CS 451 - Assignment 3

Apply MPI and MapReduce to a popular real problem, namely cluster analysis using K-Means algorithm. Compare and contrast MPI-based and MapReduce-based implementations of K-Means algorithm in terms of performance and development effort.

Cluster analysis or clustering is the task of assigning a set of objects into groups (called clusters) so that the degree of similarity can be strong between members of the same cluster and weak between members of different clusters. In summary, clustering has to define some notion of similarity among objects. The objective is to maximize intra-cluster similarity and minimize inter-cluster similarity.

Among clustering formulations that are based on minimizing a formal objective function, perhaps the most widely used and studied one is K-Means algorithm. Simply put, K- Means is an iterative algorithm that attempts to find K similar groups in a given data set via minimizing a mean squared distance function. Initial guesses of K means $(m_1, m_2, ..., m_K)$ is first made. These estimated means are then used to classify the data set objects into K clusters. Afterwards, each mean is recomputed so as to reflect the true mean of its constituent objects. The algorithm keeps iterating until the recomputed means (almost) stop varying.

How the also should work:

1. We start by deciding how many clusters we would like to form from our data. We call this value k. The value of k is generally a small integer, such as 2, 3, 4, or 5, but may be larger.

2. Next we select k points to be the centroids of k clusters which at present have no members. The list of centroids can be selected by any method (e.g., randomly from the set of data points). It is usually better to pick centroids that are far apart.

3. We then compute the Euclidean distance (the similarity function with a data set of data points) from each data point to each centroid. A data point is assigned to a cluster such that its distance to that cluster is the smallest among all other distances.

4. After associating every data point with one of k clusters, each centroid is recalculated so as to reflect the true mean of its constituent data points.

5. Steps 3 and 4 are repeated for a number of times (say μ), essentially until the centroids start varying very little.

The positive integer μ is known as number of K-Means iterations. The precise value of μ can vary depending on the initial starting cluster centroids, even on the same data set. You can use μ as the capping value to stop the algorithm.

1 Deliverables

What we want from you are the following:

1. A comparison between 3 different K-Means implementations in terms of performance and development effort - sequential, MapReduce and MPI based.
2. Two scalability studies on:
   (a) The number of processes for your MPI version with a fixed data set of 2D data points. Specifically, use 2, 4 and 8 processes.
   (b) The number of data points in your data points data set with a fixed number of processes (say 8) and 1 reducer for your MPI and MapReduce applications, respectively. Specifically, use 10 million, 20 million, and 30 million data points.
3. A discussion on:
   (a) the experience in applying MPI and MapReduce to K-Means clustering algorithm.
   (b) insights concerning the performance trade-offs of MPI and MapReduce with K-Means.
   (c) thoughts on the applicability of K-Means to MPI and MapReduce.
   (d) recommendations regarding the usage of MPI and MapReduce for algorithms similar to K-means.