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： 1903014

姓　　名： 张泽群

学　　号： 19049100002

一、实验内容

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> （注：
 br> 为换行符）
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> （注：
 br> 为换行符）
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

JAVA 程序详解

对于 map 函数和 reduce 函数实现类：

```
1.    public static class MyMapper extends Mapper<Object, Text, Text, IntWritable>
2.
3.    public void map(Object key, Text value, Context context)
4.                        throws IOException, InterruptedException{
5.
6.    public static class MyReducer extends Reducer<Text, IntWritable, Text, DoubleWritable>
7.
8.      public void reduce(Text key, Iterable<IntWritable> values, Context context)
9.                                  throws IOException, InterruptedException {
```

对于第一行 MyMapper 函数类：

这个继承了 Mapper 父类，其中四个泛型参数分别为 map 阶段的输入 key 的数据类型，map 阶段的输入 value 的数据类型，map 阶段的输出 key 的数据类型，map 阶段的输出 value 的数据类型。

对于第三行 map 的方法：

这里有三个参数，前面两个 Object key, Text value 就是输入的 key 和 value，第三个参数 Context context 这是可以记录输入的 key 和 value;此外 context 还会记录 map 运算的状态。

对于第六行 MyReducerr 函数类：

这个继承了 Reduce 父类，其中四个泛型参数分别为 reduce 阶段的输入 key 的数据类型，reduce 阶段的输入 value 的数据类型，reduce 阶段的输出 key 的数据类型，reduce 阶段的输出 value 的数据类型。

对于第八行 reduce 函数的方法：

reduce 函数的输入也是一个 key/value 的形式，不过它的 value 是一个迭代器的形式 Iterable<IntWritable> values，也就是说 reduce 的输入是一个 key 对应一组的值的 value，reduce 也有 context 和 map 的 context 作用一致。

在主函数 Main 中 ：

```
1.      Configuration conf=new Configuration();
2.      String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
3.
4.      Job job=Job.getInstance(conf, "1.1");
5.      job.setJarByClass(studentGrade .class);
6.      job.setMapperClass(MyMapper.class);
7.      job.setReducerClass(MyReducer.class);
8.
9.      job.setMapOutputKeyClass(Text.class);
10.     job.setMapOutputValueClass(IntWritable.class);
11.     job.setOutputKeyClass(Text.class);
12.     job.setOutputValueClass(DoubleWritable.class);
13.
14.     for (int i = 0; i < otherArgs.length - 1; ++i) {
15.      FileInputFormat.addInputPath(job, new Path(otherArgs[i]));
16.     }
17.     FileOutputFormat.setOutputPath(job, new Path(otherArgs[otherArgs.length - 1]));
18.     System.exit(job.waitForCompletion(true) ? 0 : 1);
```

第一行 Configuration conf = new Configuration()。该类主要是读取 mapreduce 系统配置信息，这些信息包括 hdfs 还有 mapreduce。

第二行 GenericOptionsParser 命令行解析器 是 hadoop 框架中解析命令行参数的基本类。它能够辨别一些标准的命令行参数,能够使应用程序轻易地指定 namenode, jobtracker, 以及其他额外的配置资源。

第四行就是在构建一个 job，在 mapreduce 框架里一个 mapreduce 任务也叫 mapreduce 作业也叫做一个 mapreduce 的 job，而具体的 map 和 reduce 运算就是 task 了，这里我们构建一个 job，构建时候有两个参数，一个是 conf,另一个是这个 job 的名称。

第五行就是装载程序员编写好的计算程序。

第六行和第其行就是装载 map 函数和 reduce 函数实现类了。

第九到十二这个是定义 Map 函数与 Reduce 函数输入输出的 key/value 的类型。

第十五行就是构建输入的数据文件，第十七行是构建输出的数据文件，最后一行如果 job 运行成功了，我们的程序就会正常退出。

题目 1：

1. 计算每个学生必修课的平均成绩

Mapper 和 Reducer 输入输出 key/value 的数据类型如下所示：

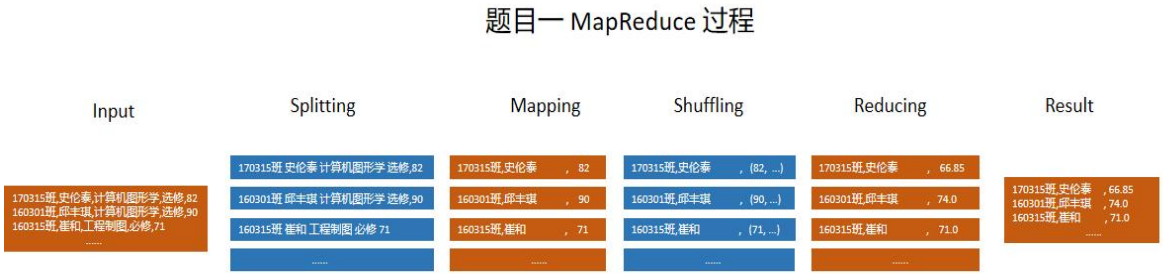
```
Mapper<Object, Text, Text, IntWritable>
Reducer<Text, IntWritable, Text, DoubleWritable>
```

在 **Map 函数**中我们将输入的字符串以“,”为分隔符进行分割，筛选出课程性质为“必修”的记录，并将“班级，姓名”作为 key，将其成绩作为 value。

在**聚集混洗阶段**，将相同 key 的记录的 value 进行聚合，形成 list(V)。

在 **Reduce 阶段**我们将每一个 key 对应 list(V)中的 value 一一求和，并进行计数，当求和完毕后，将 value 的和除以计数，得到同学必修课成绩的平均值。

MapReduce 过程如图所示：



2. 按科目统计每个班的平均成绩

Mapper 和 Reducer 输入输出 key/value 的数据类型如下所示：

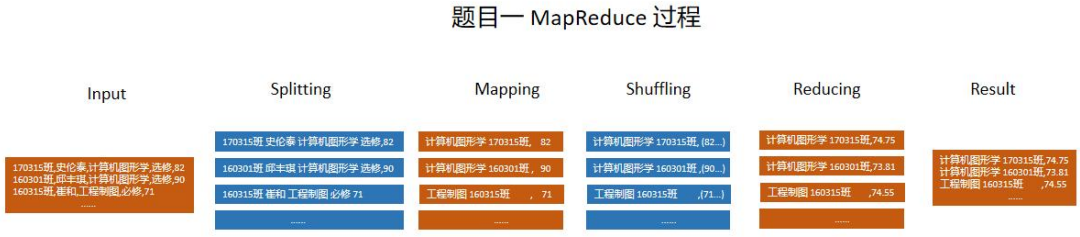
```
Mapper<Object, Text, Text, Text>
Reducer<Text, Text, Text, Text>
```

在 **Map 函数**中我们将输入的字符串以“,”为分隔符进行分割，并将“课程，班级”作为 key，将其成绩作为 value。

在**聚集混洗阶段**，将相同 key 的记录的 value 进行聚合，形成 list(V)。

在 **Reduce 阶段**我们将每一个 key 对应 list(V)中的 value 一一求和，并进行计数，当求和完毕后，将 value 的和除以计数，得到每个班每门课的成绩均值，最后以“平均值+成绩均值”的字符串形式输出。

MapReduce 过程如图所示：



题目 2：

Mapper 和 Reducer 输入输出 key/value 的数据类型如下所示：

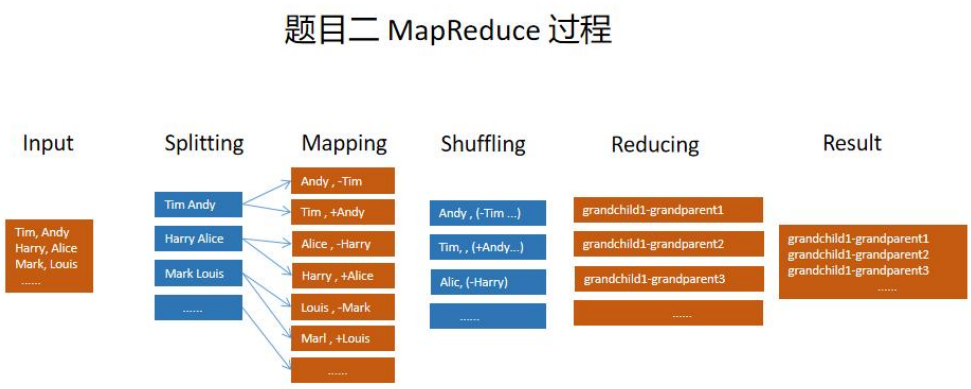
```
Mapper<Object, Text, Text, Text>
Reducer<Text, Text, Text, Text>
```

在 **Map 函数**中我们将输入的字符串以“,”为分隔符进行分割，其中第一个字符为 parent，第二个字符为 child。利用 context.write()函数，分别写入键值对（“child”，“-parent”），（“parent”，“+child”）。

在**聚集混洗阶段**，将具有相同 key 的记录的 value 进行聚合，这里的 key 不仅有某些 child 的 parent，也有某些 parent 的 child，从而形成中 list(V)，既有“-parent”，又有“+child”，这两个 parent 和 child 即为 grandparent-grandchild 的关系。

在 **Reduce 阶段**对于收到的键值对，我们判断其值是以“+”还是“-”开头，若是“-”开头，则将”parent”计入 grandparent 数组，若是“+”开头，则将”child”计入 grandchild 数组，因为此时 child 数组中的所有 child 都是 grandparent 数组中所有 parent 的 grandson，最终将输出 len(grandparent) * len(child)对爷孙关系。

MapReduce 过程如图所示：



题目 3：

其中从平均值到区间内统计的 Mapping 过程运用了一个函数，作用是根据平均值所在区间返回（对应区间，1）的键值对。

二、主要步骤和实验结果

我使用的是单节点的方式，这里开启 HDFS 分布式文件系统的步骤就不多赘述了。所有实验结果都从 hadoop 平台中进行了获取，源程序文件夹中可见。

题目 1：

command.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
hadoop fs -rm -r /input /output
hadoop fs -mkdir /input
hadoop fs -put grades.txt /input

javac -encoding GBK studentGrade.java -cp $(hadoop classpath)
jar cvf studentGrade.jar *.class
hadoop jar studentGrade.jar studentGrade /input /output
hadoop fs -get /output ./
hadoop fs -cat /output/part-r-00000
```

```
170315班,龙锐    71.85714285714286
170315班,龙锐荣  70.57142857142857
170315班,龙顺峰  72.57142857142857
170315班,龚乎    64.57142857142857
170315班,龚哲    77.0
170315班,龚姣小  79.71428571428571
170315班,龚婷杨  80.14285714285714
170315班,龚显    81.28571428571429
170315班,龚泰    62.285714285714285
170315班,龚盈    68.57142857142857
170315班,龚维    79.57142857142857
170315班,龚诗思  79.71428571428571
root@hadoopspark:~/share/1.1#
```

command.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
hadoop fs -rm -r /input /output
hadoop fs -mkdir /input
hadoop fs -put grades.txt /input

javac -encoding GBK classGrade.java -cp $(hadoop classpath)
jar cvf classGrade.jar *.class
hadoop jar classGrade.jar classGrade /input /output
hadoop fs -get /output ./
hadoop fs -cat /output/part-r-00000
```

```
root@hadoopspark:~/share/1.2# hadoop fs -get /output ./
root@hadoopspark:~/share/1.2# hadoop fs -cat /output/part-r-00000
体育 160301班    平均值 = 74.9068493150685
体育 160311班    平均值 = 74.5564738292011
体育 160312班    平均值 = 74.78428927680798
体育 160313班    平均值 = 73.83185840707965
体育 160314班    平均值 = 74.60559796437659
体育 160315班    平均值 = 73.85034965034966
体育 170301班    平均值 = 75.28333333333333
体育 170311班    平均值 = 73.79004037685061
体育 170312班    平均值 = 74.48942172073343
体育 170313班    平均值 = 73.67926988265971
体育 170314班    平均值 = 73.85092348284961
体育 170315班    平均值 = 73.70984455958549
工程优化 160301班        平均值 = 73.66754617414249
工程优化 160311班        平均值 = 74.41873278236915
工程优化 160312班        平均值 = 74.8362282878412
工程优化 160313班        平均值 = 75.19459459459459
工程优化 160314班        平均值 = 73.3598971722365
工程优化 160315班        平均值 = 74.9153005464481
工程优化 170301班        平均值 = 74.67246376811595
```

题目 2：

command.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
hadoop fs -rm -r /input /output                          //删除子目录
hadoop fs -mkdir /input
hadoop fs -put child-parent.txt /input

// del *.class                                           //删除之前的子类
// rd output /s/q                                        //强制删除非空文件夹
javac Relationship.java -cp $(hadoop classpath)
jar cvf relationship.jar *.class
hadoop jar relationship.jar Relationship /input /output  //执行
hadoop fs -get /output ./                                //在实际文件夹中获取output
hadoop fs -cat /output/part-r-00000
```

```
root@hadoopspark:~/share/2# hadoop fs -get /output ./
root@hadoopspark:~/share/2# hadoop fs -cat /output/part-r-00000
Gino    Tasha
Gino    Wendy
Katrina Christina
Katrina Jean
Katrina Olina
Katrina Randy
Katrina Charlotte
Katrina Peggy
Harrison        Cosmo
Harrison        Andrea
Harrison        Enterprise
Harrison        Duke
Christine       Garfield
```

题目 3：

command.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
hadoop fs -rm -r /input /result1 /result2
hadoop fs -mkdir /input
hadoop fs -put grades.txt /input

spark-submit Grades.py

hadoop fs -get /result1 ./
hadoop fs -cat /result1/part-00000

hadoop fs -get /result2 ./
hadoop fs -cat /result2/part-00000
```

```
('160314班 曹英', 87.71428571428571)
('170311班 侯博峰', 69.57142857142857)
('170313班 邹学', 65.42857142857143)
('160311班 田媚', 81.42857142857143)
('170301班 姚军', 71.57142857142857)
('170313班 唐天', 74.85714285714286)
('170315班 罗安昊', 73.57142857142857)
('160315班 郑妯', 76.28571428571429)
('160314班 孔笑和', 76.85714285714286)
('160312班 邵子', 83.0)
root@hadoopspark:~/share/3#
```

```
root@hadoopspark:~/share/3# hadoop fs -get /result1 ./
root@hadoopspark:~/share/3# hadoop fs -get /result2 ./
root@hadoopspark:~/share/3# hadoop fs -cat /result2/part-00000
('70~79:', 5615)
('80~89:', 1492)
('60~69:', 1865)
('90~100:', 19)
('<60:', 28)
root@hadoopspark:~/share/3#
```

四、遇到的问题及解决方法

（1）.刚安装 docker desktop 时，一直打不开该软件。

解决方法：查询网上资料。以管理员模式打开 cmd 命令行，输入 netsh winsock reset 指令。

（2）.Spark 编程不会实现。

解决方法：查询网上资料。https://blog.csdn.net/qq_56477059/article/details/124555431

（3）.Spark 编程时按照老师的方法无法读入输出。

解决方法：给 python 中输入文件的路径前面加上/input，并且在 hadoop 平台将 grades.txt 上传到/input 中。

五、心得体会

在本次实验中，我在听完老师上传的网课后根据样例对于三个题目进行了实现，总体上来说较为顺利。在此过程中，我编程实践并画出了类似 PPT 中的流程图来加深理解，对 MapReduce 模型和基于 RDD-DAG 的模型有了更深层次的理解。