重 庆 交 通 大 学

《大数据平台架构》

综合实验设计报告

班　　级：曙光2101

姓　　名：李幸洋

学　　号：632107060506

同组成员：卢裕中

设计题目：MapReduce成绩分析系统

所属课程：大数据平台架构

实验室(中心)：逸夫楼409/503

指导教师：何伟

完成时间：2023 年 6 月 18 日

评价环节与评分标准

	考核/评价环节	分值（或百分比）	考核/评价细则	学生得分
成绩构成	运行及现场答辩	60%	A、表述清楚，程序演示完全正确，界面美观，能正确回答老师问题 B、表述清楚，按要求完成90%及以上功能，界面尚可，能正确回答老师问题 C、表述较为清楚，按要求完成80%及以上功能，基本能回答老师问题 D、表述基本清楚，按要求完成70%及以上功能，能回答老师多数问题 E、存在30%以上功能未完成或抄袭	
	设计报告	40%	A、报告规范，分析清楚，严格按照要求条目书写，阐述清楚 B、报告规范，分析清楚，个别条目书写不完全符合要求，阐述基本清楚 C、报告基本规范，分析基本清楚，存在20%以内条目书写不完全符合要求 D、报告基本规范，存在30%以内条目书写不完全符合要求 E、报告不规范或存在30%以上条目书写不完全符合要求	

1 项目任务描述

假设学生的成绩放在文件中(数据可能不止存放一个文件)，其格式如下所示：

Id	name	math	English
001	Jerry	81	70
002	Rose	50	90
003	William	90	87
004	Lucy	70	88
005	Steven	62	73
……	…….		

1. 计算每门课程的平均成绩；
2. 计算每门课程学生的平均成绩，并将平均成绩从高到低输出；
3. 求课程的最高成绩；
4. 求课程的最低成绩；
5. 统计课程成绩的分布情况，如：某门课程多少人参加考试，各个分数段的人数等。
6. 查找。输入一个学生的姓名，输出该生姓名以及其参加考试的课程和成绩；
7. 求该成绩表每门课程当中出现了相同分数的分数，出现的次数，以及该相同分数的人数。

2 分析与设计

2.1 系统架构及功能模块的划分

2.1.1 系统架构

该 MapReduce 成绩分析系统架构采用业务系统中常用的 Spring MVC 架构，在此之上，使用了 MapReduce 和 HDFS 文件系统对数据进行分析和储存。

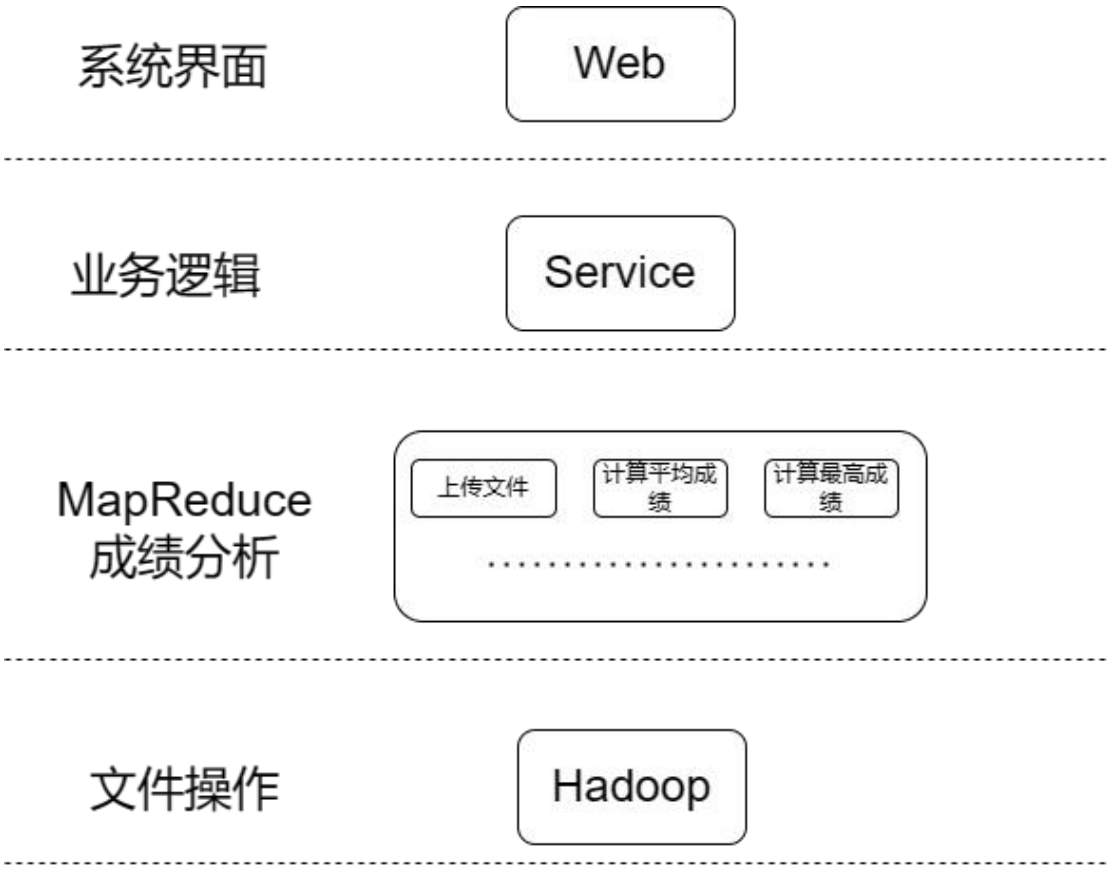

图 2.1.1 系统架构图

2.1.2 功能模块划分

系统的功能模块大致划分为两大部分，一是根据 HDFS 文件系统的文件上传、文件内容查看和文件移动等等；二是利用 MapReduce 对成绩进行分析，如计算平均值、统计成绩分布区间等等。

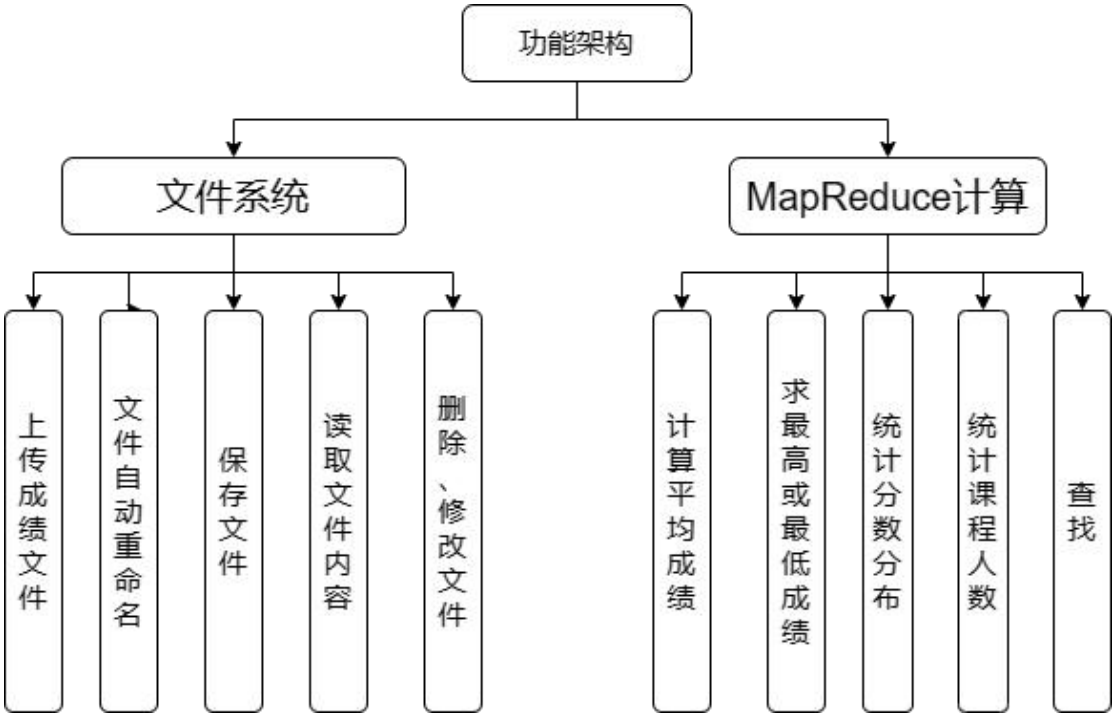

图 2.1.2 系统功能架构图

2.2 模块设计思想

将系统任务分析总结之后可以将其分为几个模块，主要分为视图、业务逻辑、文件操作（数据存储）和 MapReduce 计算四层，具体内容如下：

1. 视图层只负责处理前端的用户请求，将请求处理后发送至业务逻辑中。
2. 业务逻辑根据用户的请求进行相应的操作，并返回数据给视图，视图再返回给用户。
3. 系统将所有的文件操作方式封装为一个工具类，该工具类只专注与文件操作，同样让业务逻辑层更加专注于业务逻辑处理。
4. MapReduce 计算通过业务逻辑进行调用，由业务逻辑定义计算方式

2.3 数据存储设计

该系统采用文件存储的方式，上传的成绩文件经过系统后端上传到服务器中的 HDFS 文件系统；成绩分析的结果会由系统自动保存，同样保存到 HDFS 文件系统中。系统的前端界面可以看到上传的成绩文件和成绩分析的结果记录，存储结果

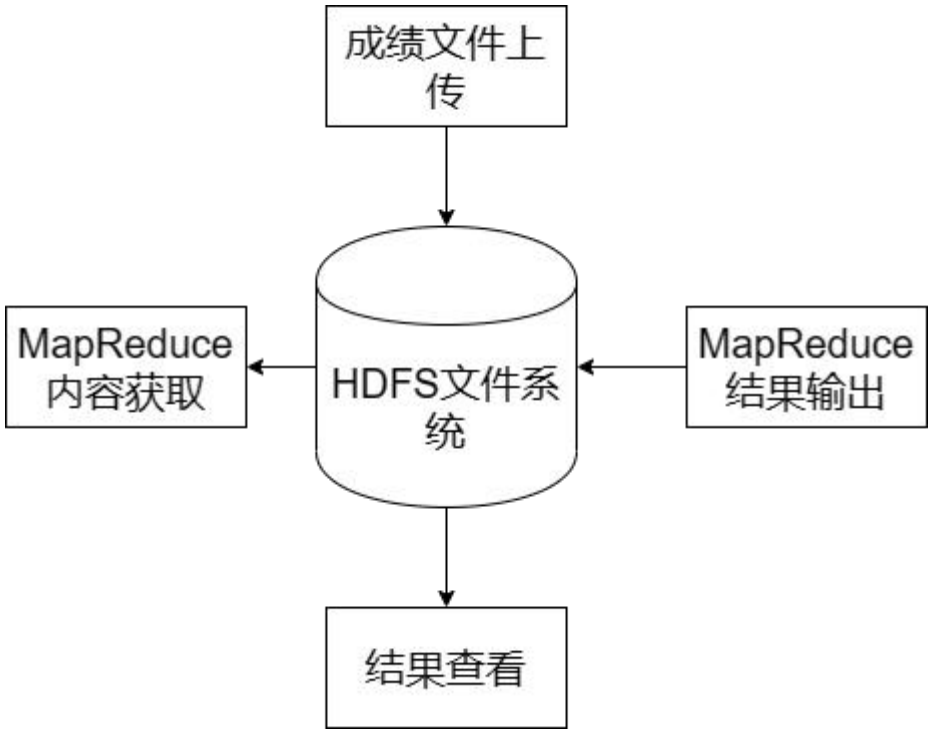

图 2.3 储存结构图

2.4 人机界面设计

系统采用 Web 形式，前后端分离的 B/S 架构。前端包含两个人机界面，分别是主界面如图 2.4.1，成绩分析结果界面如图 2.4.2。

图 2.4.1 系统主界面

图 2.4.2 成绩分析记录界面

2.5 相关类的说明

2.5.1 GFile 数据类

GFile 类主要用于前后端文件传输时所使用，系统规定了文件的存放方式，只需要 GFile 中的文件名即可找到文件，数据成员如图 2.5.1

GFile 类并没有更多的操作方法，只重写了 toString，hashcode 以及 equals 函数，用作数据类型使用。

图 2.5.1 GFile 数据成员

2.5.2 HDFSUtils 类

HDFSUtils 类主要进行 HDFS 文件系统的文件操作，其中数据成员主要是文件路径、系统配置以及文件系统类等静态变量以及常量，主要用作于连接远程服务器的 HDFS 文件系统和文件名进行拼接成完成文件路径，如图 2.5.2.1

图 2.5.2.1 HDFSUtils 数据成员(服务器 IP 马赛克)

ceateFile：创建文件方法，将传入的路径进行重名检查以及更名操作之后创建文件并将创建的文件返回，代码如下图 2.5.2.2

图 2.5.2.2 createFile 创建文件

getFiles: 获取文件方法,将传入文件路径下的所有文件转化为 GFile 文件类型并传出，代码如图 2.5.2.3

图 2.5.2.3 getFiles 获取文件

readContent: 内容与 inputContent 类似

inputContent: 将内容写入文件方法,即将字符串内容 content 写入到文件之中，代码如图 2.5.2.4

图 2.5.2.4 inputContent 写入内容到文件

2.5.3 MyInputFormat 类

MyInputFormat 继承自 FileInputFormat<Text,Text> 配合重写的 RecordReader 以实现 Mapper 读入的内容，如图 2.5.3

图 2.5.3 MyInputFormat 类

2.5.4 WholeFileRecordReader 类

Mapper 过程读入数据的类，继承自 RecordReader<Text,Text>，重写了六个函数以实现整个文件内容一起读入 Maper，其中最主要的是重写 nextKeyValue。

图 2.5.4 实现自定义读入

2.5.5 AverageOutputDesc 类

为实现成绩的排序，实现了一个 WritableComparable 的接口，作为 Mapper 的输出 Key、Reducer 的输入 Key，如图 2.5.9；同时实现了一个继承自 WritableComparator 的一个组比较类，以欺骗 Shuffle 过程中 AverageOutputDesc 的比较只按课程名来进行分组，如图 2.5.5.1、2.5.5.2

图 2.5.5.1　AverageGroupComparator

图 2.5.5.2 重定义的输出类

2.5.6 IntervalOutput 类

为了实现统计每个课程各个分数段的人数，实现了一个 WritableComparable 接口，定义其中的数据成员为课程 course 和区间 type，再通过重写 compareTo 方法，就可以将每个课程中不同的分数段在 shuffle 过程中作为同一类。实现统计每个课程各个分数段的人数。同时还重写了 toString 方法方便后续文件的阅读。

图 2.5.6.1 IntervalOutput 类数据成员

图 2.5.6.2 重写后的 compareTo 函数

图 2.5.6.3 重写后的 toString 函数

2.5.7 SameOutput 类

SameOutput 类的设计思路与 IntervalOutput 类一致，为了实现统计每个课程出现的各个分数的人数，实现了一个 WritableComparable 接口，定义其中的数据成员为课程 course 和分数 score，再通过重写 compareTo 方法，就可以将每个课程中相同的分数段在 shuffle 过程中作为同一类。实现统计每个课程出现的各个分数的人数。

图 2.5.7.1 SameOutput 类数据成员

图 2.5.7.2 重写的 compareTo 函数

3 系统实现

3.1 AverageMR 类

代码实现：

```
    public static class avgMapper extends Mapper<Object, Text,
AverageOutputDesc, DoubleWritable> {
        private final Text subj = new Text();
        private final DoubleWritable score = new
DoubleWritable();
        @Override
        protected void map(Object key, Text value, Mapper<Object,
Text, AverageOutputDesc, DoubleWritable>.Context context)
throws IOException, InterruptedException {
            String[] split = value.toString().split("\n");
            String[] headers = split[0].split("\\s+");

            for (int i = 1; i < split.length; i ++) {
                String line = split[i];
                String[] cells = line.split("\\s+");
                for (int j = 2; j < cells.length; j++) {
                    subj.set(headers[j]);
                    score.set(Double.parseDouble(cells[j]));
                    context.write(new AverageOutputDesc(subj,
score), score);
                }
            }
        }
    }
```

```
    public static class avgReducer extends
Reducer<AverageOutputDesc, DoubleWritable, AverageOutputDesc,
NullWritable> {
        @Override
        protected void reduce(AverageOutputDesc key,
Iterable<DoubleWritable> values, Reducer<AverageOutputDesc,
DoubleWritable, AverageOutputDesc, NullWritable>.Context
context) throws IOException, InterruptedException {
            int n = 0;
            double sum = 0;

            for (DoubleWritable value : values) {
                sum += value.get();
                n ++;
            }
            if(n != 0) sum /= n;
            key.setScore(new DoubleWritable(sum));
            context.write(key, NullWritable.get());
        }
    }
```

3.2 FindStudentMR 类

代码实现：

```
    public static class FindMapper extends Mapper<Object, Text,
Text, Text> {
        private final Text name = new Text();
        private final Text courseAndScore = new Text();
        @Override
        protected void map(Object key, Text value, Mapper<Object,
Text, Text, Text>.Context context) throws IOException,
InterruptedException {
```

```
            String[] split = value.toString().split("\n");
            String[] headers = split[0].split("\\s+");

            for (int i = 1; i < split.length; i ++) {
                String line = split[i];
                String[] cells = line.split("\\s+");
                for (int j = 2; j < cells.length; j++) {

        protected void reduce(Text key, Iterable<Text> values,
Reducer<Text, Text, Text, NullWritable>.Context context) throws
IOException, InterruptedException {
            if (! student.equals(key.toString())) {
                return;
            }
            context.write(new Text("学生姓名: " + student),
NullWritable.get());
            for (Text value : values) {
                output.set(value);
                context.write(output, NullWritable.get());
            }
        }
    }
```

3.3 MaxMR 类

代码实现：

```
    public static class MaxMapper extends Mapper<Object, Text,
Text, DoubleWritable> {
        private Text course = new Text();
        private DoubleWritable max = new
DoubleWritable(Double.MIN_VALUE);
        @Override
```

```
        protected void map(Object key, Text value, Mapper<Object,
Text, Text, DoubleWritable>.Context context) throws IOException,
InterruptedException {
            String[] split = value.toString().split("\n");
            String[] headers = split[0].split("\\s+");

            for (int i = 1; i < split.length; i ++) {
                String line = split[i];
                String[] cells = line.split("\\s+");
                for (int j = 2; j < cells.length; j++) {
                    course.set(headers[j]);
                    max.set(Double.parseDouble(cells[j]));
                    context.write(course,max);
                }
            }
        }
    }

    public static class MaxReduce extends Reducer<Text,
DoubleWritable, Text, DoubleWritable> {
        private DoubleWritable result = new DoubleWritable();

        @Override
        protected void reduce(Text key,
Iterable<DoubleWritable> values, Reducer<Text, DoubleWritable,
Text, DoubleWritable>.Context context) throws IOException,
InterruptedException {
            double max = Double.MIN_VALUE;
            for(DoubleWritable value : values) {
                max = Math.max(value.get(), max);
            }
```

```
            result.set(max);
            Text text = new Text(key.toString() + "  最高分:");
            context.write(text,result);
        }
    }
```

3.4 MinMR 类

代码实现：

```
    public static class MinMapper extends Mapper<Object, Text,
Text, DoubleWritable> {
        private Text course = new Text();
        private DoubleWritable min = new DoubleWritable();
        @Override
        protected void map(Object key, Text value, Mapper<Object,
Text, Text, DoubleWritable>.Context context) throws IOException,
InterruptedException {
            String[] split = value.toString().split("\n");
            String[] headers = split[0].split("\\s+");

            for (int i = 1; i < split.length; i ++) {
                String line = split[i];
                String[] cells = line.split("\\s+");
                for (int j = 2; j < cells.length; j++) {
                    course.set(headers[j]);
                    min.set(Double.parseDouble(cells[j]));
                    context.write(course,min);
                }
            }
        }
    }
```

```
    public static class MinReduce extends Reducer<Text,
DoubleWritable, Text, DoubleWritable> {
        private DoubleWritable result = new DoubleWritable();
        @Override
        protected void reduce(Text key,
Iterable<DoubleWritable> values, Reducer<Text, DoubleWritable,
Text, DoubleWritable>.Context context) throws IOException,
InterruptedException {
            double min = Double.MAX_VALUE;
            for(DoubleWritable value : values) {
                min = Math.min(value.get(), min);
            }
            result.set(min);
            Text text = new Text(key.toString() + "  最低分:");
            context.write(text,result);
        }
    }
```

3.5 CountPeopleMR 类

代码实现：

```
    public static class CountPeopleMapper extends Mapper<Object,
Text, Text, IntWritable> {
        private Text course = new Text();
        private final static IntWritable one = new
IntWritable(1);
        @Override
        protected void map(Object key, Text value, Mapper<Object,
Text, Text, IntWritable>.Context context) throws IOException,
InterruptedException {
            String[] split = value.toString().split("\n");
            String[] headers = split[0].split("\\s+");
```

```
            for (int i = 1; i < split.length; i ++) {
                String line = split[i];
                String[] cells = line.split("\\s+");
                for (int j = 2; j < cells.length; j++) {
                    course.set(headers[j]);
                    context.write(course,one);
                }
            }
        }
    }

    public static class CountPeopleReduce extends Reducer<Text,
IntWritable, Text, IntWritable> {
        private IntWritable result = new IntWritable();
        @Override
        protected void reduce(Text key, Iterable<IntWritable>
values, Reducer<Text, IntWritable, Text, IntWritable>.Context
context) throws IOException, InterruptedException {
            int sum  = 0;
            for(IntWritable value : values) {
                sum += value.get();
            }
            result.set(sum);
            Text text = new Text(key.toString() + "  选修人数:");
            context.write(text,result);
        }
    }
```

3.6 SamePeopleMR 类

代码实现：

```
    public static class SamPeoMapper extends Mapper<Object, Text,
SameOutput, IntWritable> {
        private Text course = new Text();
        private final static IntWritable one = new
IntWritable(1);
        @Override
        protected void map(Object key, Text value, Mapper<Object,
Text, SameOutput, IntWritable>.Context context) throws
IOException, InterruptedException {
            String[] split = value.toString().split("\n");
            String[] headers = split[0].split("\\s+");

            for (int i = 1; i < split.length; i ++) {
                String line = split[i];
                String[] cells = line.split("\\s+");
                for (int j = 2; j < cells.length; j++) {
                    course.set(headers[j]);
                    DoubleWritable score = new
DoubleWritable(Double.parseDouble(cells[j]));
                    context.write(new
SameOutput(course,score),one);
                }
            }
        }
    }

    public static class SamPeoReducer extends
Reducer<SameOutput, IntWritable, SameOutput, IntWritable> {
        private IntWritable result = new IntWritable();
        @Override
```

```
        protected void reduce(SameOutput key,
Iterable<IntWritable> values, Reducer<SameOutput, IntWritable,
SameOutput, IntWritable>.Context context) throws IOException,
InterruptedException {
            int sum = 0;
            for (IntWritable value : values) {
                sum += value.get();
            }
            result.set(sum);
            context.write(key, result);
        }
    }
```

3.7 StatisticInterval 类

代码实现：

```
    public static class StaIntMapper extends Mapper<Object, Text,
IntervalOutput, IntWritable> {
        private Text course = new Text();
        private final static IntWritable one = new
IntWritable(1);
        @Override
        protected void map(Object key, Text value, Mapper<Object,
Text, IntervalOutput, IntWritable>.Context context) throws
IOException, InterruptedException {
            String[] split = value.toString().split("\n");
            String[] headers = split[0].split("\\s+");

            for (int i = 1; i < split.length; i ++) {
                String line = split[i];
                String[] cells = line.split("\\s+");
                for (int j = 2; j < cells.length; j++) {
```

```
                course.set(headers[j]);
                Double score = Double.parseDouble(cells[j]);
                IntWritable type = new IntWritable();
                if(score < 60.0) {
                    type.set(1);
                } else if (score >= 60.0 && score < 70.0) {
                    type.set(2);
                } else if (score >= 70.0 && score < 80.0) {
                    type.set(3);
                } else if (score >= 80.0 && score < 90.0) {
                    type.set(4);
                } else {
                    type.set(5);
                }
                context.write(new
IntervalOutput(course,type), one);
            }
        }
    }
}

    public static class StaIntReducer extends
Reducer<IntervalOutput, IntWritable, IntervalOutput,
IntWritable> {
        private IntWritable result = new IntWritable();
        @Override
        protected void reduce(IntervalOutput key,
Iterable<IntWritable> values, Reducer<IntervalOutput,
IntWritable, IntervalOutput, IntWritable>.Context context)
throws IOException, InterruptedException {
            int sum = 0;
```

```
        for (IntWritable value : values) {
            sum += value.get();
        }
        result.set(sum);
        context.write(key, result);
    }
}
```

4 系统测试

系统	MapReduce 成绩分析系统		程序版本		V1.0	
预置条件	两个成绩文件、Hadoop 集群正常运行					
用例编号	目的	操作步骤	输入数据	期望结果	行结果	缺陷报告号
001	测试正常上传	选择两个成绩文件上传	两个成绩文件	正常上传	常	无
002	测试空上传	不选择文件直接上传	无	能提示无文件上传，反馈错误	确	无
003	测试计算平均成绩	选择两个成绩文件，计算平均成绩	两个成绩文件	能正常计算，反馈结果，并保存结果到记录中	确	无
004	测试空计算平均成绩	不选择文件直接计算	无	能反馈警告，并截断操作	确	无
004	测试查找学生	选择两个成绩文件，输入学生姓名查找学生	两个成绩文件以及学生名	能查找并反馈结果	确	无
005	测试不选择成绩文件，查找学生	不选择文件直接查找	无	能反馈警告	确	无
005	测试不输入名字查	不输入学生名字进行查找	无	能反馈警告	确	无

	找学生					
006	测试未选择文件计算成绩最大值	不选择任何成绩文件并选择计算最大值	无	反馈警告	确	无
007	测试选择文件计算成绩最大值	选择任意成绩文件并选择计算最大值	任意成绩文件	正常进行计算，并反馈计算结果，保存到分析记录中	确	无
008	测试未选择文件计算成绩最小值	不选择任何成绩文件并选择计算最小值	无	反馈警告	确	无
009	测试选择文件计算成绩最小值	选择任意成绩文件并选择计算最小值	任意成绩文件	正常进行计算，并反馈计算结果，保存到分析记录中	确	无
010	测试未选择文件进行统计选修人数	不选择任何成绩文件并选择统计各课程选修人数	无	反馈警告	确	无
011	测试选择任意文件进行统计选修人数	选择任意成绩文件并选择统计各课程选修人数	任意成绩文件	正常进行计算，并反馈计算结果，保存到分析记录中	确	无
012	测试未选择文件进	不选择任何成绩文件并选择	无	反馈警告	确	无

	行统计各个课程的出现成绩人数	统计各个课程出现的成绩的人数				
013	测试选择任意文件进行统计各个课程的出现成绩人数	选择任意成绩文件并选择统计各个课程出现的成绩的人数	任意成绩文件	正常进行计算，并反馈计算结果，保存到分析记录中	确	无
014	测试未选择文件进行统计各个课程的成绩区间人数	不选择任何成绩文件并选择统计各个课程成绩区间的人数	无	反馈警告	确	无
015	测试选择文件进行统计各个课程的成绩区间人数	选择任意成绩文件并选择统计各个课程成绩区间的人数	任意成绩文件	正常进行计算，并反馈计算结果，保存到分析记录中	确	无
016	测试能否查看历史分析记录	点击查看历史结果	无	显示所有历史分析结果条目	确	无
017	测试能否查看历史分析记录中的分析情况	点击历史记录中的任意一条历史记录的详情	无	显示该次历史分析的详细结果	确	无

5 用户使用说明

5.1 简介

MapReduce 成绩分析系统是一个用于处理学生成绩数据的工具。它利用 MapReduce 编程模型，能够高效地处理大规模的成绩数据集，并提供丰富的分析功能和可视化结果。

5.2 用户界面导览

用户在主界面可以点击上传成绩，拖拽或选择本地的成绩文件上传至 HDFS 文件系统，在上传成功后会在下方的表格中显示已在 HDFS 文件系统中。若要进行成绩分析，则选中需要进行分析的成绩文件，同时选择需要的成绩分析方式，并按要求提供相应的参数，最后点击计算即可得到结果。

图 5.2.1 系统主界面

用户在成绩分析记录界面可以查看过往的成绩分析记录，若想查看详情，则可以点击右侧详情按钮进行查看。

图 5.2.2 成绩分析记录界面

5.3 主要功能

5.3.1 文件上传

点击上传文件按钮

进入文件上传界面

选择要上传的学生成绩数据文件

点击“上传”按钮，系统将开始处理上传的数据文件

5.3.2 成绩分析

进入主界面

选择需要进行成绩分析的文件

选择分析类型，例如计算平均值、查找学生成绩等分析

根据选择的分析类型，提供相应的参数，例如选择查找学生成绩时，需要提供学生姓名。

点击"计算"按钮，系统将启动 MapReduce 任务进行成绩分析

分析结束后会显示分析结果

5.3.3 可视化查看结果

进入成绩分析历史记录可视化界面

选择需要查看的历史分析记录

点击“详情”按钮查看历史分析的详细信息

5.3 数据文件格式要求

成绩数据文件的格式如下图所示：

Id	name	math	English	…….
001	Jerry	81	70	…….
002	Rose	50	90	…….
003	William	90	87	…….
004	Lucy	70	88	…….
005	Steven	62	73	…….
……	…….			

6 总结

这次实验成功地使用了 Hadoop、MapReduce 实现了一个在线的成绩分析系统，在核心功能上使用的是 MapReduce 计算，成绩文件存储等方面是通过 Hadoop 的 HDFS 来实现。此次实验的 hadoop 平台是搭建在阿里云的服务器上的，是伪分布式的；在运行环境上，选择的是 windows 系统，windows 系统来进行远程链接操作。

这次实验最为麻烦的地方就是环境的配置，可能因为 hadoop 是一个开源的平台，不同版本、不同机器上都又有可能出现一些问题；而在 windows 上进行读取远程 hdfs 文件数据和信息又会出一些问题，原因是一些内外网通信问题。至于在后续的实际开发时，但是没有什么太大的问题。

在这次实验中，通过自己从 0 到 1 的环境配置、程序设计、代码实现的过程，巩固熟悉了一些原本已知的知识、还学习到了新的知识，比如如何自定义 mapper 的读入格式、reduce 输出格式，namenode、datanode 的通信等等知识，但我认为最核心的却是熟悉了 mapreduce 的编程思想。编程思想、程序设计思想只通过知识灌输是很难学好的，只有在实打实地解决实际问题，才可以真正地体会到。

总结一下这次实验带给我的收获是很多的。总地来所，一是解决问题的思路被拓宽了、二是学习巩固了大数据平台的相关知识。从这次实验的收获来讲，理论加实践的效果是非凡的，但是还是需要学习更多的知识、付出更多地实践才能真正地深入这一领域。

7 参考文献

前端组件库：https://www.antdv.com/components/overview

前端框架：https://cn.vuejs.org/

数据请求工具：https://www.axios-http.cn/

Hadoop 官网：https://hadoop.apache.org/

Hadoop Java Api：https://hadoop.apache.org/docs/current/api/