Practical-7

Name : Yash Marthak
Roll No. : 19BCE122
Couse Name : Big Data Analytics

Aim:

Implement any one of the analytic algorithms using MapReduce by handling larger datasets in main memory.PCY/Multi-Hash/SON algorithm Regression K-means Clustering.

Here I have implemented the K-Means algorithm as given below.

K-means on MapReduce

k-means::Map
Input: Data points D, number of clusters k and centroids
1: for each data point d Є D do
2: Assign d to the closest centroid
Output: centroids with associated data points

k-means::Reduce
Input: Centroids with associated data points
1: Compute the new centroids by calculating the average of data points in cluster
2: Write the global centroids to the disk
Output: New centroids

Code:

```
import java.io.*;
import java.util.*;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.fs.FSDataOutputStream;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
```

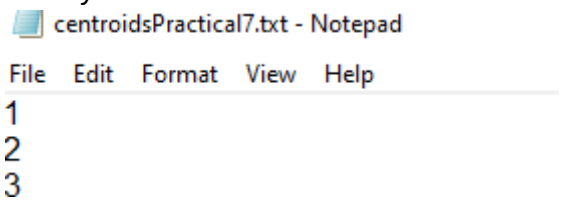
```
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class Practical7_19bce277 {

        static List<Integer> Centroids;
        static List<Integer> newCentroids;
        static Path datapoints;
        static Path centroids;
        static FileSystem fs;

  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{

    private Text word = new Text();

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        int datapoint = Integer.parseInt(word.toString());
        int min = 100000000;
        int center = 1000000000;
        for(int i=0;i<Centroids.size();i++) {
           int dist = Math.abs(datapoint-Centroids.get(i));
           if(dist<min) {
                    min=dist;
                    center = Centroids.get(i);
           }
        }
        context.write(new Text(""+center), new IntWritable(datapoint));
      }
    }
  }

  public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values,
                       Context context
                       ) throws IOException, InterruptedException {
      int sum = 0;
      int length = 0;
```

```
      for (IntWritable val : values) {
        sum += val.get();
        length++;
      }
      result.set(sum/length);
      newCentroids.add(Integer.parseInt(result.toString()));
    }
  }

  public static void main(String[] args) throws Exception {
            for(int i=0;i<3;i++) {
              Configuration conf = new Configuration();
              Centroids = new ArrayList<>();
              newCentroids = new ArrayList<>();
              datapoints=new Path("hdfs:/19bce277_first/datapointsPractical7");//Location of file
in HDFS
              centroids=new Path("hdfs:/19bce277_first/centroidsPractical7");//Location of file in
HDFS

        fs = FileSystem.get(conf);
        BufferedReader br=new BufferedReader(new InputStreamReader(fs.open(centroids)));
        String line;
        line=br.readLine();
        while (line != null){
           Centroids.add(Integer.parseInt(line));
           line=br.readLine();
        }
              Job job = Job.getInstance(conf, "word count");
              job.setJarByClass(Practical7_19bce277.class);
              job.setMapperClass(TokenizerMapper.class);
              job.setReducerClass(IntSumReducer.class);
              job.setOutputKeyClass(Text.class);
              job.setOutputValueClass(IntWritable.class);
              FileInputFormat.addInputPath(job, datapoints);
              FileOutputFormat.setOutputPath(job, new
Path("hdfs:/19bce277_first/Practical7Demo"+i));
              if(job.waitForCompletion(true)) {
                    FSDataOutputStream out = fs.create(centroids, true);
                    BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(out));
                       for(Integer itr: newCentroids) {
                             System.out.println(itr);;
                             bw.write(itr+"\n");
                       }
                       bw.close();
              }
            }
  }
}
```

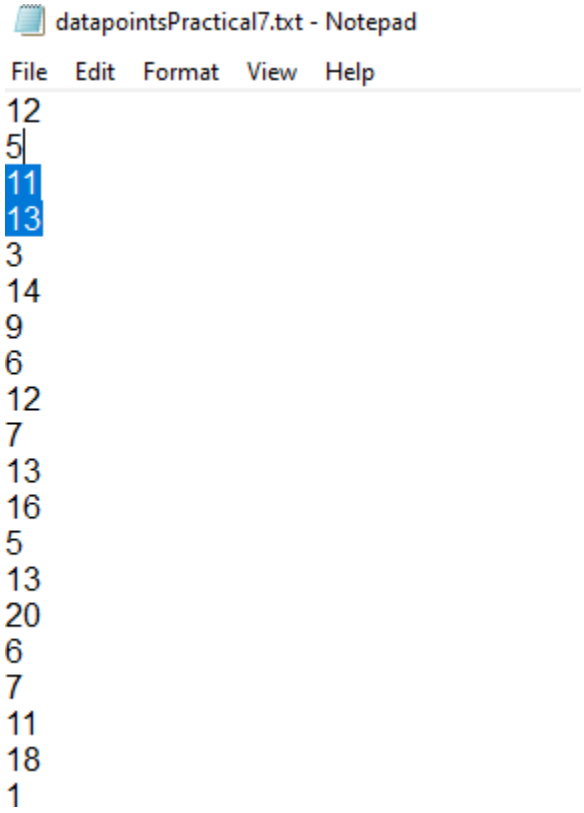
Now, I use two files:-

1) Stores the datapoints.
2) Stores the centroids.

Initially Centroids file looks like this:-



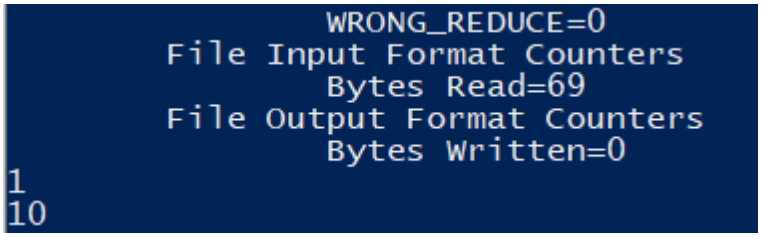
Datapoints:-



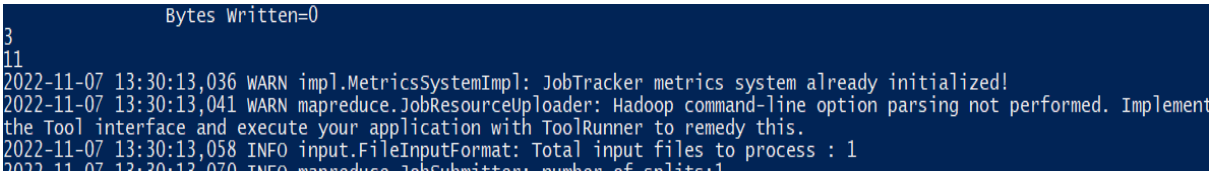
Running the MapReduce Job: -

```
2022-11-07 13:30:09,531 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2022-11-07 13:30:09,591 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s
2022-11-07 13:30:09,591 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2022-11-07 13:30:09,963 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not p
the Tool interface and execute your application with ToolRunner to remedy this.
2022-11-07 13:30:10,047 INFO input.FileInputFormat: Total input files to process : 1
2022-11-07 13:30:10,157 INFO mapreduce.JobSubmitter: number of splits:1
2022-11-07 13:30:10,275 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local1580269013_
```

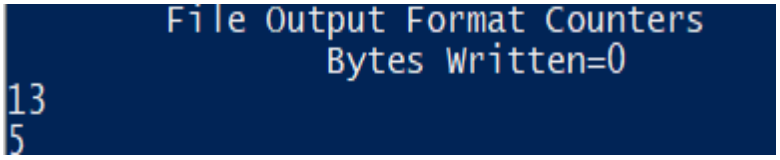
In between I have printed the intermediate centroids:-



Where 1 and 10 are the centroids after the 1st iteration.



Where 3 and 11 are the centroids after the 2nd iteration.



Where 13 and 5 are the centroids after the 3rd iteration.

The code is run just a single time for 3 iterations of MapReduce job.

Finally the centroids file looks as below:-

File information - centroidsPractical7

Download | Head the file (first 32K) | Tail the file (last 32K)

Block information – Block 0

Block ID: 1073741912

Block Pool ID: BP-1112790740-172.23.224.1-1663568697260

Generation Stamp: 1088

Size: 5

Availability:

- N511-105.mshome.net

File contents

```
13
5
```