Coursework: Lennard-Jones

Core I - High Performance Computing

Tobias Weinzierl

November 14, 2019

This assignment is to be completed and handed in via DUO. For deadlines, please consult DUO and the level/course handbook. Please read the instructions below plus the assessment description and follow them carefully as this will facilitate the accurate marking of your code. The assignment consists of two parts. The second part consists of three steps. Four pieces of work (files) are to be submitted in total. While you are allowed to do formative coursework in groups, this summative coursework has to be done individually without help from a third person.

Instructions

- This assignment is to be completed using the assignment's source code provided via DUO. Your task is to study, tune and extend this code baseline.
- While you are allowed to change any part of the code, I strongly recommend that you focus on `updateBody()`.
- You are not allowed to split up the code into multiple files. Every submission for every step of Part II has to consist of exactly one C file.
- The code is not to use any other parameters than those currently passed in on the command line and you are not allowed to make your submitted code output any additional information to the terminal (or a file). This does **not** mean that it does not make sense to add additional outputs for your own experiments, but be sure you strip the code of any additional output before you submit it. The output format of the submitted code has to be exactly the same as the output of the provided code.
- Ensure all page limitations are observed and submit PDFs only. Word files, e.g., are not accepted. Stick exactly to the submission format.

Context

The code simulates molecules (Argon) subject to the Lennard-Jones forces

$$f_{ij} = -4 \cdot \epsilon \cdot \left(-12 \cdot \left(\frac{\sigma}{dx_{ij}} \right)^{12} + 6 \cdot \left(\frac{\sigma}{dx_{ij}} \right)^{6} \right) \frac{1}{dx_{ij}}.$$

If you run the code without parameters, its usage message will tell you how to invoke a three-body simulation. It also tells you how to construct setups with larger molecule counts.

The implementation you are given computes the f-forces between all particle pairs. It is an $\mathcal{O}(n^2)$ implementations. Lennard-Jones forces between particles that are far away are very small. Many codes thus work with cut-off radii, i.e. neglect the interaction between any two particles that are far away from each other. You are allowed to use this "trick" in your code, too, but you have to identify a proper notion of "far away" yourself.

For each of the following steps, marks will be given for the correctness of the implementation. Please check that your code runs fine for many particles—indeed all the subsequent steps make sense if and only if you pass a decent number of particles into the code. These check can be visual checks, i.e. you open the output files with Paraview and you check that the outcomes are visually reasonable.

Part I: Profiling (analysis/report)

Write a report which discusses which parts of the routine `updateBody()` are well-suited for parallelisation. For this

1. profile the code (you might even think about splitting up the routine into subroutines temporarily and profile them separately);
2. set up a proper performance/upscaling model;
3. contextualise your statements with Flynn's taxonomy.

The solution to this step is to be handed in as a one-page PDF called `solution-step1.pdf`. Please support your statements in terms of well-known performance models, i.e. predict what efficiency you expect to obtain through the steps of Part II.

This part is worth 25 marks.

Part II: Parallelisation (programming and lab experiments)

Parallelise your code

1. with vectorisation
2. with OpenMP
3. with MPI

Per step, you have to hand in your source code only. Per step, you can start from the baseline code, i.e. the OpenMP version does not have to have vectorisation, and the MPI version neither has to support vectorisation nor OpenMP. Please name the submitted files `solution-vectorisation.c`, `solution-openmp.c`, and `solution-mpi.c` (all lowercase), respectively. Before you submit, ensure that you submit a scaling version, i.e. conduct experiments with lots of particles and run some performance analysis to guide your development efforts. For all tests, switch off IO. Each step is worth 25 marks. We will mark correctness, speed, and various upscaling scenarios.