Tools of high performance computing 2024

Exercise 3

Return by Wednesday 5.2.2024 23:59 to Moodle.
Exercise session: Friday 6.2.2024

Note: When measuring CPU times of programs do it many times and calculate the average of the results. (Many times ~ from tens to hundreds).

Problem 1. *(6 points)*

The Ackermann function[1] is defined in integer values of m and n as

$$A(m,n)=\begin{cases} n+1, & m=0 \\ A(m-1,1), & m>0 \text{ and } n=0 \\ A(m-1, A(m,n-1)), & m>0 \text{ and } n>0 \end{cases}.$$

Write a program that calculates values of $A(m,n)$ for desired arguments m and n. Run it under debugger with arguments $(m,n)=(4,1)$. While the program is running, press control-C (if this does not work, try control-Z). Check with `gdb` command `where` the subroutine stack. Comment on your results.

Problem 2. *(6 points)*

Attached package `mdmorse.zip` contains an atomistic molecular dynamics simulation code. There are both C (`c/`) and Fortran (`/f90`) versions with `Makefiles`. Modify the `Makefiles` to compile the code with profiling options and then run it according the instructions given in file `README.md`. Based on the `gprof` output answer the following questions.
(a) What is the *source code line* where most of the CPU time is consumed? Give the file and function/subroutine where this line is and copy-paste it to your answer.
(b) What is the *function/subroutine* where most of the CPU time is spent?

Problem 3. *(6 points)*

Compare the CPU time used for execution of attached program `ex3p3.f90` (or the C version `ex3p3.c`) using different level of compiler optimization options `-On`, where `n`=0,1,2,3. Comment the results. Remember to measure only the loops between comments

`!Begin measurement` or `/* Begin measurement */`

and

`!End measurement` or `/* End measurement */`

Problem 4. *(6 points)*

Write a program that consumes a considerable amount of CPU time (say from 1

1 For the curious ones: Check the Wikipedia page of the function.

second up; loops with math functions are good candidates) and some I/O. Measure in the program both the *CPU time* and the *elapsed* (*wall-clock*) time. Explain the difference between these two concepts. Comment on your measured values.

*Measuring **CPU time** in Fortran and C*

Fortran: `call cpu_time(t)` returns in real argument `t` the processor time used by the program so far in seconds.
Example:

```
real :: t1,t2,tcpu
…
call cpu_time(t1)
… do something …
call cpu_time(t2)
tcpu=t2-t1
```

C: Function `clock()` returns processor time used by the program so far in 'clock ticks'. To get it in seconds divide by `CLOCKS_PER_SEC`.
Example:

```
#include <time.h>
…
clock_t t1,t2,tcpu;
…
t1=clock();
… do something …
t2=clock();
tcpu=(double)(t2-t1)/(double)CLOCKS_PER_SEC;
```

*Measuring **wall-clock time** in Fortran and C*

Fortran: `call system_clock(count,rate)` returns in integer argument `count` the current value of the processor clock and in integer argument `rate` the clock count rate (ticks per second).
Example:

```
integer :: c1,c2,rate,telap
…
call system_clock(c1,rate)
… do something …
call system_clock(c2,rate)
telap=real(c2-c1)/real(rate)
```

C: Function `gettimeofday(tv,tz)` returns in its argument `tv` the processor clock value in seconds and microseconds. See `man gettimeofday`.