# **测试报告**

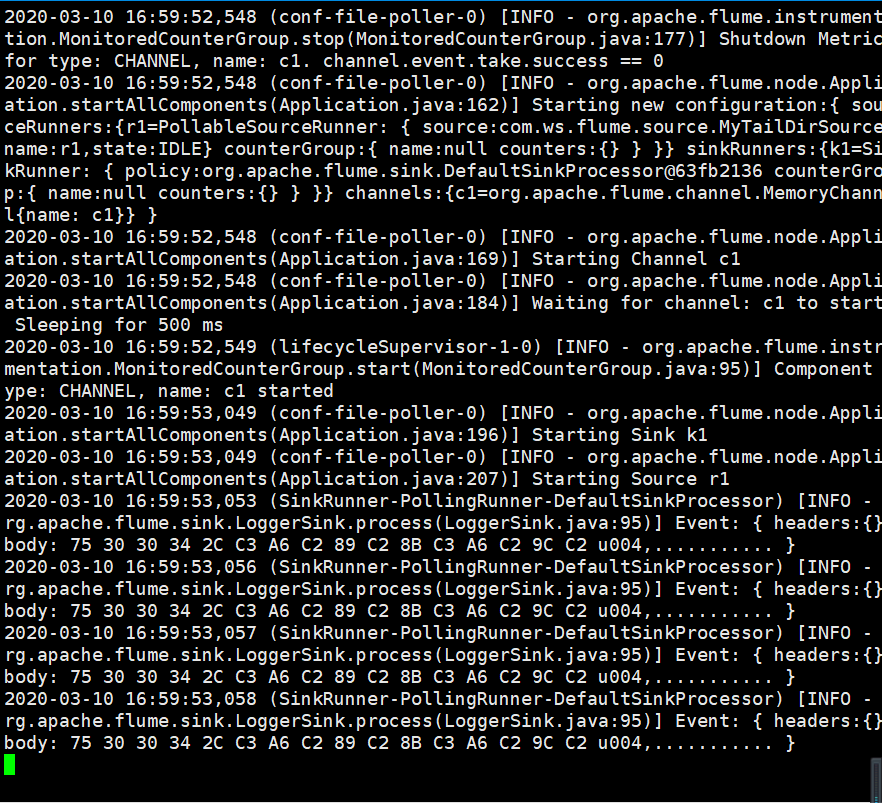
### **① Kafka+flume+数据生成：**

**预期效果**：根据需求生成数据，使用Java程序，定时产生log日志，使用flume读取日志信息，清洗数据 发送到kafka

##### **测试方法代码块效果图：**

#配置flume采集信息  
a1.sources = r1  
a1.channels = c1  
a1.sinks = k1  
​  
#a1.sources.r1.type = netcat  
#a1.sources.r1.bind = 0.0.0.0  
#a1.sources.r1.port = 6666  
#a1.sources.r1.interceptors = i1  
#a1.sources.r1.interceptors.i1.schema =user,phone,phonetype,price,counts,time,ip  
#a1.sources.r1.interceptors.i1.types = String,String,String,Double,Integer,String,String  
#a1.sources.r1.channels = c1  
​  
#使用taildir  
a1.sources.r1.type = com.bawei.flume.source.MyTailDirSource  
a1.sources.r1.channels = c1  
a1.sources.source1.channels.skipToEnd = True  
#设置位置文件的存储路径，位置文件记录着被监听的文件读取的偏移量  
a1.sources.r1.posFilePath = /root/log/flume/taildir/datas.dat  
a1.sources.r1.filePath = /root/log/flume/source/a.txt  
#设置监听的文件  
a1.sources.r1.filegroups = f1  
a1.sources.r1.filegroups.f1 = /root/log/example.log  
​  
#flume消息event的header中增加一个keyvalue值叫headerKey1=example  
a1.sources.r1.headers.f1.headerKey1 = example  
#flume消息的event的header中增加一个消息来源的file的绝对路径  
a1.sources.r1.fileHeader = true  
#source的最大batch是多少，不能大于cheannl的容量  
a1.sources.r1.maxBatchCount = 300  
a1.channels.c1.type = memory  
a1.channels.c1.capacity = 1000  
a1.channels.c1.transactionCapacity = 500  
a1.sinks.k1.type = logger  
​  
#a1.sinks.k1.type = com.ws.flume.sink.MySqlSink  
#a1.sinks.k1.url = jdbc:mysql://node1:3306/1705e  
#a1.sinks.k1.username = root  
#a1.sinks.k1.password = 123456  
a1.sinks.k1.channel = c1  
​  
#a1.sinks.k1.type = org.apache.flume.sink.kafka.KafkaSink  
#a1.sinks.k1.kafka.topic = test2  
#a1.sinks.k1.kafka.bootstrap.servers = node1:9092  
#a1.sinks.k1.kafka.flumeBatchSize = 10  
#a1.sinks.k1.kafka.producer.acks = 1  
#a1.sinks.k1.kafka.producer.linger.ms = 5  
​

### **测试结果：**



**结论**：使用flume读取日志信息

### **② 实时计算：**

**预期效果**：使用sparkstreaming读取到kafka中的消息，将数据做实时的etl，计算实时指标，存入redis集群，将数据流实时存入hdfs中，将实时计算的结果用springboot+echarts展现出来。

##### **测试方法代码块效果图：**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***存入Redis**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
//用户名商品类别商品名称单价购买数量购买时间  
stream. foreachRDD( kafkaRdd =>  
if( !kafkaRdd . isEmpty()) {  
/ /取出偏移量  
val offsetRanges = kafkaRdd . asInstanceOf [HasOf fsetRanges].of fsetRanges  
//业 务  
val lines = kafkaRdd . map(\_ . value( ))  
//用户名商品类别商品名称单价购买数量购买时间  
val resRedis = lines .map(line => {  
val arr = tine.split( regex = "，”)  
if (arr.length==6){  
((arr(0)+":"+arr(2))， arr(4))  
}else{  
数据格式错误”  
})

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***存入hdfs**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
//用户名商品类别商品名称单价购买数量购买时间  
val resHdfs = lines.map(line => {  
val arr = line.split( regex =”,” )  
if (arr. length == 6) {  
(arr(0),arr(1), arr(2), arr(3),arr(4),arr(5),arr(3). toInt\*arr(4).  
} else  
数据格式错 误"  
})  
printIn( resHdfs.collect( ). toBuffer )  
val tuples = resRedis.collect( )  
println( tuples. toBuffer)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***读取kafka消息**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
SSC . sparkContext.setLogLevel("error")  
val kafkaParams = Map[String, 0bject](  
elems =”bootstrap.servers" ->” node4:9092"  
"key .deserializer" -> classOf[Str ingDeserializer],  
"value.deserializer" -> classOf[StringDeserializer],  
" group.id" -> "use\_ a\_ separate\_ group\_ id\_ for\_ each\_ stream  
"auto.offset. reset" -> "earliest", / /latest  
"enable.auto. commit" -> ( false: java. lang . Boolean)  
)  
val topics = Array( "shopping" )  
/ /创建kafka连接  
val stream: InputDStream[ ConsumerRecord[String, String]] = KafkaUtils. crea  
SSC，  
LocationStrategies. PreferConsistent,  
ConsumerStrategies . Subscribe[String, String]( topics, kafkaParams )  
)  
//用户名商品类别商品名称单价购买数量购买时间  
stream. foreachRDD( kafkaRdd => {

### **测试结果：**

20/03/10 15:15:27 INFO BlockManager: Initialized BlockManager ArrayBuffer(( u004,食品，奶粉,199,5,2020-02-11,995) ) ArrayBuffer(( u004:奶粉,5)) ============kafka偏移量打印==================

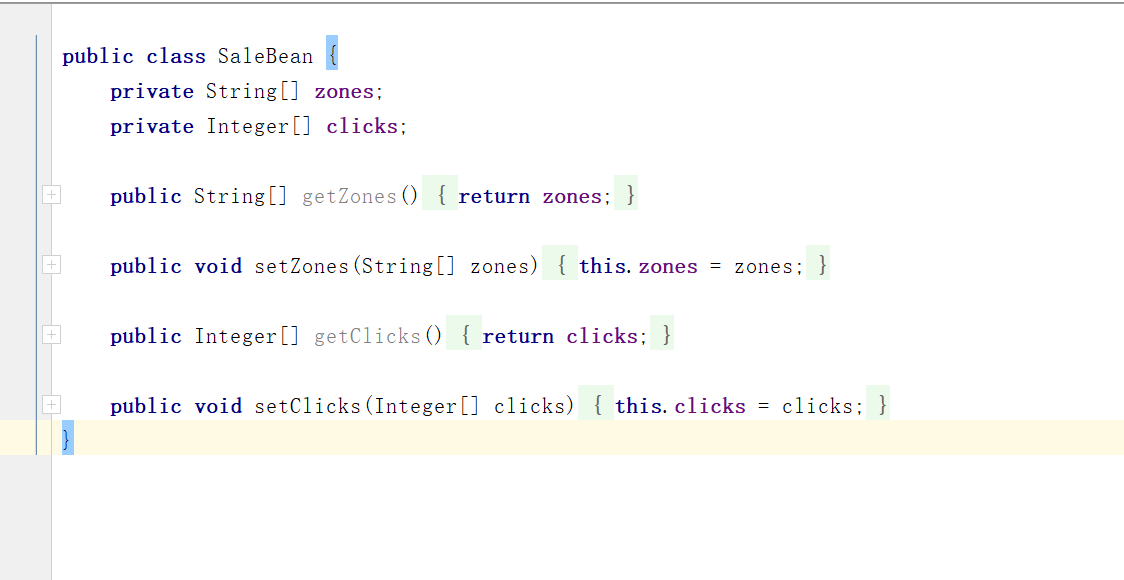
kafkapartition=0 kafkapartitionoffsets=9

**结论**：运行成功，读取到kafka中的消息，将数据做实时的etl，计算实时指标，将数据流实时存入hdfs、Redis中。

### **③ 离线计算：**

**预期效果**：读取实时计算中hdfs中的数据，使用sparkcore从hbase中读已经存在的用户画像，将计算结 果存入hbase，使用hive on spark 统计实时指标，使用sqoop将指标统计报告结果导入到 mysqL中,将离线数据结果用springboot+echarts展现出来

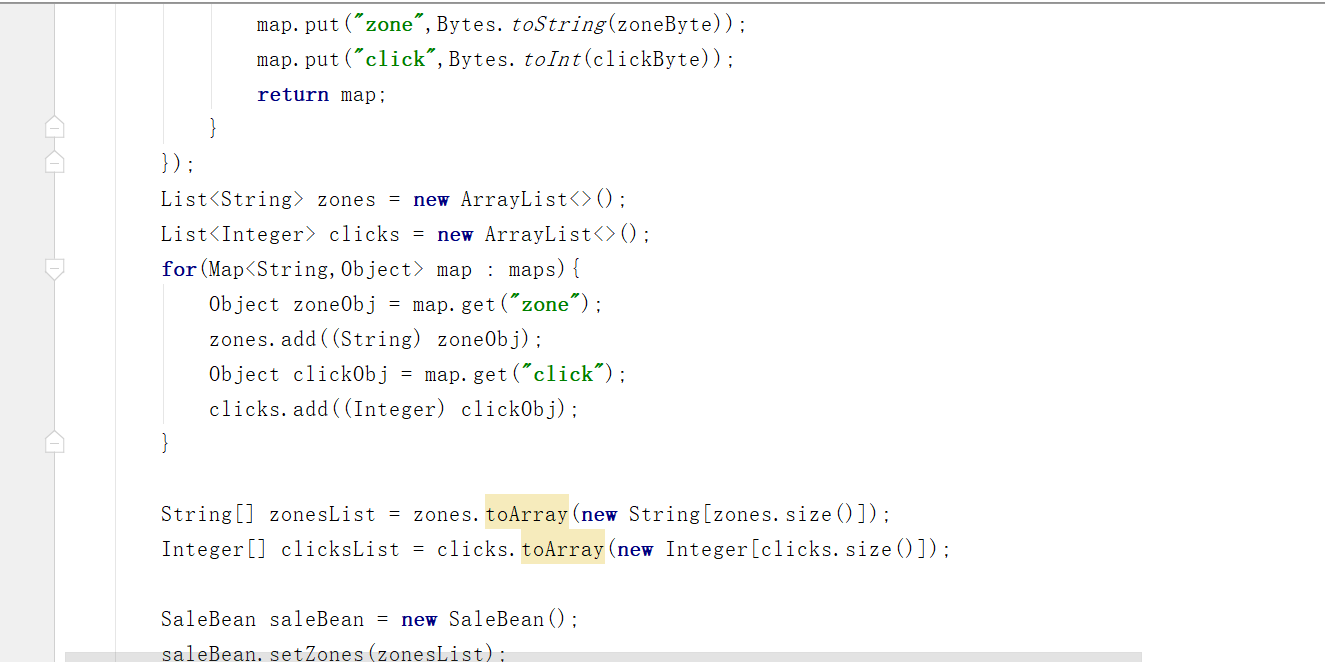
##### **测试方法代码块效果图：**

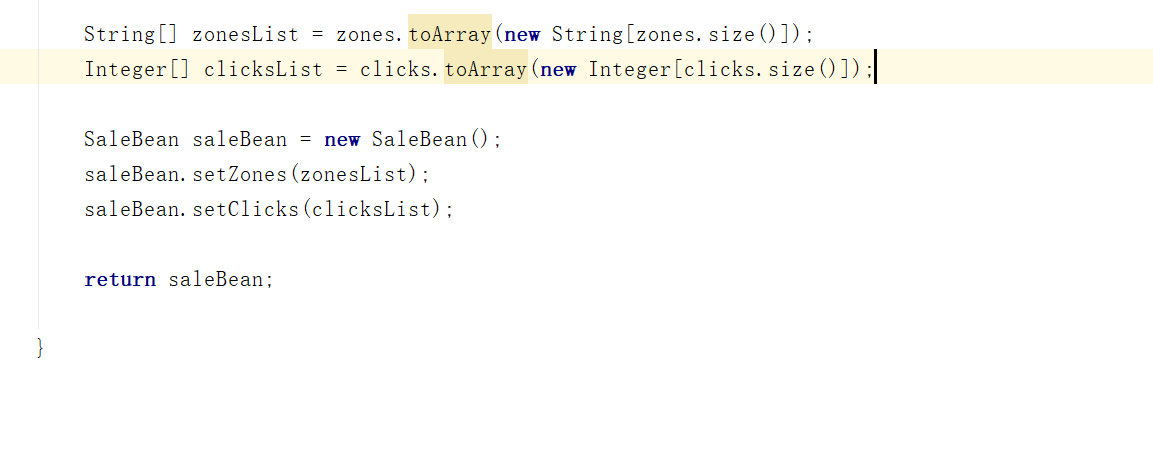


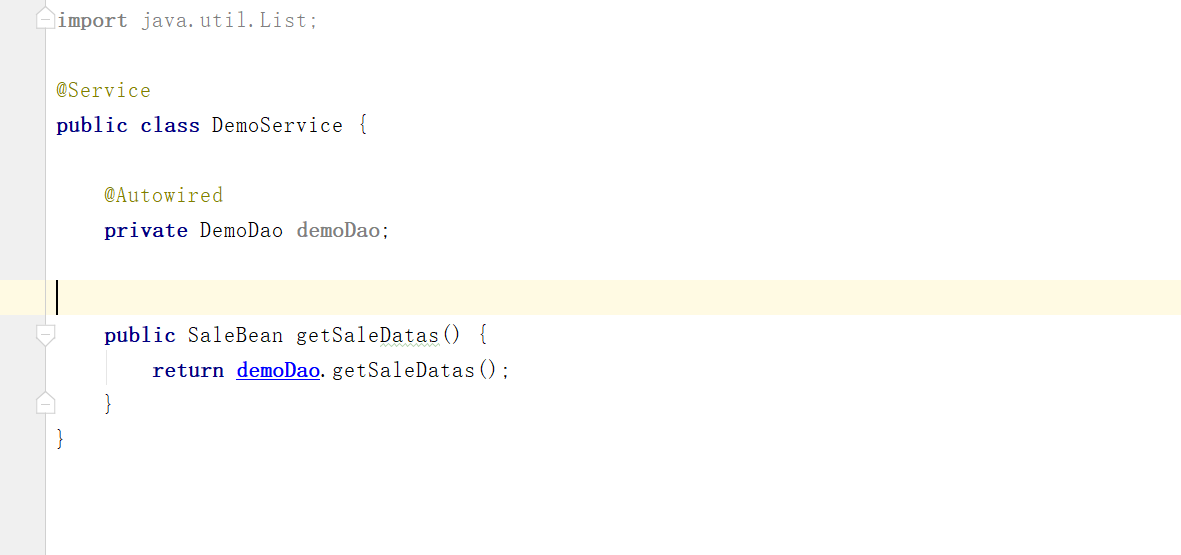








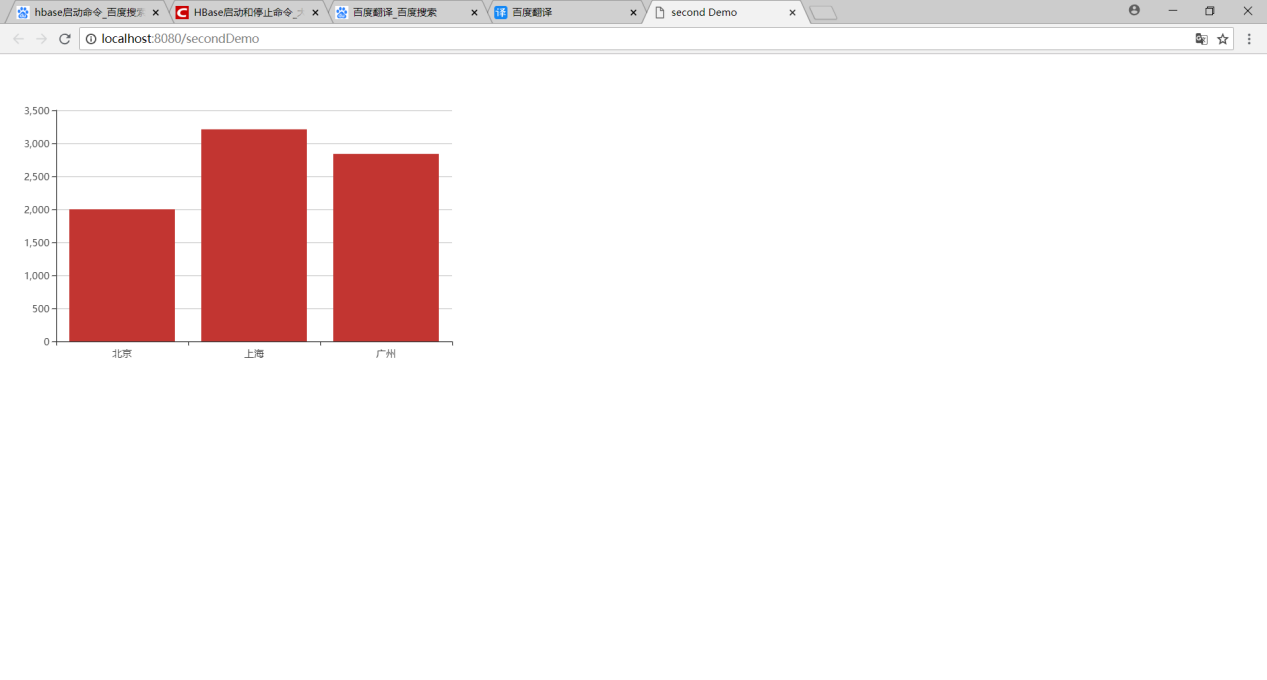








### **测试结果：**



|  |  |  |
| --- | --- | --- |
| 1 | 小明 | Sat Sep 01 08:00:00 CST 2018 |
| 2 | 小月 | Sat Sep 01 08:00:00 CST 2018 |
| 3 | 小鱼 | Wed Jun 06 08:00:00 CST 2018 |
| 4 | 小红 | Sun Aug 04 08:00:00 CST 2019 |
| 5 | 小黑 | Sun Aug 04 08:00:00 CST 2019 |

**结论**：使用hive on spark 统计实时指标，使用sqoop将指标统计报告结果导入到mysqL中,离线数据结果用springboot+echarts展现出来