

大数据处理技术

课程设计报告

|  |  |
| --- | --- |
| 设计题目 | 大数据实时分析处理 |
| 专 业 班 级 | 计算机创新实验18-1班 |
| 学生姓名及学号 | 2018213106 刘嘉伟 |
| 任 课 教 师 | 吴共庆老师 |
| 实验指导教师 | 吴共庆老师 |
| 实验地点 | 图书馆 |
| 2020 ~2021 学年第一学期 | |

[一、背景 3](#_Toc58096121)

[二、大数据处理一般流程 4](#_Toc58096122)

[1、大数据采集 4](#_Toc58096123)

[2、大数据预处理 5](#_Toc58096124)

[3、大数据存储 5](#_Toc58096125)

[4、大数据分析 6](#_Toc58096126)

[三、实时分析系统架构 8](#_Toc58096127)

[四、架构选型 9](#_Toc58096128)

[1、Flume集群 9](#_Toc58096129)

[2、Kafuka集群 11](#_Toc58096130)

[3、Flink集群 13](#_Toc58096131)

[五、系统设计流程 15](#_Toc58096132)

[1、商品交易数据的获取 15](#_Toc58096133)

[2、配置Kafka 18](#_Toc58096134)

[3、安装Flume 19](#_Toc58096135)

[4、打通Kafka和Flume 20](#_Toc58096136)

[5、MySQL中准备结果表与维度表数据 21](#_Toc58096137)

[6、创建Flink作业打通Mysql与Kafka 24](#_Toc58096138)

[7、数据可视化 28](#_Toc58096139)

[8、定时生成数据 30](#_Toc58096140)

[六、实验总结 31](#_Toc58096141)

# 一、背景

在互联网/移动互联网、物联网等应用场景中，个性化服务、用户体验提升、智能分析、事中决策等复杂的业务需求对大数据处理技术提出了更高的要求。为了满足这些需求，大数据处理系统必须在毫秒级甚至微秒级的时间内返回处理结果。以国内最大的银行卡收单机构银联商务为例，其日交易量近亿笔，需对旗下540多万个商户进行实时风险监控，在确保这些商户合规开展收单业务的同时，最大限度地保障个人用户的合法权益。这样的高并发、大数据、高实时应用需求给大数据处理系统提出了严峻的挑战。

银联商务以前使用的T+1事后风控系统存在风险侦测迟滞高(次日才能发现风险，损害已经造成)、处理时间长(十几个小时之后才能完成风险识别)、无法处理长周期历史数据(只能分析最近几日的流水数据)以及无法支持复杂规则(仅能支持累积求和等简单规则)等重大缺陷。为此，亟须研发全新的事中风控系统，以重点实现低迟滞(在1 min内甄别突发风险)、高实时(100 ms内返回处理结果)、长周期(可处理长达10年以上的历史周期数据)以及支持高复杂度规则(如方差、标准差、K阶中心矩、最大连续统计等)等目标。这一目标可以抽象为一个大数据处理科学问题：如何在一个完整的大数据集上，实现低迟滞、高实时的即席(Ad-Hoc)查询分析处理。

# 二、大数据处理一般流程

简单来说，从大数据的生命周期来看，无外乎四个方面：**大数据采集、大数据预处理、大数据存储、大数据分析**，共同组成了大数据生命周期里最核心的技术。

互联网

结构化数据

持久化存储

感兴趣的数据

分析与展示

## 1、大数据采集

对于各种来源的数据，包括移动互联网数据、社交网络的数据等，这些结构化和非结构化的海量数据是零散的，也就是所谓的数据孤岛，此时的这些数据并没有什么意义，数据采集就是将这些数据写入数据仓库中，把零散的数据整合在一起，对这些数据综合起来进行分析。数据采集包括文件日志的采集、数据库日志的采集、关系型数据库的接入和应用程序的接入等。在数据量比较小的时候，可以写个定时的脚本将日志写入存储系统，但随着数据量的增长，这些方法无法提供数据安全保障，并且运维困难，需要更强壮的解决方案。

**数据库采集：**流行的有Sqoop和ETL，传统的关系型数据库MySQL和Oracle 也依然充当着许多企业的数据存储方式。当然了，目前对于开源的Kettle和Talend本身，也集成了大数据集成内容，可实现HDFS，Hbase和主流Nosql数据库之间的数据同步和集成。

**网络数据采集：**一种借助网络爬虫或网站公开API，从网页获取非结构化或半结构化数据，并将其统一结构化为本地数据的数据采集方式。

**文件采集：**包括实时文件采集和处理技术flume、基于ELK的日志采集和增量采集等等。

## 2、大数据预处理

大数据预处理，指的是在进行数据分析之前，先对采集到的原始数据所进行的诸如“清洗、填补、平滑、合并、规格化、一致性检验”等一系列操作，旨在提高数据质量，为后期分析工作奠定基础。数据预处理主要包括四个部分：数据清理、数据集成、数据转换、数据规约。

**数据清理：**指利用ETL等清洗工具，对有遗漏数据(缺少感兴趣的属性)、噪音数据(数据中存在着错误、或偏离期望值的数据)、不一致数据进行处理。

**数据集成：**是指将不同数据源中的数据，合并存放到统一数据库的，存储方法，着重解决三个问题：模式匹配、数据冗余、数据值冲突检测与处理。

**数据转换：**是指对所抽取出来的数据中存在的不一致，进行处理的过程。它同时包含了数据清洗的工作，即根据业务规则对异常数据进行清洗，以保证后续分析结果准确性。

**数据规约：**是指在最大限度保持数据原貌的基础上，最大限度精简数据量，以得到较小数据集的操作，包括：数据方聚集、维规约、数据压缩、数值规约、概念分层等。

## 3、大数据存储

大数据存储，指用存储器，以数据库的形式，存储采集到的数据的过程，包含三种典型路线：

**1、基于MPP架构的新型数据库集群**

采用Shared Nothing架构，结合MPP架构的高效分布式计算模式，通过列存储、粗粒度索引等多项大数据处理技术，重点面向行业大数据所展开的数据存储方式。具有低成本、高性能、高扩展性等特点，在企业分析类应用领域有着广泛的应用。

较之传统数据库，其基于MPP产品的PB级数据分析能力，有着显著的优越性。自然，MPP数据库，也成为了企业新一代数据仓库的最佳选择。

**2、基于Hadoop的技术扩展和封装**

基于Hadoop的技术扩展和封装，是针对传统关系型数据库难以处理的数据和场景（针对非结构化数据的存储和计算等），利用Hadoop开源优势及相关特性（善于处理非结构、半结构化数据、复杂的ETL流程、复杂的数据挖掘和计算模型等），衍生出相关大数据技术的过程。

伴随着技术进步，其应用场景也将逐步扩大，目前最为典型的应用场景：通过扩展和封装 Hadoop来实现对互联网大数据存储、分析的支撑，其中涉及了几十种NoSQL技术。

**3、大数据一体机**

这是一种专为大数据的分析处理而设计的软、硬件结合的产品。它由一组集成的服务器、存储设备、操作系统、数据库管理系统，以及为数据查询、处理、分析而预安装和优化的软件组成，具有良好的稳定性和纵向扩展性。

## 4、大数据分析

从可视化分析、数据挖掘算法、预测性分析、语义引擎、数据质量管理等方面，对杂乱无章的数据，进行萃取、提炼和分析的过程。

**1、可视化分析**

可视化分析，指借助图形化手段，清晰并有效传达与沟通信息的分析手段。主要应用于海量数据关联分析，即借助可视化数据分析平台，对分散异构数据进行关联分析，并做出完整分析图表的过程。具有简单明了、清晰直观、易于接受的特点。

**2、数据挖掘算法**

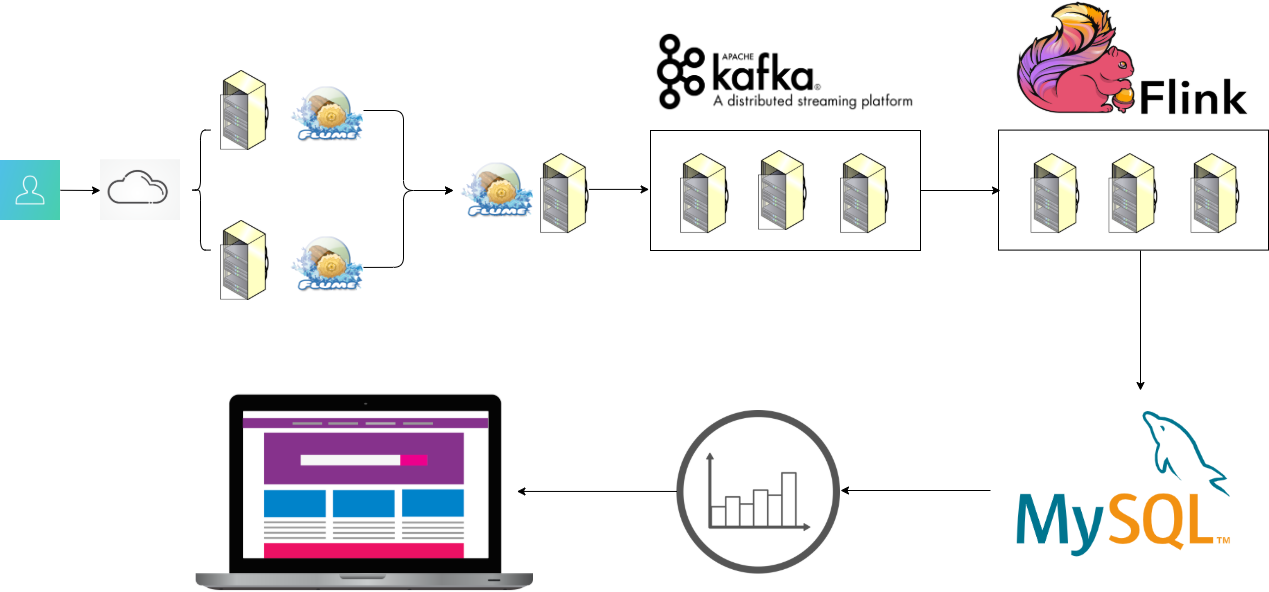
数据挖掘算法，即通过创建数据挖掘模型，而对数据进行试探和计算的，数据分析手段。它是大数据分析的理论核心。

数据挖掘算法多种多样，且不同算法因基于不同的数据类型和格式，会呈现出不同的数据特点。但一般来讲，创建模型的过程却是相似的，即首先分析用户提供的数据，然后针对特定类型的模式和趋势进行查找，并用分析结果定义创建挖掘模型的最佳参数，并将这些参数应用于整个数据集，以提取可行模式和详细统计信息。

**3、预测性分析**

预测性分析，是大数据分析最重要的应用领域之一，通过结合多种高级分析功能（特别统计分析、预测建模、数据挖掘、文本分析、实体分析、优化、实时评分、机器学习等），达到预测不确定事件的目的。帮助分用户析结构化和非结构化数据中的趋势、模式和关系，并运用这些指标来预测将来事件，为采取措施提供依据。

# 三、实时分析系统架构



**本实时分析系统架构图**

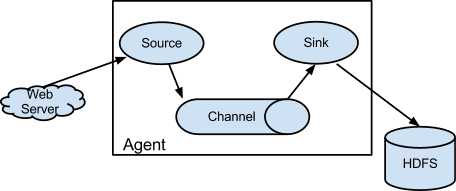
从构建实时处理系统的角度出发，我们需要做的是让数据在各个不同的集群系统之间打通（从上面的图示中也能很好地说明这一点），即需要做各个系统之前的整合，包括Flume与Kafka的整合，Kafka与Flink的整合。各个环境是否使用集群，依个人的实际需要而定，在我们的环境中，Flume、Kafka、Flink都使用集群。

# 四、架构选型

从上面的架构图我们可以看到，系统主要有以下几个部分组成

## 1、Flume集群

Flume是一种分布式，可靠且可用的，用于高效地收集，聚合和移动大量日志数据的服务。 它具有基于流数据的简单灵活的体系结构。 它具有可调整的可靠性机制以及许多故障转移和恢复机制，具有强大的功能和容错能力。 它使用一个简单的允许在线分析应用程序的可扩展数据模型。简单来说，**Flume是一个分布式、可靠、和高可用的海量日志采集、聚合和传输的系统**。支持在日志系统中定制各类数据发送方，用于收集数据；同时，Flume提供对数据进行简单处理，并写到各种数据接受方(比如文本、HDFS、Hbase等)的能力 。



**Flume工作流程**

Flume的数据流由事件(Event)贯穿始终。事件是Flume的基本数据单位，它携带日志数据(字节数组形式)并且携带有头信息，这些Event由Agent外部的Source生成，当Source捕获事件后会进行特定的格式化，然后Source会把事件推入(单个或多个)Channel中。可以把Channel看作是一个缓冲区，它将保存事件直到Sink处理完该事件。Sink负责持久化日志或者把事件推向另一个Source。

Flume具有以下特点：

* **可靠性**

当节点出现故障时，日志能够被传送到其他节点上而不会丢失。  
Flume提供了三种级别的可靠性保障，从强到弱依次分别为：

(1) end-to-end**（**收到数据agent首先将event写到磁盘上，当数据传送成功后，再删除；如果数据发送失败，可以重新发送。），

(2)  Store on failure（这也是scribe采用的策略，当数据接收方crash时，将数据写到本地，待恢复后，继续发送），  
　　(3)  Best effort（数据发送到接收方后，不会进行确认）。

* 可扩展性

Flume采用了三层架构，分别为agent，collector和storage，每一层均可以水平扩展。其中，所有agent和collector由master统一管理，这使得系统容易监控和维护，且master允许有多个（使用ZooKeeper进行管理和负载均衡），这就避免了单点故障问题。

* 可管理型

（1）所有agent和colletor由master统一管理，这使得系统便于维护。  
　　（2）多master情况，Flume利用ZooKeeper和gossip，保证动态配置数据的一致性。  
　　（3）用户可以在master上查看各个数据源或者数据流执行情况，且可以对各个数据源配置和动态加载。  
　　（4）Flume提供了web 和shell script command两种形式对数据流进行管理。

* 功能可扩展性

（1）用户可以根据需要添加自己的agent，collector或者storage。  
　　（2）此外，Flume自带了很多组件，包括各种agent（file， syslog等），collector和storage（file，HDFS等）。

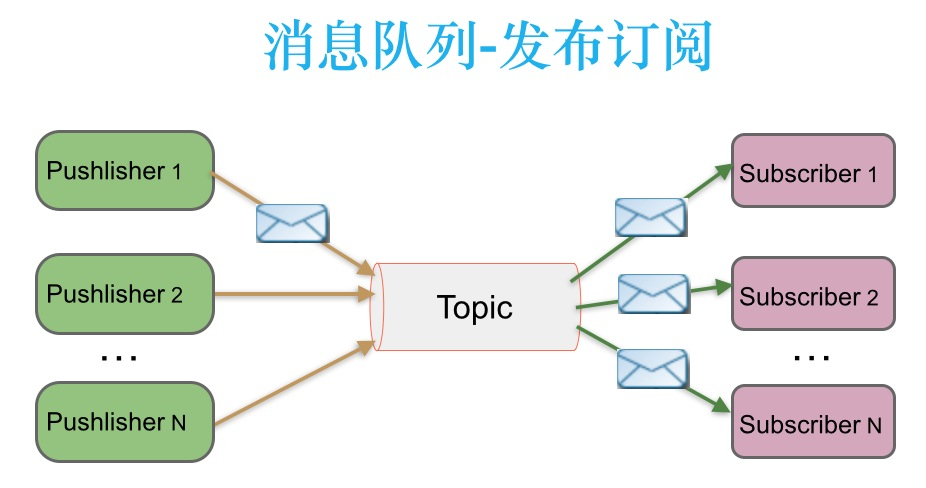
## 2、Kafka集群

**Kafka 是一个高吞吐量的分布式发布-订阅消息系统**。企业中一般使用 Kafka 做消息中间件，做缓冲缓存处理。需要zookeeper分布式协调组件管理。Kafka主要设计目标如下：

* 以时间复杂度为O(1)的方式提供消息持久化能力，即使对TB级以上数据也能保证常数时间的访问性能。
* 高吞吐率。即使在非常廉价的商用机器上也能做到单机支持每秒100K条消息的传输。
* 支持Kafka Server间的消息分区，及分布式消费，同时保证每个partition内的消息顺序传输。
* 同时支持离线数据处理和实时数据处理。
* Scale out:支持在线水平扩展

一个消息系统负责将数据从一个应用传递到另外一个应用，应用只需关注于数据，无需关注数据在两个或多个应用间是如何传递的。分布式消息传递基于可靠的消息队列，在客户端应用和消息系统之间异步传递消息。有两种主要的消息传递模式：点对点传递模式、发布-订阅模式。大部分的消息系统选用发布-订阅模式。Kafka就是一种发布-订阅模式。

在发布-订阅消息系统中，消息被持久化到一个topic中。与点对点消息系统不同的是，消费者可以订阅一个或多个topic，消费者可以消费该topic中所有的数据，同一条数据可以被多个消费者消费，数据被消费后不会立马删除。在发布-订阅消息系统中，消息的生产者称为发布者，消费者称为订阅者。该模式的示例图如下：



**KafKa工作模式**

相关术语：

1、kafka集群中的服务器都叫做broker

2、kafka有两类客户端，一类叫producer（消息生产者），一类叫做consumer（消息消费者），客户端和broker服务器之间采用tcp协议连接

3、kafka中不同业务系统的消息可以通过topic进行区分，而且每一个消息topic都会被分区，以分担消息读写的负载

4、每一个分区都可以有多个副本，以防止数据的丢失

5、某一个分区中的数据如果需要更新，都必须通过该分区所有副本中的leader来更新

## 3、Flink集群

Apache Flink是一个框架和分布式处理引擎，用于对无边界和有边界的数据流进行状态计算。Flink被设计为在运行在所有常见的集群环境中，在任何规模的数据上以内存的速度执行计算。

**Flink架构的重要特性：**

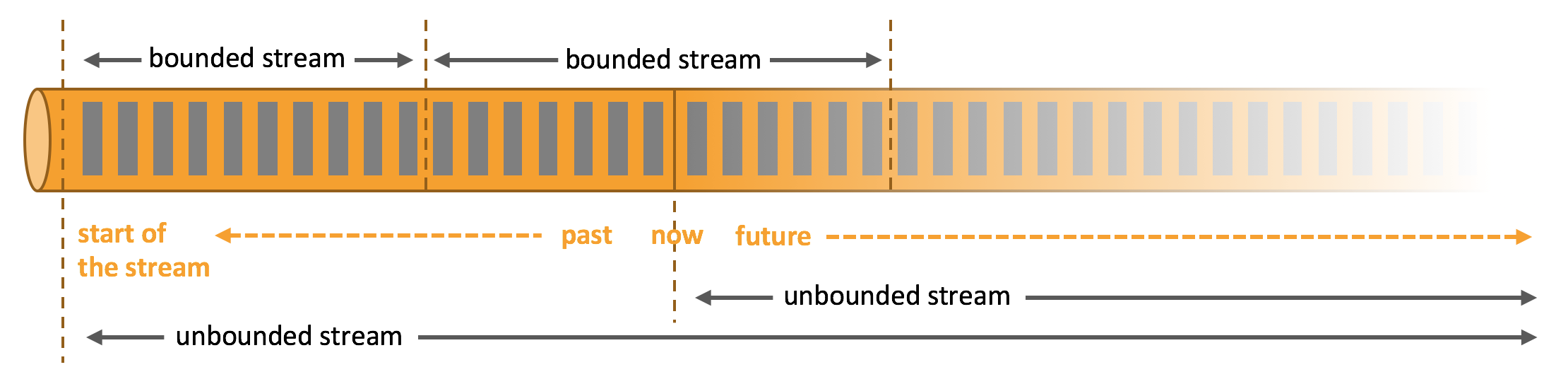
**（1）能够处理有边界和无边界数据**

任何类型的数据都是作为事件流产生的。信用卡交易，传感器测量，机器日志或网站或移动应用程序上的用户交互，所有这些数据均作为流生成。

数据可以作为无界流或有界流处理。

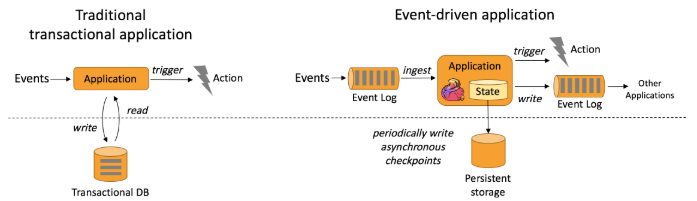
1、无边界流有一个起点，但没有定义终点。它们不会终止并在生成数据时就提供数据。无边界流必须连续处理，即，事件在被摄取后必须立即处理。不可能等待所有输入数据到达，因为输入是无界的，并且在任何时间都不会完成。处理无边界数据通常需要以特定顺序（例如事件发生的顺序）来摄取事件，以便能够推断出结果的完整性。

2、有界流具有定义的开始和结束。可以通过在执行任何计算之前提取所有数据来处理有界流。由于有界数据集始终是已经排序好的，因此不需要有序摄取即可处理有界流。有限流的处理也称为批处理。



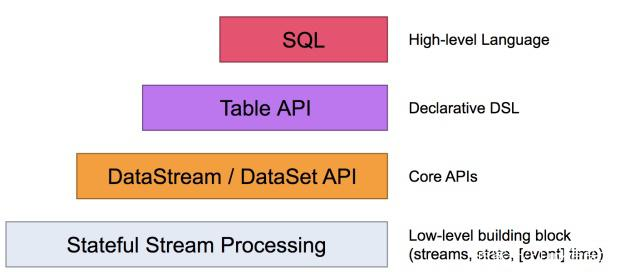
**（2）事件驱动**

事件驱动的应用程序是传统应用程序设计的发展，具有分离的计算和数据存储层。在传统体系结构中，应用程序从远程事务数据库读取数据并将数据持久化到远程事务数据库。相反，事件驱动的应用程序基于状态流处理应用程序。在这种设计中，数据和计算位于同一位置，从而可以进行本地内存或磁盘的数据访问。通过定期将检查点写入远程持久化存储来实现容错。下图描述了传统应用程序结构和事件驱动的应用程序之间的区别。



**（3）分层API**

Flink 最下面的一层API为Stateful Stream Processing，它是Flink最底层的API，控制更灵活但一般很少使用。然后上面一层就是Flink Core（核心）API，它包含DataStream和DataSet API，应用层的用户经常使用 Core API。然后再上面一层就是 Table API，它相当于在Core API中可以定义数据的Table结构，可以做table操作。最上面一层就是SQL 操作，用户可以直接使用SQL语句对数据处理，更简单更方便。本实验中使用的就是SQL层的API。



Kafka的其他特点：

1、精确一次(exactly-onece)的状态一致性保证

2、低延迟，每秒处理数百万个事件，毫秒级延迟

3、高可用，动态扩展，实现7\*24小时全天候运行

# 五、系统设计流程

## 1、商品交易数据的获取

这里的商品交易数据是通过Python脚本随机生成的，用来模拟人购买商品的行为，数据格式为：

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 行键 | 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 |

如：

|  |
| --- |
| 421564974572,Sgxrp,20,woman,152121,297.64,313015,scan,15516074528,JbiLDQmzwP@qq.com,2019-08-01 |
| 601564974572,Lbeuo,43,man,220902,533.13,313016,pv,15368753104,ezfrJSyuoR@163.com,2019-08-05 |

随即生成的代码如下：

|  |
| --- |
| 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)  # 换行写入  f.write('\n')  i += 1  f.close()  if \_\_name\_\_ == "\_\_main\_\_":  write\_record\_to\_file() |

创建一个/tmp/flume\_spooldir/目录，将该目录作为后续步骤的数据源目录，将上面脚本产生的原始数据都存放在该目录下。

命令：

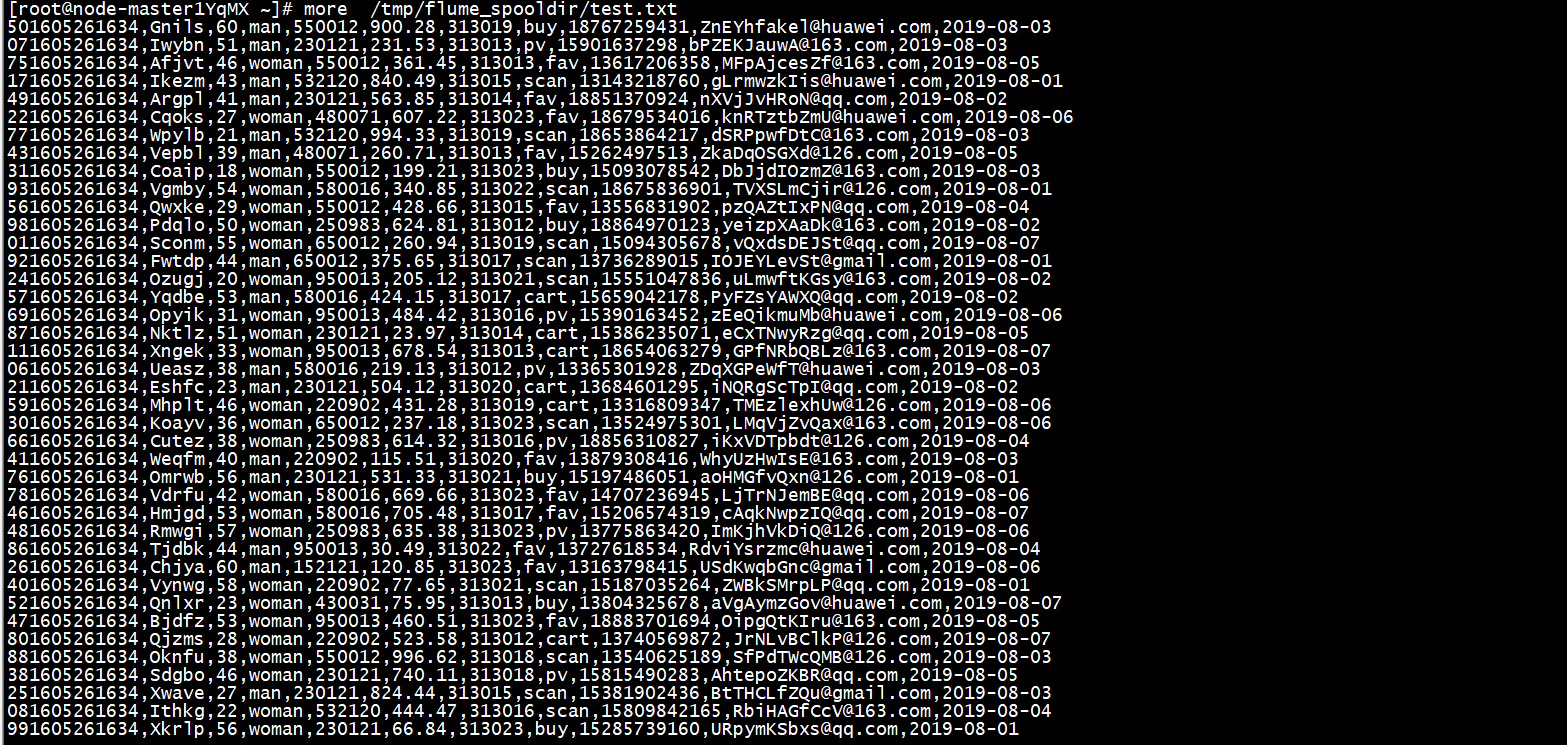
|  |
| --- |
| mkdir /tmp/flume\_spooldir/ |

执行Python脚本生成100条交易数据：

命令：

|  |
| --- |
| python autodatapython.py "/tmp/flume\_spooldir/test.txt" 100 |

以下便是随机生成的一些数据样本：



## 2、配置Kafka

Kafka是消息订阅系统，需要先定义好一个传送消息的主题topic，创建一个topic用于传输数据的命令如下：

|  |
| --- |
| /opt/client/Kafka/kafka/bin/kafka-topics.sh –create --zookeeper 192.168.0.52:2181/kafka --partitions 1 --replication-factor 1 --topic cxhcip |



## 3、安装Flume

修改配置文件，命令：

|  |
| --- |
| cd /opt/FlumeClient/fusioninsight-flume-1.6.0/  vi conf/properties.properties |

要想使用Flume来采集数据，那么必须指定数据的来源，数据传输的通道，还有数据要送到的目的地。还必须指定它们的参数，该配置文件如下，我都添加了注解：

|  |
| --- |
| client.sources = s1 #数据源的标识符  client.channels = c1 #数据通道的标识符  client.sinks = sh1 #数据接收方的标识符  # 数据源s1的配置文件  client.sources.s1.type = spooldir #数据源要采集的信息是一个目录下的文件  client.sources.s1.spoolDir = /tmp/flume\_spooldir #目录的位置  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 #数据源传输使用的通道是c1  # the channel configuration of c1 #传输通道的配置  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 #接收方的配置  client.sinks.sh1.type = org.apache.flume.sink.kafka.KafkaSink #接收方是Kafka  client.sinks.sh1.kafka.topic = cxhcip #配置Kafka的topic  client.sinks.sh1.flumeBatchSize = 1000  client.sinks.sh1.kafka.producer.type = sync  client.sinks.sh1.kafka.bootstrap.servers = 192.168.0.115:9092 #kafka的broker地址和端口  client.sinks.sh1.kafka.security.protocol = PLAINTEXT  client.sinks.sh1.requiredAcks = 0  client.sinks.sh1.channel = c1 #接收方从c1通道接收数据 |

## 4、打通Kafka和Flume

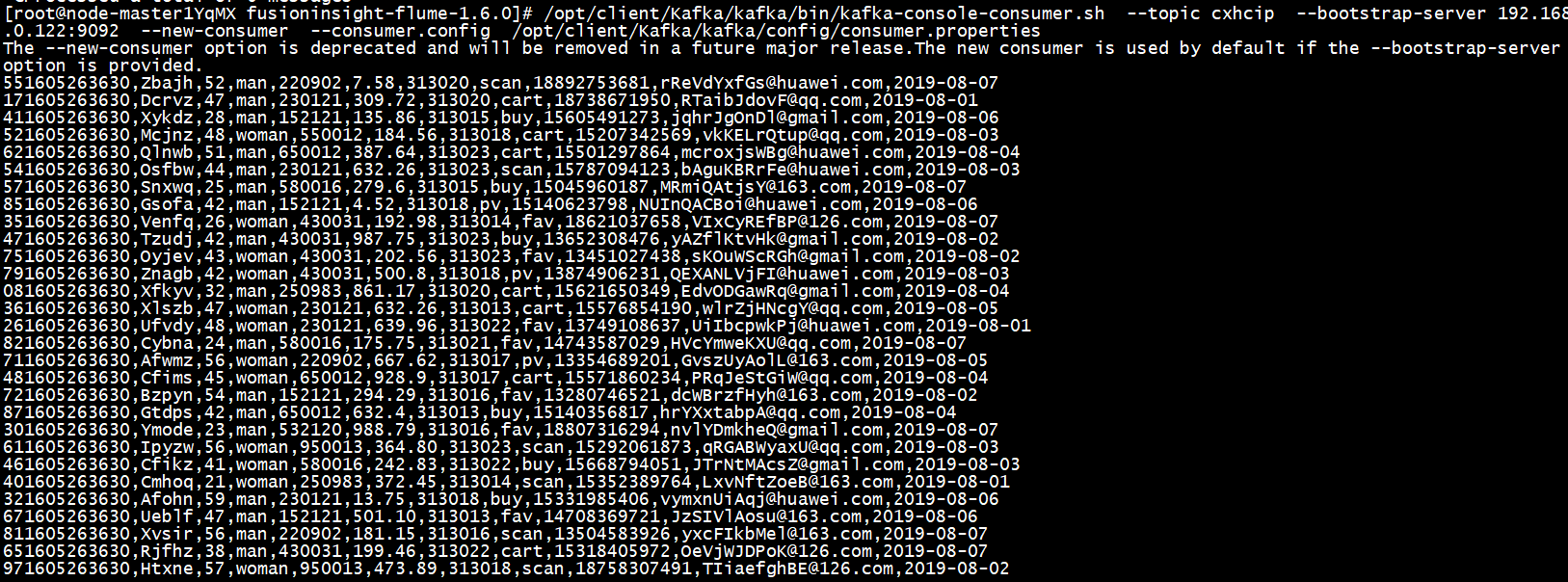
配置好flume数据采集后，就可以开启Kafka的consumer来接收消息了。

命令：

|  |
| --- |
| /opt/client/Kafka/kafka/bin/kafka-console-consumer.sh --topic cxhcip --bootstrap-server 192.168.0.86:9092 --new-consumer --consumer.config /opt/client/Kafka/kafka/config/consumer.properties |

这条命令创建了一个接收消息的客户端，并指定了接收消息的主题，地址，和该客户端的配置文件。

再去用之前写的Python脚本生成一个数据文件，观察Kafka的接收情况，由于之前指定的消息类型是文件夹类型，如果文件夹内有文件变化，那么就会将文件的数据读取出来并发送到topic中，接收方会通过该topic读取到信息。



可以看到，Python脚本在文件夹生成文件后，flume立即将产生的数据传递给了Kafka，Kafka接收到数据并显示出来了，说明Kafka和flume已经打通。

## 5、MySQL中准备结果表与维度表数据

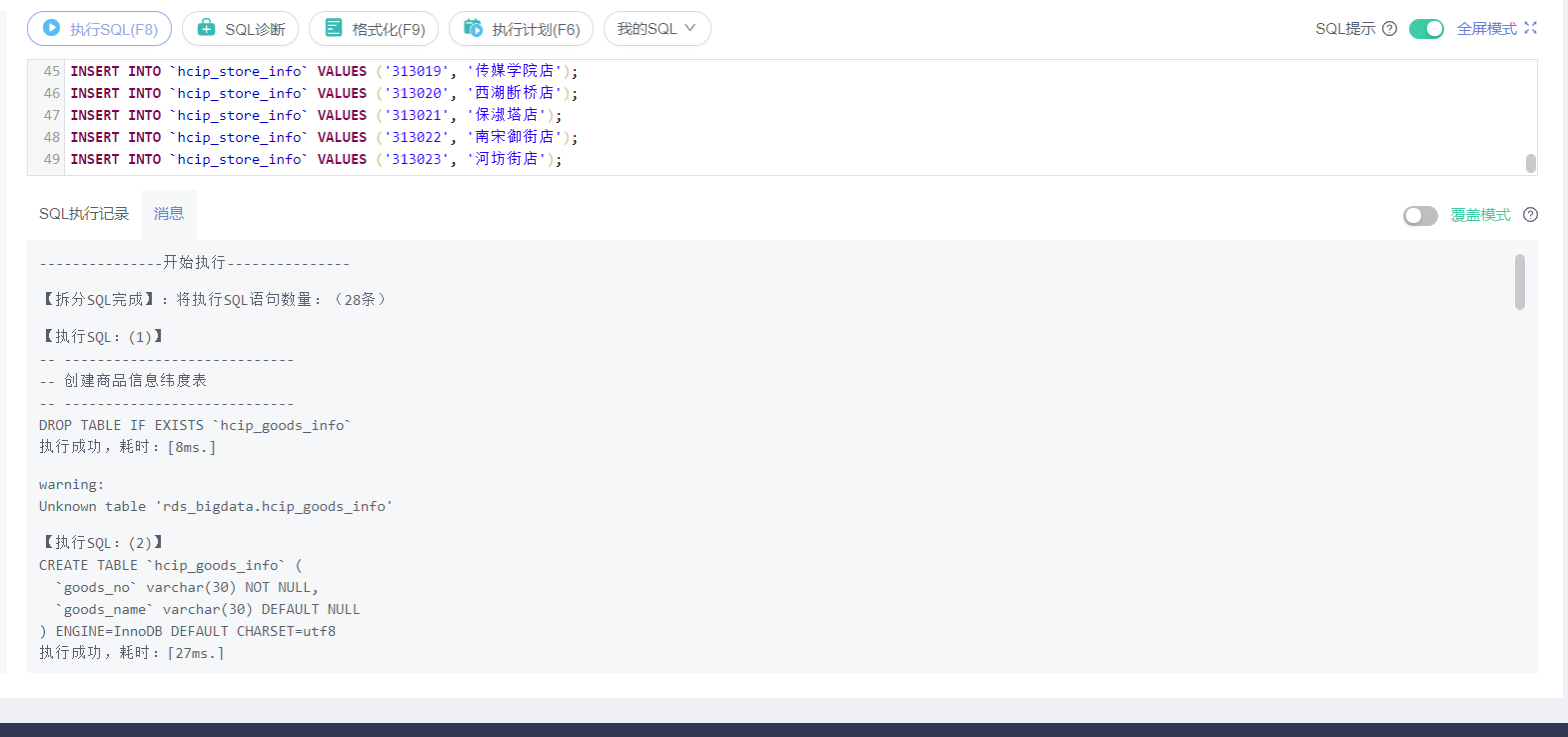


门店信息和商品信息是原本就应该确定好的，所以应该把门店信息和商品信息提前放入表中。

执行以下sql：

|  |
| --- |
| -- ----------------------------  -- 创建商品信息纬度表  -- ----------------------------  DROP TABLE IF EXISTS `hcip\_goods\_info`;  CREATE TABLE `hcip\_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 `hcip\_goods\_info` VALUES ('220902', '杭州丝绸');  INSERT INTO `hcip\_goods\_info` VALUES ('430031', '西湖龙井');  INSERT INTO `hcip\_goods\_info` VALUES ('550012', '西湖莼菜');  INSERT INTO `hcip\_goods\_info` VALUES ('650012', '张小泉剪刀');  INSERT INTO `hcip\_goods\_info` VALUES ('532120', '塘栖枇杷');  INSERT INTO `hcip\_goods\_info` VALUES ('230121', '临安山核桃');  INSERT INTO `hcip\_goods\_info` VALUES ('250983', '西湖藕粉');  INSERT INTO `hcip\_goods\_info` VALUES ('480071', '千岛湖鱼干');  INSERT INTO `hcip\_goods\_info` VALUES ('580016', '天尊贡芽');  INSERT INTO `hcip\_goods\_info` VALUES ('950013', '叫花童鸡');  INSERT INTO `hcip\_goods\_info` VALUES ('152121', '火腿蚕豆');  INSERT INTO `hcip\_goods\_info` VALUES ('230121', '杭州百鸟朝凤');  -- ----------------------------  -- 创建门店信息纬度表  -- ----------------------------  DROP TABLE IF EXISTS `hcip\_store\_info`;  CREATE TABLE `hcip\_store\_info` (  `store\_id` varchar(50) NOT NULL,  `store\_name` varchar(50) DEFAULT NULL  ) ENGINE=InnoDB DEFAULT CHARSET=utf8;  -- ----------------------------  -- 插入商品信息样例数据  -- ----------------------------  INSERT INTO `hcip\_store\_info` VALUES ('313012', '莫干山店');  INSERT INTO `hcip\_store\_info` VALUES ('313013', '定安路店');  INSERT INTO `hcip\_store\_info` VALUES ('313014', '西湖银泰店');  INSERT INTO `hcip\_store\_info` VALUES ('313015', '天目山店');  INSERT INTO `hcip\_store\_info` VALUES ('313016', '凤起路店');  INSERT INTO `hcip\_store\_info` VALUES ('313017', '南山路店');  INSERT INTO `hcip\_store\_info` VALUES ('313018', '西溪湿地店');  INSERT INTO `hcip\_store\_info` VALUES ('313019', '传媒学院店');  INSERT INTO `hcip\_store\_info` VALUES ('313020', '西湖断桥店');  INSERT INTO `hcip\_store\_info` VALUES ('313021', '保淑塔店');  INSERT INTO `hcip\_store\_info` VALUES ('313022', '南宋御街店');  INSERT INTO `hcip\_store\_info` VALUES ('313023', '河坊街店'); |

执行结果：

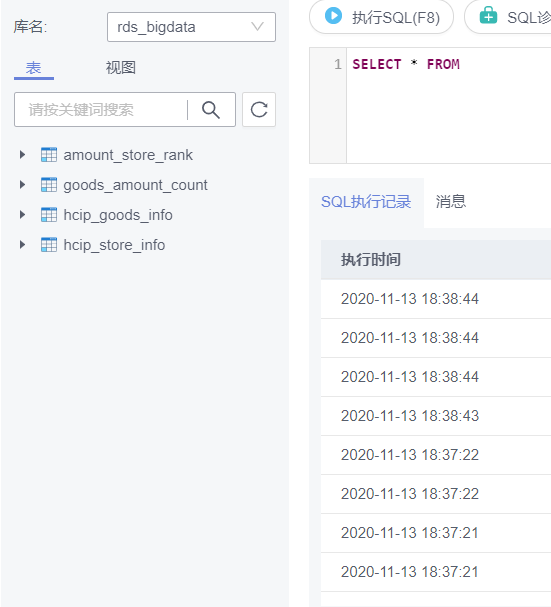


其次，我们需要将服务器产生的数据通过Flink进行处理，并将结果数据存入Mysql表中，因此，我们需要创建两个结果表以接收数据，一个是最后展示的销售额的表，还有一个是每个门店销售额的表。

建表语句如下：

|  |
| --- |
| -- ----------------------------  -- 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; |

结果：



## 6、创建Flink作业打通Mysql与Kafka

建好Mysql表后，我们需要将从Kafka接收到的数据，进行处理后存入Mysql表中，以便后续的可视化任务。这里我们使用的是Flink最顶层的API，也就是Flink-SQL，Flink-SQL的特点是编写起来非常简单，很容易使用，但是它的工作能力有限，只能完成一些简单的处理任务，不能完成非常复杂的任务，但是对于我们所做的这个设计来说已经够用了。Flink-SQL的用法与Mysql的语法很相似，看一会就能上手。

下面是Flink-SQL语句：

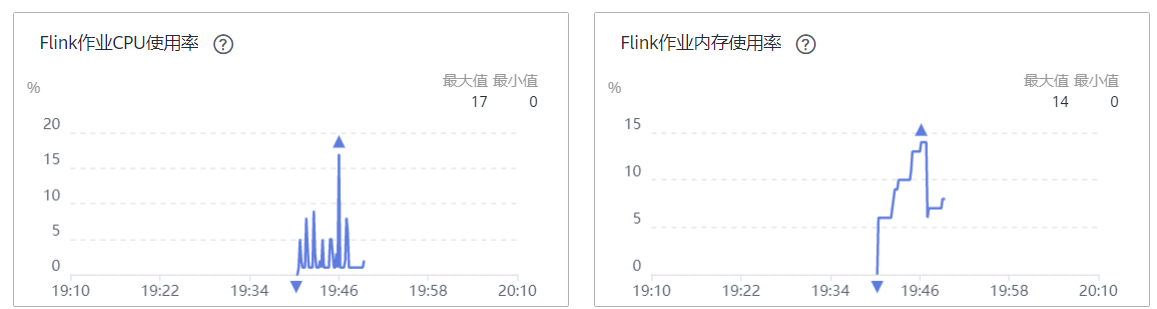
|  |
| --- |
| /\*\*  \*kafka传来的原始数据格式如下 \*kafka\_source(id,user\_name,age,gender,goods\_no,goods\_price,store\_id,shopping\_type,tel,email,shopping\_date):  \*行健，用户名，年龄，性别，商品ID，价格，门店ID，购物行为，电话，邮箱，购买日期  \*591564404906,Gkisf,59,man,550012,23.45,423012,fav,15274039826,UsPZNdxnJX@163.c\*om,2019-07-01  \*761564404906,Wupsa,55,man,250983,25.05,423013,scan,13730586914,AKVMdPveQX@gmai\*l.com,2019-07-02  \*\*/  /\*\*建立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 = "192.168.0.86:9092", /\*\* 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 = "cxhcip", /\*\*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  \*  \*\*/  /\*\*建立商品销售额的连接到MySQL的输出流\*\*/  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 = "填写密码", /\*创建MySQL实例设置的密码\*/  db\_url = "mysql://192.168.0.252:3306/rds\_bigdata", /\*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  \*  \*\*/  /\*\*建立每个门店的连接到MySql的销售额输出\*\*/  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 = "填写密码", /\*创建MySQL实例设置的密码\*/  db\_url = "mysql://192.168.0.252:3306/rds\_bigdata", /\*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  \*\*/  /\*\*创建一个MySql已经存在的门店信息表，用来关联门店信息\*\*/  CREATE TABLE table\_store\_info (  store\_id STRING,  store\_name STRING  )  WITH (  type = "rds",  region = "cn-north-1",/\* 服务所在的区域，例如华北-北京一为：cn-north-1\*/  username = "root", /\*MySQL的用户名默认是root \*/  password = "填写密码", /\*创建MySQL实例设置的密码\*/  db\_url = "mysql://192.168.0.252:3306/rds\_bigdata", /\*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 = "hcip\_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  \*\*/  /\*\*创建一个MySql已经存在的商品信息表，用来关联商品信息\*\*/  CREATE TABLE table\_goods\_info (  store\_id STRING,  store\_name STRING  )  WITH (  type = "rds",  region = "cn-north-1",/\* 服务所在的区域，例如华北-北京一为：cn-north-1\*/  username = "root", /\*MySQL的用户名默认是root \*/  password = "填写密码", /\*创建MySQL实例设置的密码\*/  db\_url = "mysql://192.168.0.252:3306/rds\_bigdata", /\*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 = "hcip\_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; |

上述Flink语句关联了Kafka的输出流与MySql的输入流，当Kafka中有数据传进来时，Flink就会自动将Kafka中的数据进行上述处理，将处理结果存入MySql中。由于Flink的低延迟特性，每秒可以处理数百万条数据，能处理大规模数据突发的状况，并能在很短时间内处理完成，送至MySql进行下一步数据分析或可视化。

下面启动该Flink任务



再执行一次最初写的那个Python脚本生成新的订单数据，这些数据会触发Flume的响应，将新生成的数据通过一个channel传输到Kafka，当Kafka收到数据后，通过Topic将数据传送给Consumer，Flink就是Kafka的一个Consumer，当Flink检测到Kafka中有数据要输出时，会将传来的数据进行处理，然后将结果输送到MySql中。



**Flink检测到数据后进行处理**

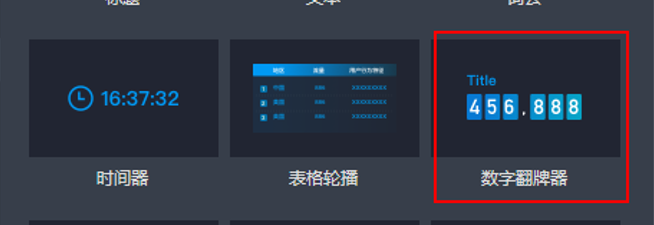
****

**MySql收到数据**

## 7、数据可视化

这里的数据可视化是使用华为云的DLV服务，直接通过Sql语句，将执行结果与华为原本就做好的模板进行关联，然后设置一个更新的时间间隔，就能在一定的时间间隔内更新可视化的结果，实现动态的数据可视化展示。

在华为云的控制台内找到DLV服务，并找到自己喜欢的可视化模板



这里我就使用实验教程上用的数字翻牌器。然后将数据库的查询结果映射到该数字翻牌器上。



Sql语句如下：

|  |
| --- |
| select amount\_total from goods\_amount\_count where sale\_date="2019-08-07" |

勾选自动更新，调整时间间隔为5s，那么每五秒展示的结果就会更新一次。

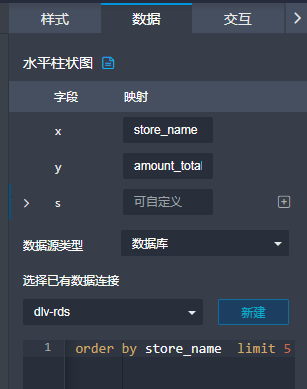


上面展示的是总销量的实时变化情况，下面将展示销售额排行前五门店的信息。

添加DLV组件常用图表中的折线柱图到大屏中，编写Sql语句如下

|  |
| --- |
| select store\_name,amount\_total,"销售额" as title from `amount\_store\_rank` order by store\_name limit 5 |

然后再添加映射关系：



勾选自动更新，并设置更新时间间隔，就能动态展示销售前五的商家的信息。



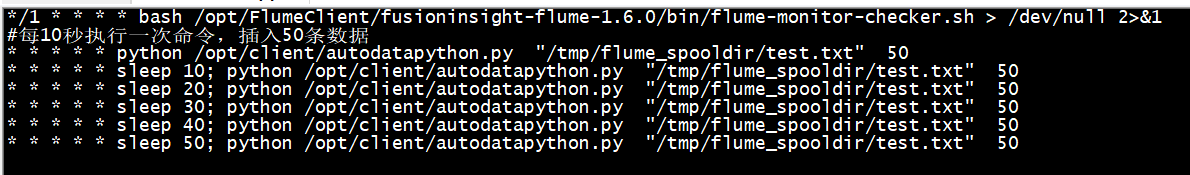
## 8、定时生成数据

由于我们只执行了一次脚本，生成了一条数据，可视化的结果不是动态的，还只是静止不动的，我们需要设置一个定时任务，每隔一段时间就执行一次脚本，相当于产生了用户购买数据。

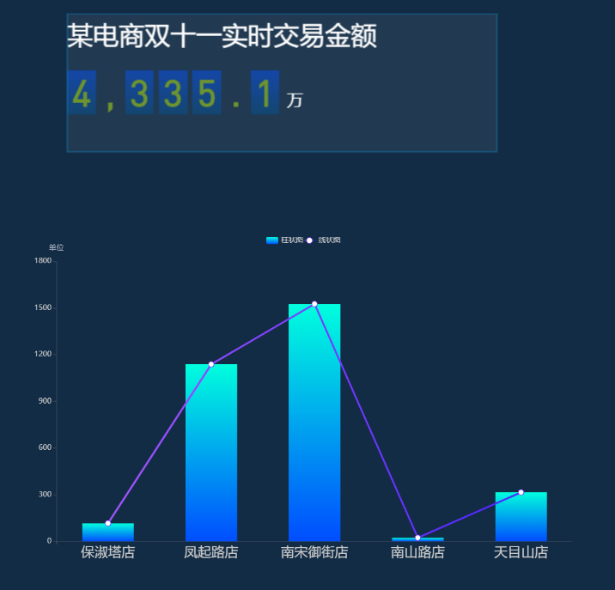
在Master中输入命令：crontab -e，并添加如下语句：

|  |
| --- |
| #每10秒执行一次命令，插入50条数据  \* \* \* \* \* python /opt/client/autodatapython.py "/tmp/flume\_spooldir/test.txt" 50  \* \* \* \* \* sleep 10; python /opt/client/autodatapython.py "/tmp/flume\_spooldir/test.txt" 50  \* \* \* \* \* sleep 20; python /opt/client/autodatapython.py "/tmp/flume\_spooldir/test.txt" 50  \* \* \* \* \* sleep 30; python /opt/client/autodatapython.py "/tmp/flume\_spooldir/test.txt" 50  \* \* \* \* \* sleep 40; python /opt/client/autodatapython.py "/tmp/flume\_spooldir/test.txt" 50  \* \* \* \* \* sleep 50; python /opt/client/autodatapython.py "/tmp/flume\_spooldir/test.txt" 50 |

这样每隔十秒，系统就会自动执行Python脚本，在目录中产生数据，最后被送入MySql中，模拟了用户购买行为。

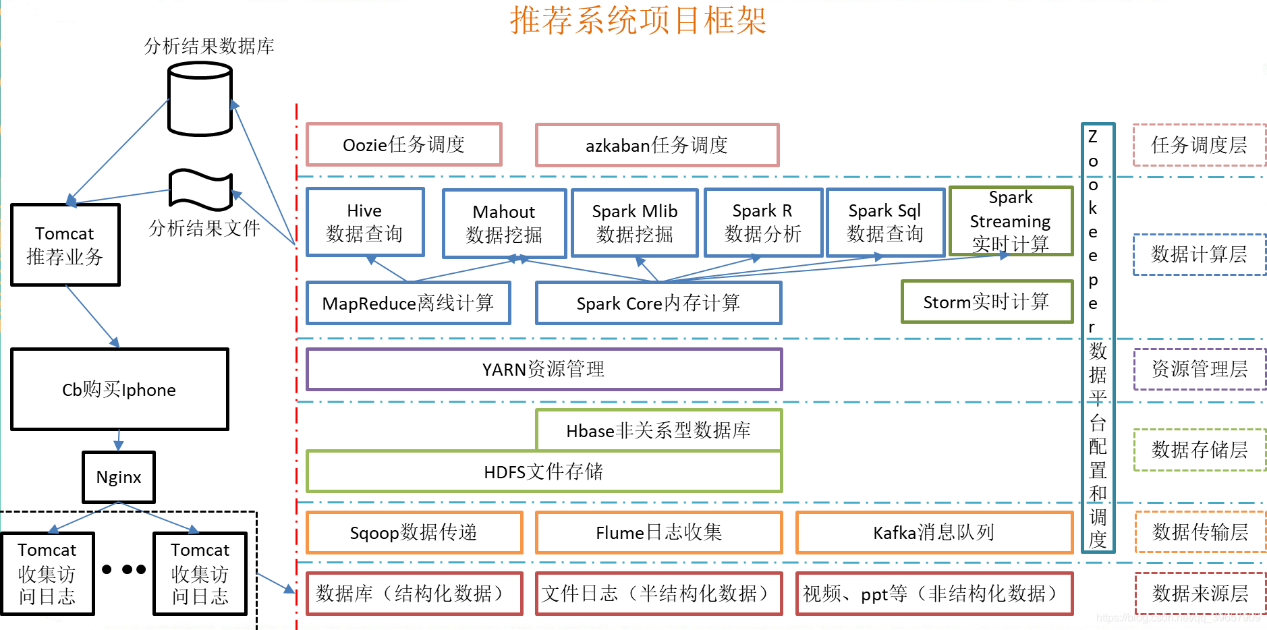


可视化结果变化如下：



# 六、实验总结

我之所以选这个大数据实时分析实验写课程设计是因为之前在准备大数据比赛的时候，看到大数据课程的最后几课学的就是Kafka和Flume，大数据比赛只考察了Hadoop，Hive还有Spark，所以当时没有去学习之后的这些课程，对Kafka和Flume也只有一个印象，并不知道这两个技术是干什么用的。刚好华为云大数据有这个实验，我就正好可以趁着这个机会去学习之前还没有学的知识。在之前自学的课程中还看见了Storm框架，通过这几天的学习，知道了Storm也是一个流数据实时计算的框架，本实验中的Flink也是流数据的处理框架，并且Flink性能比Storm还会更好。到目前为止，我也算是使用并了解过一大半的大数据处理框架了，对整个大数据的生态体系也有了较为直观的认识，并对大数据处理流程也有了一定的了解。



**大数据生态体系示意图**

除了做了这个大数据综合设计外，其他的几个综合设计我对着教程操作了一遍，

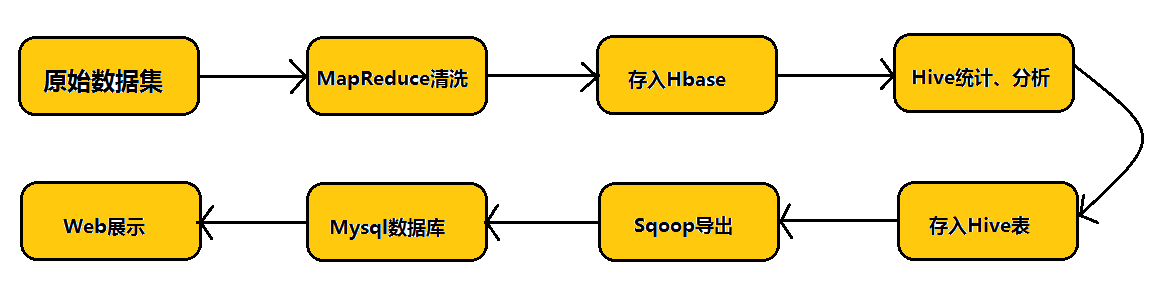
下面是我对这几个实验的一些看法与理解：

首先课程实验一，云主机实现大数据，这个实验的工作与大数据竞赛的题目几乎一模一样，是一个关于搭建Hadoop集群的实验，与我们平时搭建的Hadpoop集群不同的是，使用华为云服务器还可以将Hadoop集群与华为云的OBS存储对象服务关联起来，通过华为云的控制台，可以使用图形化界面与HDFS进行交互，文件上传与下载。

课程实验二，云服务实现大数据，这个实验就是通过华为云已经搭建好的Hadoop集群来运行MapReduce作业。我们一般编写的MapReduce程序需要打成一个Jar包送入到Master节点去进行分发任务，通过命令hadoop jar xxx来运行这个作业，而这个实验就是给我面设计好了图形界面，不需要通过命令行，直接将数据和文件上传到华为云的OBS存储对象中去，然后上传到Hadoop集群运行作业，让整个作业提交流程变得简单。

综合实验一在课程实验一的基础上增加了Spark集群的搭建，也是比较基础的一个实验。

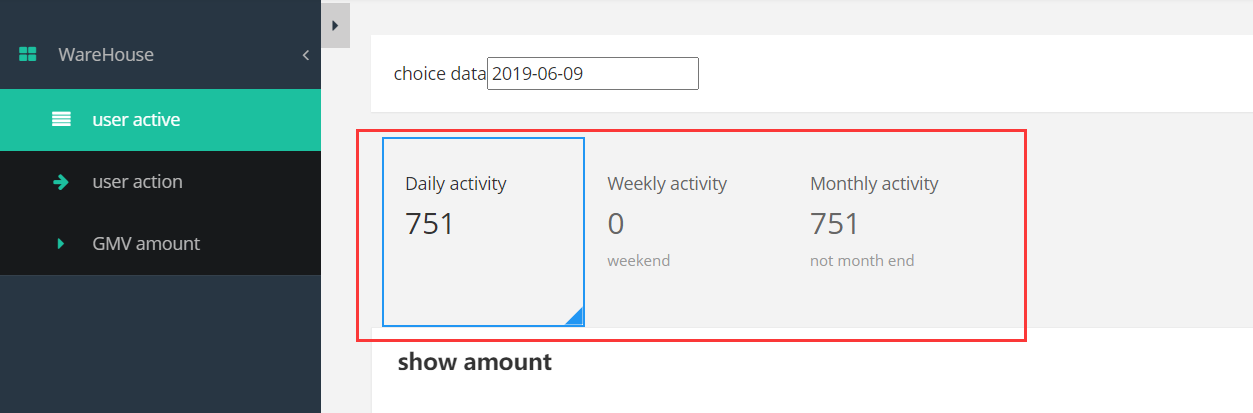
综合实验三是大数据离线分析。大数据离线分析和大数据实时分析都是电商平台对用户购买数据等数据的重要分析手段，我认为在线分析主要作用是可视化所有顾客的实施数据，比如实施成交量，实时浏览量等，让后台人员清晰的看见目前平台是什么状况，而离线分析是在一段时间内手机大量的用户信息，需要从很多的信息中挖掘出有效信息，单条信息的价值不高，但是许多信息合起来就能有很大潜在的价值，所以需要使用离线分析对大量的数据进行分析。在该实验中，主要用的工具是Hive、Myql还有Loader。其中Loader是通过Sqoop作业来进行数据库数据的传递。



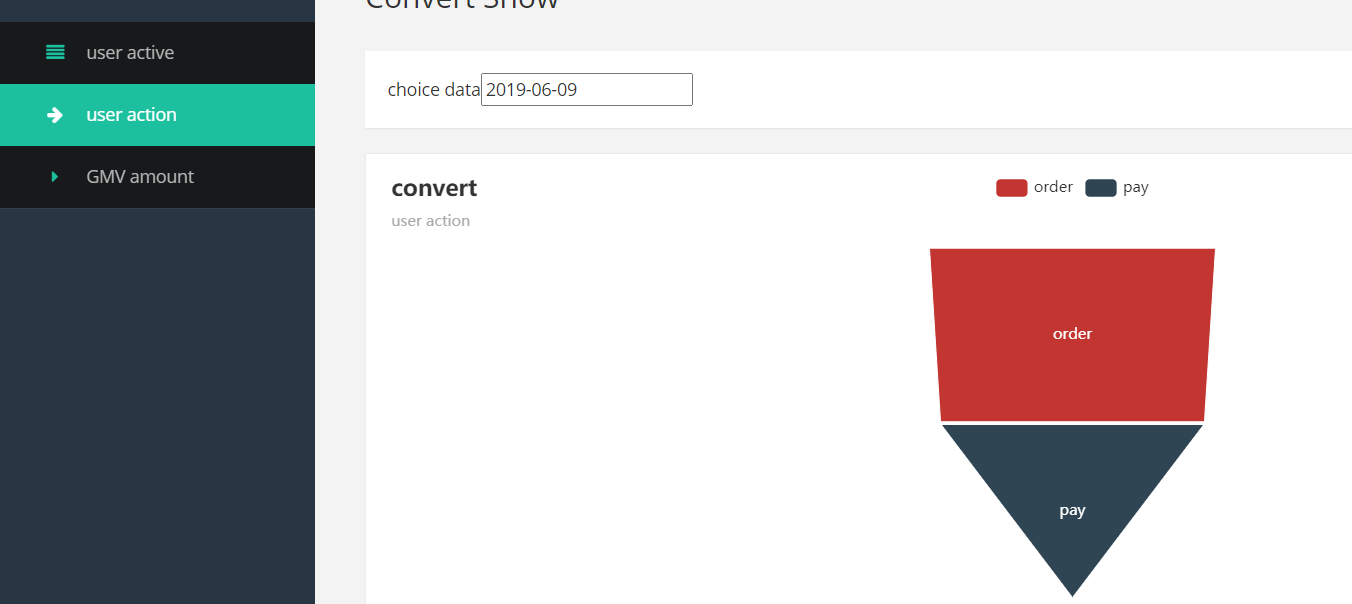
**Hadoop业务开发流程示意图**

上图展示了Sqoop在大数据分析中的地位。在实际的业务当中，我们首先对原始数据集通过MapReduce进行数据清洗，然后将清洗后的数据存入到Hbase数据库中，而后通过数据仓库Hive对Hbase中的数据进行统计与分析，分析之后将分析结果存入到Hive表中，然后通过Sqoop这个工具将我们的数据挖掘结果导入到MySql数据库中，最后通过Web将结果展示给客户。在本实验中，我们没有使用Hbase存储数据，我们直接使用已经清洗好的数据通过Hive进行统计与分析，然后利用Loader的UI界面创建Sqoop作业，将Hive内的数据输出到Mysql中数据库中，最后使用提供了的spring-boot-echarts的Jar包进行数据的展示。

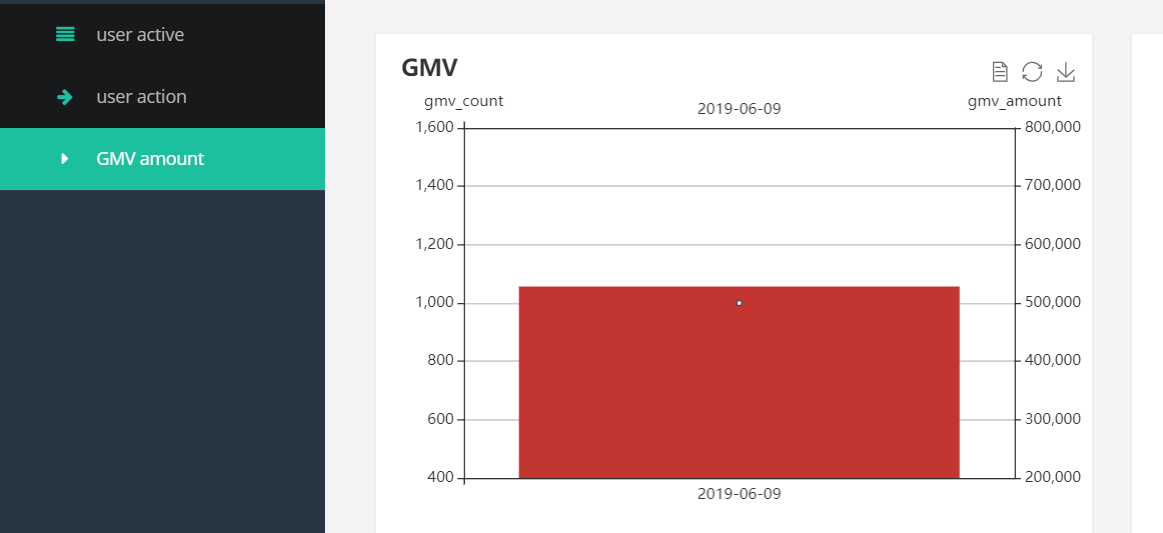
下面展示一下试验最终结果：



**日活量，周活量，月活量**



**用户形为图**



**GVM图**

综合实验四是Hive+Presto的使用，Hive是数据仓库，其作用是用Hive语句来代替编写复杂的MapReduce程序。Presto是完全基于内存的并⾏计算，分布式SQL交互式查询引擎，支持多种数据源与存储格式，也是通过sql语句来进行查询，但比Hive要快，是一种基于内存的查询引擎，通过这个实验了解了Presto工具以及它的基本使用。

综合实验六是关于Hbase的，比较简单，包含了Hbase的一些基本存储操作以及一些分区操作还有过滤操作。