GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN
INSTITUT FÜR INFORMATIK

Practical Course High-Performance Computing· SoSe 2022

Dr. Sven Bingert

Exercise Sheet #01

Preparing the development environment

You can find the programs or templates referred in the assignments in the directory in Stud.IP.

Further reading:

- Using the man-pages of GDB, gcc ,….
- https://www.openmp.org/spec-html/5.2/openmp.html
- https://hpc-tutorials.llnl.gov/openmp/
- http://hpc.gwdg.de

Exercise 1 − Basic OpenMP directives

The file `e1_1.c` contains blocks/instructions marked as *Snippets* in comments. Perform the following tasks:

a) Parallelize the "Hello World" message in *Snippet#1* by using the `parallel` directive.
b) Parallelize the vector multiplication code in *Snippet#2* by adding the `parallel` directive in combination with the `critical` directive. Use latter to serialize access to the addition operation over the `sum` shared variable.
c) Parallelize the vector multiplication code in *Snippet#3* by adding a `parallel` directive with `reduction` clause.

Note: Compile using `gcc -fopenmp -o e1_1 e1_1.c`

Exercise 2 − Reduction Clause

The file `e1_2.c` contains the computation of a sum. Parallelize it using OpenMP `for` and the `reduction` clause.

Exercise 3 − OpenMP Sections

The program in `e1_3.c` uses OpenMP `sections` for MIMD work sharing. Most of the time it does not exit cleanly (it *hangs*). Why? What is the problem with the array `c[]`? Are all barriers necessary? Fix the program.

Exercise 4 − Parallelizing matrix multiplication

The file `e1_4.c` contains code that performs multiplication of two square matrices (say A and B) with varying sizes.

a) Using OpenMP, parallelize the loop that performs the multiplication such that the computation of rows of the product matrix is parallelized among threads. Do not use default variable scope, instead use `private` and `shared` clauses explicitly.
b) Add a way to distinguish between different threads in line 74.
c) Add a way to calculate the total execution time of the matrix multiplication using omp_get_wtime(). Give the execution time in seconds!
d) Add a way to calculate the execution time for each thread separately. What do you notice?
e) Perform the multiplication of square matrices of varying sizes (say 2x2, 4x4, … , 1024x1024..) with a varying number of threads up to 4096. Plot a graph showing the dependency between the execution time and the number of threads for a 1024x1024 matrix. Why does it show that behavior?

Make sure that you don't allocate more memory than your computer has! Which size of matrix fits at maximum in 1/4 of your computers main memory? (you might look it up with the linux command `free`)
Make sure that you don't create too many threads!

Exercise 5 − Data/loop dependence

The file `e1_5.c` contains code that uses an array `x` of fixed size to store the factorial of i in an array x[i].

a) What is the difference between `#pragma omp parallel`, `#pragma omp for` and `#pragma omp parallel for`?
b) What is the problem of data or loop dependence?
c) Run the `parallel for` directive with more than 1 thread. What might go wrong?
d) How can you fix this without modifying the computation logic, i.e., using OpenMP features?
e) Modify the program logic such that the program behaves correctly with multiple threads. Use the `schedule` clause to show how the iterations are divided among threads.