High Performance Systems Programming (ECE415)
Department of Electrical & Computer Engineering
University of Thessaly

Teacher: Christos D. Antonopoulos

5th Laboratory Exercise

Objective: Parallelize and optimize NBody simulation application on GPU.

Introduction:

NBody simulations simulate the evolution of dynamic systems of bodies (particles, atoms, planets), under the influence of physical forces

(electrostatic, gravitational).

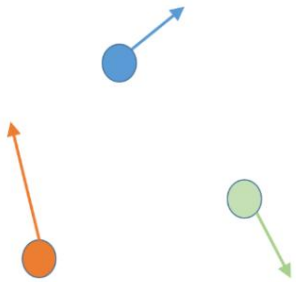

Figure 1: Snapshot of three moving bodies

The time evolution of the phenomenon is simulated in steps. At each step the bodies are characterized:

a) From their position in 3D space.
b) From their velocity vector.
In addition, depending on the nature of the phenomenon, each body can also be characterized by other physical quantities, such as e.g. its mass or charge.

As a reference and starting point you are given a sequential implementation of the C code.

Background:

The code you are given implements the simplest possible form of the algorithm.

For each time step:
a) Calculate the forces exerted on each body (and their component), due to it
of its interaction with each of the other bodies. These forces obviously depend on the distance between them and their mass/load.
b) The resultant force is used to calculate the corresponding acceleration as well as the

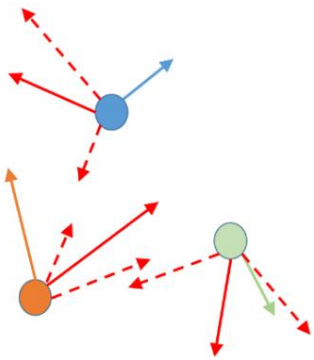

Figure 2: Step (a)

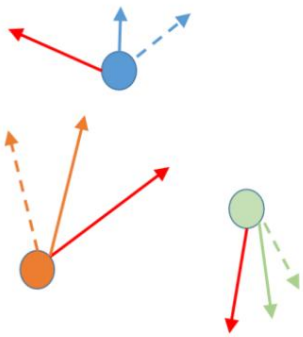

Figure 3: Step (b)

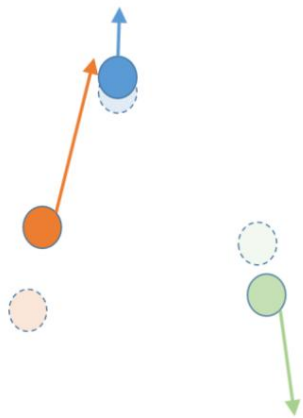

Figure 4: Step (c)

consequent change in the velocity vector of each body.
c) Based on the speed and the time interval between successive steps, the new position of the bodies is calculated.

Wanted:

In phase 1 , you should CPU-parallelize the given code using OpenMP. You probably won't need to change more than 1-2 lines of code. Note that the executable expects one parameter from the command line, the number of bodies being simulated. Values in the order of 64K (65536) or 128K (131072) are reasonable.

In a 2nd stage, you should transfer the code to the GPU. The code should be functional for any data size (provided the GPU device memory is sufficient). Make sure your code is correct by comparing the final body positions as calculated between CPU and GPU. You may consider differences beyond the 2nd decimal to be justified.

This particular code, although small, is quite amenable to optimizations that we have looked at in the lesson. Examples include the distribution of data in memory, approximate optimizations, tiling, unrolling, etc. You can also see this homework as a competition, where the fastest implementation "wins". It goes without saying that the implementation should above all be correct. For time measurements use 128K bodies.

To compile on CPU you can use the following command (respectively and for Intel Compilers if you prefer):

```
gcc -std=c99 -O3 -fopenmp -D_DEFAULT_SOURCE -o nbody nbody.c -lm
```

To compile on GPU you can use the following command:

```
nvcc -arch=sm_37 -I. -o nbody nbody.cu
```

In both cases you can of course give additional arguments to the compiler if required, depending on your implementation.

Tradition:

You must deliver:

- The final code.
- Report in which you will analyze the parallelization strategy you followed, the optimizations you applied, and their impact on runtime. Be sure to also mention any optimizations you've tried without good results in runtime (or accuracy).

Create a .tar.gz file with the above contents and name <name1>_<AEM1>_<name2>_<AEM2>_lab5.tar.gz. Send it to e-class by 23:59 on Thursday 12/1/2023.