## 模块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模块服务端说明。**  **补充说明：**各主机可通过Asbru工具或SSH客户端进行SSH访问；  请先检查ZooKeeper、Kafka、Redis端口是否已启动，若未启动则各启动命令如下：  ZK启动（netstat -ntlp查看2181端口是否打开）  /usr/zk/zookeeper-3.4.6/bin/zkServer.sh start  Redis启动（netstat -ntlp查看6379端口是否打开）  /usr/redis/bin/redis-server /usr/redis/bin/redis.conf  Kafka启动（netstat -ntlp查看9092端口是否打开）  /opt/kafka/kafka\_2.11-2.0.0/bin/kafka-server-start.sh -daemon（空格连接下一行）/opt/kafka/kafka\_2.11-2.0.0/config/server.properties  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。 |

### 任务一：实时数据采集

1. 在Master节点使用Flume采集实时数据生成器10050端口的socket数据，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对应报告中；
2. 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对应报告中。

### 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），实时计算中使用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为空值或无此属性，则使用create\_time填充，允许数据延迟5S，订单状态分别为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，统计商城实时订单实收金额（需要考虑订单状态，若有取消订单、申请退回、退回完成则不计入订单实收金额，其他状态的则累加），将key设置成totalprice存入Redis中。使用redis cli以get key方式获取totalprice值，将结果截图粘贴至对应报告中，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，监控若发现order\_status字段为退回完成, 将key设置成totalrefundordercount存入Redis中，value存放用户退款消费额。使用redis cli以get key方式获取totalrefundordercount值，将结果截图粘贴至对应报告中，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；
3. 在任务1进行的同时，使用侧边流，监控若发现order\_status字段为取消订单,将数据存入MySQL数据库shtd\_result的order\_info表中，然后在Linux的MySQL命令行中根据id降序排序，查询列id、consignee、consignee\_tel、final\_total\_amount、feight\_fee，查询出前5条，将SQL语句与执行结果截图粘贴至对应报告中。
4. 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，将结果截图粘贴至对应报告中，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；
5. 在任务4进行的同时，使用侧边流，统计每分钟申请退回订单的数量，将key设置成refundcountminute存入Redis中。使用redis cli以get key方式获取refundcountminute值，将结果截图粘贴至对应报告中，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；
6. 在任务4进行的同时，使用侧边流，计算每分钟内状态为取消订单占所有订单的占比，将key设置成cancelrate存入Redis中，value存放取消订单的占比（为百分比，保留百分比后的一位小数，四舍五入，例如12.1%）。使用redis cli以get key方式获取cancelrate值，将结果截图粘贴至对应报告中，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。
7. 使用Flink消费Kafka中的数据，实时统计商城中消费额前2的用户（需要考虑订单状态，若有取消订单、申请退回、退回完成则不计入订单消费额，其他的相加），将key设置成top2userconsumption存入Redis中（value使用String数据格式，value为前2的用户信息并且外层用[]包裹，其中按排序依次存放为该用户id:用户名称:消费总额，用逗号分割，其中用户名称为user\_info表中的name字段，可从MySQL中获取）。使用redis cli以get key方式获取top2userconsumption值，将结果截图粘贴至对应报告中，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（如有中文，需在redis-cli中展示中文）；

示例如下：

top2userconsumption：[1:张三:10020,42:李四:4540]

1. 在任务7进行的同时，使用侧边流，使用Flink消费Kafka中的订单详细信息的数据，实时统计商城中销售额前3的商品（不考虑订单状态，不考虑打折，销售额为order\_price\*sku\_num），将key设置成top3itemconsumption存入Redis中（value使用String数据格式，value为前3的商品信息并且外层用[]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，将结果截图粘贴至对应报告中，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MySQL数据库shtd\_result的orderpostiveaggr表中（表结构如下），然后在Linux的MySQL命令行中根据id降序排序，查询出前5条，将SQL语句与执行结果截图粘贴至对应报告中。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 中文含义 | 备注 |
| id | int | 订单id |  |
| orderprice | double | 订单总金额 | 统计订单信息中 final\_total\_amount字段 |
| orderdetailcount | int | 商品数量总计 | 累加订单详细信息中 sku\_num字段 |