.50 %)… (ai)

～ 途中省略 ～

< Function Information >
  Function name                 : MAIN
  Source file name              : sample.f
```

```
  Line no.                    :                 8
  Execution count             :                 1
                       (0-2) :                 1                 0                 0
                       (3-5) :                 0                 0                 0
                       (6-7) :                 0                 0
  User Time [sec]      (Max) :          0.010676 (Exe.R   0.06 %) (Par.E  12.50 %)… (aj)

～  途中省略  ～

< Function Information >
  Function name               : SUB1
  Source file name            : sub1.f
  Line no.                    :                 5
  Execution count             :                60
                       (0-2) :                60                 0                 0
                       (3-5) :                 0                 0                 0
                       (6-7) :                 0                 0
  User Time [sec]      (Max) :          4.935179 (Exe.R  26.61 %) (Par.E  12.50 %)… (ak)

～  途中省略  ～

< Function Information >
  Function name               : SUB2
  Source file name            : sub2.f
  Line no.                    :                 5
  Execution count             :                60
                       (0-2) :                60                 0                 0
                       (3-5) :                 0                 0                 0
                       (6-7) :                 0                 0
  User Time [sec]      (Max) :         10.163891 (Exe.R  54.79 %) (Par.E  12.50 %)… (al)

～  途中省略  ～

< Function Information >
  Function name               : SUB3
  Source file name            : sub3.f
  Line no.                    :                 4
  Execution count             :                60
                       (0-2) :                60                 0                 0
                       (3-5) :                 0                 0                 0
                       (6-7) :                 0                 0
  User Time [sec]      (Max) :          1.231606 (Exe.R   6.64 %) (Par.E  12.50 %)… (am)

～  途中省略  ～

< Function Information >
  Function name               : SUB4
  Source file name            : sub4.f
  Line no.                    :                 6
  Execution count             :                60
                       (0-2) :                60                 0                 0
                       (3-5) :                 0                 0                 0
                       (6-7) :                 0                 0
  User Time [sec]      (Max) :          2.207958 (Exe.R  11.90 %) (Par.E  12.50 %)… (an)

～  途中省略  ～

< Function Information >
  Function name               : SUB5
  Source file name            : sub5.f
  Line no.                    :                 4
```

```
  Execution count              :          60
                        (0-2) :          60                0                0
                        (3-5) :           0                0                0
                        (6-7) :           0                0
  User Time [sec]       (Max) :    0.000398 (Exe.R    0.00 %) (Par.E   12.50 %)… (ao)

～ 以下省略 ～
```

図 2.23　例題プログラムの実行性能情報ファイル（チューニング後）

図 2.23 から、各サブルーチンの実行時間（(aj)から(ao)）を抜粋して以下の表 2.8 にまとめます。チューニングにより、プログラム全体の実行時間が約 3 分の 1 になりました。

表 2.8　チューニング効果のまとめ

	チューニング前実行時間 [sec]	チューニング後実行時間 [sec]
MAIN	0.023781	0.010676
サブルーチン SUB1	16.845119	4.935179
サブルーチン SUB2	16.824714	10.163891
サブルーチン SUB3	9.861240	1.231606
サブルーチン SUB4	9.848676	2.207958
サブルーチン SUB5	0.000381	0.000398
合計	53.403911	18.549709

３．性能モニター使用上の注意点

本章では、性能モニターを使用する際の注意点について説明します。

3.1 ソースコードの制限

オプション-Xfuncmonitor **を指定した場合、関数／サブルーチンに対するモニター挿入位置は、図** 3.1 **に示すように処理の先頭、および末尾（**RETURN **文直前）です。性能情報はモニター関数によって得られた測定結果** a **と測定結果** b **の差分をもとに出力しています。これにより、関数／サブルーチン単位に、**CPU **時間、浮動小数点演算性能などの情報が得られます。**

```
      SUBROUTINE TEST(IFLG,A,B)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(100),B(100)
C                     <-------- モニター関数が挿入される（測定結果を a とする）
      IF (IFLG.NE.0) THEN
          WRITE(6,*) 'Error'
          STOP
      END IF
C
      DO 10 I=1,100
         A(I)=B(I)
   10 CONTINUE
C                     <-------- モニター関数が挿入される（測定結果を b とする）
```

```
      RETURN
      END
```

図 3.1　オプション-Xfuncmonitor を指定した場合のモニター挿入位置

図 3.1 のような場合、IFLG=0 でなければサブルーチンの処理の途中でプログラムが終了してしまいます。測定結果 b を採取しなくなるので正しい性能情報が得られないことになります。このように、関数／サブルーチンの処理途中で STOP 文等によりプログラムが終了してしまう場合は性能モニターは使用できません。

3.2. モニター関数のオーバーヘッド

性能モニターを挿入することにより、測定オーバーヘッドが性能測定結果に含まれてしまいます。このオーバーヘッドはモニター関数が 1 回呼び出される毎に数マイクロ秒から数十マイクロ秒です。また、LD/ST 数や Instruction count も増加します。非常に小さな値のように思えますが、次に紹介するような場合には、極端に LD/ST 数、 Instruction count や実行時間が大きく測定されてしまうことがあるので注意が必要です。

(1) 比較的軽い処理をする関数／サブルーチンを多数回呼び出す場合

次の図 3.2 および図 3.3 に示すような、2 つのソースファイルから構成されるプログラムに対して、性能モニターを用いて実行性能情報を採取する場合を例にして説明します。

```
      REAL*8 A(1000000),B(1000000),C(1000000),S
      S=0.0D0
      DO 10 I=1,1000000
         A(I)=DBLE(I)
         B(I)=3.0D0
   10 CONTINUE
      DO 20 I=1,1000000
         CALL DIVISION(A(I),B(I),C(I))
   20 CONTINUE
      DO 30 I=1,1000000
         S=S+C(I)
   30 CONTINUE
      STOP
      END
```

図 3.2　例題プログラム sample1.f

```
      SUBROUTINE DIVISION(A,B,C)
      REAL*8 A,B,C
      C=A/B
      RETURN
      END
```

図 3.3　例題プログラム sample1div.f

(1-1) プログラム全体の性能情報のみを採取する場合

```
prompt> f77 -0ss -procnum=8 -c sample1.f -Xfuncmonitor
prompt> f77 -0ss -procnum=8 -c sample1div.f
```

prompt> **f77 sample1.o sample1div.o -lpl -parallel -o sample1**
として、ロードモジュールを作成し、実行します。実行後に作成される、実行性能情報ファイルを図 3.4 に示します。

```
(Exe.R %)   :Execution ratio
(Par.E %)   :Parallel efficiency
(Max. total):Total of the Max User Time for each info (=CPU elapse time).
< Process Information >
  Node no.                        :                  0
  Process no.                     :               6653
  Load file name                  : sample1
  User Time [sec]    (Max. total) :           0.090338 (Par.E  12.50 %)  …  (a)
                          (Total) :           0.090338
                            (Max) :           0.090338
                            (0-2) :           0.090338            0.000000          0.000000
                            (3-5) :           0.000000            0.000000          0.000000
                            (6-7) :           0.000000            0.000000
  LD/ST                           :           4319348 (Par.E  12.50 %)
                            (Max) :           4319348
                            (0-2) :           4319348                   0                 0
                            (3-5) :                 0                   0                 0
                            (6-7) :                 0                   0
  Instruction count               :          19147767 (Par.E  12.50 %)
                            (Max) :          19147767
                            (0-2) :          19147767                   0                 0
                            (3-5) :                 0                   0                 0
                            (6-7) :                 0                   0
  Floating-point arithmetic count :           1125012 (Par.E  12.50 %)
                            (Max) :           1125012
                            (0-2) :           1125012                   0                 0
                            (3-5) :                 0                   0                 0
                            (6-7) :                 0                   0
  MIPS                            :        211.957620
                            (0-2) :        211.957620            0.000000          0.000000


～ 以下省略 ～
```

図 3.4 実行性能情報ファイル その 1

全体の実行時間は、(a)より 0.090338[sec]です。

(1-2) サブルーチン DIVISION の性能情報も採取する場合

prompt> **f77 -0ss -procnum=8 -c sample1.f -Xfuncmonitor**
prompt> **f77 -0ss -procnum=8 -c sample1div.f -Xfuncmonitor**
prompt> **f77 sample1.o sample1div.o -lpl -parallel -o sample1**
として、ロードモジュールを作成し実行します。実行後に作成される、実行性能情報ファイルを図 3.5 に示します。

```
(Exe.R %)   :Execution ratio
(Par.E %)   :Parallel efficiency
(Max. total):Total of the Max User Time for each info (=CPU elapse time).
< Process Information >
  Node no.                        :                  0
  Process no.                     :               6659
```

```
  Load file name                 : sample1
  User Time [sec]    (Max. total) :          2.544390 (Par.E  12.50 %)  …  (b)
                          (Total) :          2.544390
                            (Max) :          2.544390
                            (0-2) :          2.544390            0.000000          0.000000
                            (3-5) :          0.000000            0.000000          0.000000
                            (6-7) :          0.000000            0.000000
  LD/ST                          :        206315393 (Par.E  12.50 %)
                            (Max) :        206315393
                            (0-2) :        206315393                   0                 0
                            (3-5) :                0                   0                 0
                            (6-7) :                0                   0
  Instruction count              :        356124999 (Par.E  12.50 %)
                            (Max) :        356124999
                            (0-2) :        356124999                   0                 0
                            (3-5) :                0                   0                 0
                            (6-7) :                0                   0
  Floating-point arithmetic count :          1125012 (Par.E  12.50 %)
                            (Max) :          1125012
                            (0-2) :          1125012                   0                 0


～  途中省略  ～

< Function Information >
  Function name                  : DIVISION
  Source file name               : sample1div.f
  Line no.                       :                3
  Execution count                :          1000000  …  (c )
                            (0-2) :          1000000                   0                 0
                            (3-5) :                0                   0                 0
                            (6-7) :                0                   0
  User Time [sec]           (Max) :          0.726724 (Exe.R  28.56 %) (Par.E  12.50 %)
                          (Total) :          0.726724 (Exe.R  28.56 %)
                            (0-2) :          0.726724            0.000000          0.000000
                            (3-5) :          0.000000            0.000000          0.000000
                            (6-7) :          0.000000            0.000000

～  以下省略  ～
```

図 3.5　**実行性能情報ファイル その** 2

これを見ると、全体の実行時間は (b)**の** 2.544390[sec]**で、**(1-1)**で測定した全体の実行時間である** 0.090338[sec]**よりはるかに大きな値となっています。**

これは、非常に処理時間の短いサブルーチン DIVISION **を** 1,000,000 **回も実行しているため、性能モニター関数の呼び出しオーバーヘッドが無視出来なくなったためです。**

実際、この例からモニター関数の呼び出しオーバーヘッドを計算すると、１回あたり、

(2.544390[sec]- 0.090338[sec])/1000000 = 2.45[**マイクロ** sec]

と非常に小さな値ですが、このようなケースを避ける為に、処理が軽く呼び出し回数の多い関数／サブルーチンを含むソースファイルは -Xfuncmonitor **を指定しないでコンパイルすることをお勧めします。**

(2) 比較的軽い処理をする要素並列化単位を多数回呼び出す場合

次に示すようなソースファイルから構成されるロードモジュールを性能モニターを用いて、実行性能情報を採取する場合を例にして説明します。

```
      SUBROUTINE KEISAN(A,N)
      REAL*8 A(8,1000000)
      DO 110 J=1,1000000-2
         DO 120 I=1,N
            A(I,J+2)=A(I,J+1)+A(I,J)
  120    CONTINUE
         IF(A(8,J+2).EQ.0) GOTO 130
  110 CONTINUE
  130 CONTINUE
      RETURN
      END
```

図 3.6　例題プログラム sample2.f

この例のように、最内側ループ（DO 120）でしか要素並列化できない多重ループで、外側ループ（DO 110）の繰り返し回数が大きい場合も、(1)の例と同じように、性能モニター関数のオーバーヘッドの影響を受け、実行時間が実際より増加して測定されることがあります。

処理が軽く実行回数の多い要素並列化単位を含むソースファイル等は -Xparmonitor を指定しないでコンパイルすることをお勧めします。

４．おわりに

SR8000 では、プログラムチューニングを支援する機能として、性能モニター機能と診断ログメッセージ出力機能が用意されています。今回は、この 2 つの機能を使用した実際のプログラムチューニング例をご紹介させて頂きました。

これらのツールは、さらに使い易く、そして、より見やすい形で情報を提供できるよう、ユーザーインターフェースの改善等を行っています。新しい機能や出力情報については、ツールができ次第、説明させて頂きたいと思います。

以上