時間計測の方法について

東京大学情報基盤センター
黒田 久泰

Fortran 言語および C 言語において、プログラム内のある区間における経過時間や CPU 時間を計測する方法を紹介します。ここで CPU 時間とはプロセッサがプログラムコードをユーザーモードで実行した時間をいいます。ファイル入出力などのシステムコールにかかった時間は省かれます。

1. Fortran 言語の場合

1.1 経過時間の測定

xclock サービスルーチンを利用します。最初に実数型 8 バイト変数を 1 つ定義します（ここでは説明のため変数名を tt とします)。次に「call xclock(tt,7)」で、経過時間測定用タイマを起動します。この時点で変数 tt の値が変更されることはありません。そして、「call xclock(tt,8)」で経過時間測定用タイマ起動後の経過時間が変数 tt に格納されます。

自然数 1 から 10 億までの和を求める計算の経過時間を調べるプログラムは次のようになります。

(プログラム例 main1.f)

```
      program main
      integer*8 i,sum
      real*8 tt
 call xclock(tt,7)
      do i=1,1000000000
         sum=sum+i
      end do
      call xclock(tt,8)
      write(*,*) 'sum=',sum
      write(*,*) 'Elapsed time=',tt
      end
```

これを実行すると下記のように経過時間が表示されます（SR11000/J1)。

```
% f90 -Oss -noparallel main1.f
% ./a.out
 sum=   500000000500000000
 Elapsed time=   3.26630043983459473

% f90 -64 -Oss -noparallel main1.f
% ./a.out
 sum=   500000000500000000
 Elapsed time=  0.591139078140258789
```

※ 8 バイト整数型（integer*8）の変数が使われているプログラムでは、このように-64 オプションを指定すると効率的な最適化が行われ実行速度が向上することがあります。

1.2 CPU 時間の測定

CPU 時間の測定でも xclock サービスルーチンを利用します。最初に実数型 8 バイト変数を 1 つ定義します（ここでは説明のため変数名を tt とします)。次に「call xclock(tt,3)」で CPU 時間測定用タイマを起動します。この時点で変数 tt の値が変更されることはありません。そして、「call xclock(tt,5)」で CPU 時間測定用タイマ起動後の CPU 時間が変数 tt に格納されます。

自然数 1 から n（これは標準入力によって与える）までの和を求めるプログラムの CPU 時間を調べるプログラムは次のようになります。

(プログラム例 main2.f)

```
      program main
      integer*8 i,n,sum
      real*8 tt
call xclock(tt,3)
      read(5,*) n
      do i=1,n
         sum=sum+i
      end do
      call xclock(tt,5)
      write(*,*) 'sum=',sum
      write(*,*) 'CPU time=',tt
      end
```

これを実行すると下記のように CPU 時間が表示されます（SR11000/J1)。

ここでは n の値として 10 億を指定しています。

```
% f90 -64 -Oss -noparallel main2.f
% ./a.out
1000000000
 sum=    500000000500000000
 CPU time=  0.589999999999999858
```

n の値の入力にかかる時間は CPU 時間にはカウントされません。そのため、入力にかかる時間が大きく異なっても表示される CPU 時間は毎回同じような値になります。

注意事項

要素並列化により複数のプロセッサで並列処理をした場合、xclockサービスルーチンではメインスレッドで使用したCPU時間のみを返します。つまり、全プロセッサのCPU時間の総和ではありません。

```
% f90 -64 -Oss -parallel=4 main2.f      (←コンパイラによる要素並列化を行う)
% ./a.out                               (←実際はバッチジョブ上で実行)
1000000000
 sum=    500000000500000000
 CPU time=  0.700000000000002842E-001  (←メインスレッドで使用した CPU 時間が表示される)
```

2. C 言語の場合

2.1 経過時間の測定

gettimeofday システムコールを利用します。このシステムコールは 1970 年 1 月 1 日午前 0 時からの経過時間を返します。ここでは、より使いやすくするためにこの経過時間を double 型の秒数に変換する getETime 関数というサブルーチンを自前で作成する例を紹介します。この getETime 関数を測定したい区間の前後に挿入すると、2 つの getETime 関数の返す値の差が経過時間(秒数)となります。測定精度は 0.000001 秒となります。

自然数 1 から 10 億までの和を求める計算の経過時間を調べるプログラムは次のようになります。

(プログラム例 main1.c)

```
#include <stdio.h>
#include <sys/time.h>

double getETime()
{
   struct timeval tv;
   gettimeofday(&tv, NULL);
   return tv.tv_sec + (double)tv.tv_usec*1e-6;
}

int main(int argc, char *argv[])
{
   long long int i,sum=0;
   double st,en;

   st=getETime();
   for(i=0;i<=1000000000;i++){
      sum+=i;
   }
   en=getETime();
   printf("sum=%lld¥n",sum);
   printf("Elapsed Time=%.6f¥n",en-st);
   return 0;
}
```

これを実行すると下記のように経過時間が表示されます (SR11000/J1)。

```
% cc -Os -noparallel +Op main1.c
% ./a.out
sum=5000000050000000
Elapsed Time=3.258271

% cc -64 -Os -noparallel +Op main1.c
% ./a.out
sum=5000000050000000
Elapsed Time=0.525532
```

※ 8 バイト整数 (long long int 型) が使われているプログラムでは、このように-64 オプションを指定すると効率的な最適化が行われ実行速度が向上することがあります。

2.2 CPU 時間の測定

CPU 時間の測定では、getrusage システムコールを利用します。このシステムコールでは呼び出したプロセスのこれまでのリソースの使用量を返します。ru_utime.tv_sec と ru_utime.tv_usec にはこれまで使用されたユーザーモードにおける CPU 時間が格納されます。ここでは、この CPU 時間を double 型の秒数に変換する getCPUTime 関数というサブルーチンを自前で作成する例を紹介します。

(プログラム例 main2.c)

```
#include <stdio.h>
#include <sys/time.h>
#include <sys/resource.h>

double getCPUTime()
{
   struct rusage RU;
   getrusage(RUSAGE_SELF, &RU);
   return RU.ru_utime.tv_sec + (double)RU.ru_utime.tv_usec*1e-6;
}

int main(int argc, char *argv[])
{
   long long int i,sum=0;
   double st,en;

   st=getCPUTime();
   for(i=0;i<=1000000000;i++){
      sum+=i;
   }
   en=getCPUTime();
   printf("sum=%lld¥n",sum);
   printf("CPU Time=%.6f¥n",en-st);
   return 0;
}
```

これを実行すると下記のように CPU 時間が表示されます (SR11000/J1)。

```
% cc -64 -Os -noparallel +Op main2.c
% ./a.out
sum=500000000500000000
CPU Time=0.590000
```

なお複数プロセッサで並列処理した場合には、全プロセッサのCPU時間の総和になります。

```
% cc -64 -Os -parallel=4 +Op main2.c    (←コンパイラによる要素並列化を行う)
% ./a.out                               (←実際はバッチジョブ上で実行)
sum=500000000500000000
CPU Time=0.640000                       (←全プロセッサの CPU 時間の総和が表示される
                                          経過時間としては約 0.08 秒になる)
```

CPU 時間の測定は、経過時間の測定に比べると精度が高くありません。また、ファイル入出力にかかる時間は CPU 時間ではなく経過時間を測定する必要があります。こういったことから通常はプログラム中で時間を計る場合には、経過時間を利用したほうがいいでしょう。