HA8000 クラスタシステム　性能モニタ機能の利用法

（株）日立製作所

1．はじめに

平成 22 年 11 月より、HA8000 クラスタシステムにおいて、プログラム実行時の各種性能情報を採取することが可能な性能モニタ機能をご利用頂けるようになりました。この性能モニタ機能は、東京大学情報基盤センター殿に納入されている SR11000 で利用出来るものと同等な機能を有しております。以下に、概要や具体的な利用方法をご説明致します。

2．性能モニタ機能概要

性能モニタ機能は、日立最適化 Fortran および最適化 C/C++の機能の一部です。HA8000 クラスタシステムの計算ノード、HA8000-tc/RS425 は、CPU 時間、総実行命令数、浮動小数点演算数などのプログラム実行時の各種性能情報を採取するためのハードウェア機構を持っています。性能モニタ機能ではその機構を利用し、コンパイル・リンク時に特定のオプションを指定することにより、コンパイラーが性能情報を読み取る関数をオブジェクトコードに自動挿入し、読み込んだ情報を集計してファイル出力を行います。

つまり、その都度ソースコードを修正する必要がなく、コンパイル・リンケージオプションの追加のみで性能モニタを利用することができます。性能モニタ機能利用の大きな流れは、以下のようになります。

(1) 性能モニタ情報採取用オプションを指定し、プログラムをコンパイル
(2) プログラムを実行し、性能モニタ情報出力ファイルを取得
(3) 性能モニタ情報表示コマンドにより、性能モニタ情報を表示

3．性能モニタ情報採取用オプション

性能モニタ機能を利用するために指定する性能モニタ情報採取用オプションは、表 3-1 に示す 3 つです。

表 3-1 性能モニタ情報採取用オプション

性能モニタ情報取得単位	コンパイルオプション	リンケージオプション
プロセス単位	-pmproc	-pmproc
プロセス単位、および関数/手続き単位	-pmfunc	-pmfunc
プロセス単位、および要素並列化単位	-pmpar	-pmpar

-pmfunc と-pmpar オプションは同時指定が可能で、同時指定した場合にはプロセス単位、関数/手続き単位、要素並列化単位の全ての性能モニタ情報が取得できます。各オプション指定時に取得できる情報を、表 3-2 に示します。

表 3-2 オプション別性能モニタ取得情報

表示項目	内容	-pmproc	-pmfunc		-pmpar	
		プロセス単位	プロセス単位	関数単位	プロセス単位	要素並列化単位
Date	プログラム実行開始日時	○	○	―	○	―
Hostname	実行したホスト名	○	○	―	○	―
Process no	プロセス ID	○	○	―	○	―
Load module name	ロードモジュール名称	○	○	―	○	―
Func	関数/手続き名称	―	―	○	―	○
File	ソースファイル名称	―	―	○	―	○
Line	開始行番号	―	―	○	―	○
Times	実行回数	―	―	○	―	○
CPU Time	CPU 時間	○	○	○	○	○
Inst	総実行命令数	○	○	○	○	○
FLOP	浮動小数点演算数	○	○	○	○	○
MIPS	Million Instruction Per Second	○	○	○	○	○
MFLOPS	Mega Floating point Operations Per Second	○	○	○	○	○
CPU time[%]	実行比率(時間ベース)	―	―	○	―	○
Element parallelizing rate	TD※1 負荷分散比率(CPU 時間と浮動小数点演算数)	○	○	○	○	○

※1 TD=Thread(スレッド)

(凡例) ○：表示する項目

―：表示しない項目

4．性能モニタ情報表示コマンド

性能モニタ情報採取用オプションを指定しコンパイルしたプログラムを実行すると、性能モニタ情報出力ファイルが、ロードモジュール名、プログラム実行開始日時、ホスト名、プロセス番号を利用した、以下の命名規則で作成されます。

(例)　プリフィックス(固定)：pm_
　　ロードモジュール名：a.out
　　プログラム実行開始日時：11 月 11 日の 12 時 34 分
　　ホスト名：b245
　　プロセス番号：98765

上記の場合の作成ファイル名は、「pm_a.out_Nov11_1234_b245_98765」となります。

作成された性能モニタ情報出力ファイルはバイナリ形式であり、テキスト化して表示するために性能モニタ情報表示コマンド pmpr を使います。pmpr コマンドは、性能モニタ情報出力ファイルを入力ファイルとして、テキスト化した情報を標準出力に出力するコマンドです。使用例を以下に示します。

(例 1)　性能モニタ出力情報をテキスト形式で表示

```
pmpr　pm_a.out_Nov11_1234_b245_98765
```

(例 2)　性能モニタ出力情報を区切り文字コンマ(,)で区切った形式で表示

```
pmpr　-c　pm_a.out_Nov11_1234_b245_98765
```

その他オプションなど、pmpr コマンドの詳細についてはオンラインマニュアルを参照して下さい。

5．性能モニタ専用キュー

性能モニタ機能は、性能モニタ機能を利用するための専用キューのみで利用可能です。性能モニタ専用キューの名称、制限値などは表 5-1 の通りです。

表 5-1 性能モニタ専用キューの詳細

キュー名	最大ノード数	制限時間	メモリー容量	Myrinet
profile	4 ノード(64 コア)	1 時間	28GB(1 ノードあたり)	2 link

6．実行例

以下のサンプルプログラム(図 6-1)を用いたコンパイルから実行まで、性能モニタ機能利用手順を具体的に示します。サンプルプログラムは、ログインノード ha8000-[1-3]の /usr/local/ccut/sample/sample.f からコピー可能です。

```
 +1  C-----------------------------------------
 +2        PROGRAM SAMPLE
 +3        IMPLICIT REAL*8 (A-H,O-Z)
 +4        PARAMETER(N=8000000)
 +5        PARAMETER(ITER=100)
 +6        DIMENSION A(N),B(N),C(N),IN(N)
 +7  C
 +8        WRITE(6,*) 'Sample Program Start'
 +9        CALL INIT(A,B,C,N,IN)
+10        SUM=0.0D0
+11        DO 10 I=1,ITER
+12          CALL CALC(A,B,C,N,IN)
+13          CALL SUMALL(SUM,A,N)
+14          WRITE(6,*) 'SUM =',SUM
+15     10 CONTINUE
+16        WRITE(6,*) 'Sample Program End'
+17        STOP
+18        END
+19  C-----------------------------------------
+20        SUBROUTINE INIT(A,B,C,N,IN)
+21        IMPLICIT REAL*8 (A-H,O-Z)
+22        DIMENSION A(N),B(N),C(N),IN(N)
+23  C
+24        DO 20 I=1,N
+25          A(I)=1.0D0*I
+26          B(I)=2.0D0+I
+27          C(I)=3.0D0/I
+28          IN(I) = I
+29     20 CONTINUE
+30        RETURN
+31        END
+32  C-----------------------------------------
+33        SUBROUTINE CALC(A,B,C,N,IN)
+34        IMPLICIT REAL*8 (A-H,O-Z)
+35        DIMENSION A(N),B(N),C(N),IN(N)
+36  C
+37
+38        DO 30 I=1,N
+39          A(I)=A(I)+B(I)*C(I)
+40     30 CONTINUE
+41        RETURN
```

```
+42        END
+43  C-----------------------------------------
+44        SUBROUTINE SUMALL(SUM,A,N)
+45        IMPLICIT REAL*8 (A-H,O-Z)
+46        DIMENSION A(N)
+47  C
+48        DO 40 I=1,N
+49          SUM=SUM+A(I)
+50     40 CONTINUE
+51        RETURN
+52        END
+53  C-----------------------------------------
```

図 6-1 サンプルプログラム(sample.f)

6.1 コンパイルと性能モニタ情報採取

性能モニタ情報採取用オプションをコンパイル時に指定し、プログラムの性能モニタ情報を採取します。ここでは全ての性能モニタ情報を取得するため、-pmfunc と-pmpar の両方を指定します。

```
$ f90 -64 -Oss -o sample sample.f -pmfunc -pmpar
$ ls
sample   sample.f
```

図 6-2 サンプルプログラムのコンパイル

次にプログラムを実行します。上記でコンパイルしたプログラムを実行するジョブスクリプトを準備し、profile キューにジョブを投入します。キューの指定は、ジョブスクリプト内に記述することも可能です。ジョブの実行が終了すると性能モニタ情報ファイル（pm_AA_BB_CC_DD_EE）が生成されます。

```
$ qsub -q profile sample.sh
Request XXXXXX.batch1 submitted to queue: profile.
$ ls
sample   sample.f   pm_sample_Nov04_1425_b245_15367
```

図 6-3 profile キューでのジョブ実行と性能モニタ情報取得

6.2 性能モニタ情報の表示と分析

次に性能モニタ情報表示コマンド(pmpr)で性能モニタ情報を表示し、分析を行います。サンプルプログラムを実行した際の性能モニタ情報の表示例を図 6-4 に示します。チューニング方法につきましては、「ベクトル並列型スーパーコンピューターSR11000 チューニングガイド」を参照してください。

```
$ pmpr pm_sample_Nov04_1425_b245_15367
pm_sample_Nov04_1425_b245_15367:
############################################################
## Process                                                ##    (プロセス単位)
############################################################
  Date                  : Thu Nov 04 14:25:55 JST 2010      ・・・・・・・(1)
  Hostname              : b245
  Process no            : 15367
  Load module name      : ./sample
  CPU time              : 1.327422[s]
  MFLOPS                : 119.031
  MIPS                  : 306.961
  ------------------------------------------------------
          CPU time     Flop     Inst    MFLOPS      MIPS    ・・・・・・・(2)
  ------------------------------------------------------
  TD 0       1.327> 150505k> 397607k>  113.381<  299.533>
  TD 1    0.003101   500008<  657256   161.234   211.940
  TD 2    0.003148   500008<  657268   158.828   208.782
  TD 3    0.003030<  500008<  657357   165.005>  216.931
  TD 4    0.003181   500008<  657388   157.175   206.646
  TD 5    0.003180   500008<  657376   157.227   206.711
  TD 6    0.003184   500008<  657340   157.038   206.451<
  TD 7    0.003174   500008<  657280   157.513   207.057
  TD 8    0.003097   500008<  657292   161.461   212.251
  TD 9    0.003145   500008<  657342   158.992   209.021
  TD10    0.003115   500008<  657400   160.507   211.031
  TD11    0.003107   500008<  657328   160.929   211.563
  TD12    0.003150   500008<  657304   158.725   208.658
  TD13    0.003118   500008<  657364   160.384   210.858
  TD14    0.003097   500008<  657244   161.474   212.253
  TD15    0.003148   500008<  657234<  158.845   208.793
  ------------------------------------------------------
  TOTAL      1.374  158005k  407467k   119.031   306.961
  ------------------------------------------------------
   Element parallelizing rate : (TOTAL)/(Max * TDs)
        CPU time :   6.47[%] =    1.374/(   1.327*16)
        Flop     :   6.56[%] = 158005065/(150504945*16)     ・・・・・・・(3)


############################################################
## Function/Procedure                                     ##  (関数/手続き単位)
############################################################
========================================
== Function/Procedure Ranking          ==
========================================
       CPU time[%]        Times  Func(File+Line)
------------------------------------------------
```

```
[  1]  0.995457[ 74.99]      100  calc_(sample.f+37)          ・・・・・・・(4)
[  2]  0.322508[ 24.30]      100  sumall_(sample.f+48)
[  3]  0.006249[  0.47]        1  init_(sample.f+24)
[  4]  0.003200[  0.24]        1  sample_(sample.f+8)
---------------------
TOTAL     1.327[100.00]


############################################################
## Element Parallel Region                                ##  (要素並列化単位)
############################################################
========================================
== Element Parallel Region Ranking     ==
========================================
       CPU time[%]         MFLOPS      MIPS    Times  Func[Rank](File+Line)
--------------------------------------------------------------------------------
[  1]  0.982974[ 75.13]  1627.714  3459.963      100  calc_[1](sample.f+37)
[  2]  0.315813[ 24.14]  2533.185  3328.141      100  sumall_[2](sample.f+48)
[  3]  0.009549[  0.73]   837.760  8855.699        1  init_[3](sample.f+24)
---------------------                                          ・・・・・・・(5)
TOTAL     1.308[100.00]

========================================
== Element Parallel Region Detail      ==                      ・・・・・・・(6)
========================================
[  1] Func[Rank](File+Line) : calc_[1](sample.f+37)
          CPU time     Flop     Inst    MFLOPS      MIPS     Times
   ------------------------------------------------------------------
   TD 0   0.975491  100000k> 212577k>  102.512   217.918       100>
   TD 1   0.970943  100000k> 212562k   102.993   218.923       100>
   TD 2   0.974108  100000k> 212563k   102.658   218.213       100>
   TD 3   0.966845< 100000k> 212568k   103.429>  219.857>      100>
   TD 4   0.981561  100000k> 212568k   101.879   216.562       100>
   TD 5   0.982974> 100000k> 212568k   101.732<  216.250<      100>
   TD 6   0.982911  100000k> 212566k   101.739   216.262       100>
   TD 7   0.982812  100000k> 212563k   101.749   216.281       100>
   TD 8   0.971938  100000k> 212564k   102.887   218.701       100>
   TD 9   0.972373  100000k> 212565k   102.841   218.604       100>
   TD10   0.976283  100000k> 212569k   102.429   217.733       100>
   TD11   0.972437  100000k> 212566k   102.834   218.590       100>
   TD12   0.974358  100000k> 212564k   102.632   218.158       100>
   TD13   0.976864  100000k> 212567k   102.368   217.602       100>
   TD14   0.970741  100000k> 212561k   103.014   218.968       100>
   TD15   0.973357  100000k> 212561k<  102.737   218.379       100>
   ------------------------------------------------------------------
   TOTAL    15.606    1600M    3401M  1627.714  3459.963      1600
   ------------------------------------------------------------------
```

```
   Element parallelizing rate : (TOTAL)/(Max * TDs)
        CPU time :  99.23[%] =   15.606/(0.982974*16)
        Flop     : 100.00[%] = 1600000000/(100000000*16)
[  2] Func[Rank](File+Line) : sumall_[2](sample.f+48)
          CPU time     Flop     Inst    MFLOPS      MIPS     Times
   ---------------------------------------------------------------
   TD 0   0.312791   50001k>  65703k>  159.854   210.056       100>
   TD 1   0.311645   50001k>  65688k   160.442   210.779       100>
   TD 2   0.314089   50001k>  65688k   159.193   209.140       100>
   TD 3   0.310036<  50001k>  65693k   161.274>  211.889>      100>
   TD 4   0.315628   50001k>  65695k   158.417   208.139       100>
   TD 5   0.315755   50001k>  65694k   158.353   208.054       100>
   TD 6   0.315780   50001k>  65692k   158.341   208.032       100>
   TD 7   0.315813>  50001k>  65689k   158.324<  208.001<      100>
   TD 8   0.311504   50001k>  65690k   160.514   210.880       100>
   TD 9   0.314163   50001k>  65692k   159.156   209.101       100>
   TD10   0.312949   50001k>  65695k   159.773   209.923       100>
   TD11   0.311984   50001k>  65692k   160.267   210.561       100>
   TD12   0.314417   50001k>  65690k   159.027   208.927       100>
   TD13   0.313005   50001k>  65693k   159.745   209.880       100>
   TD14   0.311609   50001k>  65687k   160.460   210.801       100>
   TD15   0.314008   50001k>  65687k<  159.234   209.189       100>
   ---------------------------------------------------------------
   TOTAL     5.015  800013k    1051M  2533.185  3328.141      1600
   ---------------------------------------------------------------
   Element parallelizing rate : (TOTAL)/(Max * TDs)
        CPU time :  99.25[%] =    5.015/(0.315813*16)
        Flop     : 100.00[%] = 800012800/(50000800*16)
[  3] Func[Rank](File+Line) : init_[3](sample.f+24)
          CPU time     Flop     Inst    MFLOPS      MIPS     Times
   ---------------------------------------------------------------
   TD 0   0.005853   500000>   5285k>   85.423   902.997         1>
   TD 1   0.005767   500000>   5285k<   86.706   916.531         1>
   TD 2   0.005893   500000>   5285k    84.851   896.925         1>
   TD 3   0.005812   500000>   5285k    86.030   909.400         1>
   TD 4   0.009103   500000>   5285k    54.929   580.640         1>
   TD 5   0.009549>  500000>   5285k    52.360<  553.484<        1>
   TD 6   0.009241   500000>   5285k    54.108   571.953         1>
   TD 7   0.009475   500000>   5285k    52.768   557.791         1>
   TD 8   0.005873   500000>   5285k    85.133   899.909         1>
   TD 9   0.005829   500000>   5285k    85.775   906.701         1>
   TD10   0.005825   500000>   5285k    85.844   907.433         1>
   TD11   0.005843   500000>   5285k    85.568   904.517         1>
   TD12   0.005878   500000>   5285k    85.069   899.235         1>
   TD13   0.005753<  500000>   5285k    86.909>  918.687>        1>
   TD14   0.005834   500000>   5285k    85.706   905.968         1>
   TD15   0.005823   500000>   5285k    85.863   907.624         1>
```

```
-----------------------------------------------------------------
TOTAL  0.107351    8000k   84566k   837.760  8855.699        16
-----------------------------------------------------------------
Element parallelizing rate : (TOTAL)/(Max * TDs)
     CPU time :  70.26[%] = 0.107351/(0.009549*16)
     Flop     : 100.00[%] = 8000000/(500000*16)
```

図 6-4 性能モニタ情報の表示例

情報取得単位別の性能モニタ情報の内容詳細を以下に示します。

・プロセス単位(Process)

(1) プログラム全体の情報

プログラム実行開始日時(Date)や、プログラム全体の CPU 時間などの基本情報が表示されます。

(2) スレッド毎の詳細情報

スレッド毎の CPU 時間(CPU time)、浮動小数点演算数(FLOP)、総実行命令数(Inst)、MFLOPS、MIPS を出力し、その合計も出力します。スレッドの中での最大値には">"マークを、最小値には"<"マークを数値の右に表示します。最大値もしくは最小値が複数存在する場合は、マークが複数表示されます。ただし、全てがゼロの場合はマークが表示されません。表示例では、0 番スレッドだけ実行時間が長くなっていることがわかります。

(3) TD 負荷分散比率

CPU 時間と浮動小数点演算数について、各スレッドへの分散比率を以下の式に従って出力します。

TD 分散負荷率(%)＝全てのスレッドの合計値÷(スレッドの中での最大値＊実行スレッド数)

・関数／手続き単位(Function/Procedure)

(4) ランキング情報

-pmfunc オプションを指定してコンパイルされた関数全てを、実行に要した CPU 時間の降順に表示します。表示例では、calc 関数が全体の 74.99%、sumall 関数が全体の 24.30%を占めていることがわかります。

・要素並列化単位(Element Parallel Region)

(5) ランキング情報

-pmpar オプションを指定してコンパイルされた要素並列化単位全てを、実行に要した CPU 時間の降順に表示します。表示例では、calc 関数内のループ、sumall 関数内のループが要素並列化されていることがわかります。

(6) スレッド毎の詳細情報

ランキングされた全ての要素並列化単位について、スレッド単位の詳細情報を表示します。出力項目については、プロセス単位の(2)の「スレッド毎の性能情報」と同様です。

以上