CS677 Project Proposal K-Means

Theodore Jagodits

3/4/20

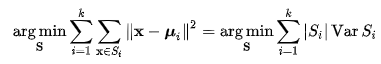
**K-Means**

Problem Description:

The computation I am trying to optimize is k-means. The idea is that one has many points that one is trying to cluster the points into different sections. This is done in several steps:

1. Randomly initialize cluster centroids that clusters center around.
2. Find the distance between the centroids and points. Assign the points with the smallest distance to the closest centroid.
3. Average the values in each cluster to find a new centroid.
4. Repeat steps 2 and 3 until clusters don’t move anymore

Mathematically this is described as:



This computation is important because it can categorize large amounts of data by similarity and unsupervised, meaning it does not need to be labelled. Which is very useful in today’s world since there is so much data but most of it is not labelled properly.

GPU Acceleration:

Since every point needs to compute the distance between each cluster and does not need to rely on other points or computations, one can see the advantage of parallelism. Having many threads each computing the distance between the clusters for one point makes it very suitable GPU use.

The data structure will most likely be a multi-dimensional array. One issue with GPU computation is that the centroids will need to be synced after every computation of the centroid since we need the current centroid values to find the next minimum distances. Also, the host needs to specify the dimensionality and how that affects the computation. In addition, the number of clusters k needs to specified by the host. This all affects the computation and how GPU handles the calculations.

The host will pass in a large array into global, then the algorithm will run and will give assignments on each data point in the array. All the computation should be done on the device and only the result will be given to the host.

Intellectual Challenges:

This project will be challenging because of how we will use what we learned in class to speed up our algorithm. We can apply tiling, shared memory and many other techniques to make this computation as fast as possible. Then I will also have to make it scalable to hundreds and thousands of data points and keep it efficient.

One bottleneck I anticipate is the choosing and implementing different kernels for different sizes of data and different sizes of k. There will be a lot of trial and error involved to get the best performance out of the machine.

Notes:

I will also be looking at some real-world applications of this program to optimize it to a specific problem. This would be interesting and would be a nice learning experience.