**赛题05**

**背景描述**

大数据时代背景下，电商经营模式发生很大改变。在传统运营模式中，缺乏数据积累，人们在做出一些决策行为过程中，更多是凭借个人经验和直觉，发展路径比较自我封闭。而大数据时代，为人们提供一种全新的思路，通过大量的数据分析得出的结果将更加现实和准确。商家可以对客户的消费行为信息数据进行收集和整理，比如消费者购买产品的花费、选择产品的渠道、偏好产品的类型、产品回购周期、购买产品的目的、消费者家庭背景、工作和生活环境、个人消费观和价值观等。通过数据追踪，知道顾客从哪儿来，是看了某网站投放的广告还是通过朋友推荐链接，是新访客还是老用户，喜欢浏览什么产品，购物车有无商品，是否清空，还有每一笔交易记录，精准锁定一定年龄、收入、对产品有兴趣的顾客，对顾客进行分组、标签化，通过不同标签组合运用，获得不同目标群体，以此开展精准推送。

因数据驱动的零售新时代已经到来，没有大数据，我们无法为消费者提供这些体验，为完成电商的大数据分析工作，你所在的小组将应用大数据技术，以Scala作为整个项目的基础开发语言，基于大数据平台综合利用Hive、Spark、Flink、Vue.js等技术，对数据进行处理、分析及可视化呈现，你们作为该小组的技术人员，请按照下面任务完成本次工作。

## 任务A：大数据平台搭建（容器环境）（15分）

**环境说明：**

|  |
| --- |
| **服务端登录地址详见各任务服务端说明。**  **补充说明：**宿主机可通过Asbru工具或SSH客户端进行SSH访问；  相关软件安装包在宿主机的/opt目录下，请选择对应的安装包进行安装，用不到的可忽略；  所有任务中应用命令必须采用绝对路径；  进入Master节点的方式为  docker exec -it master /bin/bash  进入Slave1节点的方式为  docker exec -it slave1 /bin/bash  进入Slave2节点的方式为  docker exec -it slave2 /bin/bash  三个容器节点的root密码均为123456 |

### 子任务一：Hadoop 完全分布式安装配置

本任务需要使用root用户完成相关配置，安装Hadoop需要配置前置环境。命令中要求使用绝对路径，具体要求如下:

1. 从宿主机/opt目录下将文件hadoop-3.1.3.tar.gz、jdk-8u212-linux-x64.tar.gz复制到容器Master中的/opt/software路径中（若路径不存在，则需新建），将Master节点JDK安装包解压到/opt/module路径中(若路径不存在，则需新建)，将JDK解压命令复制并粘贴至客户端桌面【Release\任务A提交结果.docx】中对应的任务序号下；
2. 修改容器中/etc/profile文件，设置JDK环境变量并使其生效，配置完毕后在Master节点分别执行“java -version”和“javac”命令，将命令行执行结果分别截图并粘贴至客户端桌面【Release\任务A提交结果.docx】中对应的任务序号下；
3. 请完成host相关配置，将三个节点分别命名为master、slave1、slave2，并做免密登录，用scp命令并使用绝对路径从master复制JDK解压后的安装文件到slave1、slave2节点（若路径不存在，则需新建），并配置slave1、slave2相关环境变量，将全部scp复制JDK的命令复制并粘贴至客户端桌面【Release\任务A提交结果.docx】中对应的任务序号下；
4. 在master将Hadoop解压到/opt/module(若路径不存在，则需新建)目录下，并将解压包分发至slave1、slave2中，其中master、slave1、slave2节点均作为datanode，配置好相关环境，初始化Hadoop环境namenode，将初始化命令及初始化结果截图（截取初始化结果日志最后20行即可）粘贴至客户端桌面【Release\任务A提交结果.docx】中对应的任务序号下；
5. 启动Hadoop集群（包括hdfs和yarn），使用jps命令查看master节点与slave1节点的Java进程，将jps命令与结果截图粘贴至客户端桌面【Release\任务A提交结果.docx】中对应的任务序号下。

### 子任务二：Flume安装配置

本任务需要使用root用户完成相关配置，已安装Hadoop及需要配置前置环境，具体要求如下：

1. 从宿主机/opt目录下将文件apache-flume-1.9.0-bin.tar.gz复制到容器master中的/opt/software路径中（若路径不存在，则需新建），将master节点Flume安装包解压到/opt/module目录下，将解压命令复制并粘贴至客户端桌面【Release\任务A提交结果.docx】中对应的任务序号下；
2. 完善相关配置设置，配置Flume环境变量，并使环境变量生效，执行命令flume-ng version并将命令与结果截图粘贴至客户端桌面【Release\任务A提交结果.docx】中对应的任务序号下；
3. 启动Flume传输Hadoop日志（namenode或datanode日志），查看HDFS中/tmp/flume目录下生成的内容，将查看命令及结果（至少5条结果）截图粘贴至客户端桌面【Release\任务A提交结果.docx】中对应的任务序号下。

### 子任务三：ClickHouse单节点安装配置

本任务需要使用root用户完成相关配置，具体要求如下：

1. 从宿主机/opt目录下将clickhouse开头的相关文件复制到容器Master中的/opt/module/clickhouse路径中（若路径不存在，则需新建），将全部解压命令复制并粘贴至客户端桌面【Release\任务A提交结果.docx】中对应的任务序号下；
2. 执行启动各个相关脚本，将全部启动命令复制并将执行结果（截取结果最后倒数15行即可）截图粘贴至客户端桌面【Release\任务A提交结果.docx】中对应的任务序号下；
3. 设置远程访问并移除默认监听文件（listen.xml），同时由于9000端口被Hadoop占用，需要将clickhouse的端口更改为9001，并启动clickhouse，启动后查看clickhouse运行状态，并将启动命令复制、查看运行状态命令复制并将执行结果截图粘贴至客户端桌面【Release\任务A提交结果.docx】中对应的任务序号下。

## 任务B：离线数据处理（25分）

**环境说明：**

|  |
| --- |
| **服务端登录地址详见各任务服务端说明。**  **补充说明：**各节点可通过Asbru工具或SSH客户端进行SSH访问；  主节点MySQL数据库用户名/密码：root/123456（已配置远程连接）；  Hive的配置文件位于/opt/apache-hive-2.3.4-bin/conf/  Spark任务在Yarn上用Client运行，方便观察日志。 |

### 子任务一：数据抽取

编写Scala代码，使用Spark将MySQL的shtd\_store库中表user\_info、sku\_info、base\_province、base\_region、order\_info、order\_detail的数据增量抽取到Hive的ods库中对应表user\_info、sku\_info、base\_province、base\_region、order\_info、order\_detail中。(若ods库中部分表没有数据，正常抽取即可)

1. 抽取shtd\_store库中user\_info的增量数据进入Hive的ods库中表user\_info。根据ods.user\_info表中operate\_time或create\_time作为增量字段(即MySQL中每条数据取这两个时间中较大的那个时间作为增量字段去和ods里的这两个字段中较大的时间进行比较)，只将新增的数据抽入，字段名称、类型不变，同时添加静态分区，分区字段为etl\_date，类型为String，且值为当前比赛日的前一天日期（分区字段格式为yyyyMMdd）。使用hive cli执行show partitions ods.user\_info命令，将结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下；
2. 抽取shtd\_store库中sku\_info的增量数据进入Hive的ods库中表sku\_info。根据ods.sku\_info表中create\_time作为增量字段，只将新增的数据抽入，字段名称、类型不变，同时添加静态分区，分区字段为etl\_date，类型为String，且值为当前比赛日的前一天日期（分区字段格式为yyyyMMdd）。使用hive cli执行show partitions ods.sku\_info命令，将结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下；
3. 抽取shtd\_store库中base\_province的增量数据进入Hive的ods库中表base\_province。根据ods.base\_province表中id作为增量字段，只将新增的数据抽入，字段名称、类型不变并添加字段create\_time取当前时间，同时添加静态分区，分区字段为etl\_date，类型为String，且值为当前比赛日的前一天日期（分区字段格式为yyyyMMdd）。使用hive cli执行show partitions ods.base\_province命令，将结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下；
4. 抽取shtd\_store库中base\_region的增量数据进入Hive的ods库中表base\_region。根据ods.base\_region表中id作为增量字段，只将新增的数据抽入，字段名称、类型不变并添加字段create\_time取当前时间，同时添加静态分区，分区字段为etl\_date，类型为String，且值为当前比赛日的前一天日期（分区字段格式为yyyyMMdd）。使用hive cli执行show partitions ods.base\_region命令，将结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下；
5. 抽取shtd\_store库中order\_info的增量数据进入Hive的ods库中表order\_info，根据ods.order\_info表中operate\_time或create\_time作为增量字段(即MySQL中每条数据取这两个时间中较大的那个时间作为增量字段去和ods里的这两个字段中较大的时间进行比较)，只将新增的数据抽入，字段名称、类型不变，同时添加静态分区，分区字段为etl\_date，类型为String，且值为当前比赛日的前一天日期（分区字段格式为yyyyMMdd）。使用hive cli执行show partitions ods.order\_info命令，将结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下；
6. 抽取shtd\_store库中order\_detail的增量数据进入Hive的ods库中表order\_detail，根据ods.order\_detail表中create\_time作为增量字段，只将新增的数据抽入，字段名称、类型不变，同时添加静态分区，分区字段为etl\_date，类型为String，且值为当前比赛日的前一天日期（分区字段格式为yyyyMMdd）。使用hive cli执行show partitions ods.order\_detail命令，将结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下。

### 子任务二：数据清洗

编写Scala代码，使用Spark将ods库中相应表数据全量抽取到Hive的dwd库中对应表中。表中有涉及到timestamp类型的，均要求按照yyyy-MM-dd HH:mm:ss，不记录毫秒数，若原数据中只有年月日，则在时分秒的位置添加00:00:00，添加之后使其符合yyyy-MM-dd HH:mm:ss。(若dwd库中部分表没有数据，正常抽取即可)

1. 抽取ods库中user\_info表中昨天的分区（子任务一生成的分区）数据，并结合dim\_user\_info最新分区现有的数据，根据id合并数据到dwd库中dim\_user\_info的分区表（合并是指对dwd层数据进行插入或修改，需修改的数据以id为合并字段，根据operate\_time排序取最新的一条），分区字段为etl\_date且值与ods库的相对应表该值相等，同时若operate\_time为空，则用create\_time填充，并添加dwd\_insert\_user、dwd\_insert\_time、dwd\_modify\_user、dwd\_modify\_time四列,其中dwd\_insert\_user、dwd\_modify\_user均填写“user1”。若该条记录第一次进入数仓dwd层则dwd\_insert\_time、dwd\_modify\_time均存当前操作时间，并进行数据类型转换。若该数据在进入dwd层时发生了合并修改，则dwd\_insert\_time时间不变，dwd\_modify\_time存当前操作时间，其余列存最新的值。使用hive cli执行show partitions dwd.dim\_user\_info命令，将结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下；
2. 抽取ods库sku\_info表中昨天的分区（子任务一生成的分区）数据，并结合dim\_sku\_info最新分区现有的数据，根据id合并数据到dwd库中dim\_sku\_info的分区表（合并是指对dwd层数据进行插入或修改，需修改的数据以id为合并字段，根据create\_time排序取最新的一条），分区字段为etl\_date且值与ods库的相对应表该值相等，并添加dwd\_insert\_user、dwd\_insert\_time、dwd\_modify\_user、dwd\_modify\_time四列,其中dwd\_insert\_user、dwd\_modify\_user均填写“user1”。若该条数据第一次进入数仓dwd层则dwd\_insert\_time、dwd\_modify\_time均填写当前操作时间，并进行数据类型转换。若该数据在进入dwd层时发生了合并修改，则dwd\_insert\_time时间不变，dwd\_modify\_time存当前操作时间，其余列存最新的值。使用hive cli查询表dim\_sku\_info的字段id、sku\_desc、dwd\_insert\_user、dwd\_modify\_time、etl\_date，条件为最新分区的数据，id大于等于15且小于等于20，并且按照id升序排序，将结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下；
3. 抽取ods库base\_province表中昨天的分区（子任务一生成的分区）数据，并结合dim\_province最新分区现有的数据，根据id合并数据到dwd库中dim\_province的分区表（合并是指对dwd层数据进行插入或修改，需修改的数据以id为合并字段，根据create\_time排序取最新的一条），分区字段为etl\_date且值与ods库的相对应表该值相等，并添加dwd\_insert\_user、dwd\_insert\_time、dwd\_modify\_user、dwd\_modify\_time四列,其中dwd\_insert\_user、dwd\_modify\_user均填写“user1”。若该条数据第一次进入数仓dwd层则dwd\_insert\_time、dwd\_modify\_time均填写当前操作时间，并进行数据类型转换。若该数据在进入dwd层时发生了合并修改，则dwd\_insert\_time时间不变，dwd\_modify\_time存当前操作时间，其余列存最新的值。使用hive cli在表dwd.dim\_province最新分区中，查询该分区中数据的条数，将结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下；
4. 抽取ods库base\_region表中昨天的分区（子任务一生成的分区）数据，并结合dim\_region最新分区现有的数据，根据id合并数据到dwd库中dim\_region的分区表（合并是指对dwd层数据进行插入或修改，需修改的数据以id为合并字段，根据create\_time排序取最新的一条），分区字段为etl\_date且值与ods库的相对应表该值相等，并添加dwd\_insert\_user、dwd\_insert\_time、dwd\_modify\_user、dwd\_modify\_time四列,其中dwd\_insert\_user、dwd\_modify\_user均填写“user1”。若该条数据第一次进入数仓dwd层则dwd\_insert\_time、dwd\_modify\_time均填写当前操作时间，并进行数据类型转换。若该数据在进入dwd层时发生了合并修改，则dwd\_insert\_time时间不变，dwd\_modify\_time存当前操作时间，其余列存最新的值。使用hive cli在表dwd.dim\_region最新分区中，查询该分区中数据的条数，将结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下；
5. 将ods库中order\_info表昨天的分区（子任务一生成的分区）数据抽取到dwd库中fact\_order\_info的动态分区表，分区字段为etl\_date，类型为String，取create\_time值并将格式转换为yyyyMMdd，同时若operate\_time为空，则用create\_time填充，并添加dwd\_insert\_user、dwd\_insert\_time、dwd\_modify\_user、dwd\_modify\_time四列，其中dwd\_insert\_user、dwd\_modify\_user均填写“user1”，dwd\_insert\_time、dwd\_modify\_time均填写当前操作时间，并进行数据类型转换。使用hive cli执行show partitions dwd.fact\_order\_info命令，将结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下；
6. 将ods库中order\_detail表昨天的分区（子任务一中生成的分区）数据抽取到dwd库中fact\_order\_detail的动态分区表，分区字段为etl\_date，类型为String，取create\_time值并将格式转换为yyyyMMdd，并添加dwd\_insert\_user、dwd\_insert\_time、dwd\_modify\_user、dwd\_modify\_time四列，其中dwd\_insert\_user、dwd\_modify\_user均填写“user1”，dwd\_insert\_time、dwd\_modify\_time均填写当前操作时间，并进行数据类型转换。使用hive cli执行show partitions dwd.fact\_order\_detail命令，将结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下。

### 子任务三：指标计算

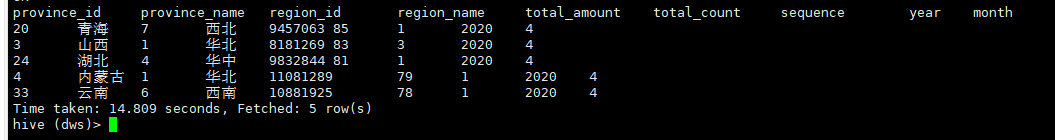
编写Scala代码，使用Spark计算相关指标。

**注：在指标计算中，不考虑订单信息表中order\_status字段的值，将所有订单视为有效订单。计算订单金额或订单总金额时只使用final\_total\_amount字段。需注意dwd所有的维表取最新的分区。**

1. 本任务基于以下2、3、4小题完成（不包括5，6题），使用Azkaban完成第2、3、4题任务代码的调度。工作流要求，使用shell输出“开始”作为工作流的第一个job（job1），2、3、4题任务为串行任务且它们依赖job1的完成（命名为job2、job3、job4），job2、job3、job4完成之后使用shell输出“结束”作为工作流的最后一个job（endjob），endjob依赖job2、job3、job4，并将最终任务调度完成后的工作流截图，将截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下；
2. 根据dwd层表统计每个省每月下单的数量和下单的总金额，并按照year，month，region\_id进行分组,按照total\_amount降序排序，形成sequence值，将计算结果存入Hive的dws数据库的province\_consumption\_day\_aggr表中（表结构如下），然后使用hive cli根据订单总数、订单总金额、省份表主键均为降序排序，查询出前5条，在查询时对于订单总金额字段将其转为bigint类型（避免用科学计数法展示），将SQL语句复制粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下，将执行结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下;

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 中文含义 | 备注 |
| province\_id | int | 省份表主键 |  |
| province\_name | string | 省份名称 |  |
| region\_id | int | 地区表主键 |  |
| region\_name | string | 地区名称 |  |
| total\_amount | double | 订单总金额 | 当月订单总金额 |
| total\_count | int | 订单总数 | 当月订单总数 |
| sequence | int | 次序 |  |
| year | int | 年 | 订单产生的年,为动态分区字段 |
| month | int | 月 | 订单产生的月,为动态分区字段 |

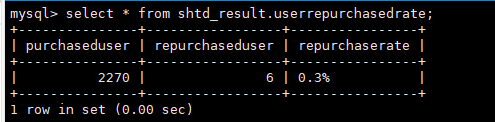
select province\_id,province\_name,region\_id,region\_name,cast(total\_amount as bigint) as total\_amount,total\_count,sequence,year,month from dws.province\_consumption\_day\_aggr order by total\_count desc, total\_amount desc, province\_id desc limit 5;



1. 根据dwd层的数据，请计算连续两天下单的用户与已下单用户的占比，将结果存入MySQL数据库shtd\_result的userrepurchasedrate表中(表结构如下)，然后在Linux的MySQL命令行中查询结果数据，将SQL语句复制粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下，将执行结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下；

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 中文含义 | 备注 |
| purchaseduser | int | 下单人数 | 已下单人数 |
| repurchaseduser | int | 连续下单人数 | 连续两天下单的人数 |
| repurchaserate | text | 百占比 | 连续两天下单人数/已下单人数百分比（保留1位小数，四舍五入，不足的补0）例如21.1%，或者32.0% |

select \* from shtd\_result.userrepurchasedrate;



1. 根据dwd层的数据，请计算每个省份累计订单量（订单信息表一条算一个记录），然后根据每个省份订单量从高到低排列，将结果打印到控制台（使用spark中的show算子，同时需要显示列名），将执行结果复制并粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下。

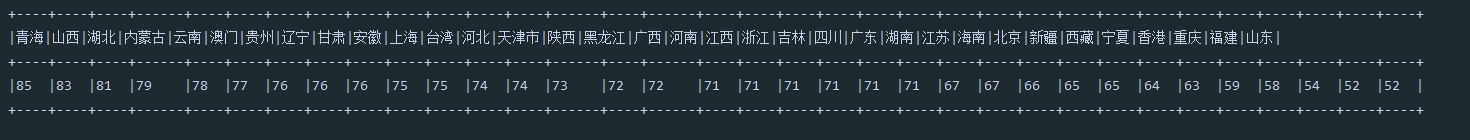
例如：可以考虑首先生成类似的临时表A：

|  |  |
| --- | --- |
| province\_name | Amount（订单量） |
| A省 | 10122 |
| B省 | 301 |
| C省 | 2333333 |

然后生成结果类似如下：其中C省销量最高，排在第一列，A省次之，以此类推。

|  |  |  |
| --- | --- | --- |
| C省 | A省 | B省 |
| 2333333 | 10122 | 301 |

**提示：可用str\_to\_map函数减轻工作量（可按自己需求使用）**



## 任务C：数据挖掘（10分）

**环境说明：**

|  |
| --- |
| **服务端登录地址详见各任务服务端说明。**  **补充说明：**各节点可通过Asbru工具或SSH客户端进行SSH访问；  主节点MySQL数据库用户名/密码：root/123456（已配置远程连接）；  Hive的配置文件位于/opt/apache-hive-2.3.4-bin/conf/  Spark任务在Yarn上用Client运行，方便观察日志。  该任务均使用Scala编写，利用Spark相关库完成。 |

### 子任务一：特征工程

剔除订单信息表与订单详细信息表中用户id与商品id不存在于现有的维表中的记录，同时建议多利用缓存并充分考虑并行度来优化代码，达到更快的计算效果。

1. 据Hive的dwd库中相关表或MySQL数据库shtd\_store中订单相关表（order\_detail、order\_info、sku\_info），对用户购买过的商品进行去重，将其转换为以下格式：第一列为用户id mapping，第二列为用户购买过的商品id mapping，按照user\_id与sku\_id进行升序排序，输出前5行，将结果截图粘贴至客户端桌面【Release\任务C提交结果.docx】中对应的任务序号下；

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 中文含义 | 备注 |
| user\_id | int | 用户id的mapping对应键 |  |
| sku\_id | int | 商品id的mapping对应键 |  |

提示：

Mapping操作：例如用户id：1、4、7、8、9，则做完mapping操作转为字典类型，键0对应用户id 1，键1对应用户id 4，以此类推

结果格式如下：

-------user\_id\_mapping与sku\_id\_mapping数据前5条如下：-------

0:0

0:89

1:1

1:2

1:3

1. 根据第1小题的结果，对其进行聚合，其中对sku\_id进行one-hot转换，将其转换为以下格式矩阵：第一列为用户id，其余列名为商品id，按照用户id进行升序排序，展示矩阵第一行前5列数据，将结果截图粘贴至客户端桌面【Release\任务C提交结果.docx】中对应的任务序号下。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 中文含义 | 备注 |
| user\_id | double | 客户key |  |
| sku\_id0 | double | 用户是否购买过商品1 | 若用户购买过该商品，则值为1，否则为0 |
| sku\_id1 | double | 用户是否购买过商品2 | 若用户购买过该商品，则值为1，否则为0 |
| sku\_id2 | double | 用户是否购买过商品3 | 若用户购买过该商品，则值为1，否则为0 |
| ..... |  |  |  |

结果格式如下：

---------------第一行前5列结果展示为---------------

0.0,1.0,0.0,0.0,0.0

### 子任务二：推荐系统

1. 根据子任务一的结果，对其进行SVD分解，对数据进行降维保留前5个奇异值信息，根据用户id为6708的用户已购买的商品分别与未购买的商品计算余弦相似度再进行累加求均值，将均值最大的5件商品id进行输出作为推荐使用。将输出结果截图粘贴至客户端桌面【Release\任务C提交结果.docx】中对应的任务序号下。

结果格式如下：

------------------------推荐Top5结果如下------------------------

相似度top1(商品id：1，平均相似度：0.983456)

相似度top2(商品id：71，平均相似度：0.782672)

相似度top3(商品id：22，平均相似度：0.7635246)

相似度top4(商品id：351，平均相似度：0.7335748)

相似度top5(商品id：14，平均相似度：0.522356)

## 任务D：数据采集与实时计算（20分）

**环境说明：**

|  |
| --- |
| **服务端登录地址详见各任务服务端说明。**  **补充说明：**各节点可通过Asbru工具或SSH客户端进行SSH访问；  Flink任务在Yarn上用per job模式（即Job分离模式，不采用Session模式），方便Yarn回收资源。 |

### 子任务一：实时数据采集

1. 在主节点使用Flume采集实时数据生成器10050端口的socket数据（实时数据生成器脚本为主节点/data\_log目录下的dj\_data\_gen脚本，该脚本为主节点本地部署且使用socket传输），将数据存入到Kafka的Topic中（Topic名称为order，分区数为4），使用Kafka自带的消费者消费order（Topic）中的数据，将前2条数据的结果截图粘贴至客户端桌面【Release\任务D提交结果.docx】中对应的任务序号下；

**注：需先启动已配置好的Flume再启动脚本，否则脚本将无法成功启动，启动方式为进入/data\_log目录执行./dj\_data\_gen(如果没有权限，请执行授权命令chmod 777 /data\_log/dj\_data\_gen)**

1. 采用多路复用模式，Flume接收数据注入kafka 的同时，将数据备份到HDFS目录/user/test/flumebackup下，将查看备份目录下的第一个文件的前2条数据的命令与结果截图粘贴至客户端桌面【Release\任务D提交结果.docx】中对应的任务序号下。

### 子任务二：使用Flink处理Kafka中的数据

编写Scala代码，使用Flink消费Kafka中Topic为order的数据并进行相应的数据统计计算（订单信息对应表结构order\_info,订单详细信息对应表结构order\_detail（来源类型和来源编号这两个字段不考虑，所以在实时数据中不会出现），同时计算中使用order\_info或order\_detail表中create\_time或operate\_time取两者中值较大者作为EventTime，若operate\_time为空值或无此列，则使用create\_time填充，允许数据延迟5s，订单状态order\_status分别为1001:创建订单、1002:支付订单、1003:取消订单、1004:完成订单、1005:申请退回、1006:退回完成。另外对于数据结果展示时，不要采用例如：1.9786518E7的科学计数法）。

1. 使用Flink消费Kafka中的数据，统计商城实时订单数量（需要考虑订单表的状态，若有取消订单、申请退回、退回完成则不计入订单数量，其他状态则累加），将key设置成totalcount存入Redis中。使用redis cli以get key方式获取totalcount值，将结果截图粘贴至客户端桌面【Release\任务D提交结果.docx】中对应的任务序号下，需两次截图，第一次截图和第二次截图间隔1分钟以上，第一次截图放前面，第二次截图放后面；
2. 在任务1进行的同时，使用侧边流，使用Flink消费Kafka中的订单详细信息数据，实时统计商城中消费额前3的商品（不考虑订单状态，不考虑打折），将key设置成top3itemconsumption存入Redis中（value使用String数据格式，value为前3的商品信息并外层用[]包裹，其中按排序依次存放商品id:销售额，并用逗号分割）。使用redis cli以get key方式获取top3itemconsumption值，将结果截图粘贴至客户端桌面【Release\任务D提交结果.docx】中对应的任务序号下，需两次截图，第一次截图和第二次截图间隔1分钟以上，第一次截图放前面，第二次截图放后面；

示例如下：

top3itemconsumption：[1:10020.2,42:4540.0,12:540]

1. 采用双流JOIN的方式（本系统稳定，无需担心数据迟到与丢失的问题,建议使用滚动窗口），结合订单信息和订单详细信息（需要考虑订单状态，若有取消订单、申请退回、退回完成则不进行统计），拼接成如下表所示格式，其中包含订单id、订单总金额、商品数，将数据存入HBase数据库(namespace)shtd\_result的的orderpositiveaggr表中（表结构如下），然后在Linux的HBase shell命令行中查询出任意5条数据，查询列orderprice、orderdetailcount，将执行结果截图粘贴至客户端桌面【Release\任务D提交结果.docx】中对应的任务序号下，将执行结果截图粘贴至客户端桌面【Release\任务D提交结果.docx】中对应的任务序号下。

表空间为：shtd\_result，表为orderpositiveaggr，列族为：info

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 中文含义 | 备注 |
| rowkey | string | HBase的主键 | 值为id |
| id | bigint | 订单id |  |
| orderprice | string | 订单总金额 | 统计订单信息中 final\_total\_amount字段 |
| orderdetailcount | string | 商品数量总计 | 统计订单详细信息中 sku\_num字段 |

## 任务E：数据可视化（15分）

**环境说明：**

|  |
| --- |
| **数据接口地址及接口描述详见各任务服务端说明。** |

### 子任务一：用柱状图展示各地区消费额的中位数

编写Vue工程代码，根据接口，用柱状图展示2020年各地区所有订单消费额的中位数，同时将用于图表展示的数据结构在浏览器的console中进行打印输出，将图表可视化结果和浏览器console打印结果分别截图并粘贴至客户端桌面【Release\任务E提交结果.docx】中对应的任务序号下。

### 子任务二：用饼状图展示各地区的平均消费能力

编写Vue工程代码，根据接口，用饼状图展示2020年各地区订单的平均消费额，同时将用于图表展示的数据结构在浏览器的console中进行打印输出，将图表可视化结果和浏览器console打印结果分别截图并粘贴至客户端桌面【Release\任务E提交结果.docx】中对应的任务序号下。

### 子任务三：用折线图展示每年上架商品数量的变化

编写Vue工程代码，根据接口，用折线图展示每年上架商品数量的变化情况，同时将用于图表展示的数据结构在浏览器的console中进行打印输出，将图表可视化结果和浏览器console打印结果分别截图并粘贴至客户端桌面【Release\任务E提交结果.docx】中对应的任务序号下。

### 子任务四：用条形图展示平均消费额最高的地区

编写Vue工程代码，根据接口，用条形图展示2020年平均消费额（四舍五入保留两位小数）最高的5个地区，同时将用于图表展示的数据结构在浏览器的console中进行打印输出，将图表可视化结果和浏览器console打印结果分别截图并粘贴至客户端桌面【Release\任务E提交结果.docx】中对应的任务序号下。

### 子任务五：用散点图展示省份平均消费额

编写Vue工程代码，根据接口，用基础散点图展示2020年最高10个省份平均消费额（四舍五入保留两位小数），同时将用于图表展示的数据结构在浏览器的console中进行打印输出，将图表可视化结果和浏览器console打印结果分别截图并粘贴至客户端桌面【Release\任务E提交结果.docx】中对应的任务序号下。

## 任务F：综合分析（10分）

### 子任务一：Flume采集数据会导致数据丢失吗？请简述其原理。

请简述Flume数据采集是否会导致数据丢失以及其原理，将内容编写至客户端桌面【Release\任务F提交结果.docx】中对应的任务序号下。

### 子任务二：Flink 任务出现很高的延迟，你会如何入手解决类似问题？

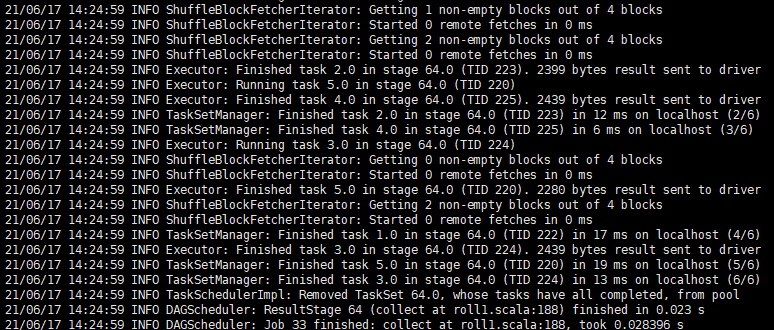
Flink 任务出现很高的延迟，你会如何入手解决类似问题，将内容编写至客户端桌面【Release\任务F提交结果.docx】中对应的任务序号下。

### 子任务三：Flink的反压机制

在使用Flink进行实时计算时通常出现反压问题，为确定具体问题所在，我们通常需要查看Flink的Web界面的Metrics菜单栏，其中可以添加关注的度量（指标）来确定是上游还是下游出现反压，我们所添加的度量有哪些。将内容编写至客户端桌面【Release\任务F提交结果.docx】中对应的任务序号下。

## 附录：补充说明

### 命令行截图样例（保证清晰）



### 表结构说明

#### 订单信息表（order\_info）

|  |  |  |
| --- | --- | --- |
| 标签 | 中文含义 | 备注 |
| id | 主键 |  |
| consignee | 收货人 |  |
| consignee\_tel | 收件人电话 |  |
| final\_total\_amount | 总金额 |  |
| order\_status | 订单状态 |  |
| user\_id | 用户id | 对应用户表id |
| delivery\_address | 送货地址 |  |
| order\_comment | 订单备注 |  |
| out\_trade\_no | 订单交易编号（第三方支付用） |  |
| trade\_body | 订单描述（第三方支付用） |  |
| create\_time | 创建时间 |  |
| operate\_time | 操作时间 |  |
| expire\_time | 失效时间 |  |
| tracking\_no | 物流单编号 |  |
| parent\_order\_id | 父订单编号 |  |
| img\_url | 图片路径 |  |
| province\_id | 省份id | 对应省份表id |
| benefit\_reduce\_amount | 优惠金额 |  |
| original\_total\_amount | 原价金额 |  |
| feight\_fee | 运费 |  |

#### 订单详细信息表（order\_detail）

|  |  |  |
| --- | --- | --- |
| 标签 | 中文含义 | 备注 |
| id | 主键 |  |
| order\_id | 订单编号 | 对应订单信息表id |
| sku\_id | 商品id | 对应商品表id |
| sku\_name | 商品名称 |  |
| img\_url | 图片路径 |  |
| order\_price | 购买价格 | 下单时的商品价格 |
| sku\_num | 购买数量 |  |
| create\_time | 创建时间 |  |
| source\_type | 来源类型 | 无实际意义 |
| source\_id | 来源编号 | 无实际意义 |

#### 1.3 商品表（sku\_info）

|  |  |  |
| --- | --- | --- |
| 标签 | 中文含义 | 备注 |
| id | 主键 |  |
| spu\_id | spuId |  |
| price | 价格 |  |
| sku\_name | 商品名称 |  |
| sku\_desc | 商品描述 |  |
| weight | 重量 |  |
| tm\_id | 品牌id |  |
| category3\_id | 三级分类id |  |
| sku\_default\_img | 默认显示图片 |  |
| create\_time | 创建时间 |  |

#### 1.4 用户表（user\_info）

|  |  |  |
| --- | --- | --- |
| 标签 | 中文含义 | 备注 |
| id | 主键 |  |
| login\_name | 用户名 |  |
| nick\_name | 用户昵称 |  |
| passwd | 用户密码 |  |
| name | 姓名 |  |
| phone\_num | 手机号 |  |
| email | 邮箱 |  |
| head\_img | 头像 |  |
| user\_level | 用户级别 |  |
| birthday | 生日 |  |
| gender | 性别M男,F女 |  |
| create\_time | 创建时间 |  |
| operate\_time | 操作时间 |  |

#### 1.5 省份表（base\_province）

|  |  |  |
| --- | --- | --- |
| 标签 | 中文含义 | 备注 |
| id | 主键 |  |
| name | 省份名称 |  |
| region\_id | 地区id | 对应地区表id |
| area\_code | 地区编码 |  |
| iso\_code | 国际编码 |  |

#### 1.6 地区表（base\_region）

|  |  |  |
| --- | --- | --- |
| 标签 | 中文含义 | 备注 |
| id | 主键 |  |
| region\_name | 地区名称 |  |