Tools of high performance computing 2024

Exercise 5

Return by Monday 19.2.2024
Exercise session: Tuesday 20.2.2022

Note: When measuring CPU times of programs do it many times and calculate the average of the results. (Many times ~ from tens to hundreds).This can be easily done using bash scripts.

Problem 1. *(6 points)*

Compare the *CPU time* used by writing a 10 000 000 element single precision floating point array to file either as formatted (i.e. ASCII) or unformatted. For formatted output use format string `g0.10` (Fortran), `%g` (C).

Unformatted output of the whole array in Fortran:

```
open(2,file='unform.datf',form='unformatted', &
      access='stream',status='replace')
…
write(2) a(i)
```

And in C:

```
fu=fopen("unform.datc","w");
…
fwrite((void *)&a[i],sizeof(a[i]),1,fu);
```

Problem 2. *(6 points)*

Compile the attached code `omit_frame_pointer.f90` with `gfortran` using the following combinations of compiler options (you might want to write a shell script to do this)

(a) `-O0 -fno-omit-frame-pointer`
(b) `-O0 -fomit-frame-pointer`
(c) `-O1 -fno-omit-frame-pointer`
(d) `-O1 -fomit-frame-pointer`
(e) `-O2 -fno-omit-frame-pointer`
(f) `-O2 -fomit-frame-pointer`

Comment your results. For information on option `-fomit-frame-pointer` see the `gcc` man page (`man gcc`). Note that the differences might be rather small. What optimization method would be applicable to a code like this? (Maybe it is applied when giving the option `-O2`. You can check out the gcc man page.)

Problem 3. *(6 points)*

Compile and run the source `mpiexample.f90`/`.c` in the MPI starter kit[1]. Do the run with number of processor 1, 2, 4, 8. As the solution, send the output files. You may run the program on any computer having MPI installed. What seems to be the order of lines printed?

Problem 4.[2] *(6 points)*

Write a program (e.g. by modifying `mpiexample.f90`/`.c`) where two processes

1 `turso_starter_kit_for_thpc.zip` on the course Moodle page.
2 We did not yet cover MPI in the lectures. We will on Monday 19.2. See the material on starting on slide 33 in Parallel computations (part 1).

send a message (one integer) back and forth twice. Each time the integer is received it is incremented by one. In the end the receiving process prints the value of the integer.