CLOUD COMPUTING

Project 8: MiniBatch KMeans

Abhishek Naik and Shree Govind Mishra

24th April, 2017

In Harp MiniBatch KMeans, we have 3 Java classes as below:

1. KMeansConstants.java;
2. KmeansMapCollective.java; and
3. KmeansMapper.java

The first one, KMeansConstants.java contains a list of all the constants that have been used throughout the application. The next one, KMeansMapCollective.java is used for configuration depending upon the parameters that are received. The last one, KmeansMapper.java actually executes the MiniBatchKmeans algorithm. It also brings about communication amongst the workers using different computational models (AllReduce, in our case).

KMeansConstants.java:

This file contains the lists of all the constants that have been used throughout the application. It mainly concerned with the storage of details related to configuration, like the batch size, vector size, the number of centroids, etc.

KmeansMapCollective.java:

This file has the following set of functions:

1. main()
   It loads the program by acting as the starting point. It in turn calls the run() method and passes an object of the KmeansMapCollective class as well as the other parameters received via the command line.
2. launch()
   It is mainly concerned with the creation of the path variables. This is done so as to store the data on HDFS. It in turn calls the configureKmeansJob().
3. runKmeans()
   It loads the centroids, calculates new ones (and creates a partial centroid table by using previous tables and values from the previous step), carries out the reduce step and collects mini batch sample points.
4. run()
   It is used for testing the count of arguments. It in turn calls the launch() method described above.
5. configureKmeansJob()
   It is used for configuration related purposes, like configuring the number of mappers and reducers, constants, etc.

KMeansMapper.java

This file has the following set of functions:

1. setup()
   This function is mostly used for setting things up, in the sense that results are stored in private class variables (e.g., the batch size).
2. runKmeans()
   It loads the data, generates the centroids, carries out broadcasting to the workers, etc.

3. mapCollective()
   This method is called for each file.
4. loadData()
   It iterators for each file in the list and for each, it defines the vectorSize(), creates a vector and adds it to a DoubleArray. Note that this method is overloaded.
5. broadcastCentroids()
   It is used for broadcasting and logging purposes.

Output files:

For generating the output, we are using the AllReduce computational model. This model uses the allreduce operation to synchronize the centroids. Thus, allreduce is passed as a parameter while running the command.

As a part of the extra credit assignment, we are using the following command format to vary the number of data points and the mini batch size:

hadoop jar harp-tutorial-app.1.0.SNAPSHOT.jar edu.iu.kmeans.common.KmeansMapCollective <numOfDataPoints> <num of Centroids> <number of MiniBatchSize> <size of vector> <number of map tasks> <number of iteration> <workDir> <localDir> <communication operation>

Note that the communication operation has been fixed to 'allreduce' in our example. The output files generated have been attached.