Project 4: OpenMPI

Michael Wang - mhw1
William Maynes - wmaynes

Section 1 - Parallel Design

Introduction

When looking at how to parallel any algorithm, one of the most direct approaches is to split the work and distribute it to multiple workers. This was the fundamental idea behind the Map-Reduce framework and is the main reason why those frameworks are useful.

When you examine the K-Means algorithm, it is simple to see that the vast majority of the work is done in the last two steps of the algorithm. In one of these steps each data value is associated with one of the means, and in the other the means are recalculated based on the points they are associated with. To parallelize the algorithm, we split up the work in each of these two steps differently.

For Data Associating:

Master:

```
Break data into n chunks
For each node i in all nodes
     Send ith chunk of data to ith node along with means
     Receive results from workers
Collate results
```

Worker:

```
For each data point
    Calculate distance to each mean
    Group each point with closest mean
Return grouped points to master
```

For Recalculating Means:

Master:

```
For each mean m
    Send full data and m to worker
Wait for results from workers
```

Worker:

```
Initialize totals and a counter
For each piece of data d
    If d is grouped with m
         Add d to totals
         counter++
return totals / counter
```

For associating the points, since each worker is only responsible for a small chunk of the points, it is faster than the sequential version where a single processor had to associate all of the points on its own.

Also, for recalculating the means, each worker is only responsible for one of the means, as opposed to all of them in the sequential version. Thus, as the number of means and clusters increases, our algorithm actually scales better.

Section 2 - Experimentation and Analysis
