HIGH PERFORMANCE PROGRAMMING
UPPSALA UNIVERSITY
SPRING 2020
ASSIGNMENT 6: USING OPENMP

Relation to previous assignments: This assignment is again connected to the previous Assignment 4; you are supposed to use your code from Assignment 4 as a starting point.

Remember that getting correct results is more important than any optimization and parallelization efforts. Therefore, before starting with this assignment you must make sure that your code from Assignment 4 works properly. If there are problems with your code for Assignment 4 you should fix that first, and then do this assignment.

This assignment is very similar to the previous Assignment 5, the main difference is that now you are supposed to do the parallelization using OpenMP instead of Pthreads.

1. Assignment

In this assignment, you should take your code from Assignment 4 and parallelize it using OpenMP. Your final code should be as efficient as possible, both regarding serial optimization and parallelization, so that it uses the multi-core computer it is run on as efficiently as possible. The number of threads to use should be specified as an input parameter to the program.

To make sure you are focusing your parallelization efforts on the most important part of your code, follow these steps:

(1) Use OpenMP to parallelize the most time-consuming part of your code.

(2) Since OpenMP makes it more convenient to parallelize code with only small changes, consider if there are some other parts of your code that you can also easily parallelize using OpenMP. For example, can you parallelize the code for updating the particle positions?

(3) In the final report for Assignments 3-6 you show the effectiveness of your parallelization, including plots of speedup when running on different numbers of cores. Include a comparison of the performance of your OpenMP-parallelized code compared to the Pthreads parallelization you did in the previous assignment. Which code performs best?

Your program should handle its input and output in precisely the same way as in Assignment 5; see those instructions for details.

Date: February 27, 2020.

Group: You should work in groups of two students/group. (If you have strong reasons why you need to or really want to work alone, discuss that with your teacher, you may be allowed to work alone.)

The group should decide on a distribution of roles during the exercise and very shortly describe it in the final report. For example, who did the most programming and debugging, measuring performance and generating figures/tables, writing the report. If both of you have contributed equally to everything, then write that. If you focused on different things, make sure each of you still understands what you have done and how each part of your code works.

Requirements regarding portability and what to include in the report are the same as for the previous assignment.

2. Deliverables

You should package your code and your final report following the instructions for the final report for Assignments 3-6.

Part of our checking of your submissions will be done using a script that automatically unpacks your file, builds your code, and runs it for some test cases, checking both result accuracy and performance. For this to work, it is necessary that your submission has precisely the requested form.

If you have used your own computer for this assignment, please verify that your code is portable to the university's Linux computers before you submit. (You may find that some change is needed to make the code portable there, e.g. adding `-lm` to link to the math library.)

If you have used your own computer for this assignment, please verify that your code is portable to the university's Linux computers before you submit. (You may find that some change is needed to make the code portable there, e.g. adding `-lm` to link to the math library.)

The section in the final where you discuss Assignment 6 report should include:

- The Solution (How did you do the parallelization? Are there other options, and why did you not use them?)
- Performance and discussion. Present experiments where you investigate the performance of the algorithm and your code. Describe optimizations you have used, or tried to use, and the measured effect of those optimization attempts. Include both a figure showing the scaling behavior as a function of N, and a couple of speedup plots showing how your OpenMP parallelization works. Also compare the performance you get now to what you got in the previous assignment, when Pthreads was used.