Cloud computing: KMeans project report
Submitted by : Prem Bhusal

Q.N 1.1:
Ans: The WordCount program was successfully compiled and tested.
I used following command to upload sample text file to HDFS:
hadoop fs -put '/home/prem/hadoopExample/wordcount/input/test.txt' /wordcount/input

I used following command to run the wordcount program :
hadoop jar '/home/prem/hadoopExample/wordcount/wordC.jar' WordCount /wordcount/input /wordcount/output

Output of :hdfs dfs -cat /wordcount/output/part-r-00000 | less
This 1
first 1
hadoop 1
is 1
my 1
program 1

Q.N 1.2:
Ans: Approximately 2 days including installation of hadoop

Q,N 1.2:
It was useful to understand the basic approach of hadoop/mapreduce environment and its implementation .

Q.N 2.1:

Convergence condition is reached when there is no change in new centroid and old centroid , i.e there is no change in cluster membership.

Q.N 2.2:
In current implementation convergence condition is achieved when hen there is change in centroid value is very less ,i.e distance between new centroid and old centroid is less than threshold value (threshold = 0.001).
Potential problem with this kind of implementation occurs when there are highly compact data points. For highly compact data points ,threshold value for eg 0.0009 might look small ie 0.0009<0.001 but it might can make huge difference in cluster membership.

Q.N 2.3:
Yes

Q.N 2.4:
For Mapper:
KEYIN:Centroids
VALUEIN: Data points
KEYOUT:Centroid label
VALUEOUT: Centroid associated data points

For Reducer:
KEYIN:Output of mapper KEYOUT
VALUEIN: output of mapper VALUEOUT
KEYOUT:Updated centroid
VALUEOUT: data points

Q.N 2.5:
Code snippet of setup function:

```
public void setup(Mapper.Context context) throws IOException{
		Configuration conf = context.getConfiguration();
		String filename = conf.get("centroids");
		ArrayList<String> cVal;
		Path p = new Path(filename);

		FileSystem fs = FileSystem.get(conf);
		BufferedReader br;
		try{
			br = new BufferedReader(new BufferedReader(new
InputStreamReader(fs.open(p))));
			nClstrs =1;
			//read the line and get no of clusters
			cVal = objCsvHelper.parseLine(br);
			nDimensions = cVal.size()-1;
			nCentroids[i] = new double[nDimensions];
			//using csvHelper read the centroid file
			int rowVal=0;
			while ((cVal = objCsvHelper.parseLine(br))!=null){
				double [] dValue = new double[cVal.size()-1];
				for (int i=0; i< cVal.size()-1; i++){
					dValue[i] = Double.parseDouble(cVal.get(i+1));
				}
				nCentroids[rowVal] = dValue;
				rowVal ++;
			}

			br.close();
		}catch(Exception e){
```

```
                    System.err.println(e);
                }

        }
```

Q.N 2.6
Ans:
All the records(i.e N records) with current centroid as key from map phase are are sent to reducer . Following happens in each map and reduce
Mapper:

- read the data points
- for each data points find the nearest centroid
- write the centroid with its data points and pass it to reducer phase.

Reducer:

- from the value received from map calculate the updated centroid
- the new centroid is written it in the file
- check the convergence, if they are not equal, repeat the process .

Q.N 2.7
Ans:
Combiner reduce the traffic by combining repeated keys in MapTask.
Combiner phase takes output key value pair from each map and then after processing it, produces key value collection pairs.

Code below is sample implementation of combiner in which input is key value pair from map. It generates the sum of values and their count and pass the result in the form centroid(as key), sum of points+count (collection) to the reducer. The implementation of combiner is similar to that of reducer.
In main function combiner is declared with following line of code.

```
Job.setCombinerClass(combine.class);
```

For N record, K cluster time complexity will be O(N*K). If we use M mapper complexity will become O(N*K/M)

For reducer phase complexity is O(M*K)

```
public void reduce(IntWritable key, Iterator<Text> values,
OutputCollector<IntWritable, Text> output,
Reporter reporter) throws IOException {
//generate partial sum and their count
String[] pValues;
int count = 0;
double p = 0;
double q = 0;
while (values.hasNext()) {
```

```
            pValues = values.next().toString().split(",");
            p =p+ Double.parseDouble(pValues[0]);
            q = q + Double.parseDouble(pValues[1]);
            count ++;
        }
        // construct the output string
        String outputValue = Double.toString(p) + "," + Double.toString(q) + "," + count;
        output.collect(key, new Text(outputValue));
        }
```

Q.N. 2.8:
Simple implementation of mapper.py is as follows which takes centroids file as parameter and returns (recordId, labels).

```
import sys,os
from sklearn.metrics.pairwise import euclidean_distances

datapoints =[]
centroids = []

#put recordID and data points as list of tuples
f = open('data.hdfs','r')
for line in f:
   line = line.split('\t')
   datapoints.append((line[0],float(line[1].split(",")[0])))


for line in sys.stdin:
   i=0
   line = line.split('\t')
   centroids.insert(i,float(line[1].split(",")[0]))

for (recId, point ) in datapoints:
   if(euclidean_distances(point, centroids[0])>euclidean_distances(point, centroids[1])):
         label =0
   else:
         label = 1
   print('%s,%s' % (recId, label))
```

Q.N 2.9:
Ans: 7-8 days

Q.N 2.10:
Ans: Very useful to learn about mapreduce/hadoop environment.

Q.N:3.1
Ans: High
Q.N 3.2:
Ans:High

Q.N 3.3:
Ans:High

Q.N 3.4:
Ans:Somewhat helpful

Q.N 3.5:
Ans:9-10 days

Q.N 3.6:
Ans:average