Clustering algorithm K Means on Hadoop (MapReduce)

The main purpose of this project was to implement the K Means algorithm using MapReduce. The MapReduce job runs on Hadoop and the Cloudera VM was used for the implementation. The project was developed by students Ilias Katsabalos and Leon kalderon in Athens University of Economics and Business, for the course of Big Data Systems.

For more in depth analysis, please check the source code. Also feel free to change any values in order to experiment with the results. These values are:

- the centers created
- the standard deviation of the points around the centers
- the distance of the old and new centers that force the algorithm to stop the iteration

1. Generating the data points – Skewed.java

The first part of the project was the generation of the data points that are used as an input for the clustering algorithm. According to the guidelines of the project, the data points will be created around three centers that the user specifies. The distribution of the data points is skewed around the centroids. Then according to the value of standard deviation that the developer sets and using a random function, all the data points are created in a text file called “points.txt “. The centers are also exported in a text file called “centers.txt”. The path that both files are generated is the project folder. The number of the data points is around one million.

1.2 File format points.txt

The points are generated as two dimensional and rounded in the second decimal position. The format of the file is the following

x,y
12.98,15.62
-100.34,60.67
.
.
.

The two dimensions are splitted using the character ‘,’ and the decimal character is ‘.’

1.3 File format for centers.txt

The centers are user specified. Their format is the following

Cx	x,y
C1	1,1
C2	50,50
C3	100,100

The first attribute is the key of each center. Later the keys are used in order to sort the new and the old centers, when comparing their difference at the end of each iteration of the algorithm.

1.4 Moving the files to hdfs

The next step is to move the files into the hdfs folder. Here is the command that you can run on the terminal

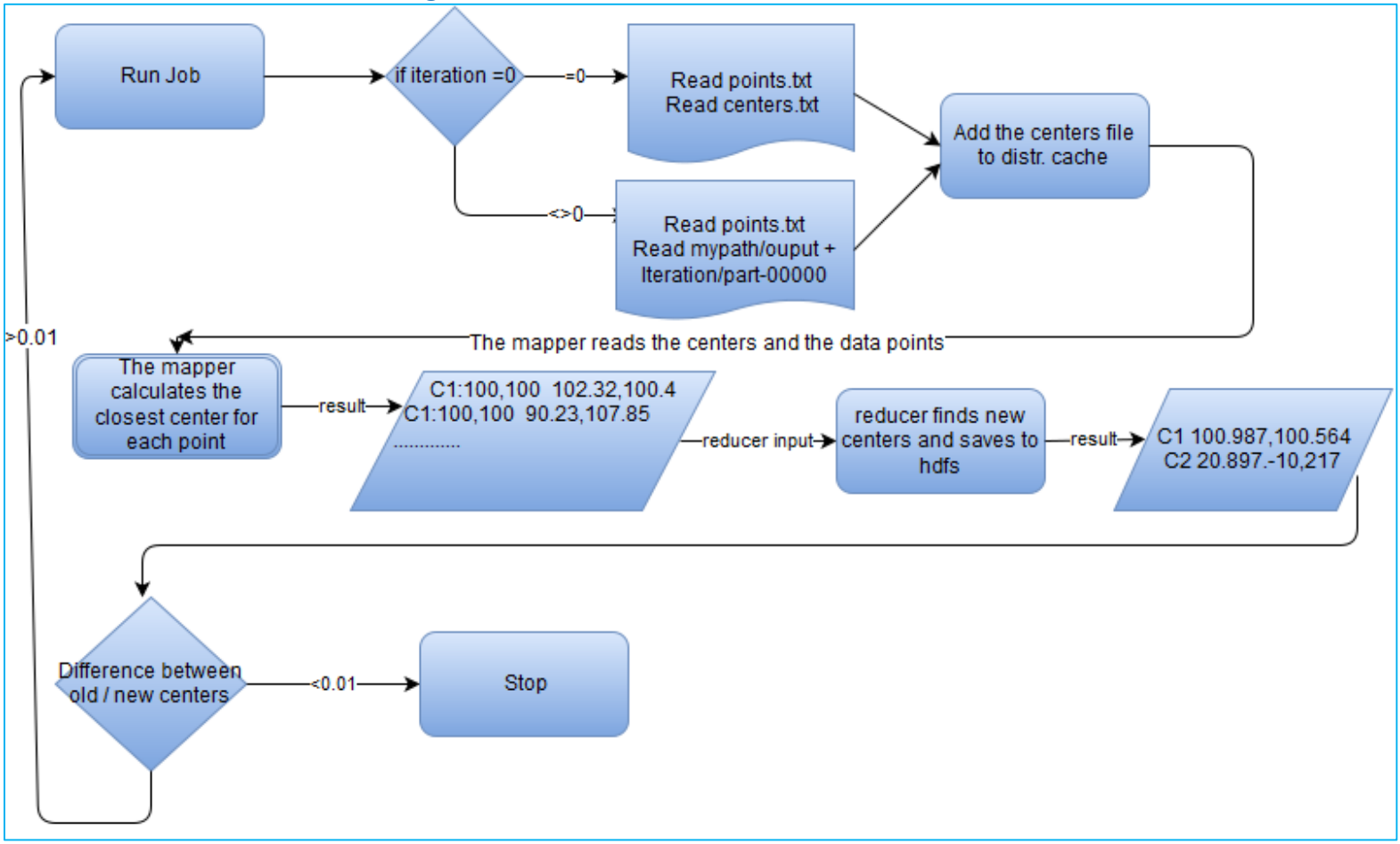
$ hadoop fs –put inputPath outputPath

2. The MapReduce process – Kmeans.java

After our files are put the hdfs, the mapreduce job is executed. The main idea is that the mapper assigns each point to the closest center, and the reducer calculates the next centers. The whole process repeats itself until the difference between the new and the old centers is less than a value that the developer specifies.

First a workflow of the process is introduced, as a general overview. Then, each step is described in detail.

2.1 Workflow Diagram



2.2 Kmeans class

The Kmeans class, contains two methods. The *main* method and the *run* method. The whole process takes place in the run method. This is where all the configuration of the m/r job is set up. We define the input and the output paths of the m/r job. The file that contains the

center is loaded in the cache, in order to be broadcasted to all the mappers that run on local machines. The whole method is wrapped in a *while* loop, which has a Boolean variable as a condition. This Boolean variable changes value, when the difference between the centers is less than the defined value. This means that the algorithm has ended.

If the flag remains *false* the whole process iterates again. For each job, a folder is created in the output path, which contains the centroids for the specific iteration. For example if we have just finished the third iteration of the algorithm, the following folders are created:

/my/path/output0
/my/path/output1
/my/path/output2

where the number at the end of the path is a user defined iterator in order to separate the different iterations of the algorithm. For each iteration, the previous centers are located in the folder /my/path/output (iterator -1).

2.3 The Map class

The map class receives as an input our points.txt file and emits the centroid with the assigned data points.

In the overridden configure method, we first read the centers that are broadcasted using the DistributedCache class.

Then the mapper method calculates the distance (Euclidean Distance) between the data points and each center, keeping the closest one. Finally, the mapper emits the closest center and the data point. The output has the following format:

CenterKey:x,y	x,y
C1:1,1	10.09,1.34
C2…..	
C3….	

The centerKey is a critical piece of information, as we need to sort the centers and keep the same offset. Thus, we need to pass it to the reducer.

2.4 The Reduce class

The reducer is responsible for calculating the new centers. The input key to the reducer is the CenterKey:x,y and the values are all the data points that are assigned to each center. After splitting our points, we calculate the new centers.

The output key of the reducer is the CenterKey and the new center.

Cn	x,y
C1	1.23,0.85
C2	50.90,43.56
C3	100.23,95.67

The new file is created in hdfs in the path my/path/output + iterator.toString()

3. Kmeans2.class

This class is another mapreduce job. The main purpose of this job is to create three files in the hdfs that contain each center as a title and all the data points that belong to each cluster.

3.1 The run method

This method is responsible for finding the last output folder that contains our final result from the Kmeans algorithm execution. This is achieved using an iterator and checking if the file exists in hdfs. If the file does not exist, then the last output folder that contains our file is my/path/output + iterator – 1.

3.2 The map class

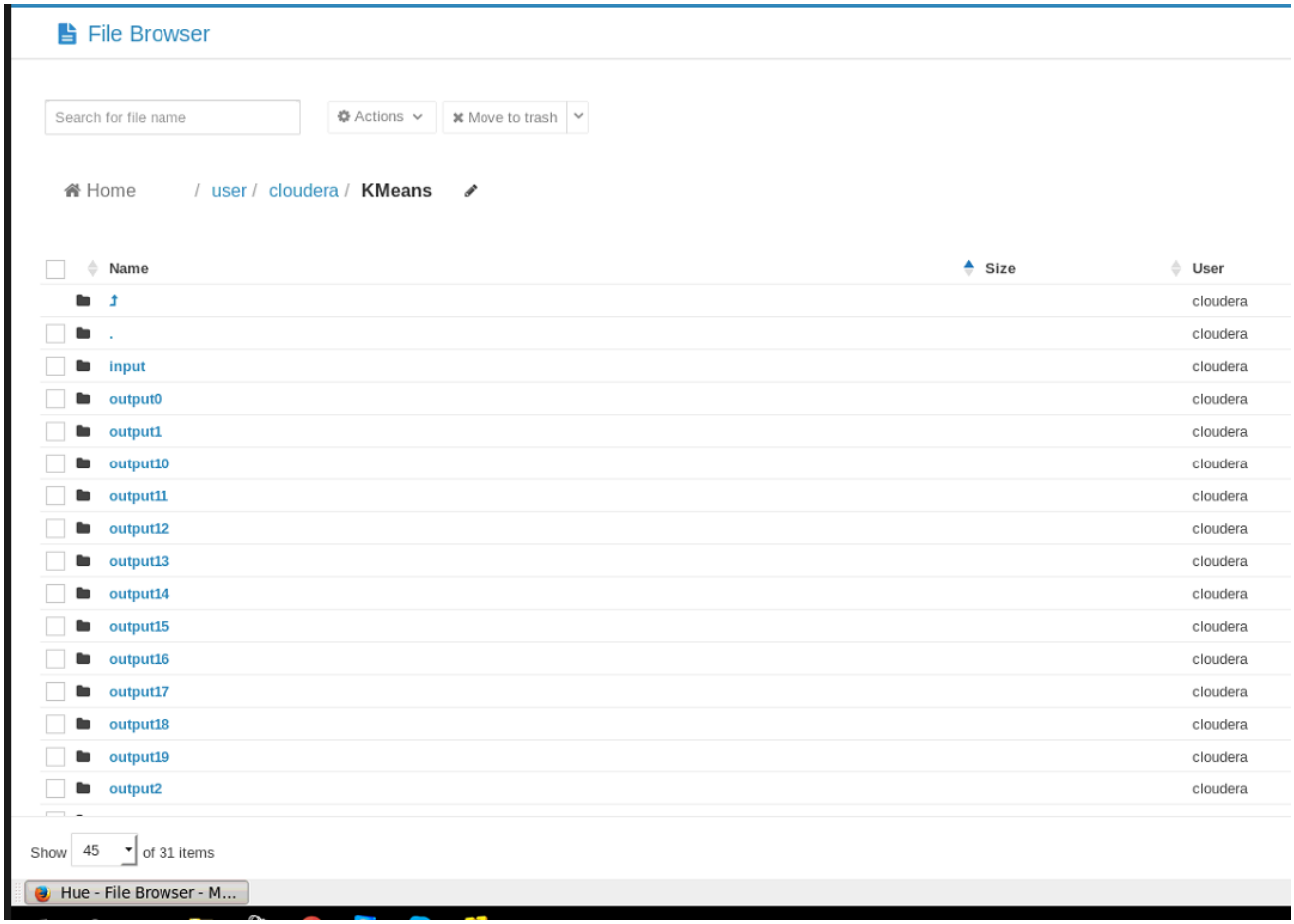
The map class is assigning each data point to the appropriate center. It emits the centers and the data points to the reducer class

3.3 The reducer class

The reducer class is responsible for creating each file for each center. Then it loops through all the values and using a BufferedWriter, writes the values to the relevant file. The name of the file is the center key.

4 Results

After Executing the KMeans class we have the following result in hdfs:



Each output folder contains the result for each iteration. The last folder contains the final clusters. You can see the final cluster centers here:

After that, we execute the KMeans2 class and we get the following results in hdfs:

The files c1, c2, c3 contain the data points of each cluster. In the next screen you will see the actual file opened:

Home / user / cloudera / KMeans2 / output / **C1**

```
-10.92,-37.09
1.95,-15.08
-4.47,-19.70
3.35,-21.60
-18.63,-43.36
3.89,-3.94
32.48,-11.30
18.97,-13.24
-1.89,-16.52
3.87,-30.42
-32.37,-14.85
-25.48,-28.34
-0.61,-7.77
8.52,-8.63
37.32,-10.04
13.41,-12.47
6.01,-50.18
1.34,-17.31
-8.76,-7.92
-16.83,-18.88
22.29,-9.88
-2.00,-13.27
17.35,-20.64
55.45,-43.31
-5.77,-39.71
-3.86,-9.48
18.24,-32.11
-27.98,-6.23
-7.57,-23.57
8.04,-17.14
```

view=/user/cloudera/KMeans/output29#