作业2

> 姓名： 吴双
>
> 学号： 10164102141

1 软件体系结构

Map

> 要求
>
> > 班级学生成绩的随机生成
> >
> > > 输入：本班同学的学号 输出：<学号，成绩>
>
> 数据
>
> > `stuID.csv文件`，每一列为一个学号，在 hdfs 上，shell input:

```
hdfs dfs put stuID.csv input
```

reduceByKey

> 要求
>
> > 求平均成绩：将全班同学每隔5号分为一组，求每组的平均成绩
> >
> > > 输入： <学号，成绩> 输出：<组号，平均分>
> >
> > 数据
> >
> > > 一个`score.csv文件`，每一列为学号和学生成绩，放入`HDFS`中，shell input:

```
hdfs dfs put score.csv input
```

NaturalJoin

> 要求：
>
> > 匹配出person中每个人所在的位置信息；
> >
> > 每条记录各个字段之间以空格为分隔符。

> 数据:
>
> > `person.txt`
> >
> > `address.txt`

Kmeans

> 数据
>
> > `k-means.dat`
> >
> > > 第一行标明K的值和数据个数N, 均为整形, 由","隔开 (如 3,10 表示K=3, N=10)。
> > >
> > > 之后N行中每行代表一个二维向量, 向量元素均为整形, 由","隔开 (如 1,2 表示向量(1, 2))。
> >
> > 输出:
> >
> > > K行, 每行是一个聚类图心的二维向量, 向量元素均为浮点型 (如 1.1,2.3)。

2 详细设计

Map:

code:

```java
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.*;
import org.apache.spark.api.java.function.Function;

import java.util.ArrayList;
import java.util.List;
import java.util.Random;

public class StuScore {

    private static Random rand = new Random();

    public static void main(String[] args) {
        SparkConf conf = new SparkConf().setAppName("StuScore");
        JavaSparkContext sc = new JavaSparkContext(conf);
        String logFile = "hdfs:///user/hadoop/input/stuID.csv";
        JavaRDD<String> stuID = sc.textFile(logFile);

        JavaRDD<String> stuScore = stuID.map(
                new Function<String, String>() {
                    @Override
                    public String call(String s) throws Exception {
                        String tmp  =  s + " " +  String.valueOf( rand.nextInt(100)+1);
                        return tmp;
                    }
                }
        );
```

```

  stuScore.saveAsTextFile("hdfs:///user/hadoop/output/spark/StuRandomScore");
    }
}
```

运行

本地运行

shell input:

```
spark-submit --class StuScore StuScore-1.0.jar
```

查看结果,shell input:

```
hdfs dfs -cat output/spark/StuRandomScore/*
```

集群上运行

同样，需要将HDFS地址更改为集群的地址，例如：

> hdfs:///user/hadoop/input/stuID.csv => hdfs://10.11.6.91:9000/user/hadoop23/input/stuID.csv

shell input:

```
spark-submit  --class StuScore StuScore-1.0.jar
```

查看运行结果,shell input:

```
hdfs dfs -cat output/spark/StuRandomScore/*
```

reducebykey

code:

```
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.*;
import org.apache.spark.api.java.function.Function;
import org.apache.spark.api.java.function.Function2;
import org.apache.spark.api.java.function.PairFunction;
import scala.Tuple2;


import java.util.Iterator;
import java.util.Random;

public class AVGScore {
```

```java
    private static Integer groupSize = 5;

    public static void main(String[] args) {
        SparkConf conf = new SparkConf().setAppName("AVGScore");
        JavaSparkContext sc = new JavaSparkContext(conf);
        sc.setLogLevel("WARN"); //http://stackoverflow.com/questions/27781187/how-to-stop-messages-displaying-on-spark-console

        String logFile = "hdfs:///user/hadoop/input/score.csv";

        JavaRDD<String> fileRDD = sc.textFile(logFile);


        /**
         * map string to (id, score) and convert to (group_id, (score,1))
         * reduceByKey => (group_id,(sumScore, count)
         * and then mapValues to avg score
         */

        JavaPairRDD<Integer, Double> stuScore = fileRDD.mapToPair(
                line -> new Tuple2<>(
                        (Integer.parseInt(line.split(",")[0]) + 1 )/5, new Tuple2<>(Double.parseDouble(line.split(",")[1]),1)))
                .reduceByKey(
                        (x,y) -> new Tuple2<>(x._1 + y._1,x._2+y._2))
                .mapValues(x -> x._1/x._2);

        stuScore.saveAsTextFile("hdfs:///user/hadoop/output/spark/AVGscore");
    }
}
```

运行

本地运行

运行程序，shell input:

```shell
spark-submit --class AVGScore AVGScore-1.0.jar
```

查看结果，同时查看行号，shell input:

```shell
hdfs dfs -cat output/spark/AVGscore/* | wc -l
```

集群上运行

需要将HDFS地址更改为集群的地址，例如：

> hdfs:///user/hadoop/input/score.csv => hdfs://10.11.6.91:9000/user/hadoop23/input/score.csv

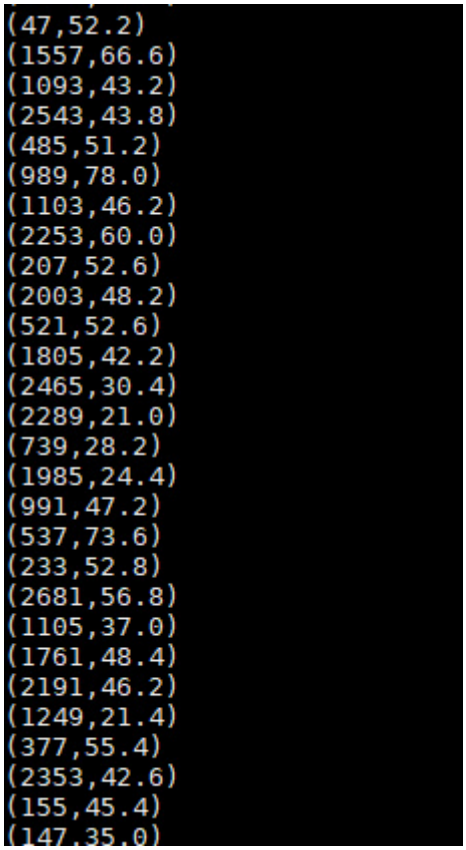
提交作业并运行，shell input:

```
spark-submit --master spark://10.11.6.91:7077 --class AVGScore AVGScore-1.0.jar
```

查看结果，shell input:

```
hdfs dfs -cat output/spark/AVGscore/*
```

output:



```
(47,52.2)
(1557,66.6)
(1093,43.2)
(2543,43.8)
(485,51.2)
(989,78.0)
(1103,46.2)
(2253,60.0)
(207,52.6)
(2003,48.2)
(521,52.6)
(1805,42.2)
(2465,30.4)
(2289,21.0)
(739,28.2)
(1985,24.4)
(991,47.2)
(537,73.6)
(233,52.8)
(2681,56.8)
(1105,37.0)
(1761,48.4)
(2191,46.2)
(1249,21.4)
(377,55.4)
(2353,42.6)
(155,45.4)
(147,35.0)
```

Natural join

code:

```
import org.apache.commons.lang.StringUtils;
import org.apache.hadoop.hdfs.protocol.DirectoryListing;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.JavaPairRDD;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.JavaSparkContext;
import org.apache.spark.api.java.function.Function;
import org.apache.spark.api.java.function.PairFunction;
import scala.Tuple2;

public class NaturalJoin {


```

```java
    public static void main(String args[]) {
        SparkConf conf = new SparkConf().setAppName("NaturalJoin");
        JavaSparkContext sc = new JavaSparkContext(conf);
        sc.setLogLevel("WARN"); //http://stackoverflow.com/questions/27781187/how-to-stop-messages-displaying-on-spark-console


        String addFile = "hdfs:///user/hadoop/input/address.txt";
        String personFile = "hdfs:///user/hadoop/input/person.txt";


        /***
         * return code,city
         */
        JavaPairRDD<Integer, String> addRDD = sc.textFile(addFile).mapToPair(
                line -> new Tuple2<>(
                        Integer.parseInt(line.split(" ")[0]), line.split(" ")[1]));

        /**
         * return return code,{ id + name }
         */
        JavaPairRDD<Integer, String> personRDD = sc.textFile(personFile).mapToPair(
                new PairFunction<String, Integer, String>() {
                    @Override
                    public Tuple2<Integer, String> call(String s) throws Exception {
                        String[] splitLines = StringUtils.split(s, " ");
                        if (splitLines.length < 3)
                            return null;
                        return new Tuple2<>(Integer.parseInt(splitLines[2]), splitLines[0] + splitLines[1]);
                    }
                }
        );

        /**
         * return code, [{id + name}, city]
         */
        JavaPairRDD<Integer, Tuple2<String,String>> resultRDD = personRDD.join(addRDD);

        resultRDD.saveAsTextFile("hdfs:///user/hadoop/output/spark/NaturalJoin");
    }
}
```

运行

本地运行

运行程序并查看结果,shell input:

```
spark-submit --class NaturalJoin NaturalJoin-1.0.jar
hdfs dfs -cat output/spark/NaturalJoin/*
```

集群上运行

需要将HDFS地址更改为集群的地址，例如：

> hdfs:///user/hadoop/input/address.txt => hdfs://10.11.6.91:9000/user/hadoop23/input/address.txt

提交作业并运行,shell input:

```
spark-submit --master spark://10.11.6.91:7077 --class NaturalJoin NaturalJoin-1.0.jar
```

查看结果,ishell input:

```
hdfs dfs -cat output/spark/NaturalJoin/*
```

Kmeans

code:

Kmeans.java

```
import org.apache.spark.api.java.JavaPairRDD;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.JavaSparkContext;
import org.apache.spark.api.java.function.Function2;
import scala.Serializable;
import scala.Tuple2;

import java.util.ArrayList;
import java.util.List;

public class Kmeans implements KmeansInterface, Serializable {

```

```java
    // every point has a cluster number and point(x,y)
    private List<Tuple2<Integer, Point>> oldCenterList = new ArrayList<>();
    private List<Tuple2<Integer, Point>> newCenterList = new ArrayList<>();
    private double threshold = 0.000001;


    /**
     * @param point
     * @return cluster belonged
     * @Method get the closest cluster for the point
     */
    public int findClosest(Point point) {
        int argmin = -1;
        double minimalDistance = Double.MAX_VALUE;
        for (Tuple2<Integer, Point> i : oldCenterList) {
            double distance = point.EuclideanDis(i._2);
            if (distance < minimalDistance) {
                minimalDistance = distance;
                argmin = i._1;
            }
        }
        return argmin;
    }


    /**
     * @param outFile string
     * @Method save center to txt
     */
    public void saveToFile(String outFile, JavaSparkContext sc) {
        List<String> outCenterList = new ArrayList<>();
//        format center points
        for (Tuple2<Integer, Point> tmp : newCenterList) {
            outCenterList.add(String.valueOf(tmp._2.getX()) + " " + String.valueOf(tmp._2.getY()));
        }

        JavaRDD<String> center = sc.parallelize(outCenterList);
        center.saveAsTextFile(outFile);
    }


    /**
     * @return False for not stable
     * @Method compare two cluster center with threshold
     */
    public boolean clusterCompare() {

        for (Tuple2<Integer, Point> oldCenter : oldCenterList) {
            int clusterNum = oldCenter._1;
            for (Tuple2<Integer, Point> newCenter : newCenterList) {
                if (newCenter._1 == clusterNum) {
                    double dis = oldCenter._2.EuclideanDis(newCenter._2);
```

```
                    if (dis > threshold)
                        return false;
                    break;
                }
            }
        }
        return true;
    }


    /**
     * @param kmeansRDD
     * @return init pointsRDD
     * @Method prepare Points RDD and select clusters randomly
     */
    public JavaPairRDD<Integer, Point> Prepare(JavaRDD<String> kmeansRDD) {
        // get the number of cluster
        String fisrtLine = kmeansRDD.first();
        int clusterCount = Integer.parseInt(fisrtLine.split(",")[0]);

//        filter first line and convert to <Point,clusternum>, init set all cluster number 1
        JavaPairRDD<Integer, Point> pointsRDD = kmeansRDD.filter(line -> !line.equals(fisrtLine)).mapToPair(
                line -> {
                    String[] splitLine = line.split(",");
                    double X = Double.parseDouble(splitLine[0]);
                    double Y = Double.parseDouble(splitLine[1]);
                    return new Tuple2<>(0, new Point(X, Y));
                }
        );

//       init center list
        oldCenterList.addAll(pointsRDD.take(clusterCount));

        for (int i = 0; i < clusterCount; i++) {
            Tuple2<Integer, Point> tmp = oldCenterList.get(i);
            oldCenterList.set(i, new Tuple2<>(i, tmp._2));
        }

        newCenterList.addAll(oldCenterList);

        return pointsRDD;

    }


    /**
     * @param pointsRDD to cluster
     * @return new classify PointsRDD
     * @method cluster and update new cluster center
     */
```

```java
    public JavaPairRDD<Integer, Point> cluster(JavaPairRDD<Integer, Point> pointsRDD) {

        JavaPairRDD<Integer, Point> newPointsRDD = pointsRDD.mapToPair(
                kv -> new Tuple2<>(findClosest(kv._2), kv._2)
        );

        JavaPairRDD<Integer, Point> newClusterRDD = newPointsRDD
                .mapValues(
                        value -> new Tuple2<>(value, 1))
                .reduceByKey(
                        new Function2<Tuple2<Point, Integer>, Tuple2<Point, Integer>, Tuple2<Point, Integer>>() {
                            @Override
                            public Tuple2<Point, Integer> call(Tuple2<Point, Integer> value1, Tuple2<Point, Integer> value2) throws Exception {
                                Point tmp = new Point(value1._1.getX() + value2._1.getX(), value1._1.getY() + value2._1.getY());
                                int count = value1._2 + value2._2;
                                return new Tuple2<>(tmp, count);
                            }
                        }
                ).mapValues(
                        v -> new Point(v._1.getX() / v._2, v._1.getY() / v._2)
                );

        oldCenterList.clear();
        oldCenterList.addAll(newCenterList);
        // convert to list to store
        newCenterList.clear();
        newCenterList.addAll(newClusterRDD.collect());

        return newPointsRDD;
    }

}
```

kmeansRun.java

```java
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.JavaPairRDD;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.JavaSparkContext;
import scala.Tuple2;

import java.util.ArrayList;
import java.util.List;

public class kmeansRun {

    public static void main(String args[]) {
        Kmeans kmeans = new Kmeans();

```

```java
        SparkConf conf = new SparkConf().setAppName("Kmeans");
        JavaSparkContext sc = new JavaSparkContext(conf);

        String kmeansFile = "hdfs://10.11.6.91:9000/user/hadoop23/input/k-means.dat";
        String outFile = "hdfs://10.11.6.91:9000/user/hadoop23/output/spark/kmeans";

        int counter = 1, maxIteration = 500;
        boolean result = false;

        JavaRDD<String> kmeansRDD = sc.textFile(kmeansFile).cache();

//        init centerList and Points RDD
        JavaPairRDD<Integer, Point> PointsRDD = kmeans.Prepare(kmeansRDD);

        while (!result && counter < maxIteration) {
            PointsRDD = kmeans.cluster(PointsRDD);

            result = kmeans.clusterCompare();
            System.out.println("*******KMEANS finished iteration:>> " + counter + " || means stable: " + result);
            counter++;
        }

        kmeans.saveToFile(outFile,sc);

    }
}
```

运行

本地运行

运行程序, shell input:

```shell
spark-submit --class kmeansRun Kmeans-1.0.jar 2>&1 | grep "KMEANS finished iteration:"
```

output:

```
*******KMEANS finished iteration:>> 1 || means stable: false
*******KMEANS finished iteration:>> 2 || means stable: false
*******KMEANS finished iteration:>> 3 || means stable: false
*******KMEANS finished iteration:>> 4 || means stable: false
*******KMEANS finished iteration:>> 5 || means stable: false
*******KMEANS finished iteration:>> 6 || means stable: false
*******KMEANS finished iteration:>> 7 || means stable: false
*******KMEANS finished iteration:>> 8 || means stable: true
```

查看结果, shell input:

```shell
hdfs dfs -cat output/spark/kmeans/*
```

output:

```
98.31,803.89
496.11650485436894,207.3398058252427
493.2277227722772,798.3267326732673
91.05208333333333,206.19791666666666
```

集群上运行

需要将HDFS地址更改为集群的地址，例如：

> hdfs:///user/hadoop/input/k-means.dat => hdfs://10.11.6.91:9000/user/hadoop23/input/k-means.dat

提交作业并运行, shell input：

```
spark-submit --master spark://10.11.6.91:7077 --class kmeansRun Kmeans-1.0.jar 2>&1 |
grep "KMEANS finished iteration:"
```

output:

```
*******KMEANS finished iteration:>> 1 || means stable: false
*******KMEANS finished iteration:>> 2 || means stable: false
*******KMEANS finished iteration:>> 3 || means stable: false
*******KMEANS finished iteration:>> 4 || means stable: false
*******KMEANS finished iteration:>> 5 || means stable: false
*******KMEANS finished iteration:>> 6 || means stable: false
*******KMEANS finished iteration:>> 7 || means stable: false
*******KMEANS finished iteration:>> 8 || means stable: true
```

查看结果：

```
18/11/01 22:22:50 WARN util.NativeCodeLoader: Unable to load nativ
. using builtin-java classes where applicable
98.31 803.89
493.2277227722772 798.3267326732673
496.11650485436894 207.3398058252427
91.05208333333333 206.19791666666666
```