# 分布式计算 MapReduce 模型

## 实验内容

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, 成绩1 <br>
* （注：<br> 为换行符）  
  班级2, 姓名2, 科目1, 成绩2 <br>  
  班级1, 姓名1, 科目2, 成绩3 <br>  
  ………., ………, ………, ……… <br>  
  编写Hadoop平台上的MapReduce程序，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。

1. **题目2：**

* 输入文件的每一行为具有父子/父母关系的一对人名，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。

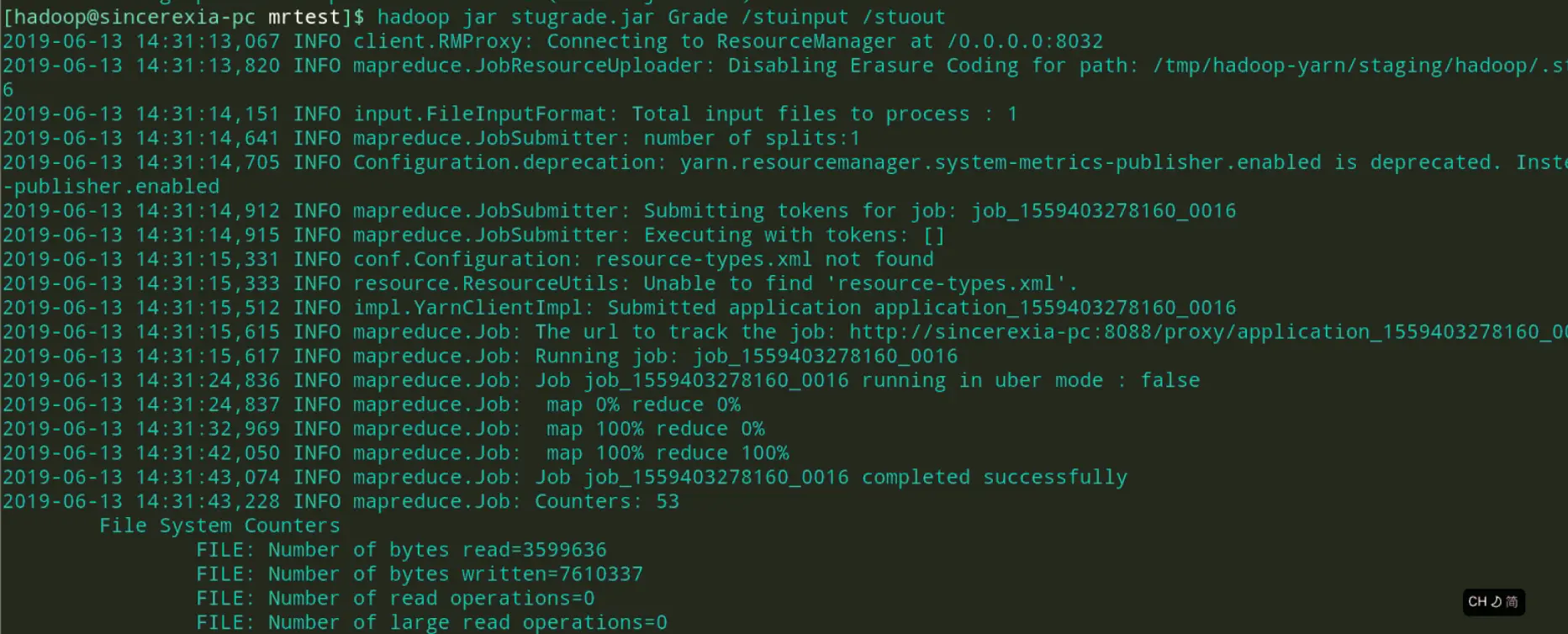
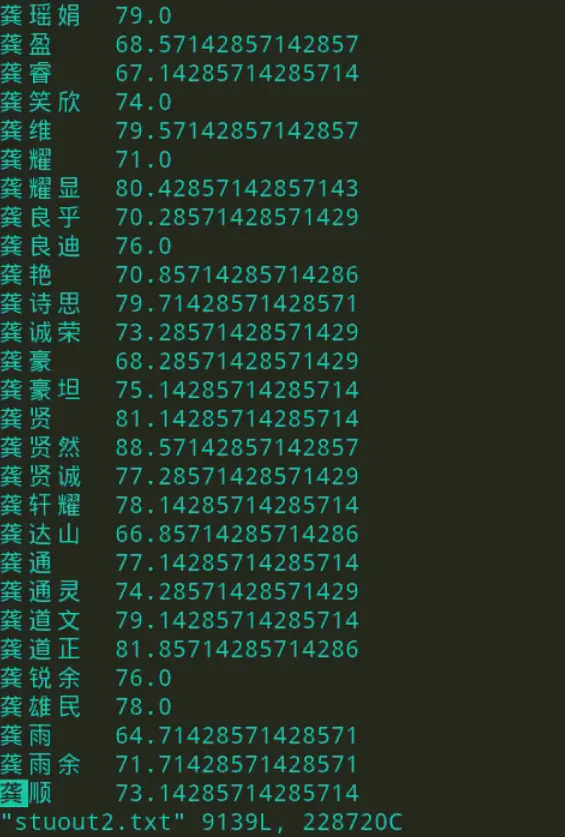
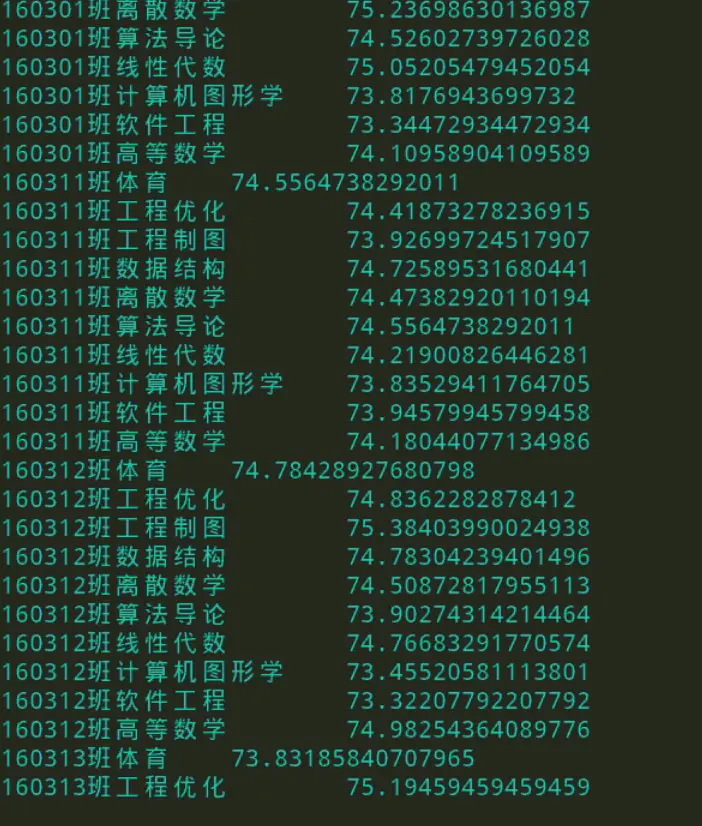
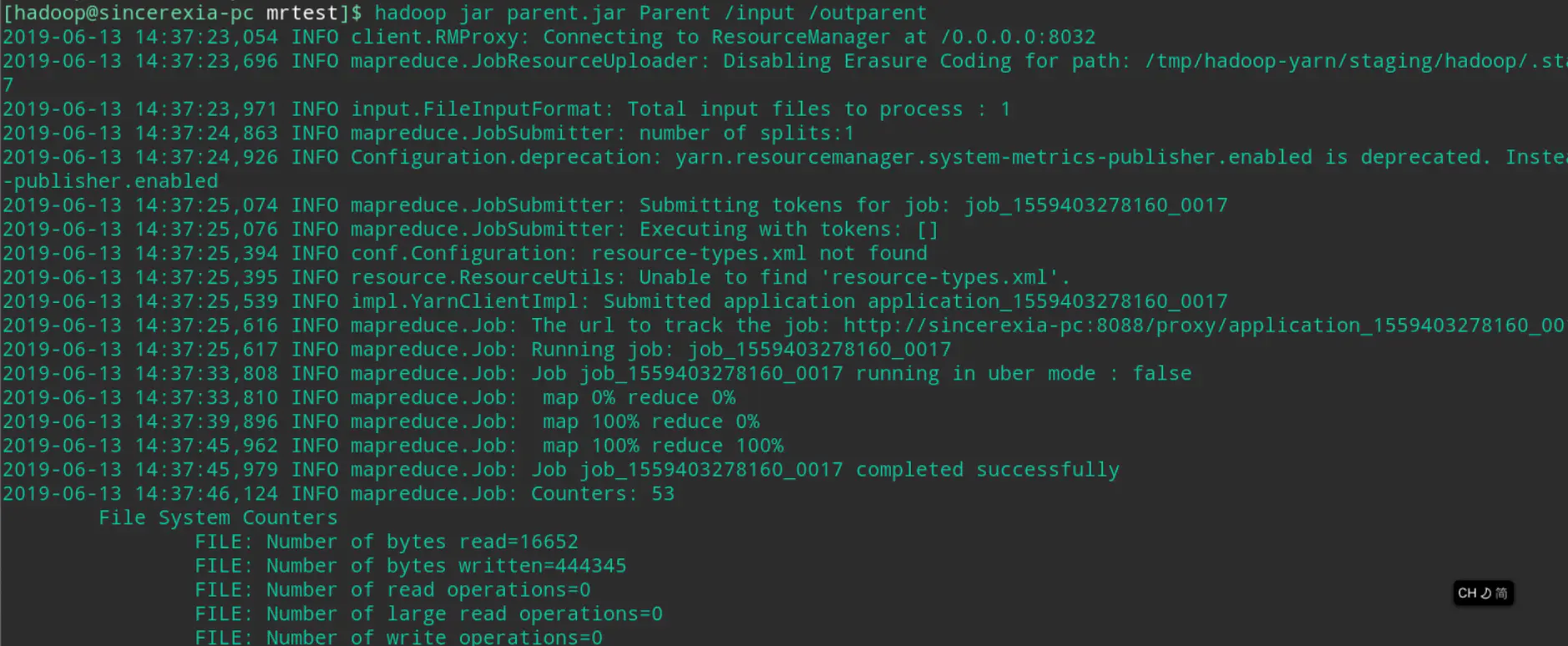
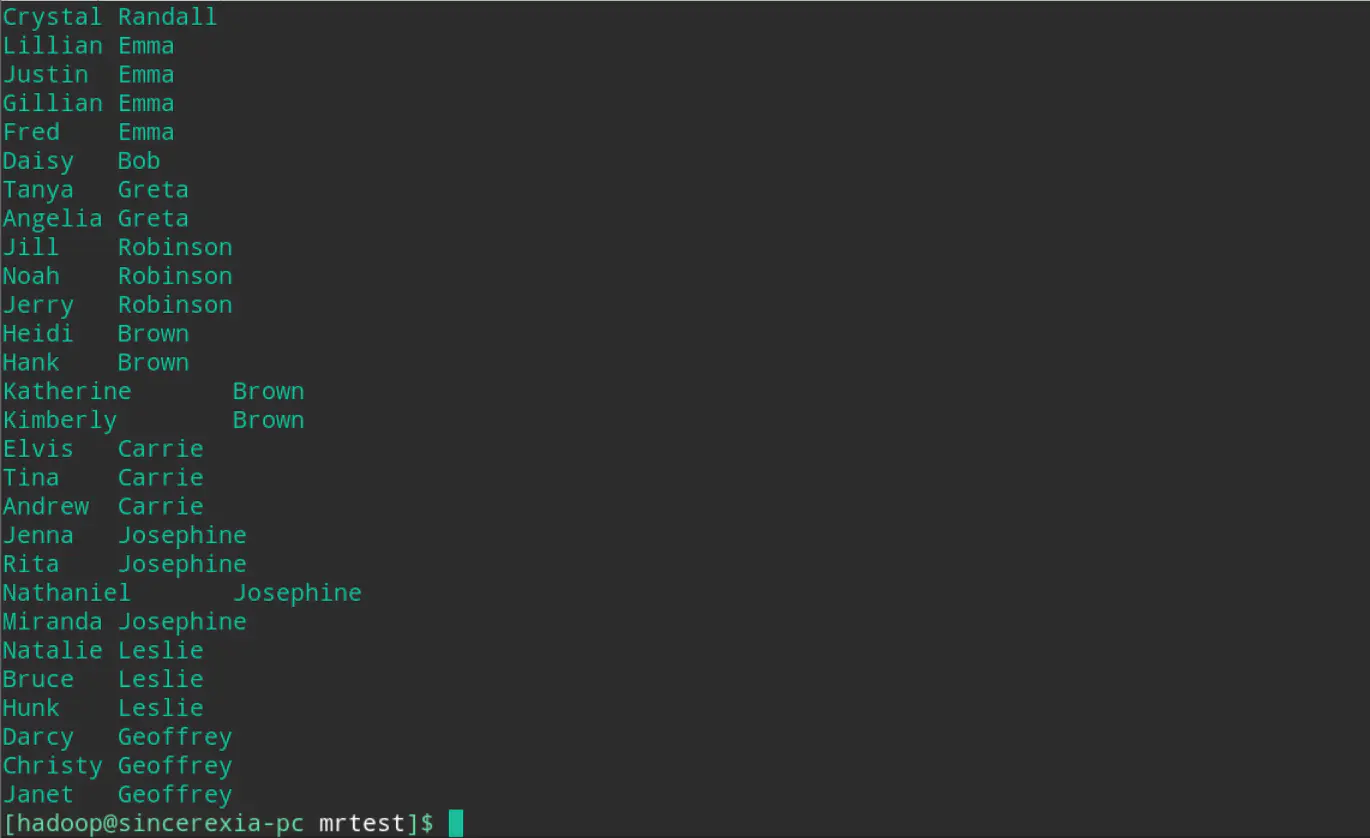
## 设计思想

* 题目一
  1. MR 方式计算学生必修课的平均成绩：
  + 分析给出的文件格式，学生的每科成绩按照 班级2, 姓名2, 科目1, 成绩2 的格式给出，因此，在 map 阶段对文件中的每一行进行处理，按照逗号将字段进行分割。从中提取出有用信息
  + String[] rs = value.toString().split(",");  
    String className = rs[0];  
    String SName = rs[1];  
    String CName = rs[2];  
    boolean isM = rs[3].equals("必修");  
    double G = Double.parseDouble(rs[4]);
  + 为了计算学生必修课的平均成绩，需要对每一行进行判断，看这一科是否为必修。若为必修，则将学生的姓名作为 key 值，学生这一科的分数作为 value ，构建 key, value 键值对
  + if (isM){  
     context.write(new Text(SName), new DoubleWritable(G));  
    }
  + 在 reduce 阶段，键值对按照 key 值聚合，对聚合的key 值求和，获取其平均值，计算出学生必修课的平均成绩
  + public void reduce(Text key, Iterable<DoubleWritable> values,  
     Context context  
     ) throws IOException, InterruptedException {  
     System.out.print("start to reduce!");  
      
     double num = 0;  
     int sum = 0;  
     for (DoubleWritable i : values){  
     sum += i.get();  
     num += 1;  
     }  
     double rs = (double)sum/num;  
     context.write(key, new DoubleWritable(rs));  
     }
  1. 统计每个班的平均成绩。
  + 将每一行数据的班号+课程名作为 key 值，分数作为 value 构建键值对
  + context.write(new Text(className+CName), new DoubleWritable(G));
  + 使用与之前相同的reduce 方法，就可以计算出班级科目的平均值
* 题目二
* 找出所有具有grandchild-grandparent关系的人名组的关键，是寻找一个 中间人，中间人的父亲，与中间人的儿子之间就会构成祖孙关系。
* 因此，对于输入数据中的每一行，生成两个键值对：
  + son, @father
  + father, son
* 这样，输入数据中的每一行中的两个人名，都让其做了中间人，分别找到的父子关系和子父关系。
* public static class PeopleMapper  
   extends Mapper<LongWritable, Text, Text, Text>{  
    
    
   public void map(LongWritable key, Text value, Context context  
   ) throws IOException, InterruptedException {  
   String son = value.toString().split(",")[0];  
    
   String father=value.toString().split(",")[1];  
    
   context.write(new Text(son), new Text("@"+father));  
   context.write(new Text(father), new Text(son));  
   System.out.println("add!"+ son + father);  
    
   }  
   }
* 在 reduce 阶段，hadoop 按照中间人进行聚类，value 值前面加了 @ 符号的代表中间人的父亲，否则为中间人的儿子，遍历一遍 values ，得到中间人父亲和中间人儿子两个列表，两个列表中任意两人名两两配对，构成祖孙关系。
* public void reduce(Text key, Iterable<Text> values,  
   Context context  
   ) throws IOException, InterruptedException {  
   System.out.print("start to reduce!");  
    
   ArrayList<String> sons = new ArrayList<>();  
   ArrayList<String> grandfathers = new ArrayList<>();  
    
   for (Text val1 : values) {  
   String people = val1.toString();  
   if (people.startsWith("@")){  
   grandfathers.add(people.substring(1));  
   }else {  
   sons.add(people);  
   }  
    
   }  
   for (String son:sons){  
   for (String grandfather:grandfathers){  
   context.write(new Text(son), new Text(grandfather));  
   }  
   }  
    
   }

## 遇到的问题及解决方案

* + 使用 MR 模型进行祖孙关系计算时，没有对应的输出
  + 这是因为在 reduce 之前，进行了局部的聚集 combine，因此 reduce 时没有得到需要的包含祖孙信息的键值对，将 combine 过程去除即可。
  + 计算学生的平均成绩和班级的平均成绩时，其 reduce 的过程本质上是相同的，因此，只要在 map 过程中，产生对应不同的键值，就可以在 reduce 过程中同时完成学生平均成绩的计算和班级平均成绩的计算。

## 实验结果

* + 题目1 学生成绩结果
  + 
  + 
  + 按科目统计的平均成绩
  + 
  + 题目2 祖孙关系查询
  + 
  + 

## 心得体会

* 通过本次实验，我完成 hadoop 分布式计算平台的搭建，掌握了 hadoop 平台在 windows 和 linux 上的搭建方法。并且自己编写了求平均成绩和求祖孙关系的 mapreduce 程序，使我对 mapreduce 编程模型有了具体的体会和更深的理解。