**西安电子科技大学**

**分布式计算**

**(2022年度)**

**实**

**验**

**报**

**告**

**实验名称：**基于MapReduce和Spark的分布式算法设计

**班 级：** 1903015

**姓 名：** 贾舜宇

**学 号：** 19030100383

一、实验内容

## 题目1：

题目1 输入文件为学生成绩信息，包含了必修课与选修课成绩，格式如下：

班级1, 姓名1, 科目1, 必修, 成绩1 <br>

班级2, 姓名2, 科目1, 必修, 成绩2 <br>

班级1, 姓名1, 科目2, 选修, 成绩3 <br>

………., ………, ………, ……… <br>

编写两个Hadoop平台上的MapReduce程序，分别实现如下功能：

1. 计算每个学生必修课的平均成绩。

2. 按科目统计每个班的平均成绩。

## 题目2：

输入文件的每一行为具有父子/父女/母子/母女/关系的一对人名，例如：

Tim, Andy <br>

Harry, Alice <br>

Mark, Louis <br>

Andy, Joseph <br>

………, ………… <br>

假定不会出现重名现象。 编写Hadoop平台上的MapReduce程序，找出所有具有grandchild-grandparent关系的人名组。

## 题目3：

输入文件为学生成绩信息，包含了必修课与选修课成绩，格式如下：

班级1, 姓名1, 科目1, 必修, 成绩1 <br>

班级2, 姓名2, 科目1, 必修, 成绩2 <br>

班级1, 姓名1, 科目2, 选修，成绩3 <br>

………., ………, ………, ……… <br>

编写一个Spark程序，同时实现如下功能：

1. 计算每个学生必修课的平均成绩。

2. 统计学生必修课平均成绩在：90~100,80~89,70~79,60~69和60分以下这5个分数段的人数。

二、设计思想

## 题目1：

MapReduce并行计算模型的计算任务主要划分为三个阶段：Map阶段，聚集混洗阶段、Reduce阶段。

**任务一**

任务一要求计算每个学生必修课的平均成绩，所以在map阶段首先要过滤掉选修课的数据;

**map**

public static class MyMapper extends org.apache.hadoop.mapreduce.Mapper<LongWritable, Text, Text, Text> {//输入的key，value；输出的key，value类型  
       Text KeyOut = new Text();  
       Text ValueOut = new Text();  
​  
       @Override  
       protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text, Text>.Context context)  
               throws IOException, InterruptedException {  
           String line = value.toString();  
           String[] splited = line.split(",");  
           //通过判断课程类型，来过滤掉选修的数据  
           if (splited[3].equals("必修")) {  
               KeyOut.set(splited[1]); //name  
               ValueOut.set(splited[4]); //score  
               context.write(KeyOut, ValueOut);  
          }  
      }  
  }

mappe阶段，输出的<key,value> 为

key:name  
value:score

在聚集混洗阶段，不同Map子任务输出的<key,value>数组按照**key值**进行聚集。

**reduce**

对于reduce阶段，输入的<key,values>为

key：name  
values：某一学生对应的所有必修课分数score的一个迭代器

public static class MyReducer extends org.apache.hadoop.mapreduce.Reducer<Text, Text, Text, Text> {  
       Text ValueOut = new Text();  
       List<Integer> scoreList = new ArrayList<>();  
​  
       @Override  
       protected void reduce(Text k2, Iterable<Text> v2s,  
                             Reducer<Text, Text, Text, Text>.Context context) throws IOException, InterruptedException {  
           scoreList.clear();  
           for (Text v2 : v2s) {  
               scoreList.add(Integer.valueOf(v2.toString()));  
          }  
​  
           int sum\_score = 0;  
           for (int score : scoreList) {  
               sum\_score += score;  
          }  
​  
           double avg\_score = sum\_score \* 1D / scoreList.size();  
​  
           ValueOut.set(String.valueOf(avg\_score));  
           context.write(k2, ValueOut);  
      }  
  }

根据题目需求计算每一个学生必修课的平均成绩，即对每一个key-value键值对，对value迭代求总和再取均值即可。

## 题目2：

实验要求我们根据数据集中的child-parent联系，推出数据集中含有的grandchild-grandparent关系；每一行具有父子/父女/母子/母女/关系的一对人名，在未经推理的情况下，每个人既有可能成为grandchild或中间辈，也有可能成为grandparent或中间辈；为了确保Reduce阶段保留统一的key值，在map阶段分别将child-parent与parent-child一起输出，通过对value值添加”-“或”+“标识来区别value代表的人物与key代表的人物之间的关系。

例如对于 Tim,Andy 经过map阶段后会输出两个键值对：

（key,value）: (Tim,-Andy) (Andy,+Tim)

表达的含义分别是：对于Tim而言Andy是它的父辈，对于Andy而言Tim是它的子代

而在reduce阶段，输入的key，value值分别代表：

key：某一个人  
value：所有和key代表的人物有关系的人物的name构成的迭代器

reduce阶段对于每一个key-value键值对，分割value中key代表人物的子代与父辈，将子代与父辈分别作为reduce输出的key与value。

**map**

public static class MyMapper extends org.apache.hadoop.mapreduce.Mapper<LongWritable, Text, Text, Text>{//输入的key，value；输出的key，value类型  
   @Override  
   protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text, Text>.Context context)  
           throws IOException, InterruptedException {  
       //分割数据，左列为child，右列是parent  
       String child = value.toString().split(",")[0];  
       String parent = value.toString().split(",")[1];  
​  
       //产生正序与逆序的key-value对，同时写入context  
       context.write(new Text(child), new Text("-" + parent));  
       context.write(new Text(parent), new Text("+" + child));  
       //Tim,Andy ->（key,value）: (Tim,-Andy) (Andy,+Tim)  
       //Andy, Joseph -> (key,value): (Andy,-Joseph) (Joseph,+Andy)  
  }  
}

**reduce**

public static class MyReducer extends org.apache.hadoop.mapreduce.Reducer<Text, Text, Text, Text>{  
   @Override  
   protected void reduce(Text k2, Iterable<Text> v2s,  
                         Reducer<Text, Text, Text, Text>.Context context) throws IOException, InterruptedException {  
       ArrayList<Text> grandparent = new ArrayList<>();  
       ArrayList<Text> grandchild = new ArrayList<>();  
​  
       for (Text v2:v2s){  
​  
           String s = v2.toString();  
​  
           if (s.startsWith("-")){//startsWith()方法用来判断当前字符串是否是以另外一个给定的子字符串“开头”的，根据判断结果返回 true 或 false。  
               grandparent.add(new Text(s.substring(1))); //截取字符串，从序1开始（包含1）  
          } else {  
               grandchild.add(new Text(s.substring(1)));  
          }  
      }  
       //对于key代表的人物，至少有一个子代和一个父辈才会被写入  
       for (int i = 0; i < grandchild.size(); i++ ){  
           for (int j = 0; j < grandparent.size(); j++){  
               context.write(grandchild.get(i), grandparent.get(j));  
          }  
      }  
  }  
}

## 题目3：

本实验重点在于RDD算子间的转换

任务一要求计算每个学生必修课的平均成绩，首先通过RDD转换算子filter过滤掉选修相关的数据，再将key-value设置为学生名字与其必修课成绩，最后进行取平均。

任务二要求统计学生在五个分数段的人数，首先将学生划分给不同的分数段并将分数段作为key值，再统计每个分数段人数。

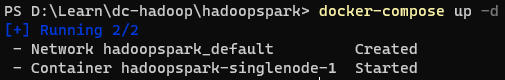


三、主要步骤和实验结果

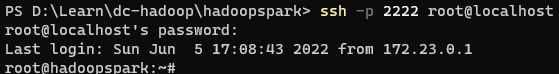
## 题目1

### 主要步骤

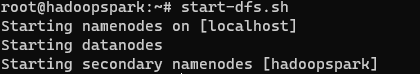
**启动实验环境**

****

通过ssh协议从本机（宿主机）远程登陆到hadoopspark\_singlenode虚拟机：

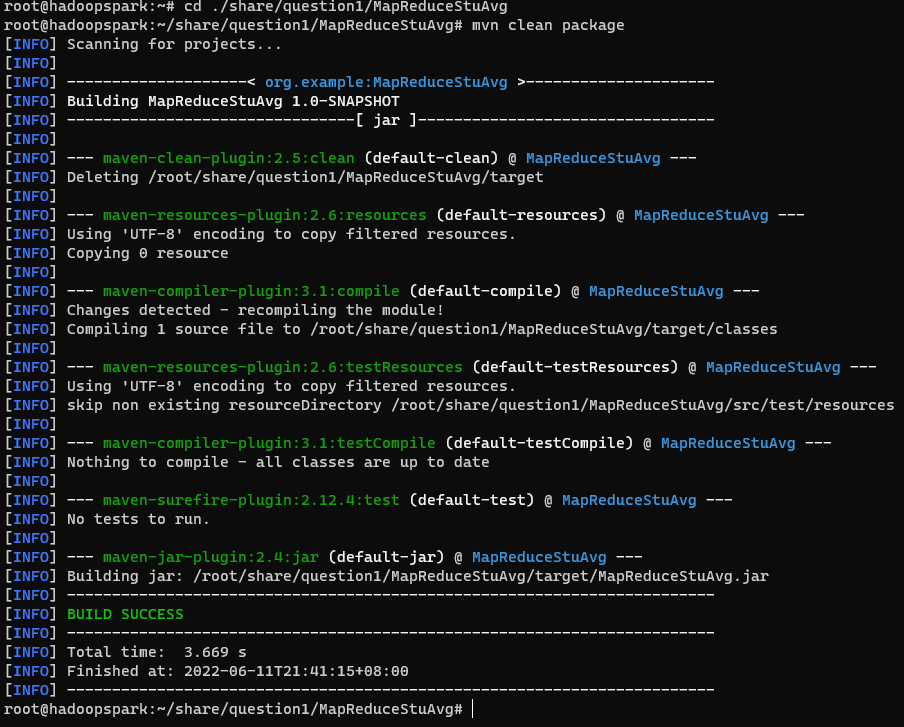


启动HDFS分布式文件系统：

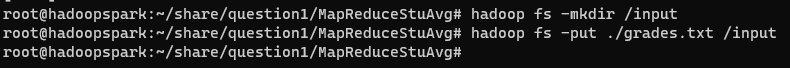


**任务一**

切换到任务一mapreduce程序所在目录，进行编译打包：



将实验所需的数据上传至hdfs文件系统中：

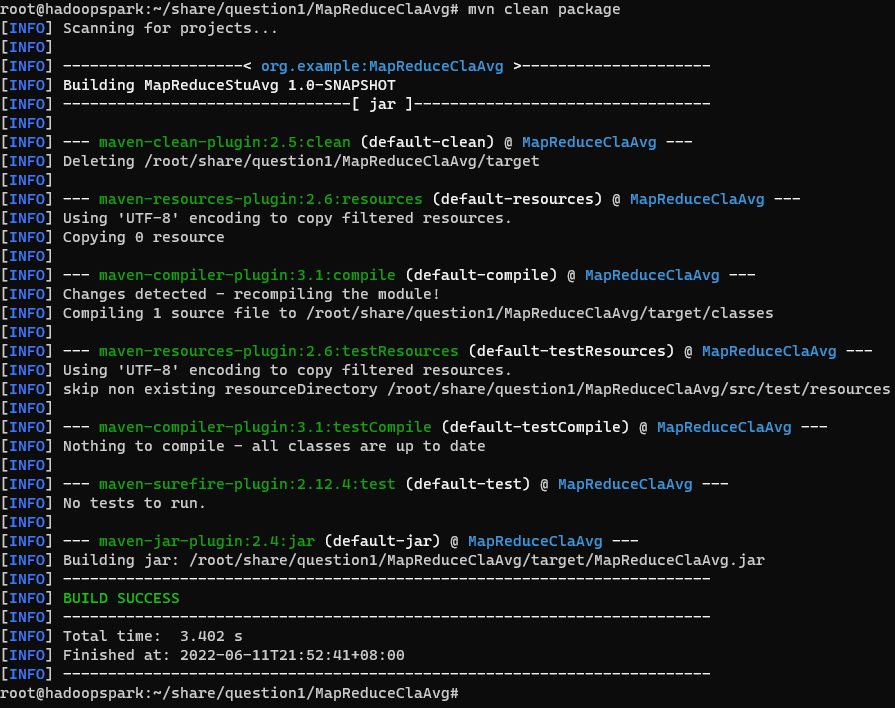


运行mapreduce程序：



**任务二**

切换到任务二mapreduce程序所在目录，进行编译打包：



腾空任务一结果占用的ouput文件夹：



运行mapreduce程序：

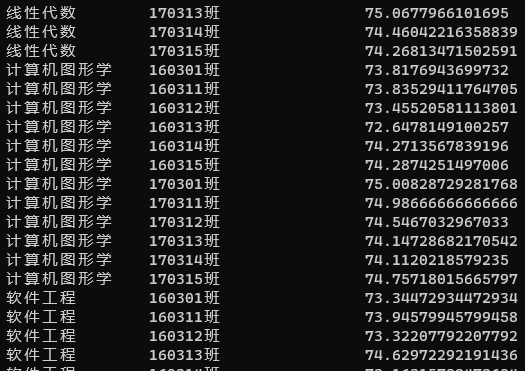


### 实验结果

任务一部分结果展示：



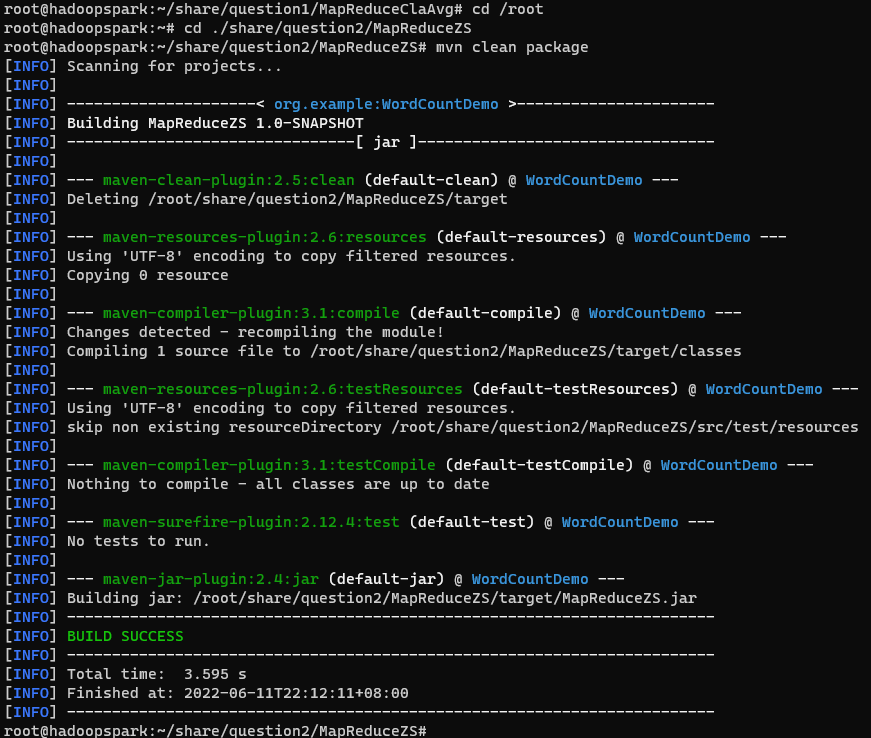
任务二部分结果展示：



## 题目2

### 主要步骤

切换到mapreduce程度所在目录，通过maven编译打包：



上传实验需要的数据集：



腾空实验一结果占用的ouput文件夹：



执行mapreduce程序：



### 结果展示

部分结果展示：

左列是grandchild，右列是grandparent

### 题目3

### 主要步骤

切换到pyspark程序所在目录：

image-20220614123204233

运行pyspark程序：

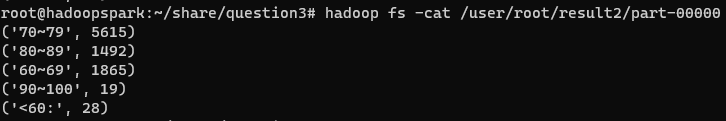
image-20220614123228256

### 结果展示

任务一，部分结果展示：



任务二，结果展示：



四、遇到的问题及解决方法

1. 在实验一完成代码阶段任务后，进行测试时，发现map过程中output：0 records；通过throws抛出错误的方法进行验证，发现`if (splited[3].equals("必修"))`这一if子句中的语句没有被执行，而且mvn package指令warning了部分内容的ACSII编码问题，怀疑是mvn的编码格式问题。

在pom.xml中添加了指定UTF-8格式编码，问题成功解决！

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

2. 在spark实验中上传数据文件与查看结果文件中，遇到了文件不存在的情况，经过查询资料和同学帮助，了解到spark计算模型在这方面与mapreduce存在差别，所以按照前两次是中mapreduce的思考方式去做会出现错误。

3. hadoopsparkv2虚拟机用docker-compose down命令关掉之后保存到HDFS中的实验数据会丢失。要想保留实验数据可以用docker stop命令把虚拟机暂停掉，下次用时用docker start启动。

4. Java语言中equal方法比较字符串，==比较地址。

五、心得体会

本次实验是基于分布式课程知识的综合性实验，以Hadoop分布式文件系统(HDFS)为核心，通过MapReduce与Apache Spark两个计算模型的使用考察，让我对三者有了更加清晰的认识，对于分布式计算的基础知识有了更好的学习和了解；我也学习到了spark与MapReduce的差别，进一步体会到了分布式计算对于大数据领域的应用情况，学习并实践了分布式存储与分布式计算模型的联系与应用。