

## 功能 2：统计相同订单优先级的订单中订单发货优先级最高的订单信息

---

### 一、设计思路

#### (1) Mapper 类：

##### (a) 输入

- Key 类型：输入的键是 `Object`，这通常是 Hadoop 的默认值，但实际上它不被使用。这里的关键是文件中的行号，而不是行内容。
- Value 类型：输入的值是 `Text`，代表了文件中的一行内容。

##### (b) 输出

- Key 类型：MyPair\_t（由订单键和负的订单优先级）
- Value 类型：记录 ID

思路：

1. 每一行输入数据表示一个订单的记录，字段之间使用 `|` 分隔。
2. 通过 `|` 将每一行数据分割成多个字段（order\_key, cust\_key, order\_status, total\_price, order\_data, order\_prority, clerk, ship\_priority 和 comment）
3. 提取相关字段：第一个字段 (result[0]) 表示订单的标识符（order\_key），第六个字段 (result[5]) 表示订单的优先级（order\_priority）和第八个字段（result[7]）表示订单发货的优先级（ship\_priority）。
4. 将订单发货的优先级取复制，为了实现降序排序
5. 构建键值对 MyPair\_t 里面包含了订单键和订单优先级
6. 将 MyPair\_t 和记录 Id 作为值的键值对列表

#### (2) Partitioner 类

##### (c) 输入

- Key 类型：MyPair\_t（从 Mapper 输出的键）
- Value 类型：记录 ID

##### (d) 输出

- Key 类型：int 类型，计算每个键值对应的分区号，确保相同订单键的记录发送给同一个 Reducer

思路：

1. 使用 HashPartitioner 对 order\_key 进行哈希分区

### (3) Reducer 类

#### (e) 输入

- Key 类型: `MyPair_t` (从 Mapper 输出的键)
- Value 类型: 和 `MyPair_t` 键对应的值, 记录 ID 的迭代器

#### (f) 输出

- Key 类型: 'Text', 由记录 ID, 订单键和订单优先级重新组合的成本
- Value 类型: 空值

思路 (去重):

1. 初始化一个字符串变量 'elem' 用于记录上一个处理的订单键
2. 对每个 `MyPair_t` 键值对, 先获取订单键和订单优先级
3. 如果当前订单键和 elem 记录的上一个订单键不同, 则更新 elem 并继续处理
4. 如果当前订单键和 elem 记录的上一个订单键一样, 则使用 `Desrializer_t` 静态类将其序列化并输出

MapReduce 中 Map\_t , Partitioner\_t 和 Reduce\_t 的伪代码。

Map 的伪代码:

```
public static class Mapper_t extends Mapper<Object, Text, MyPair_t, Text> {
    private static String separator = "|"; //按|拆分为多个字段

    public static String[] extractFields(String str) {
        // extracts fields from a line

        if (str == null) {
            throw new IllegalArgumentException(s:"Arguments cannot be null or empty");
        }

        int capacity = 0;
        int start = 0;
        int end;

        // 计算分隔符出现的次数
        while ((end = str.indexOf(separator, start)) != -1) {
            capacity++;
            start = end + 1;
        }

        //最后一个分隔符后面的部分
        if (start < str.length()) {
            capacity++;
        }

        // 创建数组
        String[] result = new String[capacity];
        capacity = 0;
        start = 0;
        while ((end = str.indexOf(separator, start)) != -1) {
            result[capacity++] = str.substring(start, end);
            start = end + separator.length();
        }

        if (start < str.length()) {
            result[capacity++] = str.substring(start);
        }
        //提取所需字段
        String[] fields = new String[3];
        fields[0] = result[0]; //order_key 订单的标识符
        fields[1] = result[5]; //order_priority 订单优先级
        fields[2] = result[7]; //ship_priority 订单发货优先级

        return fields;
    }

    @Override
    public void map(Object key, Text value, Context context)
        throws IOException, InterruptedException {
        String[] fieldsList = extractFields(value.toString());
        Text orderKey = new Text(fieldsList[1]);
        IntWritable orderPriority = new IntWritable(-Integer.parseInt(fieldsList[2]));
        MyPair_t priorityKeyPair =
            new MyPair_t(
                orderKey,
                orderPriority
            );

        context.write(
            priorityKeyPair,
            new Text(fieldsList[0]));
    }
}
```

## Partitioner

```
public static class Partitioner_t extends Partitioner<MyPair_t, Text> {
    private final HashPartitioner<Text, Text> hashPartitioner = new HashPartitioner<>();

    @Override
    public int getPartition(MyPair_t myPair, Text text, int i) {
        return hashPartitioner.getPartition(myPair.getKey(), text, i);
    }
}
```

## Reducer

```
@Override
public void reduce(MyPair_t key, Iterable<Text> values, Context context)
    throws IOException, InterruptedException {
    int shipPrior = key.getValue().get();
    String key_str = key.getKey().toString();

    if (this.elem.equals(key_str) == false) { //只处理不同的订单键，去重
        this.elem = key_str;

        for (Text t : values) {
            context.write(
                Desirializer_t.deserialize(t, key_str, shipPrior),
                NullWritable.get()
            );
        }
    }
}
```


## 一、实验结果

输出文件路径：/output\_lab3-2

(1) 输出结果文件的部分截图

```
hive> SELECT * FROM task2_res1 LIMIT 20;
OK
4097    1-URGENT    7    NULL    NULL
3107    1-URGENT    7    NULL    NULL
15553   1-URGENT    7    NULL    NULL
6534    1-URGENT    7    NULL    NULL
6535    1-URGENT    7    NULL    NULL
4608    1-URGENT    7    NULL    NULL
6595    1-URGENT    7    NULL    NULL
7489    1-URGENT    7    NULL    NULL
6693    1-URGENT    7    NULL    NULL
7456    1-URGENT    7    NULL    NULL
13155   1-URGENT    7    NULL    NULL
12642   1-URGENT    7    NULL    NULL
4614    1-URGENT    7    NULL    NULL
12710   1-URGENT    7    NULL    NULL
7299    1-URGENT    7    NULL    NULL
7232    1-URGENT    7    NULL    NULL
13028   1-URGENT    7    NULL    NULL
6918    1-URGENT    7    NULL    NULL
17891   1-URGENT    7    NULL    NULL
7072    1-URGENT    7    NULL    NULL
Time taken: 2.615 seconds, Fetched: 20 row(s)
```

(2) Yarn Resource Manager 的 WebUI 执行报告内容



Application application\_1678703754602\_10805

Logged in as: dr.wht

Cluster

About

Nodes

Node Labels

Applications

NEW

NEW SAVING

SUBMITTED

ACCEPTED

RUNNING

FINISHED

FAILED

KILLED

Scheduler

Tools

Application Overview

User: 2024stu\_10

Name: ShipmentTask2

Application Type: MAPREDUCE

Application Tags:

Application Priority: 0 (Higher integer value indicates higher priority)

YarnApplicationState: FINISHED

Queue: 2024stu0000class2

FinalStatus Reported by AM: SUCCEEDED

Started: Sat May 25 21:33:20 +0800 2024

Launched: Sat May 25 21:33:20 +0800 2024

Finished: Sat May 25 21:36:59 +0800 2024

Elapsed: 3mins, 39sec

Tracking URL: [History](#)

Log Aggregation Status: TIME\_OUT

Application Timeout (Remaining Time): Unlimited

Diagnostics:

Unmanaged Application: false

Application Node Label expression: <Not set>

AM container Node Label expression: <DEFAULT\_PARTITION>

Application Metrics

Total Resource Preempted: <memory:0, vCores:0>

Total Number of Non-AM Containers Preempted: 0

Total Number of AM Containers Preempted: 0

Resource Preempted from Current Attempt: <memory:0, vCores:0>

Number of Non-AM Containers Preempted from Current Attempt: 0

Aggregate Resource Allocation: 678008 MB-seconds, 434 vcore-seconds, 0 yam.io/gpu-seconds

Aggregate Preempted Resource Allocation: 0 MB-seconds, 0 vcore-seconds

Show 20 entries

Attempt ID	Started	Node	Logs	Nodes blacklisted by the app	Nodes blacklisted by the system
appattempt_1678703754602_10805_000001	Sat May 25 21:33:20 +0800 2024	<a href="http://slave009.8042">http://slave009.8042</a>	Logs	0	0

Showing 1 to 1 of 1 entries

First Previous 1 Next Last