

金融大数据第二次实验报告

221275025 张可

可以用[在线文档](#)查看

1.任务一

1.1 任务描述

根据 `user_balance_table` 表中的数据，编写 MapReduce 程序，统计所有用户每日的资金流入与流出情况。资金流入意味着申购行为，资金流出为赎回行为。缺失值视为 0。

1.2 设计思路

本实验的目标是根据 `user_balance_table` 表中的数据，通过 MapReduce 程序统计每个用户的每日资金流入与流出情况。设计思路如下：

1.2.1 Map 类设计

解析输入数据，将每行记录按逗号分隔为字段数组。判断数据是否符合最低字段数量要求（字段个数 ≥ 10 ）。过滤掉表头行，通过 `report_date` 字段来识别。

```
Plain Text
if (!date.equals("report_date"))
    dateKey.set(date);
```

提取 `report_date` 作为 key，并从 `total_purchase_amt`（字段索引 4）和 `total_redeem_amt`（字段索引 8）中提取资金流入和流出数据。

```
JavaScript
// 处理资金流入量
String purchaseStr = fields[4];
int purchaseAmt = 0;
if (!purchaseStr.isEmpty() && isNumeric(purchaseStr)) {
```

```
        purchaseAmt = Integer.parseInt(purchaseStr);
    }
```

将资金流入量和流出量分别输出，流入量以正值表示，流出量以负值表示。对缺失值进行处理，视为零交易。

```
JavaScript
// 输出流入和流出
context.write(dateKey, new IntWritable(purchaseAmt)); // 流入
context.write(dateKey, new IntWritable(-redeemAmt)); // 流出
```

1.2.2 Reduce 类 设计

接收来自 Mapper 的键值对，根据 `report_date` 进行汇总。将正值视为资金流入，累加得到总的资金流入量。将负值视为资金流出，通过累加负值的绝对值得到总的资金流出量。输出格式为 `<日期> TAB <资金流入量>,<资金流出量>`，方便后续的数据分析与展示。

```
JavaScript
for (IntWritable val : values) {
    if (val.get() > 0) {
        inflow += val.get(); // 资金流入
    } else {
        outflow += -val.get(); // 资金流出
    }
}
```

1.2.3 数据校验与处理

针对输入数据的空值处理：若 `total_purchase_amt` 或 `total_redeem_amt` 为空，将其视为零。使用正则表达式判断字符串是否为数字，确保数据的有效性。

```
JavaScript
private boolean isNumeric(String str) {
    return str != null && str.matches("\\d+");
}
```

1.3 遇到的问题及解决方案

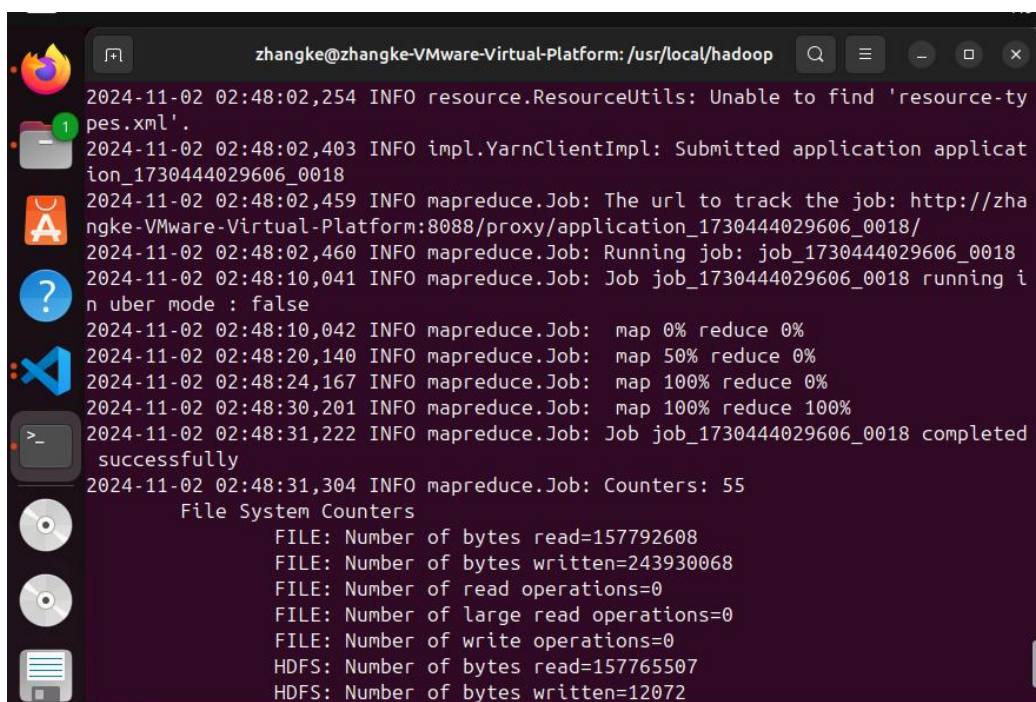
运行时报出以下错误：

Bash

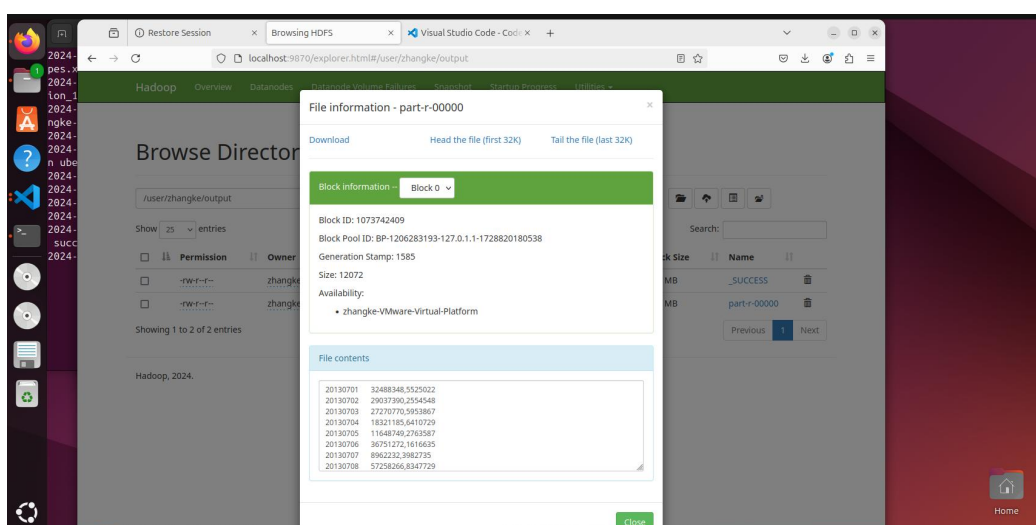
```
Error: java.lang.NumberFormatException: For input string:
"total_purchase_amt"
```

经检查发现是由于表头不是数字，无法转换为整数导致的，所以后续在代码中加上了判断取到的东西是否符合预期的判断用以筛选。

1.4 运行结果部分截图



```
zhangke@zhangke-VMware-Virtual-Platform: /usr/local/hadoop
2024-11-02 02:48:02,254 INFO resource.ResourceUtils: Unable to find 'resource-ty
pes.xml'.
2024-11-02 02:48:02,403 INFO impl.YarnClientImpl: Submitted application applicat
ion_1730444029606_0018
2024-11-02 02:48:02,459 INFO mapreduce.Job: The url to track the job: http://zha
ngke-VMware-Virtual-Platform:8088/proxy/application_1730444029606_0018/
2024-11-02 02:48:02,460 INFO mapreduce.Job: Running job: job_1730444029606_0018
2024-11-02 02:48:10,041 INFO mapreduce.Job: Job job_1730444029606_0018 running i
n uber mode : false
2024-11-02 02:48:10,042 INFO mapreduce.Job: map 0% reduce 0%
2024-11-02 02:48:20,140 INFO mapreduce.Job: map 50% reduce 0%
2024-11-02 02:48:24,167 INFO mapreduce.Job: map 100% reduce 0%
2024-11-02 02:48:30,201 INFO mapreduce.Job: map 100% reduce 100%
2024-11-02 02:48:31,222 INFO mapreduce.Job: Job job_1730444029606_0018 completed
successfully
2024-11-02 02:48:31,304 INFO mapreduce.Job: Counters: 55
File System Counters
FILE: Number of bytes read=157792608
FILE: Number of bytes written=243930068
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=157765507
HDFS: Number of bytes written=12072
```



2.任务二

2.1 任务描述

该任务要求编写一个 MapReduce 程序来统计一周七天中每天的平均资金流入与流出情况，并按照资金流入量从大到小进行排序。输入数据包含日期和金额（可能包括流入和流出），输出格式为“<weekday> TAB <资金流入量>,<资金流出量>”。

2.2 设计思路

2.2.1 Map 类设计

首先读取输入数据，因为任务一的输出是以 TAB 作为分隔的，所以在 Map 中转义为“\t”。

```
JavaScript
String[] fields = value.toString().split("\t");
```

然后解析日期并转换为对应的星期几。

```
JavaScript
Calendar calendar = Calendar.getInstance();
calendar.setTime(date);
int dayOfWeek = calendar.get(Calendar.DAY_OF_WEEK); // 返回 1-7 (1 = 星期日)
```

将金额分割为流入和流出两部分。对于每个日期（转换为星期几后），输出两个键值对。到了 reduce 中区分这两个输入：用正数表示资金流入，负数表示资金流出（这个和任务一的处理一样）。

2.2.2 Reduce 类设计

接收 Mapper 输出的键值对，其中键是星期几，值是金额（正数表示流入，负数表示流出）。对每个星期几，分别累加流入和流出的总金额，同时分别用 sum1 和 sum2 表示在星期几资金流入和流出的总天数。

```
JavaScript
for (IntWritable val : values) {
    if (val.get() > 0) {
        inflow += val.get(); // 资金流入
        sum1++;
    } else {
        outflow += -val.get(); // 资金流出
        sum2++;
    }
}
```

```
}  
}  
inflow = inflow/sum1;  
outflow = outflow/sum2;
```

最后就是根据资金输入量进行排序，这里采用的是 Treemap 中的降序函数。

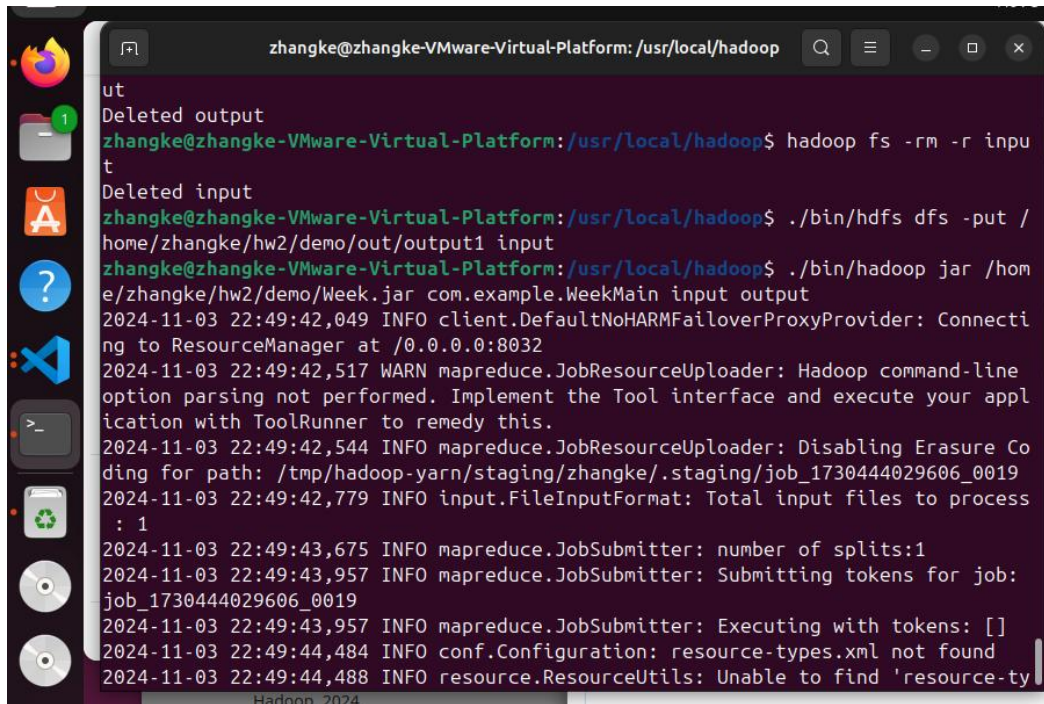
JavaScript

```
for (Map.Entry<Long, String> entry : flow.entrySet()) {  
    context.write(new Text(entry.getValue()), new Text(""));  
}
```

2.3 出现的错误及解决方案

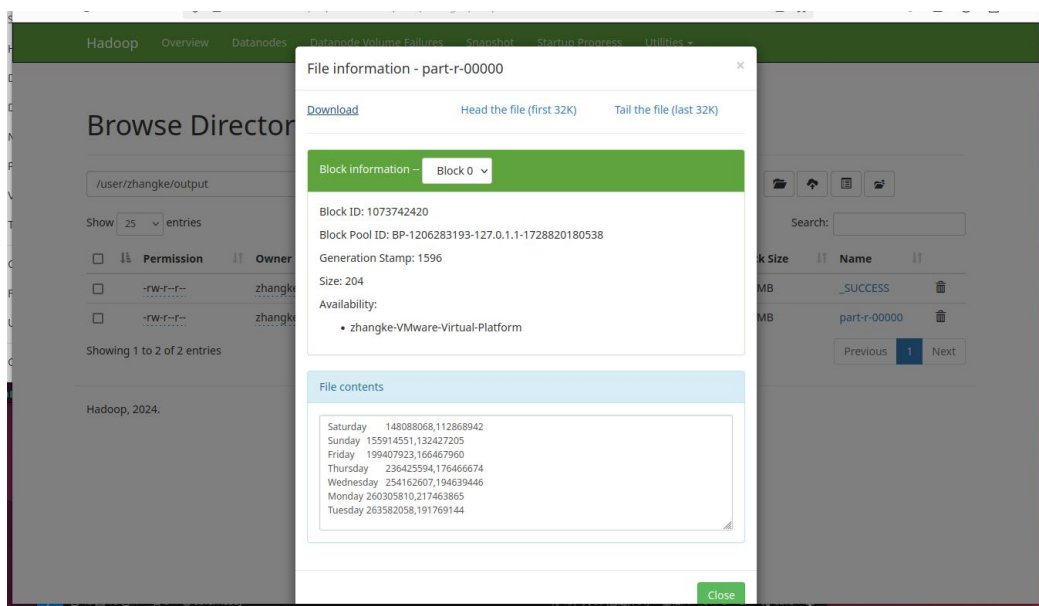
第一次输出的资金流入和流出的数值都过大，不符合预期，经检查发现题目要求的是一周七天中每天的平均资金，而我算的是一周七天中每天的总资金。所以添加 sum1 和 sum2 记录每个星期几资金流入和流出的总天数，最后用总资金除以天数来求得均值。

2.4 运行结果部分截图



The image shows a terminal window with the following content:

```
ut  
Deleted output  
zhangke@zhangke-VMware-Virtual-Platform: /usr/local/hadoop$ hadoop fs -rm -r input  
Deleted input  
zhangke@zhangke-VMware-Virtual-Platform: /usr/local/hadoop$ ./bin/hdfs dfs -put /home/zhangke/hw2/demo/out/output1 input  
zhangke@zhangke-VMware-Virtual-Platform: /usr/local/hadoop$ ./bin/hadoop jar /home/zhangke/hw2/demo/Week.jar com.example.WeekMain input output  
2024-11-03 22:49:42,049 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032  
2024-11-03 22:49:42,517 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.  
2024-11-03 22:49:42,544 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/zhangke/.staging/job_1730444029606_0019  
2024-11-03 22:49:42,779 INFO input.FileInputFormat: Total input files to process : 1  
2024-11-03 22:49:43,675 INFO mapreduce.JobSubmitter: number of splits:1  
2024-11-03 22:49:43,957 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1730444029606_0019  
2024-11-03 22:49:43,957 INFO mapreduce.JobSubmitter: Executing with tokens: []  
2024-11-03 22:49:44,484 INFO conf.Configuration: resource-types.xml not found  
2024-11-03 22:49:44,488 INFO resource.ResourceUtils: Unable to find 'resource-ty
```



3.任务三

3.1 任务描述

根据 `user_balance_table` 表中的数据，编写 MapReduce 程序，统计每个用户的活跃天数，并按照活跃天数降序排列。判断是否活跃：当用户当日有直接购买（`direct_purchase_amt` 字段大于 0）或赎回行为（`total_redeem_amt` 字段大于 0）时，则该用户当天活跃。

3.2 设计思路

3.2.1 Map 类设计

解析每一行输入数据，将每个用户的 ID 作为键，如果用户当日有活跃行为，即存在 `direct_purchase_amt` 字段大于 0 或者 `total_redeem_amt` 字段大于 0 的情况则输出键值对，值为 1。

```
JavaScript
String[] temps = value.toString().split(",");
if(!temps[5].isEmpty() && isNumeric(temps[5])){
    if(Integer.parseInt(temps[5])>0){
        userid.set(temps[0]);
        context.write(userid, one);
        return;
    }
}
```



```

else if(!temps[8].isEmpty() && isNumeric(temps[8])){
    if(Integer.parseInt(temps[8])>0){
        userid.set(temps[0]);
        context.write(userid, one);
        return;
    }
}

```

3.2.2 Reduce 类设计

Reducer 接收来自 Mapper 的键值对，对每个用户的活跃天数进行汇总。使用 `TreeMap` 存储用户及其活跃天数，以便在最后输出时能够利用其降序函数对活跃天数进行排序。

```

JavaScript
    protected void reduce(Text key, Iterable<IntWritable>
values, Context context) throws IOException,
InterruptedException{
        int sum = 0;
        for(IntWritable value : values){
            sum += value.get();
        }
        sortedusers.put(sum, key.toString());
    }

    @Override
    protected void cleanup(Context context) throws
IOException, InterruptedException{
        for(Map.Entry<Integer, String> entry:
sortedusers.descendingMap().entrySet()){
            String output = entry.getValue() + "\t" +
entry.getKey();
            context.write(new Text(output), null);
        }
    }

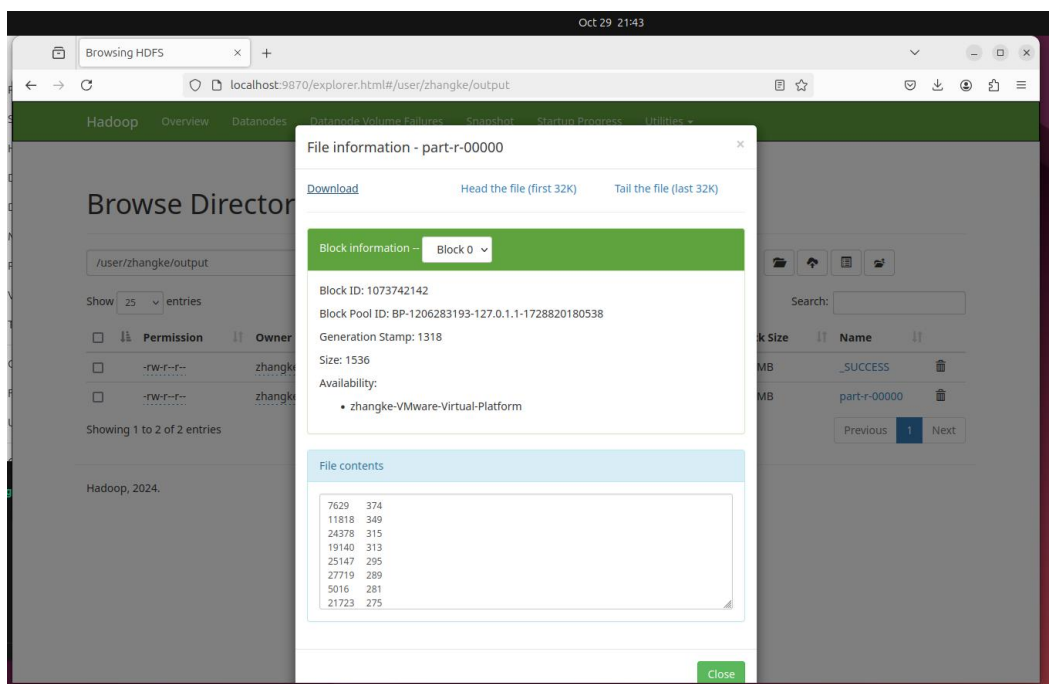
```

3.2.3 主类设计

在主类中确定 Map 类和 Reduce 类、变量的类型和输入、输出的路径。

3.3 运行结果部分截图

```
zhangke@zhangke-VMware-Virtual-Platform: /usr/local/hadoop
2024-10-29 21:34:12,849 INFO impl.YarnClientImpl: Submitted application applicat
ion_1730050316950_0012
2024-10-29 21:34:12,888 INFO mapreduce.Job: The url to track the job: http://zha
ngke-VMware-Virtual-Platform:8088/proxy/application_1730050316950_0012/
2024-10-29 21:34:12,889 INFO mapreduce.Job: Running job: job_1730050316950_0012
2024-10-29 21:34:16,987 INFO mapreduce.Job: Job job_1730050316950_0012 running i
n uber mode : false
2024-10-29 21:34:16,988 INFO mapreduce.Job: map 0% reduce 0%
2024-10-29 21:34:23,055 INFO mapreduce.Job: map 50% reduce 0%
2024-10-29 21:34:25,068 INFO mapreduce.Job: map 100% reduce 0%
2024-10-29 21:34:28,089 INFO mapreduce.Job: map 100% reduce 100%
2024-10-29 21:34:28,099 INFO mapreduce.Job: Job job_1730050316950_0012 completed
successfully
2024-10-29 21:34:28,161 INFO mapreduce.Job: Counters: 54
File System Counters
FILE: Number of bytes read=1954993
FILE: Number of bytes written=4834729
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=157765507
HDFS: Number of bytes written=1714
HDFS: Number of read operations=11
HDFS: Number of large read operations=0
```



4.任务四

4.1 任务描述

将七日年化收益率按数值大小分为三类，小于 5 的为第一类，5~6 是第二类，大于 6 的归为第三类，这三个档次七日年化收益率的日均支付宝余额购买量。

4.2 设计思路

将整个任务分成两步骤：Job1 和 Job2。Job1 负责将 user_balance 表中的日期与 mfd_day_share_interest 表中七日年化收益率所处的档次对应起来。Job2 则负责统计每个档次的日均支付宝余额购买量。

4.2.1 Job1 设计

Map 类（分为两个 Map）：

Map1 处理 user_balance 表，获取日期（索引为 2）和支付宝余额购买量（索引为 7），前者作为键，后者作为值传给 Reduce。

```
JavaScript
context.write(dateKey, new DoubleWritable(purchaseAmt));
```

Map2 处理 mfd_day_share_interest 表，获取日期和七日年化收益率，前者为键，后者加上负号作为值，传给 Reduce。

```
JavaScript
context.write(dateKey, new DoubleWritable(-interest));
```

Reduce 类：

接收 Mapper 输出的键值对，键是日期，若值为正数则是支付宝余额购买量，将其累加；如果值为负数就是该日的七日年化收益率，根据这个负数的大小确定档次，作为 Reduce 输出的键，而累加得到的支付宝余额购买量则作为值。传给 Job2。

```
JavaScript
for (DoubleWritable val : values) {
    if (val.get() >= 0) {
        buy += val.get();
    } else {
        interest = -val.get();
        if (interest >= 0 && interest < 5) {
            categoryKey.set("1");
        } else if (interest >= 5 && interest < 6) {
            categoryKey.set("2");
        } else if (interest > 6) {
            categoryKey.set("3");
        }
    }
}

// 输出分类后的 key 和购买量
```

```
context.write(new Text(categoryKey), new DoubleWritable(buy));
```

4.2.2 Job2 设计

Map 类:

主要是负责接收 Job1 的 Reduce 输出，不对其进行任何处理，传递给 Reduce。

Reduce 类:

将同一档次的支付宝余额购买量相加，同时用 sum 统计每次遍历的那个档次出现的天数，后续作为计算此档次日均支付宝余额购买量的除数。

```
JavaScript
double totalBuy = 0.0;
double sum = 0;
for (DoubleWritable val : values) {
    totalBuy += val.get();
    sum ++;
}
totalBuy = totalBuy/sum;

context.write(key, new DoubleWritable(totalBuy));
```

4.2.3 主类设计

在主类中部署两个 Job 的进行顺序及其对应的输入输出文件路径。

```
JavaScript
Job job1 = Job.getInstance(conf, "First Analyze Job");

job1.setJarByClass(AnalyzeMain.class);

// 设置 Mapper 和 Reducer 类
MultipleInputs.addInputPath(job1, new Path(args[0] +
"/user_balance_table.csv"),
org.apache.hadoop.mapreduce.lib.input.TextInputFormat.class,
AnalyzeMapper1.class);
MultipleInputs.addInputPath(job1, new Path(args[0] +
"/mfd_day_share_interest.csv"),
org.apache.hadoop.mapreduce.lib.input.TextInputFormat.class,
AnalyzeMapper2.class);

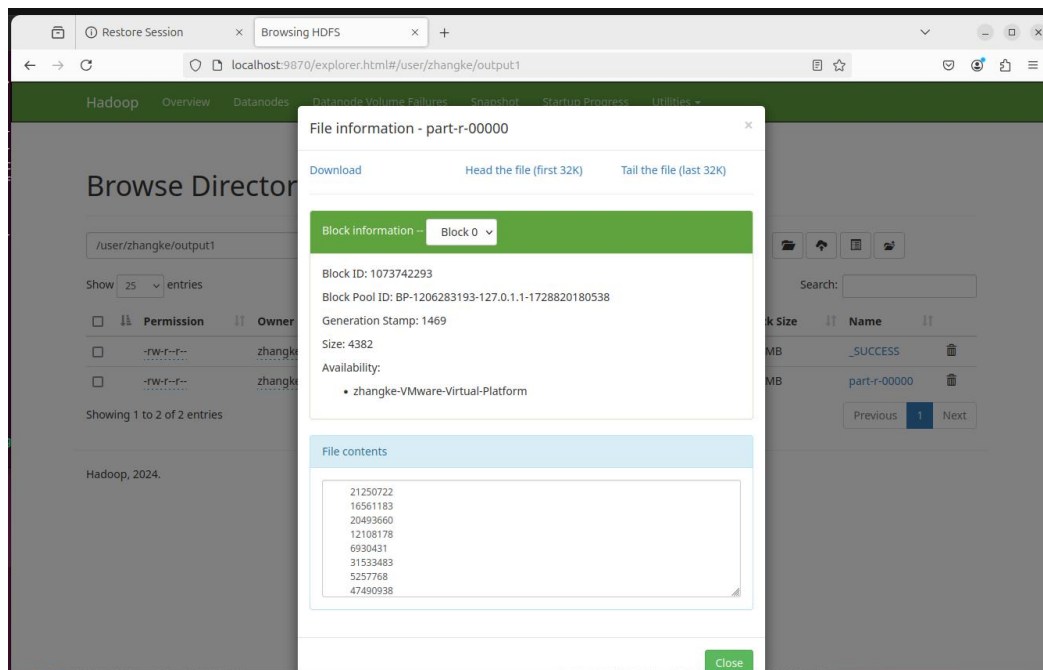
job1.setReducerClass(AnalyzeReducer.class);
```

```
JavaScript
if (!job1.waitForCompletion(true)) {
    System.exit(1);
}

// 设置第二个作业
Job job2 = Job.getInstance(conf, "Final Analyze Job");
job2.setJarByClass(AnalyzeMain.class);
job2.setMapperClass(Job2Mapper.class);
job2.setReducerClass(FinalReducer.class);
```

4.3 遇到的问题及解决方案

1. Job1 的输出没有进行分类，只有支付宝余额购买量（如下图）。



经检查发现是因为七日年化收益率中的数值多为小数，在判断是否为整数时，被筛掉了，也就造成 Reduce 接收到的值都为正，没有可以用来分类的依据。解决方法就是，在原代码基础上加上是小数也符合条件的筛选。

```
JavaScript
private boolean isNumeric(String str) {
    return str != null && str.matches("\\d+(\\.\\d+)?"); // 支持小数
}
```

2. 一开始没有在 Job2 中加入 Map 类时，运行时报错：

Bash

```
Error: java.io.IOException: Type mismatch in key from map:
expected org.apache.hadoop.io.Text, received
org.apache.hadoop.io.LongWritable
```

这是因为，当任务中不存在 Map 类时，程序会运行默认的 Map 类，虽然不会做任何处理，但是由于默认 Map 输入的类型为 Text，所以在接收到 Job1 中 Reduce 的输出包含数值型就生成报错。所以我通过添加一个不做任何处理但可接收输入和 Reduce 输出类型匹配的 Map 类。

JavaScript

```
public class Job2Mapper extends Mapper<LongWritable, Text,
Text, DoubleWritable>
```

3. 运行时报错:

Bash

```
Exception in thread "main"
org.apache.hadoop.mapred.FileAlreadyExistsException: Output
directory hdfs://localhost:9000/user/zhangke/output already
exists
```

由于可以确认运行前已经将 output 文件夹删除了，通过查看日志发现，运行 Job1 时程序正常，该报错是在运行 Job2 时产生的。所以最终确定是因为 Job1 的输出文件和 Job2 的输出文件路径都是 output，当 Job1 运行完，输出文件会自动创建 output 文件夹，此时再运行 Job2，output 文件夹已存在，故出现报错。

解决方案：将 Job1 和 Job2 的输出文件路径分开。

JavaScript

```
Path outputPathJob1 = new Path(args[1]);
FileOutputFormat.setOutputPath(job1, outputPathJob1);
```

JavaScript

```
Path outputPathJob2 = new Path(args[2]);
FileOutputFormat.setOutputPath(job2, outputPathJob2);
```

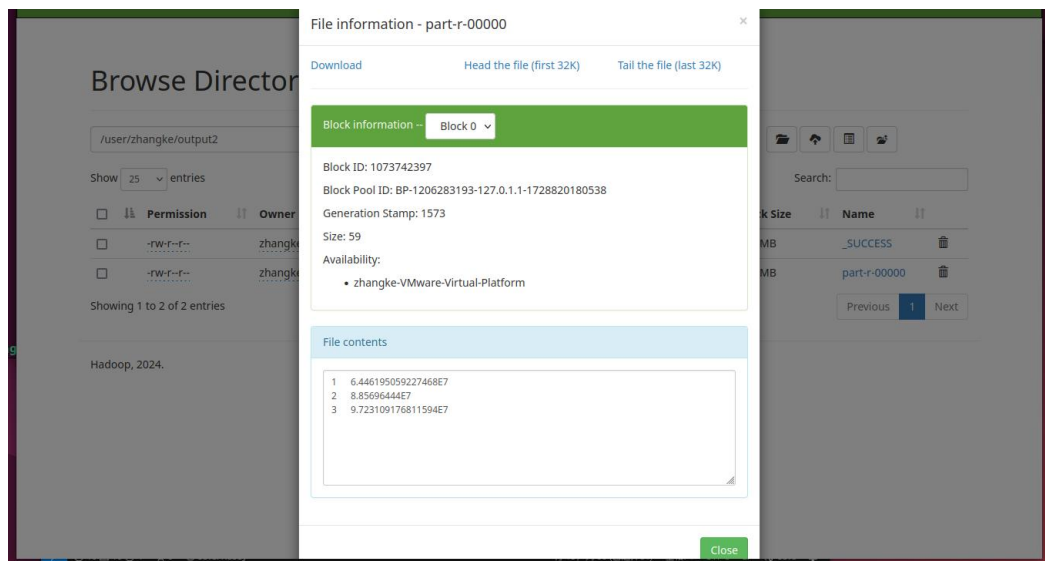
最后通过下面的命令运行 MapReduce 程序:

JavaScript

```
./bin/hadoop jar /home/zhangke/hw2/demo/Analyze.jar
com.example.AnalyzeMain input output1 output2
```

4.4 运行结果与分析

运行结果截图：



结果分析：

由上述结果可知，余额宝的七日年化收益率所处档次越高，支付宝余额宝购买量就越多，这也与我的预期相符。即余额宝的七日年化收益率与支付宝余额宝购买量之间存在正相关关系。这种现象可以从以下多个方面进行进一步分析：

- 收益动机：用户通常倾向于将资金投入收益更高的理财产品中，以追求更好的投资回报。因此，当余额宝的七日年化收益率较高时，用户的购买意愿和资金流入量都会增加。
- 风险偏好：较高的收益率往往会吸引风险承受能力较强的投资者，他们希望通过短期的高收益来获得额外的利益。这可能导致更多的用户选择余额宝作为其资金管理的工具。
- 市场环境：如果市场整体的利率水平较低，用户可能会更积极地寻找收益较高的理财产品。此时，余额宝的吸引力会进一步增强，促进用户增加投资。

5.可能的改进之处

- 当前代码对数据格式的处理较为简单，也许可以增强异常处理能力，如捕捉可能出现的解析异常，以保证程序的健壮性。
- 对于大规模数据，使用 **TreeMap** 可能会导致性能下降，可以考虑使用其他高效的数据结构或直接排序的方法。可在之后进行实现。
- 可以考虑支持更多的输入格式和数据清洗功能，例如处理缺失值或异常值，以提高程序的适用性。