

## 实验二实验报告

221275023 朱奕冰

### 任务一：每日资金流入流出统计

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。

程序设计思路如下：

map任务：在map任务中，利用split函数对每行进行分割，随后取出相应的日期以及资金流入流出列，将流入与流出的资金拼接为字符串作为Text对象，利用context.write进行传输即可

```
public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
    String line = value.toString();
    String[] fields = line.split(","); // 假设字段是用逗号分隔的
    if (fields.length < 6) { // 确保有足够的字段
        return; // 忽略无效行
    }
    String mfd_date = fields[1].trim(); // 日期
    String total_purchase_amt = fields[4].trim(); // 资金流入
    String total_redeem_amt = fields[5].trim(); // 资金流出
    // 如果字段缺失，视为零
    if (total_purchase_amt.isEmpty()) {
        total_purchase_amt = "0";
    }
    if (total_redeem_amt.isEmpty()) {
        total_redeem_amt = "0";
    }
    // 输出格式: <日期>      <资金流入量,资金流出量>
    dateKey.set(mfd_date);
    flowValues.set(total_purchase_amt + "," + total_redeem_amt);
    context.write(dateKey, flowValues);
}
}
```

在reduce任务中，只需获取资金流入/流出值，进行累加求和即可。最终利用context.write作为最终输出。


```
try {
    // 尝试解析资金流入和流出的值
    double inflow = Double.parseDouble(flowAmounts[0]);
    double outflow = Double.parseDouble(flowAmounts[1]);

    // 仅在解析成功后累加
    totalInflow += inflow;
    totalOutflow += outflow;
} catch (NumberFormatException e) {
    System.err.println("Skipping invalid numbers: " +
val.toString());
    continue; // 跳过无法解析的值
}
```

```
    }

    // 输出最终的资金流入和流出总额
    result.set(totalInflow + "," + totalOutflow);
    context.write(key, result);
}
```

```
user@user-virtual-machine:~/local/code/lab2$ hadoop jar target/my-mapreduce-project-1.0-SNAPSHOT.jar com.example.hadoop.Fund /user/lab2/user_balance_table.csv /user/lab2/output1
2024-10-20 09:26:19,388 INFO client.DefaultHadoopFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-10-20 09:26:19,736 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/user/.staging/job_1729387495542_0001
2024-10-20 09:26:20,670 INFO input.FileInputFormat: Total input files to process : 1
2024-10-20 09:26:21,549 INFO mapreduce.JobSubmitter: number of splits:2
2024-10-20 09:26:21,659 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1729387495542_0001
2024-10-20 09:26:21,659 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-10-20 09:26:21,768 INFO conf.Configuration: resource-types.xml not found
2024-10-20 09:26:21,769 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'
2024-10-20 09:26:22,285 INFO impl.YarnClientImpl: Submitted application application_1729387495542_0001
2024-10-20 09:26:22,321 INFO mapreduce.Job: The url to track the job: http://user-virtual-machine:8088/proxy/application_1729387495542_0001/
2024-10-20 09:26:22,322 INFO mapreduce.Job: Running job: job_1729387495542_0001
2024-10-20 09:26:33,417 INFO mapreduce.Job: Job job_1729387495542_0001 running in uber mode : false
2024-10-20 09:26:33,418 INFO mapreduce.Job: map 0% reduce 0%
2024-10-20 09:26:40,483 INFO mapreduce.Job: map 50% reduce 0%
2024-10-20 09:26:43,494 INFO mapreduce.Job: map 100% reduce 0%
2024-10-20 09:26:46,509 INFO mapreduce.Job: map 100% reduce 100%
2024-10-20 09:26:48,534 INFO mapreduce.Job: Job job_1729387495542_0001 completed successfully
2024-10-20 09:26:48,616 INFO mapreduce.Job: Counters: 55
File System Counters
  FILE: Number of bytes read=46277029
  FILE: Number of bytes written=93382585
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
```

All Applications 

91%  
+ 16.5K/s  
CPU 82°C

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Used Resources	Total Resources
4	0	0	4	0	<memory:0 B, vCores:0>	<memory:8 GB, vCores:8>

Cluster Nodes Metrics


Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes
1	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>	<memory:8192, vCores:4>

Show 20 ▼ entries

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores
application_1729387495542_0004	user	weekly fund flow	MAPREDUCE		default	0	Sun Oct 20 09:55:24 +0800 2024	Sun Oct 20 09:55:24 +0800 2024	Sun Oct 20 09:55:36 +0800 2024	FINISHED	SUCCEEDED	N/A	N/A

c0a66d397247a83042c8e0a7b503ed7

## 任务二：星期交易量统计

基于任务一的结果，编写MapReduce程序，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排序。

程序设计思路如下：

定义由日期向星期的转换函数，将其作为map的辅助方法。Java的包time.format.DateTimeFormatter可以实现日期的解析。

```
private String getWeekday(String dateStr) {
    DateTimeFormatter formatter =
    DateTimeFormatter.ofPattern("yyyyMMdd");
    LocalDate date = LocalDate.parse(dateStr, formatter);
    return date.getDayOfWeek().name(); // 获取星期几的名称
}
```

由于利用的是任务1的输出文件，所以将输入类型确定为KeyValueTextInputFormat.class，有利于后续处理。

```
job.setInputFormatClass(KeyValueTextInputFormat.class);
```

在map任务中，需要调用该函数，将日期转换为相应的星期，进行传输

```
mapContext.write(new Text(dateStr), new Text(getWeekday(dateStr)));
```

```

public void map(Text key, Text value, Context context) throws IOException,
InterruptedException {
    String dateStr = key.toString().trim(); // 日期 (key)
    String[] flowAmounts = value.toString().split(","); // 输入格式 v,v

    if (flowAmounts.length < 2) {
        return; // 忽略无效行
    }
    String total_purchase_amt = flowAmounts[0].trim(); // 资金流入
    String total_redeem_amt = flowAmounts[1].trim(); // 资金流出
    try {
        String weekday = getWeekday(dateStr);
        weekdayKey.set(weekday);
        flowValues.set(total_purchase_amt + "," + total_redeem_amt);
        context.write(weekdayKey, flowValues);
    } catch (DateTimeParseException e) {
        // 记录异常, 但不使任务失败
        System.err.println("Invalid date format for input: " + dateStr);
    }
}

```

为了实现输出的排序, 定义辅助类以便后续处理, 包含星期以及平均资金流入/流出。并且定义列表, 用于存储reduce时统计的数据信息

```

public static class WeekData {
    String week;
    double avgInflow;
    double avgOutflow;

    WeekData(String week, double avgInflow, double avgOutflow) {
        this.week = week;
        this.avgInflow = avgInflow;
        this.avgOutflow = avgOutflow;
    }
}

```

在reduce任务中进行统计求和, 并且求平均, 将结果存入list当中

```

public void reduce(Text key, Iterable<Text> values, Context context) throws
IOException, InterruptedException {
    double totalInflow = 0.0;
    int count = 0;
    double totalOut = 0.0;
    for (Text val : values) {
        String[] flowAmounts = val.toString().split(",");
        double inflow = Double.parseDouble(flowAmounts[0]); // 获取资金流入
        double outflow = Double.parseDouble(flowAmounts[1]);
        totalInflow += inflow;
        totalOut += outflow;
        count++;
    }

    // 计算平均值
}

```

```
double avgInflow = totalInflow / count;
double avgOut = totalOut / count;

// 将平均值和星期添加到列表中
weekDataList.add(new WeekData(key.toString(), avgInflow, avgOut));
}
```

由于总共7个数据，排序开销小，因此直接在cleanup阶段，调用sort函数利用数组实现排序即可。

```
@Override
protected void cleanup(Context context) throws IOException,
InterruptedException {
    // 对列表进行排序，按 avgInflow 从大到小排序
    Collections.sort(weekDataList, new Comparator<WeekData>() {
        public int compare(WeekData w1, WeekData w2) {
            return Double.compare(w2.avgInflow, w1.avgInflow); // 降序排序
        }
    });


    // 输出排序后的结果
    for (WeekData weekData : weekDataList) {
        result.set(weekData.avgInflow + "," + weekData.avgOutflow);
        context.write(new Text(weekData.week), result);
    }
}
```

```
user@user-virtual-machine:~/local/code/lab2$ hadoop jar target/my-mapreduce-project-1.0-SNAPSHOT.jar main.java.com.example.hadoop.WeeklyFundFlow /user/lab2/input2 /user/lab2/output2
2024-10-20 10:31:21,805 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-10-20 10:31:22,015 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2024-10-20 10:31:22,026 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/user/.staging/job_1729387495542_0007
2024-10-20 10:31:22,256 INFO input.FileInputFormat: Total input files to process : 1
2024-10-20 10:31:22,721 INFO mapreduce.JobSubmitter: number of splits:1
2024-10-20 10:31:23,207 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1729387495542_0007
2024-10-20 10:31:23,207 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-10-20 10:31:23,340 INFO conf.Configuration: resource-types.xml not found
2024-10-20 10:31:23,341 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2024-10-20 10:31:23,375 INFO impl.YarnClientImpl: Submitted application application_1729387495542_0007
2024-10-20 10:31:23,403 INFO mapreduce.Job: The url to track the job: http://user-virtual-machine:8088/proxy/application_1729387495542_0007/
2024-10-20 10:31:23,403 INFO mapreduce.Job: Running job: job_1729387495542_0007
2024-10-20 10:31:28,502 INFO mapreduce.Job: Job job_1729387495542_0007 running in uber mode : false
2024-10-20 10:31:29,504 INFO mapreduce.Job: map 0% reduce 0%
2024-10-20 10:31:32,581 INFO mapreduce.Job: map 100% reduce 0%
2024-10-20 10:31:36,626 INFO mapreduce.Job: map 100% reduce 100%
2024-10-20 10:31:37,662 INFO mapreduce.Job: Job job_1729387495542_0007 completed successfully
2024-10-20 10:31:37,725 INFO mapreduce.Job: Counters: 54
File System Counters
  FILE: Number of bytes read=15093
  FILE: Number of bytes written=582657
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=14722
  HDFS: Number of bytes written=340
  HDFS: Number of read operations=8
```

← → ↺

localhost:8088/cluster

🔍 ⭐ 🏠 📄



All Applications 

93%

0.8K/sCPU 46°C

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Used Resources	Total Resources
7	0	0	7	0	<memory:0 B, vCores:0>	<memory:8 GB, vCores:8>

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes
1	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>	<memory:8192, vCores:4>

Show 20 entries

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU, VCore
application_1729387495542_0007	user	weekly fund flow	MAPREDUCE		default	0	Sun Oct 20 10:31:23 +0800 2024	Sun Oct 20 10:31:23 +0800 2024	Sun Oct 20 10:31:36 +0800 2024	FINISHED	SUCCEEDED	N/A	N/A

278074c0e9430be7a6dc07f2f406849

### 任务三：用户活跃度分析

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。

该任务与任务一类似，根据map阶段读取的数据，判断当天用户是否活跃，若活跃将key=userId，value=1，进行写出即可。

```
userKey.set(user);
if (directPurchaseAmt > 0 || totalRedeemAmt > 0) {
    context.write(userKey, new Text("1")); // 活跃用户输出 "1"
}
```


reduce阶段，进行求和与计数

```
public void reduce(Text key, Iterable<Text> values, Context context) throws
IOException, InterruptedException {
    int sum = 0;
    for (Text val : values) {
        sum += Integer.parseInt(val.toString()); // 将 Text 转换为 String,
再解析为 int
    }
    userActivityList.add(new UserActivity(key.toString(), sum));
}
```

在最终输入的排序上，可以再编写一个mapreduce任务，利用key-value反转进行排序，也可以在cleanup时原地排序，这里采用直接原地排序的方法

```
@Override
protected void cleanup(Context context) throws IOException,
InterruptedException {
    // 对用户按活跃天数进行降序排序
    Collections.sort(userActivityList, new Comparator<UserActivity>() {
        @Override
        public int compare(UserActivity u1, UserActivity u2) {
            return Integer.compare(u2.activeDays, u1.activeDays); // 降序
        }
    });
    // 输出排序后的结果
    for (UserActivity userActivity : userActivityList) {
        context.write(new Text(userActivity.userId), new
Text(String.valueOf(userActivity.activeDays)));
    }
}
```

```
user@user-virtual-machine:~/local/code/lab2$ hadoop jar target/my-mapreduce-project-1.0-SNAPSHOT.jar main.java.com.example.hadoop.Active /user/lab2/user_balance
table.csv /user/lab2/output3
2024-10-20 11:11:36,542 INFO client.DefaultNoHARMAFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-10-20 11:11:36,748 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/user/.staging/job_1729387495542_0009
2024-10-20 11:11:36,949 INFO input.FileInputFormat: Total input files to process : 1
2024-10-20 11:11:37,831 INFO mapreduce.JobSubmitter: number of splits:2
2024-10-20 11:11:38,330 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1729387495542_0009
2024-10-20 11:11:38,330 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-10-20 11:11:38,476 INFO conf.Configuration: resource-types.xml not found
2024-10-20 11:11:38,476 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2024-10-20 11:11:38,513 INFO impl.YarnClientImpl: Submitted application application_1729387495542_0009
2024-10-20 11:11:38,539 INFO mapreduce.Job: The url to track the job: http://user-virtual-machine:8088/proxy/application_1729387495542_0009/
2024-10-20 11:11:38,539 INFO mapreduce.Job: Running job: job_1729387495542_0009
2024-10-20 11:11:43,653 INFO mapreduce.Job: Job job_1729387495542_0009 running in uber mode : false
2024-10-20 11:11:43,655 INFO mapreduce.Job: map 0% reduce 0%
2024-10-20 11:11:47,725 INFO mapreduce.Job: map 50% reduce 0%
2024-10-20 11:11:49,745 INFO mapreduce.Job: map 100% reduce 0%
2024-10-20 11:11:52,785 INFO mapreduce.Job: map 100% reduce 100%
2024-10-20 11:11:53,822 INFO mapreduce.Job: Job job_1729387495542_0009 completed successfully
2024-10-20 11:11:53,894 INFO mapreduce.Job: Counters: 54
File System Counters
FILE: Number of bytes read=3363569
FILE: Number of bytes written=7555755
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
```



All Application92%

0.76%

CPU 46°C

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Used Resources	Total Resources
9	0	0	9	0	<memory:0 B, vCores:0>	<memory:8 GB, vCores:8>

Cluster Nodes Metrics


Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes
1	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	(memory-mb (unit=Mi), vcores)	<memory:1024, vCores:1>	<memory:8192, vCores:4>

Show 20 entries

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCore
application_1729387495542_0009	user	Active user count	MAPREDUCE		default	0	Sun Oct 20 11:11:38 +0800 2024	Sun Oct 20 11:11:38 +0800 2024	Sun Oct 20 11:11:52 +0800 2024	FINISHED	SUCCEEDED	N/A	N/A

a3917a1f76f8064edd4e875c18090b0

### 任务四：交易行为影响因素分析

在本任务中尝试基于任务三的结果与user\_profile进行结合分析，探究星座对用户活跃程度的影响，即统计每个星座用户的平均活跃天数。

为了实现由用户id到星座的映射，将user\_profile文件存到hdfs中，并构建用用户id到用户星座的映射表  
此操作在setup阶段进行

```
@Override
protected void setup(Context context) throws IOException,
InterruptedException {
    Configuration conf = context.getConfiguration();
    Path path = new Path(conf.get("zodiac.file.path"));
    FileSystem fs = FileSystem.get(conf);
    FSDataInputStream inputStream = fs.open(path);
    BufferedReader reader = new BufferedReader(new
InputStreamReader(inputStream));
    String line;
    while ((line = reader.readLine()) != null) {
        String[] fields = line.split(",");
        if (fields.length >= 4) { // 确保有足够的字段
            String userId = fields[0];
            String zodiacSign = fields[3]; // 假设星座在第四列
            zodiacMap.put(userId, zodiacSign);
        } else {
            system.err.println("Invalid line format: " + line);
        }
    }

    reader.close();
    inputStream.close();
}
```

```
}
```

在map阶段，将key从用户id映射至其星座

```
@Override
    protected void map(Text key, Text value, Context context) throws
IOException, InterruptedException { // 修改 LongWritable 为 Text
        String userId = key.toString();
        long activeDays = Long.parseLong(value.toString()); // 将 Text 转换为
long

        String zodiacSign = zodiacMap.get(userId);

        if (zodiacSign != null) {
            context.write(new Text(zodiacSign), new
LongWritable(activeDays));
        }
    }
}
```

reduce阶段只需求均值即可

```
public static class AvgActiveDaysReducer extends Reducer<Text, LongWritable,
Text, Text> {
    @Override
    protected void reduce(Text key, Iterable<LongWritable> values, Context
context) throws IOException, InterruptedException {
        long totalActiveDays = 0;
        int count = 0;

        for (LongWritable value : values) {
            totalActiveDays += value.get();
            count++;
        }

        double averageActiveDays = count > 0 ? (double) totalActiveDays /
count : 0;
        context.write(key, new Text(String.format("%.2f",
averageActiveDays)));
    }
}
```

为了实现全局路径参数传递，需要在main类中设置user\_profile的路径参数

```
conf.set("zodiac.file.path", args[2]); // 从命令行获取星座文件路径

Job job = Job.getInstance(conf, "Zodiac Average Active Days");
```

最终执行结果为f56b5e00b2b0c0013a3a7b51a7f62b4

c65090c9d0623ae80cd480305832e01

由此可见星座对用户的活跃水平影响并不显著，所有星座的用户活跃天数均值都在（20，24）天内。