Status Report: November 13^{th} 2017

Benchmarking different hardware architectures using a parallel implementation of the N-Body Simulation algorithm

Jan Mrázek

CIS 750 - Advanced Computer Architecture Experiments

1 Gantt chart of estimated schedule

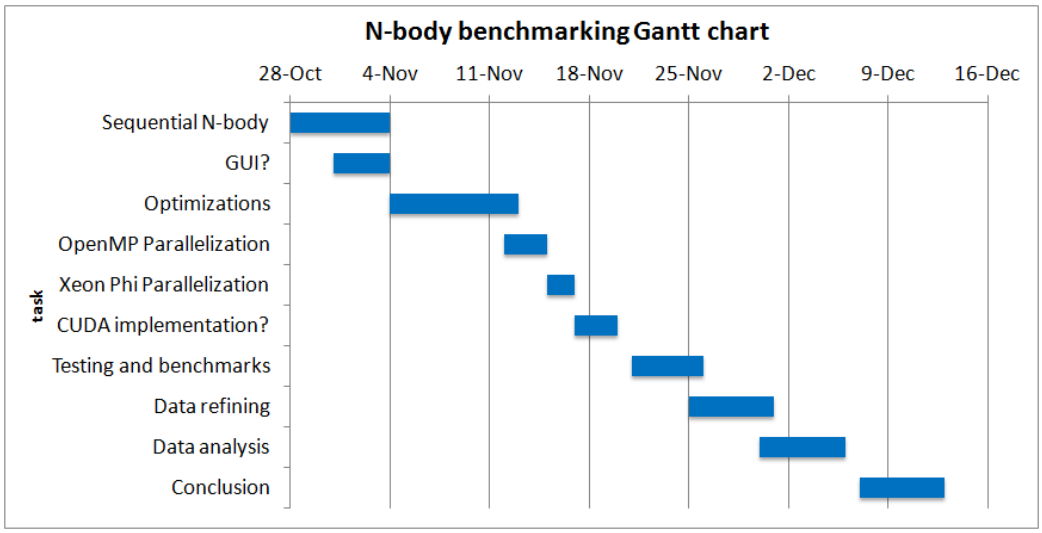

Figure 1: Gantt chart of an estimated project working schedule

2 Tasks completed to date

Since I begun my work on the semestral project for the CIS 750 course, I've been keeping up with the schedule I set for myself (see Figure 1) without too many big hiccups.

I've managed to successfully implement a sequential version of my proposed algorithm.

For the GUI portion of my project, I decided to utilize a very simple GIF-making library created by Charlie Tangora. [1]

Next up, I took a closer look at producing multiple optimized versions of my implementation. First, I overhauled my `updatePositions()` method, the purpose of which is the calculation of every body's new velocity and position vectors. This overhaul resulted in a very good speedup.

I only benchmarked the differences very briefly and with limited resources. I ran some tests on a Ubuntu distribution installed in a dual-thread virtual machine on my computer. The measured speedup between the very naive first implementation and the optimized one was very high, with a value of

about 5 (running times of 32 seconds vs 6 seconds respectively for an N-body system of 1000 bodies and 1000 time iterations).

3 Tasks currently underway

Next, I focused on taking advantage of the OpenMP parallel programming interface.

I have only tried a few things in order to speed up my implementation, and so far I've seen solid performance improvements. After running a really quick test (again, in my less-than-ideal VM environment), I could see a speedup of about 1.73 when running the same simulation as above on two threads.

Although this task is still only in the "work in progress" phase, it already looks very promising when it comes to performance improvements.

4 Tasks soon to start

After further optimizing and tuning my OpenMP parallel implementation, I would like to continue my work with the task of optimizing the implementation for the Intel Xeon Phi coprocessor. I found out that I will probably have to reactivate my account that I will use to log into a machine that contains a Xeon Phi card. I will also probably have to renew my license for the Intel Compiler, which I'm going to use to compile the source code for Xeon Phi.

References

[1] C. Tangora, "gif-h," 2014. [Online; accessed 7-November-2017], available at `https://github.com/ginsweater/gif-h`.