作业4

> 姓名： 吴双
>
> 学号： 10164102141

1 软件体系结构

> `person.txt`
>
> `address.txt`
>
> `README.md`
>
> `map.java`
>
> `ReduceByKey.java`
>
> `NaturalJoin.java`

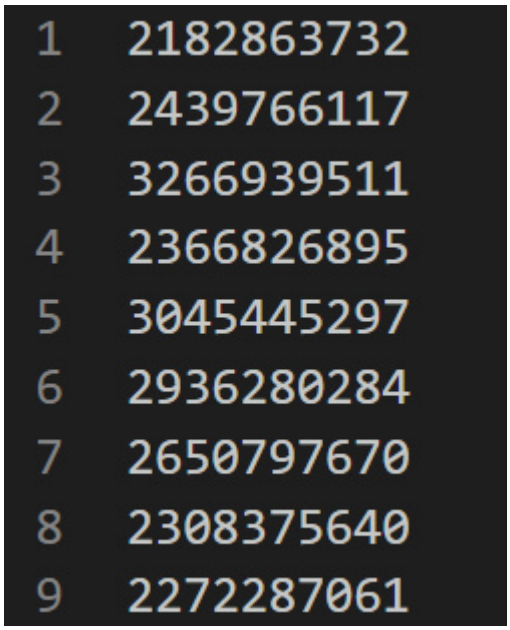
> `KMeans`
>
> > `k-means.dat`
> >
> > `README_k-means.md`
> >
> > `point.java`
> >
> > `kmeansRun.java`

2 详细设计

> Map
>
> 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>

数据准备

1. 首先需要一个 stuID.csv文件， 每一列为一个学号：



2. 然后将文件放入 HDFS 中：

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

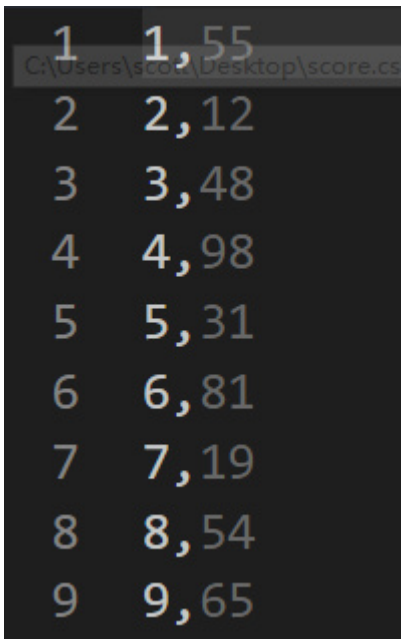
ReduceByKey

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>

数据准备

1. 首先需要一个`score.csv文件`，每一列为学号和学生成绩：



> Natural join
>
> > person.txt
> >
> > > Aaron 210000
> > >
> > > Abbott 214000
> > >
> > > Abel 221000
> > >
> > > Abner 215000
> > >
> > > Abraham 226000
> > >
> > > Adair 225300
> > >
> > > Adam 223800
> > >
> > > Addison 224000
> > >
> > > Adolph 223001
>
> > address.txt
> >
> > > 210000 Nanjing
> > >
> > > 214000 Wuxi
> > >
> > > 221000 Xuzhou
> > >
> > > 213000 Changzhou
>
> 要求以code为连接属性，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。

> KMeans
>
> 输入：
>
> - 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)。

2.4 编码实现及问题分析

Map部分：

```
import org.apache.flink.api.common.functions.MapFunction;
import org.apache.flink.api.java.DataSet;
import org.apache.flink.api.java.ExecutionEnvironment;
import org.apache.flink.api.java.tuple.Tuple2;
import org.apache.flink.api.java.utils.ParameterTool;
import java.util.Random;

```

```java
public class StuScore {
    private static Random rand = new Random();

    public StuScore(){}

    public static void main(String[] args) throws Exception {
        ParameterTool params = ParameterTool.fromArgs(args);
        ExecutionEnvironment env = ExecutionEnvironment.getExecutionEnvironment();
        env.getConfig().setGlobalJobParameters(params);
        DataSet<String> text;

        if(params.has("input")){
            text = env.readTextFile(params.get("input"));
        }else{
            System.out.println("Please confirm input keywords!");
            return;
        }


        DataSet<Tuple2<String,Integer>> stuscore =  text.map(new MapFunction<String, Tuple2<String, Integer>>() {
            @Override
            public Tuple2<String, Integer> map(String s) throws Exception {
                return new Tuple2<>(s,rand.nextInt(100) +1);
            }
        });

        //如果没有指定输出，则默认打印到控制台
        if(params.has("output")){
            stuscore.writeAsCsv(params.get("output"),"\n", ",");
            env.execute();
        }else{
            System.out.println("Printing result to stdout. Use --output to specify output path.");
            stuscore.print();
        }

    }
}
```

ReduceByKey部分：

```java
import org.apache.commons.compress.archivers.dump.DumpArchiveEntry;
import org.apache.flink.api.common.typeinfo.Types;
import org.apache.flink.api.java.DataSet;
import org.apache.flink.api.java.ExecutionEnvironment;
import org.apache.flink.api.java.tuple.Tuple2;
import org.apache.flink.api.java.utils.ParameterTool;

public class AVGscore {
    private static Integer groupSize = 5;

    public static void main(String[] args) throws Exception {
```

```
        ParameterTool params = ParameterTool.fromArgs(args);
        ExecutionEnvironment env = ExecutionEnvironment.getExecutionEnvironment();
        env.getConfig().setGlobalJobParameters(params);
        DataSet<Tuple2<Integer, Double>> fileDataSet;

        if (params.has("input")) {
            fileDataSet = env.readCsvFile(params.get("input"))
                    .types(Integer.class, Double.class);
        } else {
            System.out.println("Please confirm input keywords!");
            return;
        }

        /**
         * map string to (id, score) and convert to (group_id, (score,1))
         * GroupBy and reduce == reduceByKey
         * and then map to avg score
         */
        DataSet<Tuple2<Integer, Double>> stuAVGscore = fileDataSet
                .map(line -> Tuple2.of(
                        (line.f0-1)/ 5, Tuple2.of(line.f1, 1)))
                .returns(Types.TUPLE(Types.INT, Types.TUPLE(Types.DOUBLE,
Types.INT)))
                .groupBy(0)
                .reduce(
                        (kv1, kv2) -> Tuple2.of(kv1.f0, Tuple2.of(kv1.f1.f0 +
kv2.f1.f0, kv1.f1.f1 + kv2.f1.f1)))
                .returns(Types.TUPLE(Types.INT, Types.TUPLE(Types.DOUBLE,
Types.INT)))
                .map(
                        line -> Tuple2.of(line.f0, line.f1.f0 / line.f1.f1)
                ).returns(Types.TUPLE(Types.INT, Types.DOUBLE));

        //如果没有指定输出，则默认打印到控制台
        if (params.has("output")) {
            stuAVGscore.writeAsCsv(params.get("output"), "\n", ",");
            env.execute();
        } else {
            System.out.println("Printing result to stdout. Use --output to specify
output path.");
            stuAVGscore.print();
        }

    }
}
```

NaturalJoin部分:

```
import org.apache.flink.api.common.functions.FlatJoinFunction;
import org.apache.flink.api.common.typeinfo.Types;
import org.apache.flink.api.java.DataSet;
import org.apache.flink.api.java.ExecutionEnvironment;
import org.apache.flink.api.java.tuple.Tuple2;
```

```java
import org.apache.flink.api.java.tuple.Tuple3;
import org.apache.flink.api.java.tuple.Tuple4;
import org.apache.flink.api.java.utils.ParameterTool;
import org.apache.flink.util.Collector;
import scala.Int;

import java.lang.reflect.Type;

public class NaturalJoin {

    public static void main(String args[]) throws Exception{
        ParameterTool params = ParameterTool.fromArgs(args);
        ExecutionEnvironment env = ExecutionEnvironment.getExecutionEnvironment();
        env.getConfig().setGlobalJobParameters(params);

        // code, city
        DataSet<Tuple2<Integer, String>> addDataSet;
        // id ,name, code
        DataSet<Tuple3<Integer, String, Integer>> personDataSet;

        if (params.has("addinput")) {
            addDataSet = env.readCsvFile(params.get("addinput"))
                    .fieldDelimiter(" ")
                    .ignoreInvalidLines()
                    .types(Integer.class, String.class);
        } else {
            System.out.println("Please confirm input keywords!");
            return;
        }

        if (params.has("personinput")) {
            personDataSet = env.readCsvFile(params.get("personinput"))
                    .fieldDelimiter(" ")
                    .ignoreInvalidLines()
                    .types(Integer.class, String.class, Integer.class);
        } else {
            System.out.println("Please confirm input keywords!");
            return;
        }

        DataSet<Tuple4<Integer, String, Integer, String>> result =
personDataSet.join(addDataSet)
                .where(2)
                .equalTo(0)
                .with(
                        (x, y) -> Tuple4.of(x.f0, x.f1, x.f2, y.f1)

).returns(Types.TUPLE(Types.INT,Types.STRING,Types.INT,Types.STRING));

        personDataSet.print();

        //如果没有指定输出，则默认打印到控制台
        if (params.has("output")) {
```

```
            result.writeAsCsv(params.get("output"), "\n", ",");
            env.execute();
        } else {
            System.out.println("Printing result to stdout. Use --output to specify output path.");
            result.print();
        }

    }
}
```

KMeans部分：

point.java:用于自定义point类（POJO对象)

```
import java.io.Serializable;

public class Point implements Serializable {
    public double x,y;

    public Point() {

    }

    public Point(double x, double y) {
        this.x = x;
        this.y = y;
    }

    public Point add(Point other) {
        x += other.x;
        y += other.y;
        return this;
    }

    public Point div(long val) {
        x /= val;
        y /= val;
        return this;
    }

    public double euclideanDistance(Point other) {
        return Math.sqrt((x - other.x) * (x - other.x) + (y - other.y) * (y - other.y));
    }

    public String toString() {
        return x + " " + y;
    }

}
```

KMeansRun.java:

```java
import org.apache.flink.api.common.functions.*;
import org.apache.flink.api.java.DataSet;
import org.apache.flink.api.java.ExecutionEnvironment;
import org.apache.flink.api.java.functions.FunctionAnnotation;
import org.apache.flink.api.java.operators.IterativeDataSet;
import org.apache.flink.api.java.tuple.Tuple2;
import org.apache.flink.api.java.tuple.Tuple3;
import org.apache.flink.api.java.utils.ParameterTool;
import org.apache.flink.configuration.Configuration;

import java.io.BufferedReader;
import java.io.FileReader;
import java.util.ArrayList;
import java.util.Collection;

public class Kmeans {

    public static void main(String[] args) throws Exception {
        // Checking input parameters
        final ParameterTool params = ParameterTool.fromArgs(args);
        // set up execution environment
        ExecutionEnvironment env = ExecutionEnvironment.getExecutionEnvironment();
        env.getConfig().setGlobalJobParameters(params); // make parameters available in the web interface

        // get input data:
        DataSet<Point> points = getPoint(params, env);
        if (points == null)
            return;
        DataSet<Centroid> centroids = getCentroid(params, env);


        // set number of bulk iterations for KMeans algorithm
        IterativeDataSet<Centroid> loop = centroids.iterate(params.getInt("iterations", 100));

        DataSet<Centroid> newCentroids = points
                // compute closest centroid for each point
                .map(new SelectNearestCenter()).withBroadcastSet(loop, "centroids")
                // count and sum point coordinates for each centroid
                .map(new CountAppender())
                .groupBy(0).reduce(new CentroidAccumulator())
                // compute new centroids from point counts and coordinate sums
                .map(new CentroidAverager());


        // feed new centroids back into next iteration
        DataSet<Centroid> finalCentroids = loop.closeWith(newCentroids, newCentroids.filter(new thresholdFilter()).withBroadcastSet(loop,"centroids"));

//        DataSet<Tuple2<Integer, Point>> clusteredPoints = points
//                // assign points to final clusters
```

```
//                .map(new SelectNearestCenter()).withBroadcastSet(finalCentroids,
"centroids");

        // emit result
        if (params.has("output")) {
            finalCentroids.writeAsCsv(params.get("output"), "\n", " ");
            env.execute();
        } else {
            System.out.println("Printing result to stdout. Use --output to specify
output path.");
            finalCentroids.print();
        }
    }

    private static DataSet<Point> getPoint(ParameterTool params,
ExecutionEnvironment env) {
        DataSet<Point> points;
        if (params.has("input")) {
            // read points from CSV file
            points = env.readCsvFile(params.get("input"))
                    .ignoreFirstLine()
                    .pojoType(Point.class, "x", "y");
        } else {
            System.out.println("Use --input to specify file input.");
            return null;
        }
        return points;
    }

    private static DataSet<Centroid> getCentroid(ParameterTool params,
ExecutionEnvironment env) throws Exception {
        ArrayList<Centroid> centroidArrayList = new ArrayList<>();
        BufferedReader br = new BufferedReader(new FileReader(params.get("input")));
        String text = br.readLine();
        int k = Integer.parseInt(text.split(",")[0]);

        while (k != 0) {
            text = br.readLine();
            double x = Double.parseDouble(text.split(",")[0]);
            double y = Double.parseDouble(text.split(",")[1]);
            centroidArrayList.add(new Centroid(k, x, y));
            k--;
        }

        DataSet<Centroid> centroids = env.fromCollection(centroidArrayList);
        return centroids;
    }

    /**
     * Determines the closest cluster center for a data point.
     */
    @FunctionAnnotation.ForwardedFields("*->1")
```

```
    public static final class SelectNearestCenter extends RichMapFunction<Point,
Tuple2<Integer, Point>> {
        private Collection<Centroid> centroids;

        /**
         * Reads the centroid values from a broadcast variable into a collection.
         */
        @Override
        public void open(Configuration parameters) throws Exception {
            this.centroids = getRuntimeContext().getBroadcastVariable("centroids");
        }

        @Override
        public Tuple2<Integer, Point> map(Point p) throws Exception {

            double minDistance = Double.MAX_VALUE;
            int closestCentroidId = -1;

            // check all cluster centers
            for (Centroid centroid : centroids) {
                // compute distance
                double distance = p.euclideanDistance(centroid);

                // update nearest cluster if necessary
                if (distance < minDistance) {
                    minDistance = distance;
                    closestCentroidId = centroid.id;
                }
            }

            // emit a new record with the center id and the data point.
            return new Tuple2<>(closestCentroidId, p);
        }
    }

    /**
     * Appends a count variable to the tuple.
     */
    @FunctionAnnotation.ForwardedFields("f0;f1")
    public static final class CountAppender implements MapFunction<Tuple2<Integer,
Point>, Tuple3<Integer, Point, Long>> {
        @Override
        public Tuple3<Integer, Point, Long> map(Tuple2<Integer, Point> t) {
            return new Tuple3<>(t.f0, t.f1, 1L);
        }
    }

    /**
     * Sums and counts point coordinates.
     */
    @FunctionAnnotation.ForwardedFields("0")
    public static final class CentroidAccumulator implements
ReduceFunction<Tuple3<Integer, Point, Long>> {
```

```
148
149         @Override
150         public Tuple3<Integer, Point, Long> reduce(Tuple3<Integer, Point, Long>
    val1, Tuple3<Integer, Point, Long> val2) {
151             return new Tuple3<>(val1.f0, val1.f1.add(val2.f1), val1.f2 + val2.f2);
152         }
153     }
154
155     /**
156      * Computes new centroid from coordinate sum and count of points.
157      */
158     @FunctionAnnotation.ForwardedFields("0->id")
159     public static final class CentroidAverager implements
    MapFunction<Tuple3<Integer, Point, Long>, Centroid> {
160         @Override
161         public Centroid map(Tuple3<Integer, Point, Long> value) {
162             // id, X/num Y/num
163             return new Centroid(value.f0, value.f1.div(value.f2));
164         }
165     }
166
167     /**
168      * Filter that filters vertices where the centorid difference is below a
    threshold.
169      */
170     public static final class thresholdFilter extends RichFilterFunction<Centroid> {
171         private Collection<Centroid> centroids;
172         private double threshold = 1e-5;
173
174         /**
175          * Reads the centroid values from a broadcast variable into a collection.
176          */
177         @Override
178         public void open(Configuration parameters) throws Exception {
179             this.centroids = getRuntimeContext().getBroadcastVariable("centroids");
180         }
181
182         @Override
183         public boolean filter(Centroid centroid) {
184             for (Centroid oldcentroid : centroids) {
185                 if (centroid.id == oldcentroid.id) {
186                     // compute distance
187                     double distance = centroid.euclideanDistance(oldcentroid);
188                     return (distance > this.threshold);
189                 }
190             }
191             return true;
192         }
193     }
194 }
```

结果：

Map部分：

```
flink run -c StuScore StuScore.jar --input
/home/hadoop/Documents/distribution/Flink/StuScore/stuID.csv
```

output:

```
(3076722373,20)
(407826062,3)
(2115951107,75)
(3116395451,48)
(651866235,8)
(245285626,29)
Program execution finished
Job with JobID 6fcaba071f7e23671ad6c7ba4e836ec5 has finished.
Job Runtime: 475 ms
Accumulator Results:
- 8ab08a881ddb93127a7f75ca2aba7270 (java.util.ArrayList) [14094
```

ReduceByKey部分：

```
flink run -c AVGscore AVGscore.jar --input
/home/hadoop/Documents/distribution/Flink/AVGscore/score.csv
```

output:

```
(2813,38.2)
(2814,51.6)
(2815,71.0)
(2816,45.2)
(2817,39.4)
(2818,55.0)
Program execution finished
```

Natural join部分：

```
flink run -c NaturalJoin NaturalJoin.jar --addinput
/home/hadoop/Documents/distribution/Flink/NaturalJoin/address.txt --personinput
/home/hadoop/Documents/distribution/Flink/NaturalJoin/person.txt
```

output:

```
(70,Cliff,222000,Lianyungang)
(71,Cleveland,214000,Wuxi)
(72,Clyde,226000,Nantong)
(73,Colin,222000,Lianyungang)
(74,Conrad,226000,Nantong)
(75,Cornelius,224000,Yancheng)
Program execution finished
Job with JobID 6ab7c20915728a60c808fca6
Job Runtime: 2176 ms
```

Kmeans部分：

```
flink run -c Kmeans Kmeans.jar --input
/home/hadoop/Documents/distribution/Flink/kmeans/k-means.dat
```

output:

```
Printing result to stdout. Use --output to specify output path.
1 91.05208333333333 206.19791666666666
2 493.2277227722772 798.3267326732673
3 496.11650485436894 207.3398058252427
4 98.31 803.89
Program execution finished
Job with JobID 1dbe3cd013615653e999d27047f8971e has finished.
Job Runtime: 1405 ms
```