GEBZE TECHNICAL UNIVERSITY

Computer Engineering Department

CSE 611-CSE 458

Big Data Analytics

Homework 1

ONUR

SEZER

121044074

April – 2017
Gebze – KOCAELİ

- Hadoop kurulup, datalar indirildi ardından datalar hdfs' e atıldı.
  - Hadoop burdan kuruldu -> http://www.bogotobogo.com/Hadoop/BigData_hadoop_Install_on_ubuntu_single_node_cluster.php
  - Hadoop kurulduktan sonra terminalden "start-all.sh" komutu ile nodelar açıldı.
  - "hadoop fs –mkdir /filghtData" komutu ile hadoop ta klasör oluşturuldu.
  - "hadoop fs –copyFromLocal /home/hduser/data /flightData" komutu ile datalar hdfs'e atıldı.
  - loacalhost:50070 den yüklenen datalara bakıldı.
  - Javada yazılan kod terminalden yazılan aşağıdaki kodlarla derlendi.

    export HADOOP_CLASSPATH=/usr/lib/jvm/java-7-openjdk-amd64/lib/tools.jar

    hadoop com.sun.tools.javac.Main WordCount.java
    jar cf wc.jar WordCount*.class
    hadoop jar wc.jar WordCount /Documan1 /cikti

- MapReduce kullanılarak zamanında kalkan uçuşların oranı bulundu.

Toplam Uçuş	Zamanında Kalkışların Sayısı	Oran
1.23534991E8	**7.9847987E7**	**64.63592732200063**

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.lang.Math;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
```

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.TextInputFormat;
import org.apache.hadoop.mapred.TextOutputFormat;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class FlightRatings extends Configured implements Tool {
   public static class MapClass extends MapReduceBase implements Mapper<LongWritable,
Text, Text, Text> {
      private Text loc = new Text();
      private Text rating = new Text();

      @Override
      public void map(LongWritable key, Text value, OutputCollector<Text, Text> output,
Reporter reporter)
            throws IOException {
         String[] rows = value.toString().split(",");
         if (rows.length == 29) {
            String depTime = rows[4]; // actual departure time
            String crsDepTime = rows[5]; // scheduled departure time

            loc.set("fligt");

            rating.set(1 + "\t" + depTime + "\t" + crsDepTime);
            output.collect(loc, rating);
         }
      }
   }

   public static class Reduce extends MapReduceBase implements Reducer<Text, Text, Text,
Text> {
      @Override
      public void reduce(Text key, Iterator<Text> values, OutputCollector<Text, Text> output,
Reporter reporter)
            throws IOException {
         double ontime = 0;
         double numTotal = 0;
         double rating = 0;

         while (values.hasNext()) {
            String tokens[] = (values.next().toString()).split("\t");
```

```
        int tot = Integer.parseInt(tokens[0]);
        if(onTime(tokens[1], tokens[2]))
            ++ontime;
        numTotal = numTotal + tot;
    }

    rating = (ontime / numTotal) * 100;

    output.collect(key, new Text(numTotal + "\t" + ontime + "\t" + rating));
}

public boolean onTime(String str1, String str2) {
    if(!(str1.length() < 3 || str2.length() < 3))
    {
        if (str1.length() == 3) {
            StringBuilder build = new StringBuilder();
            build.append('0');
            build.append(str1.charAt(0));
            build.append(":");
            build.append(str1.charAt(1));
            build.append(str1.charAt(2));
            build.append(":00");
            str1 = build.toString();
        }
        if (str1.length() == 4) {
            StringBuilder build = new StringBuilder();
            build.append(str1.charAt(0));
            build.append(str1.charAt(1));
            build.append(":");
            build.append(str1.charAt(2));
            build.append(str1.charAt(3));
            build.append(":00");
            str1 = build.toString();
        }
        if (str2.length() == 3) {
            StringBuilder build = new StringBuilder();
            build.append('0');
            build.append(str2.charAt(0));
            build.append(":");
            build.append(str2.charAt(1));
            build.append(str2.charAt(2));
            build.append(":00");
            str2 = build.toString();
        }
        if (str2.length() == 4) {
            StringBuilder build = new StringBuilder();
            build.append(str2.charAt(0));
            build.append(str2.charAt(1));
            build.append(":");
            build.append(str2.charAt(2));
            build.append(str2.charAt(3));
            build.append(":00");
            str2 = build.toString();
        }
```

```
            String time1 = str1;
            String time2 = str2;

            SimpleDateFormat format = new SimpleDateFormat("HH:mm:ss");
            try {
               Date date1 = format.parse(time1);
               Date date2 = format.parse(time2);
               long diff = date2.getTime() - date1.getTime();

               long diffMinutes = diff / (60 * 1000) % 60;
               long diffHours = diff / (60 * 60 * 1000) % 24;

               if (diffMinutes >= -5 && diffMinutes <= 5 && diffHours == 0) {
                  return true;
               }
            } catch (ParseException e) {
               e.printStackTrace();
               System.out.println("Parse errorr.");
               System.out.println("str1 : " + str1);
               System.out.println("str2 : " + str2);
            }
         }

         return false;
      }
   }

   static int printUsage() {
      System.out.println("FlightRatings [-m <maps>] [-r <reduces>] <input> <output>");
      return 0;
   }

   @Override
   public int run(String[] args) throws IOException {
      return 0;
   }

   public static void main(String[] args) throws IOException {
      JobConf conf = new JobConf(FlightRatings.class);
      conf.setJobName("FlightRatings");

      conf.setOutputKeyClass(Text.class);
      conf.setOutputValueClass(Text.class);

      conf.setMapperClass(MapClass.class);
      conf.setReducerClass(Reduce.class);

      conf.setInputFormat(TextInputFormat.class);
      conf.setOutputFormat(TextOutputFormat.class);

      FileInputFormat.setInputPaths(conf, new Path(args[0]));
      FileOutputFormat.setOutputPath(conf, new Path(args[1]));

      JobClient.runJob(conf);
   }
```

}

- Her bir uçak için zamanında kalkma oranları bulundu.

KuyrukNum.	ToplamUç.	ZamanındaK.	Oran
-N037M	327.0	243.0	74.31192660550458
-N047M	1170.0	820.0	70.08547008547008
-N107D	1624.0	1043.0	64.22413793103449
-N108D	1611.0	1022.0	63.43885785226567
-N109D	1668.0	1035.0	62.05035971223022
-N110D	1632.0	1028.0	62.99019607843137
-N110H	1385.0	984.0	71.04693140794224
-N111D	1680.0	1078.0	64.16666666666667
-N112D	1662.0	1107.0	66.60649819494586
-N114D	1671.0	1050.0	62.83662477558348
-N116D	1716.0	1055.0	61.48018648018648
-N117D	1697.0	1053.0	62.05067766647024
-N118D	1667.0	1053.0	63.16736652669466
-N119D	1608.0	954.0	59.32835820895522
…	….	….	…..

Toplam uçak sayısı : 13151

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.lang.Math;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.TextInputFormat;
```

```
import org.apache.hadoop.mapred.TextOutputFormat;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class FindEachCarrier extends Configured implements Tool {
   public static class MapClass extends MapReduceBase implements Mapper<LongWritable,
Text, Text, Text> {
      private Text loc = new Text();
      private Text rating = new Text();

      @Override
      public void map(LongWritable key, Text value, OutputCollector<Text, Text> output,
Reporter reporter)
            throws IOException {
         String[] rows = value.toString().split(",");
         if (rows.length == 29) {
            String depTime = rows[4]; // actual departure time
            String crsDepTime = rows[5]; // scheduled departure time

            loc.set(rows[10]);  // tail number
            rating.set(1 + "\t" + depTime + "\t" + crsDepTime);
            output.collect(loc, rating);
         }
      }
   }

   public static class Reduce extends MapReduceBase implements Reducer<Text, Text, Text,
Text> {
```

```
    @Override
    public void reduce(Text key, Iterator<Text> values, OutputCollector<Text, Text> output,
Reporter reporter)
            throws IOException {
        double ontime = 0;
        double numTotal = 0;
        double rating = 0;

        while (values.hasNext()) {
            String tokens[] = (values.next().toString()).split("\t");
            int tot = Integer.parseInt(tokens[0]);
            String tailNumber = key.toString();

            if(!(tailNumber.equals("NA"))){
                if(onTime(tokens[1], tokens[2]))
                    ++ontime;
                numTotal = numTotal + tot;
            }
        }

        rating = (ontime / numTotal) * 100;

        output.collect(key, new Text(numTotal + "\t" + ontime + "\t" + rating));
    }

    public boolean onTime(String str1, String str2) {
        if(!(str1.length() < 3 || str2.length() < 3))
        {
            if (str1.length() == 3) {
                StringBuilder build = new StringBuilder();
                build.append('0');
                build.append(str1.charAt(0));
                build.append(":");
                build.append(str1.charAt(1));
                build.append(str1.charAt(2));
                build.append(":00");
                str1 = build.toString();
```

```
}
if (str1.length() == 4) {
    StringBuilder build = new StringBuilder();
    build.append(str1.charAt(0));
    build.append(str1.charAt(1));
    build.append(":");
    build.append(str1.charAt(2));
    build.append(str1.charAt(3));
    build.append(":00");
    str1 = build.toString();
}
if (str2.length() == 3) {
    StringBuilder build = new StringBuilder();
    build.append('0');
    build.append(str2.charAt(0));
    build.append(":");
    build.append(str2.charAt(1));
    build.append(str2.charAt(2));
    build.append(":00");
    str2 = build.toString();
}
if (str2.length() == 4) {
    StringBuilder build = new StringBuilder();
    build.append(str2.charAt(0));
    build.append(str2.charAt(1));
    build.append(":");
    build.append(str2.charAt(2));
    build.append(str2.charAt(3));
    build.append(":00");
    str2 = build.toString();
}

String time1 = str1;
String time2 = str2;

SimpleDateFormat format = new SimpleDateFormat("HH:mm:ss");
try {
```

```
            Date date1 = format.parse(time1);
            Date date2 = format.parse(time2);
            long diff = date2.getTime() - date1.getTime();

            long diffMinutes = diff / (60 * 1000) % 60;
            long diffHours = diff / (60 * 60 * 1000) % 24;

            if (diffMinutes >= -5 && diffMinutes <= 5 && diffHours == 0) {
               return true;
            }
         } catch (ParseException e) {
            e.printStackTrace();
            System.out.println("str1 : " + str1);
            System.out.println("str2 : " + str2);
         }
      }

      return false;
   }
}

static int printUsage() {
   System.out.println("FindEachCarrier [-m <maps>] [-r <reduces>] <input> <output>");
   return 0;
}

@Override
public int run(String[] args) throws IOException {
   return 0;
}

public static void main(String[] args) throws IOException {
   JobConf conf = new JobConf(FindEachCarrier.class);

   conf.setOutputKeyClass(Text.class);
   conf.setOutputValueClass(Text.class);
```

```
        conf.setMapperClass(MapClass.class);
        conf.setReducerClass(Reduce.class);

        conf.setInputFormat(TextInputFormat.class);
        conf.setOutputFormat(TextOutputFormat.class);

        FileInputFormat.setInputPaths(conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(conf, new Path(args[1]));

        JobClient.runJob(conf);
    }
}
```

- MapReduce üzere iki ana fonksiyondan oluşur. Birincisi map; bir yığının tüm üyelerini sahip olduğu fonksiyon ile işleyip bir sonuç listesi döndürür. İkincisi reduce; paralel bir şekilde çalışan iki veya daha fazla map fonksiyonundan dönen sonuçları harmanlar ve çözer. Hem map() hem de reduce() paralel bir şekilde çalışırlar.Map fonksiyonunda her bir key'e valu'e atanır, key sayısı 1 ile n arasında olabilir, reduce fonksiyonunda ise aynı keyleri toplar ve value'larıyla yapılması gereken gerekli işlemleri yapar.
- K-Means algoritması kullanılarak MapReduce yapıldı.mapdan gelen değerler reduce kısmında alındı ve K-Means uygulandı.

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.lang.Math;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;

import org.apache.hadoop.conf.Configuration;
```

```
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.TextInputFormat;
import org.apache.hadoop.mapred.TextOutputFormat;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class FlightRatings extends Configured implements Tool {
   public static class MapClass extends MapReduceBase implements Mapper<LongWritable, Text,
Text, Text> {
      private Text loc = new Text();
      private Text rating = new Text();

      @Override
      public void map(LongWritable key, Text value, OutputCollector<Text, Text> output, Reporter
reporter)
            throws IOException {
         String[] rows = value.toString().split(",");
         if (rows.length == 29) {
            String depTime = rows[4]; // actual departure time
            String depDelay = rows[15]; // actual departure time
            String origin = rows[16]; // actual departure time

            loc.set("fligt");

            rating.set(1 + "\t" + depTime + "\t" + depDelay + "\t" + origin);
            output.collect(loc, rating);
         }
      }
```

```
    }

    public static class Reduce extends MapReduceBase implements Reducer<Text, Text, Text, Text> {
        @Override
        public void reduce(Text key, Iterator<Text> values, OutputCollector<Text, Text> output, Reporter
reporter)
                throws IOException {
            double ontime = 0;
            double numTotal = 0;
            double rating = 0;

            while (values.hasNext()) {
                String tokens[] = (values.next().toString()).split("\t");
                int tot = Integer.parseInt(tokens[0]);
                if(onTime(tokens[1], tokens[2]))
                    ++ontime;
                numTotal = numTotal + tot;
            }

            rating = (ontime / numTotal) * 100;

            output.collect(key, new Text(numTotal + "\t" + ontime + "\t" + rating));
        }

        public boolean onTime(String str1, String str2) {
            if(!(str1.length() < 3 || str2.length() < 3))
            {
                if (str1.length() == 3) {
                    StringBuilder build = new StringBuilder();
                    build.append('0');
                    build.append(str1.charAt(0));
                    build.append(":");
                    build.append(str1.charAt(1));
                    build.append(str1.charAt(2));
                    build.append(":00");
                    str1 = build.toString();
                }
                if (str1.length() == 4) {
                    StringBuilder build = new StringBuilder();
                    build.append(str1.charAt(0));
```

```
    build.append(str1.charAt(1));
    build.append(":");
    build.append(str1.charAt(2));
    build.append(str1.charAt(3));
    build.append(":00");
    str1 = build.toString();
}
if (str2.length() == 3) {
    StringBuilder build = new StringBuilder();
    build.append('0');
    build.append(str2.charAt(0));
    build.append(":");
    build.append(str2.charAt(1));
    build.append(str2.charAt(2));
    build.append(":00");
    str2 = build.toString();
}
if (str2.length() == 4) {
    StringBuilder build = new StringBuilder();
    build.append(str2.charAt(0));
    build.append(str2.charAt(1));
    build.append(":");
    build.append(str2.charAt(2));
    build.append(str2.charAt(3));
    build.append(":00");
    str2 = build.toString();
}

String time1 = str1;
String time2 = str2;

SimpleDateFormat format = new SimpleDateFormat("HH:mm:ss");
try {
    Date date1 = format.parse(time1);
    Date date2 = format.parse(time2);
    long diff = date2.getTime() - date1.getTime();

    long diffMinutes = diff / (60 * 1000) % 60;
    long diffHours = diff / (60 * 60 * 1000) % 24;
```

```
            if (diffMinutes >= -5 && diffMinutes <= 5 && diffHours == 0) {
                return true;
            }
        } catch (ParseException e) {
            e.printStackTrace();
            System.out.println("Parse errorr.");
        }
    }

    return false;
  }
}

static int printUsage() {
  System.out.println("FlightRatings [-m <maps>] [-r <reduces>] <input> <output>");
  return 0;
}

@Override
public int run(String[] args) throws IOException {
  return 0;
}

public static void main(String[] args) throws IOException {
  JobConf conf = new JobConf(FlightRatings.class);
  conf.setJobName("FlightRatings");

  conf.setOutputKeyClass(Text.class);
  conf.setOutputValueClass(Text.class);

  conf.setMapperClass(MapClass.class);
  conf.setReducerClass(Reduce.class);

  conf.setInputFormat(TextInputFormat.class);
  conf.setOutputFormat(TextOutputFormat.class);

  FileInputFormat.setInputPaths(conf, new Path(args[0]));
  FileOutputFormat.setOutputPath(conf, new Path(args[1]));

  JobClient.runJob(conf);
```

```
    }
    public abstract class Clustering {

        protected List<Point> points = new ArrayList<Point>();
        protected List<Point> centers = new ArrayList<Point>();

        protected abstract void initialize(int num);
        protected abstract void show();
        protected abstract void performClustering();

        protected void loadFromFile()
        {
            points = ReadFile.readFile("data");
        }
        public final void clustering(int num)
        {
            loadFromFile();
            initialize(num);
            performClustering();
            show();
        }
    }
    import java.security.SecureRandom;
import java.util.*;

public class Kmeans extends Clustering{


    static final String AB =
"0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";
    static SecureRandom rnd = new SecureRandom();

    private String randomString( int len ){
       StringBuilder sb = new StringBuilder( len );
       for( int i = 0; i < len; i++ )
          sb.append( AB.charAt( rnd.nextInt(AB.length()) ) );
       return sb.toString();
    }

    @Override
```

```
protected void initialize(int numOfGroup) {
    int x=-1,y=-1;
    String z = "";
    Random rand = new Random();
    for (int i = 0; i < numOfGroup; i++) {
        rand = new Random();
        for (int j = 0; j < 3; j++) {
            if(j == 0)
                x = rand.nextInt(250);
            else if(j == 1)
                y = rand.nextInt(250);
            else if(j == 2)
                z = randomString(4);
        }
        centers.add(new Point(x,y,z));
    }
}
@Override
protected void show() {
    System.out.println("For kmeans the clusters are : ");
    for (int i = 0; i < centers.size(); i++) {
        if(i == centers.size() - 1)
            System.out.println(centers.get(i));
        else
            System.out.println(centers.get(i) + ", ");
    }
}
@Override
protected void performClustering() {

    List<Point> temp = new ArrayList<Point>();
    for(Point p : centers)
        temp.add(new Point(p.getX(), p.getY(), p.getZ()));
    do{
        findGroups();
        int count = 0;
        for (int i = 0; i < centers.size(); i++)
            if(centers.get(i).equals(temp.get(i)))
                count++;
        if(count == centers.size())
```

```
            break;
        temp.clear();
        for(Point p : centers)
            temp.add(new Point(p.getX(), p.getY(), p.getZ()));
    }while(true);

}
private double getDistance(Point p1, Point p2) {
    return Math.sqrt( Math.pow(p1.getX() - p2.getX(), 2) +
                    Math.pow(p1.getY() - p2.getY(), 2) +
                    Math.pow((Integer.parseInt(p1.getZ()) - Integer.parseInt(p2.getZ())), 2));
}
private void findGroups()
{
    for (int i = 0; i < points.size(); i++) {
        double minDistance = getDistance( points.get(i), centers.get(0) );
        points.get(i).setGroup(0);
        for (int j = 1; j < centers.size(); j++) {
            double dis = getDistance( points.get(i), centers.get(j) );
            if ( dis < minDistance ) {
                    minDistance = dis;
                    points.get(i).setGroup(j);
            }
        }
    }
    for (int j = 0; j < centers.size(); j++) {
        int x = 0, y = 0, z = 0, count = 1;
        for (int i = 0; i < points.size(); i++) {
            if(points.get(i).getGroup() == j){
                x += points.get(i).getX();
                y += points.get(i).getY();
                z += Integer.parseInt(points.get(i).getZ());
                count++;
            }
        }
        Integer num = z / count;
        centers.get(j).setX(x / count);
        centers.get(j).setY(y / count);
        centers.get(j).setZ(num.toString());
        count = 1;
```

```
        }
    }


}

public class Point {

    private int depTime;
    private int depDel;
    private String origin;
    private int group;

    public Point(int x, int y, String z) {
        super();
        this.depTime = x;
        this.depDel = y;
        this.origin = z;
    }

    public int getGroup() {
        return group;
    }

    public void setGroup(int group) {
        this.group = group;
    }

    public int getX() {
        return depTime;
    }
    public void setX(int x) {
        this.depTime = x;
    }
    public int getY() {
        return depDel;
    }
    public void setY(int y) {
        this.depDel = y;
    }
```

```
    public String getZ() {
        return origin;
    }
    public void setZ(String z) {
        this.origin = z;
    }



    @Override
    public String toString() {
        return "Point [x=" + depDel + ", y=" + depTime + ", z=" + origin + "]";
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        Point other = (Point) obj;
        if (depTime != other.depTime)
            return false;
        if ( depDel != other.depDel)
            return false;
        if (!origin.equals(other.origin))
            return false;
        return true;
    }

}

}
```