Tyler Rose

Professor Annexstein

CS6068

5 December 2018

Analysis of Beer Classification in Parallel

**Introduction:**

The purpose of this project is to test the conversion of a classification of a beer’s style based on tags that were predetermined and pulled from the beer’s description to parallel. The original concept of parsing a beer’s description for certain predetermined tags, separating it to a data set, then training a neural network over the features is a non-trivial problem that would be ideal for parallel operations. This project was worth investigating due to the high usage of neural networks in tech companies today and due to the high performance gains that were achievable in the data parsing part in particular.

The first part of the program is the parsing of data which is easily separable by beer entry and can use a simple Pipe-and-Filter pattern. The actual algorithm used to parse the description used one thread per tag and each tag searched the description for a match. Since this project encompassed both running sequential and in parallel to find comparative performance, the algorithm was modified to only use thread 0 in sequential mode. This means that the total computations is the number of beer entries times the length of the descriptions times the length of the tag. Since it was predetermined to make descriptions a maximum of 2000 characters and tags a maximum of 20 characters there is a max of 40000 computations. This means that we have a work complexity of O(n). When this is done in parallel, there are 85 threads going at a time (one for each tag) so we have a step complexity of O(n) as well. The main reason for this cutback to only use

**Design and Optimization Approach:**

. The main concern when parsing the data is the amount of memory on the Graphics Processing Unit (GPU) and the memory required for storing all the descriptions, tags, and result. To handle this issue, I assumed based on a quick overview of the raw data that the descriptions would not be more than 2000 characters. If they were, they became truncated at 2000 characters. The tags were predetermined by me and were limited to 20 characters and there were 85 tags. To save space on the result, the result was stored as an 11-byte bitmask to store a binary representation of if each tag was present for the given beer. Since all this was predetermined, it was then known that there would be 1700 bytes required for the tags and 2011 bytes per beer for the results and the description. This made it easy to calculate how many beer’s the GPU could parse at a time.

**Application Performance Analysis and Project Results:**

This is stuff

**Division of Work:**

This is stuff

**Bibliography:**

This is stuff

**Code Appendix:**

<https://github.uc.edu/roset3/BeerClassification>