

**大数据处理实验报告**

实验四：大数据实时分析

专业班级： 计算机2011班

学 号： U202010755

姓 名： 路昊东

指导教师： 郑渤龙

报告日期： 2022年 4 月 7 日

**计算机科学与技术学院**

**《大数据处理》课程实验报告**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 实验地点 | | 南一楼804 | 课程名称 | | 大数据处理 | | |
| 实验题目 | | 大数据实时分析 | 成绩 |  | | 指导教师 |  |
| 教师评价 | □ 实验过程正确； □ 源程序/实验内容提交； □ 程序结构/实验步骤合理；  □ 实验结果正确； □ 语法、语义/命令正确； □ 报告规范；  其他： | | | | | | |
| **一、实验目的**   1. 了解大数据实时分析的用途 2. 掌握大数据实时分析的基本命令   **二、实验内容**   1. 实验环境配置 2. Python脚本生成测试数据 (20’) 3. 配置Kafka (10’) 4. 安装Flume客户端 (10’) 5. 配置Flume采集数据 (20’) 6. MySQL中准备结果表与维度表数据 (10’) 7. 使用DLI中的Flink作业进行数据分析 (20’) 8. 资源释放 9. 实验总结 （10’）   **三、实验环境**  **一. 华为云环境：**  1. 区域：“华北-北京四”  2. MRS集群：  （1）名称：mrs\_dong  （2）集群版本：1.9.2  （3）类型：流式集群  （4）组件：Kafka、KafkaManager、Storm、Flume  （5）计费模式：按需计费  （6）可用区：可用区1  （7）弹性公网IP绑定：124.70.44.145  （8）CPU架构：鲲鹏计算（集群高可用关闭；core节点数：1）  （9）委托、数据盘加密默认，告警“关闭”  （10）kerberos认证：关闭  （11）登录方式：密码  （12）master节点绑定IP：192.168.0.83  （13）虚拟私有云：vpc-dong  （14）子网：subnet-dong(192.168.0.0/24)  （15）安全组：mrs\_mrs\_dong\_fVmq (入方向: TCP/9022，9022，9022，9022，9022，9022，9022，9022，9022，9022，9022，9022，9022 | 出方向: --)  3. 虚拟私有云：默认创建: vpc-dong  4. 弹性公网 IP：  （1）计费模式：按需计费  （2）线路：全动态 BGP  （3）公网带宽：按流量计费  （4）带宽大小：50M  （5）购买数量：2  5. 云数据库服务 RDS：  （1）计费模式：按需计费  （2）区域：北京四  （3）实例名称：rds-dong  （4）数据库引擎：MySQL  （5）数据库版本：5.7  （6）实例类型：单机  （7）可用区：可用区一  （8）性能规格：鲲鹏通用增强型|2vCPUs|4GB,最大连接数:1200,TPS/QPS:340|7100  （9）存储类型：SSD云盘  （10）存储空间：40 GB  （11）时区：UTC+08:00  （12）磁盘加密：不加密  （13）虚拟私有云：vpc-dong  （14）子网：subnet-dong(192.168.0.0/24)  （15）内网地址：192.168.0.198  （16）安全组：mrs\_mrs\_dong\_fVmq (入方向: TCP/9022，9022，9022，9022，9022，9022，9022，9022，9022，9022，9022，9022，9022 | 出方向: --)  （17）数据库端口默认端口：3306  （18）参数模板：Default-MySQL-5.7  6. 数据湖探索服务（DLI）：  （1）队列区域：北京四  （2）队列名称：queue\_dong  （3）队列类型：通用队列  （4）队列专属资源模式：是  （5）队列AZ策略：单AZ  （6）队列CPU架构：鲲鹏  （7）队列规格：16 CUs  （8）队列计费模式：按需计费  （9）服务授权：VPC Administrator  （10）跨源连接名称：DLI\_RDS  （11）跨源连接绑定队列：queue\_dong  （12）虚拟私有云：与 MRS 服务相同  （13）子网：与 MRS 服务相同  7. 数据迁移服务（CDM）：  （1）区域：北京四  （2）集群名称：cdm-dong  （3）版本：2.9.1.100  （4）实例类型：cdm.large  （5）CPU：8 vCPUs  （6）内存：16 GB  （7）可用区：可用区1  （8）安全组：与 MRS 服务相同  （9）虚拟私有云：与 MRS 服务相同  （10）子网：与 MRS 服务相同  8. 数据可视化服务（DLV）：开通试用  **二.实验的本地软硬件环境如下：**  1.CPU型号：AMD Ryzen 7 4800H  2.CPU主频：2.9GHz  3.核心/线程数：八核心/十六线程  4.内存容量：16GB（8GB×2）  5.操作系统：ubuntu20.04 & windows10  **四、实验过程或步骤（源程序）**  下面是本次实验的具体内容及步骤、实验的详细记录、实验结果分析：  **1. 实验环境配置**  （1）服务购买：  ①登录控制台  ②申请弹性公网 IP  ③开通 MapReduce 服务  ④开通云数据库服务 RDS  ⑤开通数据湖探索服务（DLI）  ⑥开通数据迁移服务（CDM）  ⑦开通数据可视化服务（DLV）  **2.Python 脚本生成测试数据**  **（1）登录 MRS 的 master 节点服务器**  **①** 打开**ubuntu**终端，使用**ssh**登录服务器，命令行输入：  **ssh root@124.70.44.145**  **②** 终端截图如下：    **（2）编写 Python 脚本**  **①** 进入**/opt/client/**目录，使用 **vi** 命令编写 **Python** 脚本：**autodatagen.py**  命令行输入：  **cd /opt/client/**  **vim autodatagen.py**  **②** 终端截图如下：    脚本代码为：  **#coding:utf-8**  **###########################################**  **# rowkey：随机的两位数 + 当前时间戳，并要确保该 rowkey 在表数据中唯一。**  **# 列定义：行健，用户名，年龄，性别，商品 ID，价格，门店 ID，购物行为，电话，邮箱，购买日期**  **# 421564974572,Sgxrp,20,woman,152121,297.64,313015,scan,15516056688,JbwLDQmzwP@qq.com,2019-08-01**  **# 601564974572,Lbeuo,43,man,220902,533.13,313016,pv,15368953106,ezfrJSluoR@163.com,2019-08-05**  **###########################################**  **import random**  **import string**  **import sys**  **import time**  **# 大小写字母**  **alphabet\_upper\_list = string.ascii\_uppercase**  **alphabet\_lower\_list = string.ascii\_lowercase**  **# 随机生成指定位数的字符串**  **def get\_random(instr, length):**  **# 从指定序列中随机获取指定长度的片段并组成数组，例如:['a', 't','f', 'v', 'y']**  **res = random.sample(instr, length)**  **# 将数组内的元素组成字符串**  **result = ''.join(res)**  **return result**  **# 放置生成的并且不存在的 rowkey**  **rowkey\_tmp\_list = []**  **# 制作 rowkey**  **def get\_random\_rowkey():**  **import time**  **pre\_rowkey = ""**  **while True:**  **# 获取 00~99 的两位数字，包含 00 与 99**  **num = random.randint(00, 99)**  **# 获取当前 10 位的时间戳**  **timestamp = int(time.time())**  **# str(num).zfill(2)为字符串不满足 2 位，自动将该字符串补 0**  **pre\_rowkey = str(num).zfill(2) + str(timestamp)**  **if pre\_rowkey not in rowkey\_tmp\_list:**  **rowkey\_tmp\_list.append(pre\_rowkey)**  **break**  **return pre\_rowkey**  **# 创建用户名**  **def get\_random\_name(length):**  **name = string.capwords(get\_random(alphabet\_lower\_list, length))**  **return name**  **# 获取年龄**  **def get\_random\_age():**  **return str(random.randint(18, 60))**  **# 获取性别**  **def get\_random\_sex():**  **return random.choice(["woman", "man"])**  **# 获取商品 ID**  **def get\_random\_goods\_no():**  **goods\_no\_list = ["220902","430031","550012","650012","532120","230121","250983","480071","580016","950013","152121","230121"]**  **return random.choice(goods\_no\_list)**  **# 获取商品价格（浮点型）**  **def get\_random\_goods\_price():**  **# 随机生成商品价格的整数位，1~999 的三位数字，包含 1 与 999**  **price\_int = random.randint(1, 999)**  **# 随机生成商品价格的小数位，1~99 的两位数字，包含 1 与 99**  **price\_decimal = random.randint(1, 99)**  **goods\_price = str(price\_int) + "." + str(price\_decimal)**  **return goods\_price**  **# 获取门店 ID**  **def get\_random\_store\_id():**  **store\_id\_list = ["313012", "313013", "313014","313015","313016","313017","313018","313019","313020","313021","313022", "313023"]**  **return random.choice(store\_id\_list)**  **# 获取购物行为类型**  **def get\_random\_goods\_type():**  **goods\_type\_list = ["pv", "buy", "cart", "fav","scan"] #点击、购买、加购、收藏、浏览**  **return random.choice(goods\_type\_list)**  **# 获取电话号码**  **def get\_random\_tel():**  **pre\_list = ["130", "131", "132", "133", "134", "135", "136", "137", "138", "139", "147", "150", "151", "152", "153", "155", "156", "157", "158", "159", "186", "187", "188"]**  **return random.choice(pre\_list) + ''.join(random.sample('0123456789', 8))**  **# 获取邮箱名**  **def get\_random\_email(length):**  **alphabet\_list = alphabet\_lower\_list + alphabet\_upper\_list**  **email\_list = ["163.com", "126.com", "qq.com", "gmail.com", "huawei.com"]**  **return get\_random(alphabet\_list, length) + "@" + random.choice(email\_list)**  **# 获取商品购买日期（统计最近 7 天数据）**  **def get\_random\_buy\_time():**  **buy\_time\_list = ["2019-08-01", "2019-08-02", "2019-08-03", "2019-08-04", "2019-08-05", "2019-08-06", "2019-08-07"]**  **return random.choice(buy\_time\_list)**  **# 生成一条数据**  **def get\_random\_record():**  **return get\_random\_rowkey() + "," + get\_random\_name(5) + "," + get\_random\_age() + "," + get\_random\_sex() + "," + get\_random\_goods\_no() + ","+get\_random\_goods\_price() + "," + get\_random\_store\_id()+","+get\_random\_goods\_type() +","+get\_random\_tel() + "," + get\_random\_email(10) + "," +get\_random\_buy\_time()**  **# 获取随机整数用于休眠**  **def get\_random\_sleep\_time():**  **return random.randint(5, 10)**  **# 将记录写到文本中**  **def write\_record\_to\_file():**  **# 覆盖文件内容，重新写入**  **f = open(sys.argv[1], 'w')**  **i = 0**  **while i < int(sys.argv[2]):**  **record = get\_random\_record()**  **f.write(record)**  **f.write('\n')**  **i += 1**  **f.close()**  **if \_\_name\_\_ == "\_\_main\_\_":**  **write\_record\_to\_file()**  输入 **:wq** 保存并退出 **vim** 编辑器。脚本编辑截图如下：    **（3）创建存放测试数据的目录**  **①** 使用 **mkdir** 命令在**/tmp** 下创建目录 **flume\_spooldir**，我们把 **Python** 脚本模拟生成的数据放到此目录下，后面 **Flume** 就监控这个文件下的目录，以读取数据。命令行输入：  **mkdir /tmp/flume\_spooldir**  **②**终端截图如下：    **（4）执行脚本测试**  **①** 执行 **Python** 命令，测试生成 **100** 条数据,命令行输入：  **python autodatagen.py "/tmp/flume\_spooldir/test.txt" 100**  **②** 使用 **more** 命令查看生成的数据（**enter** 键向下一行，**space** 键向下一屏，**b** 向  上一屏，**q** 键退出）。命令行输入：  **more /tmp/flume\_spooldir/test.txt、**  此处使用**more**指令。此命令类似于 **cat** 命令，但显示时会以页的形式进行翻屏方便阅读。按空白键（**space**）向下翻一页，按b键往回（**back**）翻一页。其中more语句有一些参数：比如**-num**表示一次要显示的行数；**-f**表示以文件实际的行数计算行数， 而不是自动换行过后的行数；**-p**表示先清除屏幕再显示每一页内容， 而不是以卷动的方式显示；**-s**表示遇到有连续两行以上的空白行时替换为一行的空白行等等。  **②** 终端截图如下：        **3. 配置Kafka**  **（1）设置环境变量**  **①** 使用**ssh**登录**MRS**的**master**节点服务器后，首先使用**source**命令进行环境  变量的设置使得相关命令可用。命令行输入：  **source /opt/client/bigdata\_env**  此处使用**source**指令。使 shell 读入指定的文件并依次执行文件中的所有语句。 shell 通常用“.”代替，被称为点命令。通常用于重新执行刚修改的初始化文件，使之立即生效，而无需注销并重新登录。  **②**终端截图如下：    **（2）在kafka中创建topic**  **①** 首先在控制台中获取**Zookeeper**的**ip**，如图所示：    **②** 命令行输入：  **/opt/client/Kafka/kafka/bin/kafka-topics.sh --create --zookeeper 192.168.0.83:2181/kafka --partitions 1 --replication-factor 1 --topic fludesc**  **②**终端截图如下：    此处使用**kafka**的指令。在单节点配置中，已知 ZooKeeper 和代理 id 实例。**Kafka**提供了一个名为**kafka-topics.sh**的命令行实用程序，用于在服务器上创建主题我们刚刚创建了一个名为**fludesc**的主题，其中包含一个分区和一个副本因子。  创建主题后，我们可以在**Kafka**代理终端窗口中获取通知，并在**config / server.properties** 文件中的“**/ tmp / kafka-logs /**"中指定的创建主题的日志。  **（3）查看topic信息**  **①** 命令行输入：  **/opt/client/Kafka/kafka/bin/kafka-topics.sh --list --zookeeper 192.168.0.83:2181/kafka**  **②**终端截图如下：    此处指令意为获取 **Kafka**服务器中的主题列表。  由于我们已经创建过主题，它将仅列出创建的主题名，其中就有之前创建的**fludesc**。  此外**Kafka**指令还有很多。比如启动生产者以发送消息：**bin/kafka-console- producer.sh --broker-list localhost:9092 --topic topic-name；**  启动消费者以接收消息：**bin/kafka-console-consumer.sh --zookeeper localhost:2181 —topic topic-name --from-beginning**  还可以启动Zookeeper服务器进行单节点多代理配置。  **4. 安装Flume客户端**  **（1）进入MRS Manager集群管理**  **①** 在**mrs\_dong**集群页面，点击“前往**Manager**”，如图所示：    **②** 输入用户名和密码，点击“登录”，进入到**MRS Manager**界面。  变  **（2）下载Flume客户端**  **①** 在**MRS Manager**集群管理界面，点击“**服务管理**”，点击“**Flume**”，进入**Flume**服务，点击“**下载客户端**”按钮。点击“**确定**”后等待下载。          下载完成后会有弹出框提示下载到哪一台服务器上（这台机器就是**master**节点），路径就是**/tmp/MRS-client**。  **（2）校验下载的客户端文件包**  **①** 使用ssh登录到master节点服务器上，进入/tmp/MRS-client目录，命令行输入：  **cd/tmp/MRS-client/**  **ll**  **②** 终端截图如下：    **总用量 546180**  **-rw-------. 1 omm wheel 559288320 4月 6 23:18 MRS\_Flume\_Client.tar**  这里使用cd命令进入相应目录，使用ll指令列出目录下信息。  **③** 执行以下命令，解压压缩包获取校验文件与客户端配置包  **Tar -xvf MRS\_Flume\_Client.tar**  这里使用**tar**命令外加参数**-xvf**，其中tar指令用于备份文件，可以建立或者还原备份文件。扩展名为.tar的文件通常称为tar包。其中常用选项： **-A** 或**--catenate** 新增文件到已存在的备份文件； **-c** 或**--create** 创建新的备份文件； **-C<目录>** 解压到指定的目录；**-f<备份文件>** 指定备份文件**；-r** 新增文件到已存在的备份文件的结尾部分；**-t** 列出备份文件的内容；**-v** 显示指令执行过程；**-x** 从备份文件中还原文件；**-z** 通过 gzip 指令处理备份文件。  **④** 终端截图如下：  、  这里可见压缩包解压之后有两个文件。  **⑤** 执行命令，校验文件包。  **sha256sum -c MRS\_Flume\_ClientConfig.tar.sha256**  **⑥** 终端截图如下：    **MRS\_Flume\_ClientConfig.tar: 成功**  界面显示如上信息，表明文件包校验成功。  **（2）安装Flume运行环境**  **①** 解压“**MRS\_Flume\_ClientConfig.tar**”文件，然后查看解压文件。命令行输入：  **tar-xvf MRS\_Flume\_ClientConfig.tar**  **ll**  **②** 终端截图如下：      **总用量 1092356**  **drwx------. 4 root root 340 4月 6 23:18 MRS\_Flume\_ClientConfig**  **-rw-------. 1 root root 559278080 4月 6 23:18 MRS\_Flume\_ClientConfig.tar**  **-rw-------. 1 root root 92 4月 6 23:18 MRS\_Flume\_ClientConfig.tar.sha256**  **-rw-------. 1 omm wheel 559288320 4月 6 23:18 MRS\_Flume\_Client.tar**  **③** 安装客户端运行环境到目录“/opt/Flume\_env”（安装时自动生成目录），命令行输入：  **sh /tmp/MRS-client/MRS\_Flume\_ClientConfig/install.sh /opt/Flume\_env**  **④** 终端截图如下：      **[22-04-06 23:22:52]: JDK installation is complete.**  **[22-04-06 23:22:52]: Components client installation is complete.**  界面显示如上信息，表明表示客户端运行环境安装成功。  **⑤** 执行命令配置环境变量，命令行输入：  **source /opt/Flume\_env/bigdata\_env**  **⑥** 终端截图如下：    **（3）安装Flume客户端**  **①** 执行命令解压Flume客户端文件。命令行输入：  **cd /tmp/MRS-client/MRS\_Flume\_ClientConfig/Flume**  **ll**  **tar-xvf FusionInsight-Flume-1.6.0.tar.gz**  **②** 终端截图如下：      **③** 安装**Flume**到目录“**/opt/FlumeClient**”（**安装时自动生成目录，-d：表示**  **Flume客户端安装路径**），命令行输入如下：  **sh /tmp/MRS-client/MRS\_Flume\_ClientConfig/Flume/install.sh -d /opt/FlumeClient**  **④** 终端截图如下：  **CST 2022-04-06 23:26:36 [flume-client install]: install flume client successfully.**  系统显示以上结果表示客户端运行环境安装成功。  **（4）重启Flume服务**  **①** 执行以下命令重启Flume的服务。命令行输入：  **cd/opt/FlumeClient/fusioninsight-flume-1.6.0**  **sh bin/flume-manage.sh restart**  **②** 终端截图如下：    **Stop Flume PID=13321 successful.**  **Start flume successfully,pid=17413.**  服务重启成功，安装结束！  **5. 配置Flume采集数据**  **（1）修改配置文件**  **①** 进入Flume安装目录，在conf目录下编辑文件properties.properties，命令行输入：  **cd /opt/FlumeClient/fusioninsight-flume-1.6.0/**  **ll**  **vi conf/properties.properties**  **②** 终端截图如下：    **③** 找到**kafka**的**broker**的**业务IP**，如图所示：    **④** 根据上面获得的IP，在vi打开的文件**conf/properties.properties中**加入如下内容:  **client.sources = s1**  **client.channels = c1**  **client.sinks = sh1**  **# the source configuration of s1**  **client.sources.s1.type = spooldir**  **client.sources.s1.spoolDir = /tmp/flume\_spooldir**  **client.sources.s1.fileSuffix = .COMPLETED**  **client.sources.s1.deletePolicy = never**  **client.sources.s1.trackerDir = .flumespool**  **client.sources.s1.ignorePattern = ^$**  **client.sources.s1.batchSize = 1000**  **client.sources.s1.inputCharset = UTF-8**  **client.sources.s1.deserializer = LINE**  **client.sources.s1.selector.type = replicating**  **client.sources.s1.fileHeaderKey = file**  **client.sources.s1.fileHeader = false**  **client.sources.s1.basenameHeader = true**  **client.sources.s1.basenameHeaderKey = basename**  **client.sources.s1.deserializer.maxBatchLine = 1**  **client.sources.s1.deserializer.maxLineLength = 2048**  **client.sources.s1.channels = c1**  **# the channel configuration of c1**  **client.channels.c1.type = memory**  **client.channels.c1.capacity = 10000**  **client.channels.c1.transactionCapacity = 1000**  **client.channels.c1.channlefullcount = 10**  **client.channels.c1.keep-alive = 3**  **client.channels.c1.byteCapacityBufferPercentage = 20**  **# the sink configuration of sh1**  **client.sinks.sh1.type = org.apache.flume.sink.kafka.KafkaSink**  **client.sinks.sh1.kafka.topic = fludesc**  **client.sinks.sh1.flumeBatchSize = 1000**  **client.sinks.sh1.kafka.producer.type = sync**  **client.sinks.sh1.kafka.bootstrap.servers = 192.168.0.118:9092**  **client.sinks.sh1.kafka.security.protocol = PLAINTEXT**  **client.sinks.sh1.requiredAcks = 0**  **client.sinks.sh1.channel = c1**  如下图所示：    之后使用**:wq**保存。  **（2）创建消费者消费kafka中的数据**  **①** 使用ssh登录master节点后，执行命令（此处bootstrap-server的ip对  应的是Kafka的Broker的IP）：  **kafka-console-consumer.sh --topic fludesc --bootstrap-server 192.168.0.118:9092 --new-consumer --consumer.config /opt/client/Kafka/kafka/config/consumer.properties**  **②** 终端截图如下:    **③** 终端中使用**ctrl+shift+t**新开一个终端会话窗口，然后ssh登录节点，命令行输入：  **ssh root@124.70.44.145**  **④** 终端截图如下：    输入密码重新进入master节点  **⑤** 进入Python脚本所在目录，执行python脚本，再生成一份数据，命令行输入：  **cd /opt/client/**  **python autodatagen.py "/tmp/flume\_spooldir/test.txt" 100**  **⑥** 终端截图如下：    **⑦** 查看原窗口，可以看到已经消费出了数据：    有数据产生，表明**Flume**到**Kafka**目前是打通的。  **⑧** 测试完毕，在新打开的窗口输入**exit**关闭窗口，在原窗口输入**Ctrl+c**退出  进程，如下所示：    **[root@node-master1fxOt client]# exit**  **注销**  **Connection to 124.70.44.145 closed.**    **^CProcessed a total of 100 messages**  **[root@node-master1fxOt fusioninsight-flume-1.6.0]#**  **6. MySQL中准备结果表与维度表数据**  **（1）登录MySQL的DAS管理平台**  **①** 在控制台进入到云数据库RDS实例管理界面，点击实例后面的“**登录**”按钮。  输入用户名**root**和密码，勾选“**记住密码**”，开启“**定时采集**”和“**SQL执行记录**”，  然后点击“**测试连接**”，成功后点击“**登录**”按钮。截图如下：      **（2）创建数据库**  **①** 点击“**新建数据库**”，输入名称“**rds\_desc**”，字符集选择**utf8**，点击“确定”。      **（3）进入SQL界面**  **①** 点击数据库后面的“SQL查询”，进入到SQL执行界面。    **（4）创建维度表并插入数据**  **①** 清除查询器中原有内容，复制下面的SQL语句粘贴到SQL查询中：  **-- ----------------------------**  **-- 创建商品信息纬度表**  **-- ----------------------------**  **DROP TABLE IF EXISTS `desc\_goods\_info`;**  **CREATE TABLE `desc\_goods\_info` (**  **`goods\_no` varchar(30) NOT NULL,**  **`goods\_name` varchar(30) DEFAULT NULL**  **) ENGINE=InnoDB DEFAULT CHARSET=utf8;**  **-- ----------------------------**  **-- 插入商品信息样例数据**  **-- ----------------------------**  **INSERT INTO `desc\_goods\_info` VALUES ('220902', '杭州丝绸');**  **INSERT INTO `desc\_goods\_info` VALUES ('430031', '西湖龙井');**  **INSERT INTO `desc\_goods\_info` VALUES ('550012', '西湖莼菜');**  **INSERT INTO `desc\_goods\_info` VALUES ('650012', '张小泉剪刀');**  **INSERT INTO `desc\_goods\_info` VALUES ('532120', '塘栖枇杷');**  **INSERT INTO `desc\_goods\_info` VALUES ('230121', '临安山核桃');**  **INSERT INTO `desc\_goods\_info` VALUES ('250983', '西湖藕粉');**  **INSERT INTO `desc\_goods\_info` VALUES ('480071', '千岛湖鱼干');**  **INSERT INTO `desc\_goods\_info` VALUES ('580016', '天尊贡芽');**  **INSERT INTO `desc\_goods\_info` VALUES ('950013', '叫花童鸡');**  **INSERT INTO `desc\_goods\_info` VALUES ('152121', '火腿蚕豆');**  **INSERT INTO `desc\_goods\_info` VALUES ('230121', '杭州百鸟朝凤');**  **-- ----------------------------**  **-- 创建门店信息纬度表**  **-- ----------------------------**  **DROP TABLE IF EXISTS `desc\_store\_info`;**  **CREATE TABLE `desc\_store\_info` (**  **`store\_id` varchar(50) NOT NULL,**  **`store\_name` varchar(50) DEFAULT NULL**  **) ENGINE=InnoDB DEFAULT CHARSET=utf8;**  **-- ----------------------------**  **-- 插入门店信息样例数据**  **-- ----------------------------**  **INSERT INTO `desc\_store\_info` VALUES ('313012', '莫干山店');**  **INSERT INTO `desc\_store\_info` VALUES ('313013', '定安路店');**  **INSERT INTO `desc\_store\_info` VALUES ('313014', '西湖银泰店');**  **INSERT INTO `desc\_store\_info` VALUES ('313015', '天目山店');**  **INSERT INTO `desc\_store\_info` VALUES ('313016', '凤起路店');**  **INSERT INTO `desc\_store\_info` VALUES ('313017', '南山路店');**  **INSERT INTO `desc\_store\_info` VALUES ('313018', '西溪湿地店');**  **INSERT INTO `desc\_store\_info` VALUES ('313019', '传媒学院店');**  **INSERT INTO `desc\_store\_info` VALUES ('313020', '西湖断桥店');**  **INSERT INTO `desc\_store\_info` VALUES ('313021', '保淑塔店');**  **INSERT INTO `desc\_store\_info` VALUES ('313022', '南宋御街店');**  **INSERT INTO `desc\_store\_info` VALUES ('313023', '河坊街店');**  这里创建两张表**desc\_goods\_info**和**desc\_store\_info**，并向其中分别插入了内容。  **②** 点击“**执行SQL**”执行上面的语句，执行成功后可以在下面看到执行消息：      执行成功后可以在下面看到执行记录和执行消息，如下图所示：    **（5）创建Flink作业的结果表**  **①** Flink执行的结果数据是存储在MySQL中，现在需要提前把表创建好。  步骤同上，清除SQL窗口中原有代码（若不清除原有代码则需要选中本次要  执行的代码后再执行），粘贴并执行下面建表语句：  **-- ----------------------------**  **-- 创建商品总销售额表--goods\_amount\_count**  **-- ----------------------------**  **DROP TABLE IF EXISTS `goods\_amount\_count`;**  **CREATE TABLE `goods\_amount\_count` (**  **`amount\_total` float NOT NULL,**  **`sale\_date` date PRIMARY KEY**  **) ENGINE=InnoDB DEFAULT CHARSET=utf8;**  **-- ----------------------------**  **-- 创建销售总额前 5 的门店排行表--amount\_store\_rank**  **-- ----------------------------**  **DROP TABLE IF EXISTS `amount\_store\_rank`;**  **CREATE TABLE `amount\_store\_rank` (**  **`store\_id` int PRIMARY KEY,**  **`store\_name` varchar(50) DEFAULT NULL,**  **`amount\_total` float DEFAULT NULL**  **) ENGINE=InnoDB DEFAULT CHARSET=utf8;**    执行成功后，点击**左侧的刷新按钮**可以看到已经创建的表：    这里可以发现数据库中先后一共创建了四张表，其中后创建的两张表：`goods\_amount\_count`和 `amount\_store\_rank’没有插入内容。  **7. 使用DLI中的Flink作业进行数据分析**  **（1）创建Flink作业**  **①** 进入数据湖探索服务（DLI）的控制台，点击左侧“作业管理”菜单，选择  “Flink作业”，然后点击右上角的“创建作业”。截图如下：    类型选择“**Flink SQL**”，名称输入“**desc\_order\_count**”（可以自定义），模板、  标签默认，点击“确定”。    **（2）编辑Flink作业的SQL脚本**  **①** 首先查看MRS集群中**Kafka**的**Broker**的**IP**和端口号、**kafka**中创建的主题名称、MySQL的登录密码和内网IP、服务所在的区。  **②** 复制下面的脚本到编辑框中：  **/\*\***  **\* 该示例为 CS 服务使用对等连接对接用户在华为云自有的 Kafka 集群的SQL 模板。数据的输入源为用户的 Kafka 集群提供**  **\* >>>>>>>>>请务必确保您的 Kafka 集群中已创建了您配置的 Kafka 主题<<<<<<<<<<**  **\***  **\* >>>>>样例输入<<<<<**  **\* 流名:**  **\*kafka\_source(id,user\_name,age,gender,goods\_no,goods\_price,store\_id,shopping\_type,tel,email,shopping\_date):**  **\*行健，用户名，年龄，性别，商品 ID，价格，门店 ID，购物行为，电话，邮箱，购买日期**  **\*591564404906,Gkisf,59,man,550012,23.45,423012,fav,15274039826,UsPZNdxnJX@163.c\*om,2019-07-01**  **\*761564404906,Wupsa,55,man,250983,25.05,423013,scan,13730586914,AKVMdPveQX@gmai\*l.com,2019-07-02**  **\*\*/**  **/\*\*kafka 输入流\*\*/**  **CREATE SOURCE STREAM kafka\_source (**  **id STRING,**  **use\_rname STRING,**  **age int,**  **gender STRING,**  **goods\_no STRING,**  **goods\_price Float,**  **store\_id int,**  **shopping\_type STRING,**  **tel STRING,**  **email STRING,**  **shopping\_date Date**  **) WITH (**  **type = "kafka",**  **kafka\_bootstrap\_servers = "192.168.0.118:9092",**  **/\*\* kafka 的连接端口，需要确保能连通（可以使用对等连接的方式打通 CS 集群和 kafka 集群）\*\*/**  **kafka\_group\_id = "sourcegroup1",**  **/\*\* group id\*\*/**  **kafka\_topic = "fludesc",**  **/\*\*读取的 topic\*\*/**  **encode = "csv",**  **/\*\* 解码格式，支持 json/csv\*\*/**  **field\_delimiter = ","**  **);**  **/\***  **\* >>>>>1-商品总的销售额（按天统计）数据输出<<<<<**  **\* 流名: goods\_amount\_count**  **\***  **\*\*/**  **CREATE SINK STREAM goods\_amount\_count(**  **amount\_total Float,**  **/\* 销售额\*/**  **sale\_date date**  **/\* 销售日期\*/**  **) WITH (**  **type = "rds",**  **username = "root",**  **/\*MySQL 的用户名默认是 root \*/**  **password = "LHD20020516lhd-odin",**  **/\*创建 MySQL 实例设置的密码\*/**  **db\_url = "mysql://192.168.0.198:3306/rds\_desc",**  **/\*此处 IP 改成 MySQL 的内网 IP，端口号不需要改，默认就是 3306，IP 可以在 MySQL 页面查看\*/**  **table\_name = "goods\_amount\_count",**  **primary\_key = "sale\_date"**  **);**  **/\***  **\* >>>>>2-销售总额前 5 的门店排行数据输出<<<<<**  **\* 流名: amount\_store\_rank**  **\***  **\*\*/**  **CREATE SINK STREAM amount\_store\_rank(**  **store\_id int,**  **/\* 门店 ID\*/**  **store\_name STRING,**  **/\* 门店名称\*/**  **amount\_total Float**  **/\* 销售额\*/**  **) WITH (**  **type = "rds",**  **username = "root",**  **/\*MySQL 的用户名默认是 root \*/**  **password = "LHD20020516lhd-odin",**  **/\*创建 MySQL 实例设置的密码\*/**  **db\_url = "mysql://192.168.0.198:3306/rds\_desc",**  **/\*此处 IP 改成 MySQL 的内网 IP，端口号不需要改，默认就是 3306，IP 可以在 MySQL 页面查看\*/**  **table\_name = "amount\_store\_rank",**  **primary\_key = "store\_id"**  **);**  **/\***  **创建数据维表 1，用于和输入流连接，实现字段回填**  **本实验中用到的是关联维度表门店信息表，根据流中的门店 id,关联出门店**  **的名称**  **\*\*/**  **CREATE TABLE table\_store\_info (store\_id STRING, store\_name STRING) WITH (**  **type = "rds",**  **region = "cn-north-4",**  **/\* 服 务 所 在 的 区 域 ， 例 如 华 北 - 北 京 四 为 ：cn-north-4\*/**  **username = "root",**  **/\*MySQL 的用户名默认是 root \*/**  **password = "LHD20020516lhd-odin",**  **/\*创建 MySQL 实例设置的密码\*/**  **db\_url = "mysql://192.168.0.198:3306/rds\_desc",**  **/\*此处 IP 改成 MySQL的内网 IP，端口号不需要改，默认就是 3306，IP 可以在 MySQL 页面查看\*/**  **table\_name = "desc\_store\_info"**  **/\* 关联的维度表名称\*/**  **);**  **/\***  **创建数据维表 2，用于和输入流连接，实现字段回填**  **本实验中用到的是关联维度表商品信息表，根据流中的商品 id,关联出商品**  **的名称**  **\*\*/**  **CREATE TABLE table\_goods\_info (goods\_no STRING, goods\_name STRING) WITH (**  **type = "rds",**  **region = "cn-north-4",**  **/\* 服 务 所 在 的 区 域 ， 例 如 华 北 - 北 京 四 为 ：cn-north-4\*/**  **username = "root",**  **/\*MySQL 的用户名默认是 root \*/**  **password = "LHD20020516lhd-odin",**  **/\*创建 MySQL 实例设置的密码\*/**  **db\_url = "mysql://192.168.0.198:3306/rds\_desc",**  **/\*此处 IP 改成 MySQL的内网 IP，端口号不需要改，默认就是 3306，IP 可以在 MySQL 页面查看\*/**  **table\_name = "desc\_goods\_info"**  **/\* 关联的维度表名称\*/**  **);**  **/\* 1-商品总的销售额数据插入 \*/**  **INSERT INTO**  **goods\_amount\_count**  **SELECT**  **sum(goods\_price) as amount\_total,**  **shopping\_date as sale\_date**  **FROM**  **kafka\_source**  **WHERE**  **shopping\_type = 'buy'**  **group BY**  **shopping\_date;**  **/\* 2-销售总额前 5 的门店排行数据插入 \*/**  **INSERT INTO**  **amount\_store\_rank**  **SELECT**  **t1.store\_id,**  **t2.store\_name,**  **sum(t1.goods\_price) as amount\_total**  **FROM**  **kafka\_source t1**  **left join table\_store\_info as t2 on t1.store\_id = t2.store\_id**  **WHERE**  **t1.shopping\_type = 'buy'**  **group BY**  **t1.store\_id,**  **t2.store\_name;**  编辑完后点击“语义校验”，如果校验无错误则进行下一步骤操作，如果有误  则根据报错信息对SQL脚本进行修改后重新校验。      **（3）测试网络连通性**  **①** 测试DLI与Kafka网络是否连通，输入kafka\_bootstrap\_servers地址，测试连通性。      **（4）配置Flink作业的集群信息**  **①** 进入Flink作业，选择“运行参数”，设置CU数量为2，选择所属队列“queue\_flink”（必选，为开通DLI服务创建的队列），其他默认即可：    **（5）运行Flink作业**  **①** 点击右上角的“启动”，在启动Flink作业页面点击右下角的“立即启动”，回到Flink作业界面，状态变为“提交中”：      **（6）查看作业运行详情**  **①** 点击作业管理中的作业名称desc\_order\_count可以进入作业详情页，当作业变为“运行中”时可以进行下一步骤操作。（作业提交需要一定的时间，显示“运行中”后就可以查看监控及运行日志）：（注：日志中如果出现error字样，则运行错误，请按照对应错误检查相关设置，网络设置、包括安全组是否设置正确，SQL脚本数据库连接是否拼写正确，数据库中是否已经建好相应的数据库和表等问题。）    任务列表正确显示，截图如下：    运行日志正确显示，截图如下：    **（7）验证数据分析**  **①** 使用ssh登录到MRS的master节点后，进入Python脚本所在目录，再次执行Python脚本生成测试数据，命令行输入：  **cd /opt/client/**  **python autodatagen.py "/tmp/flume\_spooldir/test.txt" 100**  **②** 终端截图如下：    **③** 回到Flink作业管理界面，点击“作业监控”按钮，进入作业监控页面后，通过指标图表可以看到数据正常处理（各指标图表顺序可以调整）：    **④** 登录MySQL后点击数据库的名称或后面的“**库管理**”进入到库管理界面，可以看到结果表中有数据进来：    **⑤** 点击表名后面的“SQL查询”可以看到表中的数据：    依次使用sql语句对四张表查询如下：      **⑥** 至此，整个数据处理流程完成。下面步骤将对数据进行可视化：  **（注：由于之前漏掉可视化任务提前释放了资源，本步骤以及定时操作步骤均为重新配置资源在windows系统下操作，其中python脚本命名为autodata.py，master节点配置为master1QXfW ）**   1. 首先进入Flink作业，在作业管理中配置好相关问题，之后配好cdm集群相关。 2. 然后打开数据可视化DLV界面，在“管理中心”中打开“管理vpc连接”，创建相应的连接； 3. 在“我的数据”中配置好与云数据库MySQL的数据连接 4. 在“我的大屏”中创建新大屏，这里使用组件“键值表格”进行可视化，点击组件，在“数据”一栏中，选择“静态数据”一栏中的“云数据库MySQL”下的数据库资源，输入select查询语句，更改键值映射之后，即可在大屏上显示四张数据表格的相关内容。   **部分过程截图以及四张表的可视化显示如下图所示：**  -ea74efdd54df57a  43cf157a2577c30  **显示成功，证明可视化成功实现！**  **⑦** 待可视化完成后，我们新增定时任务，让生成测试数据的Python脚本每隔10秒钟运行一次，在可视化页面上就可以看到统计数据在不断的变化。 **1.** 这里我们使用了crontab定时任务，命令行输入：  **crontab -e**  **2.** 在打开的编辑页面中输入： **\* \* \* \* \* python /opt/client/ autodata.py "/tmp/flume\_spooldir/test.txt" 100 \* \* \* \* \* sleep 10; python /opt/client/ autodata.py "/tmp/flume\_spooldir/test.txt" 100 \* \* \* \* \* sleep 20; python /opt/client/ autodata.py "/tmp/flume\_spooldir/test.txt" 100 \* \* \* \* \* sleep 30; python /opt/client/ autodata.py "/tmp/flume\_spooldir/test.txt" 100 \* \* \* \* \* sleep 40; python /opt/client/ autodata.py "/tmp/flume\_spooldir/test.txt" 100 \* \* \* \* \* sleep 50; python /opt/client/ autodata.py "/tmp/flume\_spooldir/test.txt" 100**  **3.** 使用**:wq**保存退出，然后查看脚本内容如下：  **crontab  -l**  **4.** 截图如下所示：（这里使用windows系统，使用putty登录节点服务器）  15868ce274f14fb6  IMG_256  上图是有关**crontab**的执行原理图，根据上图，在linux中 crontab的最小执行单位是分钟，没法直接实现单位秒的运行，所以得通过其他方式来处理。故每10秒运行一次，那就运行一次后睡眠10秒，10秒后再睡眠10秒，依次类推。  **5.** 除了在可视化界面上看到数据不断变化之外，还可以在Flink作业监控中看到相关变化，如下图所示：  19c968fe6fa3a681  7f6498e33450ce5a  -5b6b64338310d4e  至此，定时显示数据变化的功能成功实现！  **五、出现的问题与解决方案**  本次实验的主要问题出现在数据可视化的配置上。由于没有教程属于自主探索，于是在华为云上搜索到有关配置文档，尤其是新建数据连接的时候，总是找不到数据库，最终经过多次尝试，创建了相应的私有云，解决了数据连接问题。后续创建大屏的时候哦，由于映射关系出错，也是认真地检查了一些时间。  **六、实验总结**  本次实验给予了我们一个接触数据分析的良机。无论是对于kafka的使用、对于Flume的初次尝试、创建实例配合MySQL进行存储、使用DLI中的Flink作业进行数据分析都使我对大数据处理这一新领域有了更细致更具体的理解和体会。  结合之前三次实验、老师上课时的讲解以及自己的了解，我再次深刻体会到了数据，以及数据分析的重要程度。就以其中的决策为例，通常我们做决策都是靠过往的经验，常说的一句话是”这样搞应该没问题“，但没有数据支撑的往往不够精准，大概率会出问题，所以我们要从“经验决策”到“实时数据驱动决策”，让一切行动以数据为事实。而具体来讲，总结起来其实就是三点：  1：现状分析，其实就是告诉我们过去发生了什么；  2：原因分析，可以说是告诉我们为什么这些现状会发生；在对第一阶段的现状进行分析之后，我们对数据对象的基本情况有了基本的了解，但是我们不知道哪方面更好，差异是什么，以及原因是什么。这时，我们需要进行原因分析，以进一步确定业务变更的具体原因；  3：预测分析，这就是告诉我们未来会发生什么。  而实时分析，当然是数据分析中的重中之重。这里实时分析系统是最常见的一种场景。通过搜集资料我了解到，传统上，常常使用数据仓库来承担这一任务，数据仓库通过创建索引来应对多维度复杂查询。传统数据仓库也存在很明显的缺点，扩展性不强，索引创建成本高，索引易失效等等。当查询条件复杂时，传统领域和hadoop目前都没有一个特别好的解决方案。维度如果不固定，就无法创建索引或者索引代价太高，通常只能通过全盘暴力SCAN的方法来解决。  目前来完美解决实时分析的系统还在探索中，而经过这段时间的学习，我们发现hadoop领域中的确有一些很好的解决方案：hive、hbase、impala等等。逐一分析的话，hive就是一个数据仓库工具，可以将结构化的数据文件映射为一张数据库表，并提供完整的sql查询功能，可以将sql语句转换为MapReduce任务进行运行。Hive支持HSQL，是一种类SQL，当然也正是由于这种机制导致Hive最大的缺点是慢。而HBase则是一个分布式的、面向列的开源数据库，不同于一般的关系数据库，它是一个适合于非结构化数据存储的数据库。核心是将数据抽象成表，表中只有rowkey和column family。Rowkey是记录的主键，通过key /value很容易找到。Colum family中存储实际的数据。仅能通过主键(row key)和主键的range来检索数据，仅支持单行事务(可通过hive支持来实现多表join等复杂操作)。关于impala，现阶段还没涉猎，后续会细致了解这个SQL大数据查询工具。  有实时分析当然有实时计算。查询资料我了解到：实时计算一般都是针对海量数据进行的，并且要求为秒级。由于大数据兴起之初，Hadoop并没有给出实时计算解决方案，随后Storm，SparkStreaming，Flink等实时计算框架应运而生，而Kafka，ES的兴起使得实时计算领域的技术越来越完善，而随着物联网，机器学习等技术的推广，实时流式计算将在这些领域得到充分的应用。  经过这次实验和课余的学习，我了解到一个实时计算的架构：  我们当然知道，数据从Web平台中产生，通过数据同步系统，导入到大数据平台，由于数据源不同，这里的数据同步系统实际上是多个相关系统的组合。数据库同步通常用 Sqoop，日志同步可以选择 Flume等，不同的数据源产生的数据质量可能差别很大，数据库中的格式化数据直接导入大数据系统即可，而日志和爬虫产生的数据就需要进行大量的清洗、转化处理才能有效使用。关于内部的数据存储，该层对原始数据、清洗关联后的明细数据进行存储，基于统一的实时数据模型分层理念，将不同应用场景的数据分别存储在 Kafka、HDFS、Kudu、 Clickhouse、Hbase等存储中。关于数据计算，主要使用 Flink、Spark、Presto 以及 ClickHouse 自带的计算能力等四种计算引擎，Flink 计算引擎主要用于实时数据同步、 流式 ETL、关键系统秒级实时指标计算场景，Spark SQL 主要用于复杂多维分析的准实时指标计算需求场景，Presto 和 ClickHouse 主要满足多维自助分析、对查询响应时间要求不太高的场景。关于实时应用——以统一查询服务对各个业务线数据场景进行支持，业务主要包括实时大屏、实时数据产品、实时 OLAP、实时特征等。  综合来讲，这四次大数据实验让我由浅入深对大数据处理的一些宏观概念得到了具体的理解，也有了一定程度上的实践，仍然十分感谢老师提供的机会和工具，之后的我会继续学习有关实现原理，争取得到更多的提高。 | | | | | | | |