Nelson Batista, Max Inciong, and Francesca Truncale

Senior Project II — Fall 2017

Professor Jianting Zhang

Report for Sept. 25

Our project consists of implementing a solution for the subgraph isomorphism problem in CUDA, to be run on a GPU. This should lead to a significant performance improvement over more common CPU implementations, which should be especially noticeable when operating on larger graphs (with tens of thousands of nodes). As for the graph, we will be testing our program on two graphs from the Stanford Network Analysis Project (SNAP), one of which contains data on Facebook friend's lists and consists of 4039 nodes and 88234 edges. The other is a significantly larger graph consisting of "circles" from Twitter, with 81306 nodes and 1768149 edges. This data is consistent with the goal of our project to be able to quickly identify individuals with common interests on social media and recommend them to each other as friends or followers. The graphs are also large enough to make the performance gains from a GPU implementation very apparent. The Facebook graph is available at `https://snap.stanford.edu/data/egonets-Facebook.html`, and the Twitter graph is available at `https://snap.stanford.edu/data/egonets-Twitter.html`.

We will be using the 2010 Ullmann algorithm for the CPU implementation of our project. We have found a working Python implementation and have almost completed an equivalent C++ implementation. Previously, we had allocated a rather large amount of time to implementing a CPU version a solution to the subgraph isomorphism problem. However, it became apparent that this would take too long, so we have instead settled on finding a working CPU version and instead converting it into a version that can run on a GPU. The goal thus becomes focused more on recording the performance improvements of the GPU implementation in a realistic scenario by running our completed project using the CPU implementation and then the GPU implementation in the background on a suitably large graph, then comparing the performances of each.

Currently, the main hurdle is creation of an adequate data structure to represent a graph. We could not find a way to successfully compile code which uses the `BasicGraph` class provided by the Stanford `cslib` C++ library (see `https://stanford.edu/~stepp/cppdoc/BasicGraph-class.html`), due to numerous cryptic errors during the installation process. In the interest of not spending too much time on this relatively small detail, we have chosen instead to use the Boost library's adjacency list structure for our purposes. We are currently finishing a program to test the various features of this data structure to ensure it is suitable for our project, as well as to build familiarity with how it works.

In the coming two weeks, we hope to have completed a working CPU implementation of, at the very least, the back end of the CPU implementation, and have completed or nearly completed some type of basic

user interface, likely terminal-based. We will then begin work on the GPU implementation after verifying the CPU implementation is fully functional.