Project Proposal for 15-418, Spring 2018:

A Parallel Sketching Library for GPUs

Eliot Robson erobson@andrew.cmu.edu
Taisuke Yasuda taisukey@andrew.cmu.edu

April 9, 2018

Abstract

In this course project, we will provide sequential and parallel implementations of various sketching algorithms and their applications. The sequential code will be written in C++ and the parallel code will be written for NVIDIA GPUs.

1 Background

1.1 Introduction

Sketching algorithms are a class of randomized approximation algorithms designed to approximate large matrices with ones with less rows and/or columns. These algorithms are known to have provable approximation guarantees with high probability and can be applied to countless downstream applications for asymptotically faster approximation algorithms, including but most certainly not limited to linear regression, low rank approximations, and preconditioning. We refer the audience to a monograph by David Woodruff for an excellent overview of sketching algorithms [W+14], especially as applied to linear algebraic operations.

Some of the sketching algorithms we may implement are

- Gaussian sketch
- Subsampled randomized Hadamard transform
- Count sketch
- Leverage score sampling sketch

and some of the applications we may implement are

- linear regression
- k rank approximation
- k means clustering
- principal component analysis

2 The Challenge

We expect the main challenge of the project to be the initial implementation of correct sequential versions of our algorithms, and the subsequent optimization on GPUs. Although the algorithms themselves are not incredibly complex, deciding on a framework to work in and setting everything up may be difficult in the first phase of the project. We expect to make use of existing code, as described in section 3, as well as libraries for linear algebraic operations. However, we still expect this to be one of the more challenging parts of the project. After the set up, the main objective and challenge of the project will be speeding up the code on GPUs. As described before, some parallelization strategies are known already. However, parallel programming on GPUs is an art that can be wildly different from parallelization strategies for distributed computation, and we expect this to take the majority of our time on this project.

3 Resources

Sketching algorithms for streaming problems have been implemented successfully, most notably (and the only one to our knowledge) by Yahoo in their Java sketching library, Data Sketches (`https://datasketches.github.io/`). Although this library supports parallel computation in the form of distributed computation, it has not yet been parallelized at the level of GPUs. Other attempts (`https://github.com/jaykar/matrix_sketching`) have been made by smaller organizations to implement these sketching algorithms on the GPU, to our knowledge, they have not been successful.

The two projects mentioned above give us an excellent platform from which we can kick off our project. Although Data Sketches does not allow us to check correctness since it is a randomized algorithm whose seed we cannot set, it gives us an excellent reference implementation for a sequential version of some of the algorithms we are interested in with respect to the speedup we may achieve. In addition, the website for Data Sketches also includes a discussion for the parallelization tricks used to support distributed computation, so we can base our strategies off of their reports. The unsuccessful GPU code included a C++ implementation of some of the sketching algorithms we are interested in, which helps speed up the process of developing sequential versions of our code.

4 Goals and Deliverables

5 Platform Choice

We will be developing and testing on GPUs on the GHC machines.

6 Schedule

References

[W+14] David P Woodruff et al. Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science*, 10(1–2):1–157, 2014.