# kafka

## 实验目的

安装kafka,并整合flume和kafka,整合sparkStreaming和kafka

## 实验内容

## 一、安装kafka,并整合flume和kafka,

### 1.安装kafka

下载

复制kafka到/apps下，进行解压，并删除压缩包

```
cp ~/big_data_tools/kafka_2.12-2.6.0.tgz /apps/
tar -xzvf /apps/kafka_2.12-2.6.0.tgz -C /apps/
mv /apps/kafka_2.12-2.6.0 kafka
rm -rf /apps/kafka_2.12-2.6.0.tgz
```

### 2.使用Kafka

#### 启动zookeeper服务

Kafka 自带了 ZooKeeper，直接使用脚本启动单节点的 ZooKeeper 即可。

```
/apps/kafka/bin/zookeeper-server-start.sh -daemon
/apps/kafka/config/zookeeper.properties
```

加-daemon 参数，在后台启动 Zookeeper，输出的信息保存在执行目录的 logs/zookeeper.out 文件中

#### 启动kafka服务

```
/apps/kafka/bin/kafka-server-start.sh /apps/kafka/config/server.properties
```

```
chen@ubuntu:~$ /apps/kafka/bin/kafka-server-start.sh /apps/kafka/config/server.properties
[2020-12-06 21:53:54,246] INFO Registered kafka:type=kafka.Log4jController MBean (kafka.utils.Log4
jControllerRegistration$)
[2020-12-06 21:53:56,604] INFO Setting -D jdk.tls.rejectClientInitiatedRenegotiation=true to disab
le client-initiated TLS renegotiation (org.apache.zookeeper.common.X509Util)
[2020-12-06 21:53:56,844] INFO Registered signal handlers for TERM, INT, HUP (org.apache.kafka.com
mon.utils.LoggingSignalHandler)
[2020-12-06 21:53:56,870] INFO starting (kafka.server.KafkaServer)
[2020-12-06 21:53:56,873] INFO Connecting to zookeeper on localhost:2181 (kafka.server.KafkaServer
)
[2020-12-06 21:53:56,970] INFO [ZooKeeperClient Kafka server] Initializing a new session to localh
ost:2181. (kafka.zookeeper.ZooKeeperClient)
[2020-12-06 21:53:57,001] INFO Client environment:zookeeper.version=3.5.8-f439ca583e70862c3068a1f2
a7d4d068eec33315, built on 05/04/2020 15:53 GMT (org.apache.zookeeper.ZooKeeper)
[2020-12-06 21:53:57,001] INFO Client environment:host.name=ubuntu (org.apache.zookeeper.ZooKeeper
)
[2020-12-06 21:53:57,001] INFO Client environment:java.version=1.8.0_191 (org.apache.zookeeper.Zoo
Keeper)
```

**创建一个主题**

另外打开一个终端，创建一个名为“first_topic”的主题，只包含一个分区，只有一个副本， 命令如下：

```
/apps/kafka/bin/kafka-topics.sh --create \
--bootstrap-server localhost:9092 \
--replication-factor 1 \
--partitions 1 \
--topic first_topic
```

```
1 chen...   2 chen...
chen@ubuntu:~$ /apps/kafka/bin/kafka-topics.sh --create \
> --bootstrap-server localhost:9092 \
> --replication-factor 1 \
> --partitions 1 \
> --topic first_topic
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore ('_') could
collide. To avoid issues it is best to use either, but not both.
Created topic first_topic.
chen@ubuntu:~$
```

**查看创建的主题**

查看 Kafka 中有哪些已创建的主题，可以用以下命令

```
/apps/kafka/bin/kafka-topics.sh --list \
--bootstrap-server localhost:9092
```

```
chen@ubuntu:~$ /apps/kafka/bin/kafka-topics.sh --list \
> --bootstrap-server localhost:9092
first_topic
chen@ubuntu:~$
```

**删除主题(需要时执行)**

```
/apps/kafka/bin/kafka-topics.sh --delete \
--bootstrap-server localhost:9092 \
--topic first_topic
```

**发送消息到服务中**

运行producer，然后输入一些消息发送到broker。默认情况下，每行将作为单独的消息发送。

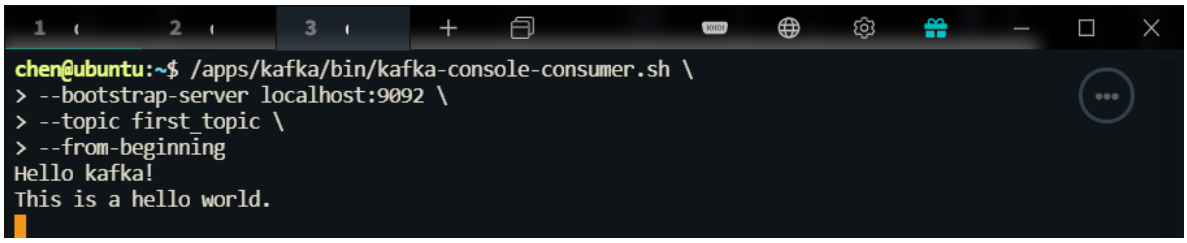
```
/apps/kafka/bin/kafka-console-producer.sh \
--broker-list localhost:9092 \
--topic first_topic
```

```
chen@ubuntu:~$ /apps/kafka/bin/kafka-console-producer.sh \
> --broker-list localhost:9092 \
> --topic first_topic
>Hello kafka!
>This is a hello world.
>
```

**从服务中获取消息**

再打开一个终端运行consumer，从broker中获取已有的消息

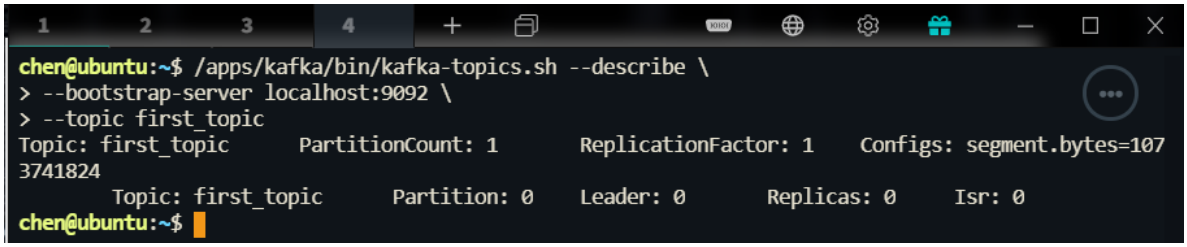
```
/apps/kafka/bin/kafka-console-consumer.sh \
--bootstrap-server localhost:9092 \
--topic first_topic \
--from-beginning
```




**查看运行信息**

另外再打开一个终端，运行以下命令查看运行信息

```
/apps/kafka/bin/kafka-topics.sh --describe \
--bootstrap-server localhost:9092 \
--topic first_topic
```




第一个行显示所有 Partition 的一个总结，以下每一行给出一个 Partition 中的信息，我们 只有一个 Partition，所以只显示一行。

**Leader**：是在给出的所有 Partitons 中负责读写的节点，每个节点都有可能成为 Leader

**Replicas**：显示给定 Partiton 所有副本所存储节点的节点列表，不管该节点是否是 Leader 或者是否存活。

**Isr**：副本都已同步的节点集合，这个集合中的所有节点都是存活状态，并且跟 Leader 同步

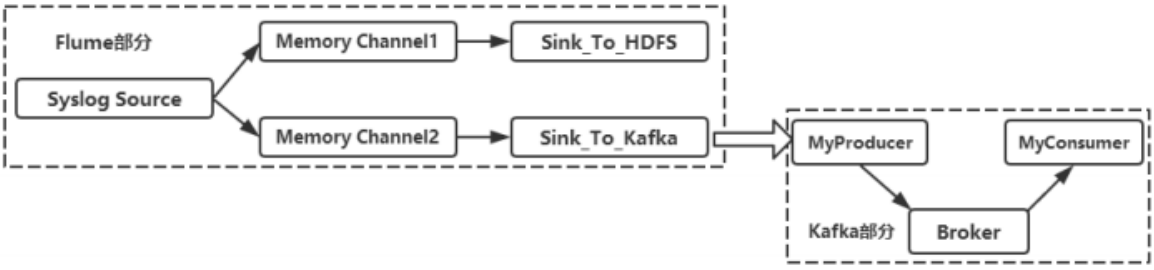
**关闭zookeeper**

需要时再关闭，这里可以先不关闭。

```
/apps/kafka/bin/zookeeper-server-stop.sh -daemon config/zookeeper.properties
```

## 3.flume传输数据给kafka

新建 Flume 的配置文件/apps/flume/conf/syslog_mem_hdfskafka.conf，使用 Flume 抓 取 syslog 端口的日志数据，使用 mem 作为 Channel，一个输出是将数据存储到 HDFS 中 的/myflume/目录下，作为持久存储；另一个输出是将数据传递给 Kafka 进行使用，Kafka 端启用 console-consumer 来消费数据，并输出到屏幕上。




**创建 Flume 的配置文件/apps/flume/conf/syslog_mem_hdfsandkafka.conf**

```
#定义各个组件
agent1.sources = src
agent1.channels = ch_hdfs ch_kafka
agent1.sinks = des_hdfs des_kafka
#把组件关联起来
agent1.sources.src.channels = ch_hdfs ch_kafka
agent1.sinks.des_hdfs.channel = ch_hdfs
agent1.sinks.des_kafka.channel = ch_kafka
#配置 source
agent1.sources.src.type = syslogtcp
agent1.sources.src.bind = localhost
agent1.sources.src.port = 6666
#配置 channel
agent1.channels.ch_hdfs.type = memory
agent1.channels.ch_kafka.type = memory
#配置 hdfs sink
agent1.sinks.des_hdfs.type = hdfs
agent1.sinks.des_hdfs.hdfs.path = hdfs://localhost:9000/myflume/
agent1.sinks.des_hdfs.hdfs.useLocalTimeStamp = true
#设置 flume 临时文件的前缀为_
agent1.sinks.des_hdfs.hdfs.inUsePrefix=_
#设置 flume 写入文件的前缀
agent1.sinks.des_hdfs.hdfs.filePrefix = q7
agent1.sinks.des_hdfs.hdfs.fileType = DataStream
agent1.sinks.des_hdfs.hdfs.writeFormat = Text
#配置 kafka sink
agent1.sinks.des_kafka.type = org.apache.flume.sink.kafka.KafkaSink
agent1.sinks.des_kafka.brokerList = localhost:9092
agent1.sinks.des_kafka.topic = flumekafka
agent1.sinks.des_kafka.batchSize=100
agent1.sinks.des_kafka.requiredAcks=1
```

**启动hadoop**

```
/apps/hadoop/sbin/start-all.sh
```

```
chen@ubuntu:~$ /apps/hadoop/sbin/start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as chen in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [ubuntu]
Starting resourcemanager
Starting nodemanagers
```

**启动zookeeper(如果还没有开启)**

```
/apps/kafka/bin/zookeeper-server-start.sh \
-daemon /apps/kafka/config/zookeeper.properties
```

**启动kafka-server(如果还没有开启)**

```
/apps/kafka/bin/kafka-server-start.sh /apps/kafka/config/server.properties
```

**创建topic**

再打开一个终端，在 Kafka 中创建名为 flumekafka 的 topic

```
/apps/kafka/bin/kafka-topics.sh --create \
--bootstrap-server localhost:9092 \
--replication-factor 1 \
--partitions 1 \
--topic flumekafka
```

```
chen@ubuntu:~$ /apps/kafka/bin/kafka-topics.sh --create \
> --bootstrap-server localhost:9092 \
> --replication-factor 1 \
> --partitions 1 \
> --topic flumekafka
Created topic flumekafka.
```

**启动flume**

```
flume-ng agent --conf conf \
--conf-file /apps/flume/conf/syslog_mem_hdfsandkafka.conf \
--name agent1 \
-Dflume.root.logger=DEBUG,console
```

```
chen@ubuntu:~$ flume-ng agent --conf conf --conf-file /apps/flume/conf/syslog_mem_hdfsandkafka.con
f --name agent1 -Dflume.root.logger=DEBUG,console
Warning: JAVA_HOME is not set!
Info: Including Hadoop libraries found via (/apps/hadoop/bin/hadoop) for HDFS access
/apps/hadoop/bin/../libexec/hadoop-functions.sh: line 2326: HADOOP_ORG.APACHE.FLUME.TOOLS.GETJAVAP
ROPERTY_USER: bad substitution
/apps/hadoop/bin/../libexec/hadoop-functions.sh: line 2421: HADOOP_ORG.APACHE.FLUME.TOOLS.GETJAVAP
ROPERTY_OPTS: bad substitution
Info: Including HBASE libraries found via (/apps/hbase/bin/hbase) for HBASE access
Info: Including Hive libraries found via (/apps/hive) for Hive access
+ exec /apps/java/bin/java -Xmx20m -Dflume.root.logger=DEBUG,console -cp 'conf:/apps/flume/lib/*:/
apps/hadoop/etc/hadoop:/apps/hadoop/share/hadoop/common/lib/*:/apps/hadoop/share/hadoop/common/*:/
apps/hadoop/share/hadoop/hdfs:/apps/hadoop/share/hadoop/hdfs/lib/*:/apps/hadoop/share/hadoop/hdfs/
*:/apps/hadoop/share/hadoop/mapreduce/*:/apps/hadoop/share/hadoop/yarn:/apps/hadoop/share/hadoop/y
```

**启动consumer**

另外打开一个终端，启动 Kafka 的 console consumer 来消费数据

```
/apps/kafka/bin/kafka-console-consumer.sh \
--bootstrap-server localhost:9092 \
--topic flumekafka \
--from-beginning
```

**发送数据**

再打开一个终端使用 nc 命令向 6666 端口发送数据

```
echo "hello can you hear me?" | nc localhost 6666
```

**查看consumer输出**

查看 Kafka 的 console consumer 是否有内容输出

```
chen@ubuntu:~$ /apps/kafka/bin/kafka-console-consumer.sh \
> --bootstrap-server localhost:9092 \
> --topic first_topic \
> --from-beginning
Hello kafka!
This is a hello world.
^CProcessed a total of 2 messages
chen@ubuntu:~$ /apps/kafka/bin/kafka-console-consumer.sh \
> --bootstrap-server localhost:9092 \
> --topic flumekafka \
> --from-beginning
hello can you hear me?
```

**查看flume-ng输出**

```
2020-12-06 22:26:59,388 WARN source.SyslogUtils: Event created from Invalid Syslog data.
2020-12-06 22:27:01,774 INFO hdfs.HDFSDataStream: Serializer = TEXT, UseRawLocalFileSystem = false
2020-12-06 22:27:02,041 INFO hdfs.BucketWriter: Creating hdfs://localhost:9000/myflume//_q7.160732
2421775.tmp
2020-12-06 22:27:02,745 INFO clients.Metadata: Cluster ID: UraDUtA0TuidxcEKsLM1jw
2020-12-06 22:27:35,778 INFO hdfs.HDFSEventSink: Writer callback called.
2020-12-06 22:27:35,779 INFO hdfs.BucketWriter: Closing hdfs://localhost:9000/myflume//_q7.1607322
421775.tmp
2020-12-06 22:27:35,864 INFO hdfs.BucketWriter: Renaming hdfs://localhost:9000/myflume/_q7.1607322
421775.tmp to hdfs://localhost:9000/myflume/q7.1607322421775
```

**查看hdfs结果**

```
hadoop fs -cat /myflume/*
```

```
chen@ubuntu:~$ hadoop fs -cat /myflume/*
hello can you hear me?
chen@ubuntu:~$
```

# 二、整合sparkStreaming和kafka

## 1.准备工作

1.将 spark-streaming-kafka-0-8-assembly_2.11-2.4.3.jar 复制到/apps/spark/jars

```
cp ~/big_data_tools/spark-streaming-kafka-0-8-assembly_2.11-2.4.3.jar
/apps/spark/jars/
```

注：spark-streaming-kafka-0-8-assembly_2.11-2.4.3.jar 为 Kafka 作为 Spark Streaming 的数据源依赖的库。下载地址为： https://search.maven.org/artifact/org.apache.spark/spark-streaming-kafka-0-8- assembly_2.11/2.4.3/jar

2. 在目录/apps/spark/jars 中创建文件夹 kafka，并将 Kafka 安装目录 libs 下的所有 jar 文件复制其中

```
mkdir /apps/spark/jars/kafka
cp /apps/kafka/libs/*.jar /apps/spark/jars/kafka
```

3. 安装 Python 的 Kafka 库

```
sudo pip3 install kafka -i https://pypi.tuna.tsinghua.edu.cn/simple
```

4. 安装 Python 连接 MySQL 的模块 PyMySQL

```
sudo pip3 install PyMySQL -i https://pypi.tuna.tsinghua.edu.cn/simple
```

```
chen@ubuntu:/apps/spark/jars/kafka$ sudo pip3 install PyMySQL -i  https://pypi.tuna.tsinghua.edu.c
n/simple
The directory '/home/chen/.cache/pip/http' or its parent directory is not owned by the current use
r and the cache has been disabled. Please check the permissions and owner of that directory. If ex
ecuting pip with sudo, you may want sudo's -H flag.
The directory '/home/chen/.cache/pip' or its parent directory is not owned by the current user and
 caching wheels has been disabled. check the permissions and owner of that directory. If executing
 pip with sudo, you may want sudo's -H flag.
Collecting PyMySQL
  Retrying (Retry(total=4, connect=None, read=None, redirect=None, status=None)) after connection
broken by 'ReadTimeoutError("HTTPSConnectionPool(host='pypi.tuna.tsinghua.edu.cn', port=443): Read
 timed out. (read timeout=15)",)': /simple/pymysql/
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/1a/ea/dd9c81e2d85efd03cfbf808736dd055bd9e
a1a78aea9968888b1055c3263/PyMySQL-0.10.1-py2.py3-none-any.whl (47kB)
    100% |████████████████████████████████| 51kB 2.4MB/s
Installing collected packages: PyMySQL
Successfully installed PyMySQL-0.10.1
chen@ubuntu:/apps/spark/jars/kafka$
```

## 2.Kafka 作为 Streaming 数据源

1.启动hadoop

```
/apps/hadoop/sbin/start-all.sh
```

2.启动 ZooKeeper 服务

```
cd /apps/kafka
bin/zookeeper-server-start.sh -daemon config/zookeeper.properties
```

3.启动kafka

```
bin/kafka-server-start.sh config/server.properties
```

4.创建主题

另外打开一个终端，创建一个名为“sparkapp”的主题，只包含一个分区，只有一个副本

```
bin/kafka-topics.sh --create \
--bootstrap-server localhost:9092 \
--replication-factor 1 \
--partitions 1 \
--topic sparkapp
```

```
chen@ubuntu:~$ /apps/kafka/bin/kafka-topics.sh --create \
> --bootstrap-server localhost:9092 \
> --replication-factor 1 \
> --partitions 1 \
> --topic sparkapp
Created topic sparkapp.
```

5.确认一下主题是否创建成功 （基于内存，下次开机就没有了）

```
chen@ubuntu:~$ /apps/kafka/bin/kafka-topics.sh --list \
> --bootstrap-server localhost:9092
sparkapp
chen@ubuntu:~$
```

6.编写代码

在~/pyspark-workspace/streaming/中新建目录 kafka

```
cd ~/pyspark-workspace/streaming/
mkdir kafka
```

在其中创建文件 StreamingKafka.py

```
cd kafka
vim StreamingKafka.py
```

写入下面的代码

```
#!/usr/bin/env python
# coding=utf-8
from pyspark import SparkContext,SparkConf
from pyspark.streaming import StreamingContext
from pyspark.streaming.kafka import KafkaUtils
conf = SparkConf()
conf.setAppName('StreamingKafka').set('spark.io.compression.codec','snappy')
conf.setMaster('local[2]')
sc = SparkContext(conf = conf)
ssc = StreamingContext(sc, 5)
brokers = 'localhost:9092'
topic = 'sparkapp'

kafka_streaming_rdd = KafkaUtils.createDirectStream(ssc,[topic],
{"metadata.broker.list":brokers})
lines_rdd = kafka_streaming_rdd.map(lambda x: x[1])
counts = lines_rdd.flatMap(lambda line: line.strip().split(" ")) \
          .map(lambda word: (word,1)) \
          .reduceByKey(lambda a,b:a+b)
counts.pprint()
ssc.start()
ssc.awaitTermination()
```

7.提交任务

注意提交任务时要指定依赖的库

```
nohup spark-submit --jars /apps/spark/jars/spark-streaming-kafka-0-8-
assembly_2.11-2.4.3.jar StreamingKafka.py>file 2>&1
```

启动成功后，Streaming 进入循环监听状态

```
tail -F file # 实时显示文件的更新
```

```
file ▯                                                                    buffers
nohup: ignoring input
20/12/08 20:09:25 WARN Utils: Your hostname, ubuntu resolves to a loopback address: 127.0.1.1;
20/12/08 20:09:25 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
20/12/08 20:09:25 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform
-------------------------------------------
Time: 2020-12-08 20:09:30
-------------------------------------------

-------------------------------------------
Time: 2020-12-08 20:09:35
-------------------------------------------

-------------------------------------------
Time: 2020-12-08 20:09:40
-------------------------------------------
```

8.生成数据

再打开一个终端，运行 Kafka producer，然后输入一些消息发送到服务器

```
/apps/kafka/bin/kafka-console-producer.sh \
--broker-list localhost:9092 \
--topic sparkapp
```

```
chen@ubuntu:~/pyspark-workspace/streaming/kafka$ /apps/kafka/bin/kafka-console-producer.sh \
> --broker-list localhost:9092 \
> --topic sparkapp
>hello world
>
```

9.查看结果

```
Time: 2020-12-09 00:18:50
-------------------------------------------
('world', 1)
('hello', 1)
```

10. 编写python程序创建producer

接下来，使用 Python 的 kafka 包编写 Python 程序创建 Producer。在 ~/pysparkworkspace/streaming/kafka 中创建文件 kafka_producer.py 写入以下内容

```
from kafka import KafkaProducer
import time
producer = KafkaProducer()
with open("/data/testfile") as f:
    for line in f.readlines():
        time.sleep(1)
        producer.send("sparkapp",line.encode('utf-8'))
        print(line)
        producer.flush()
```

11.运行kafka_producer.py

```
python3 kafka_producer.py
```

```
>^Cchen@ubuntu:~/pyspark-workspace/streaming/kafka$ python3 kafka_producer.py
hello python

hello flume

hello spark streaming

hello world

chen@ubuntu:~/pyspark-workspace/streaming/kafka$
```

12.再次查看结果

由于kafka_producer.py把文件里的内容读入到kafka的producer中，所以可以被streaming监测到。

在 StreamingKafka.py 运行窗口就可以看到词频统计结果，看到文件前两行处理一次。

```
-------------------------------------------
Time: 2020-12-09 00:24:55
-------------------------------------------
('python', 1)
('hello', 2)
('flume', 1)

-------------------------------------------
Time: 2020-12-09 00:25:00
-------------------------------------------
('streaming', 1)
('world', 1)
('hello', 2)
('spark', 1)

-------------------------------------------
Time: 2020-12-09 00:25:05
```

使用 Ctrl-c，终止 StreamingKafka.py 程序，其它终端窗口和进程不要关闭，下面还要使 用。

## 3.将Streaming结果存储到mysql

1.进入数据库

```
mysql -u root -p
```

2.创建数据库spark

```
create database spark;
use spark;
```

3.新建表 wordcount 用于存储词频统计结果

```
create table wordcount (word char(20), count int(4));
```

4.查看表结构，

```
desc wordcount;
```

```
mysql> desc wordcount;
+-------+----------+------+-----+---------+-------+
| Field | Type     | Null | Key | Default | Extra |
+-------+----------+------+-----+---------+-------+
| word  | char(20) | YES  |     | NULL    |       |
| count | int(4)   | YES  |     | NULL    |       |
+-------+----------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

5.编写程序

在~/pyspark-workspace/streaming/kafka 中创建文件 StreamingKafkaToMysql.py，写入以下内容

```
#!/usr/bin/env python
# coding=utf-8
from pyspark import SparkContext,SparkConf
from pyspark.streaming import StreamingContext
from pyspark.streaming.kafka import KafkaUtils
import pymysql

##############################################################################
############
def dbfunc(records):
    db=pymysql.connect("localhost","root","123456","spark")
    cursor=db.cursor()

    def doinsert(p):
        sql="insert into wordcount(word,count) values ('%s','%s')" %
(str(p[0]),str(p[1]))

        try:
            cursor.execute(sql)
            db.commit()
        except:
            db.rollback()

    for item in records:
        doinsert(item)

def func(rdd):
    repartitionedRDD=rdd.repartition(3)
    repartitionedRDD.foreachPartition(dbfunc)
##############################################################################
##############
conf=SparkConf()
conf.setAppName('StreamingKafka').set('spark.io.compression.codec','snappy')
conf.setMaster('local[2]')
sc=SparkContext(conf=conf)
ssc=StreamingContext(sc,5)
brokers='localhost:9092'
topic='sparkapp'
# 使用streaming直接模式消费kafka
kafka_streaming_rdd=KafkaUtils.createDirectStream(ssc,[topic],
{"metadata.broker.list":brokers})
lines_rdd=kafka_streaming_rdd.map(lambda x:x[1])

counts=lines_rdd.flatMap(lambda line:line.strip().split(" ")).map(lambda word:
(word,1)).reduceByKey(lambda a,b:a+b)

counts.pprint()
```

```
# 与StreamingKafka.py不同的地方
# 写入mysql数据库
counts.foreachRDD(func)
ssc.start()
ssc.awaitTermination()
```

6.启动spark streaming

```
spark-submit --jars /apps/spark/jars/spark-streaming-kafka-0-8-assembly_2.11-2.4.3.jar StreamingKafkaToMysql.py
```

7.运行kafka_producer.py

```
python3 kafka_producer.py
```

8.查看结果

在 StreamingKafka.py 运行窗口就可以看到词频统计结果

```
chen@ubuntu:~/pyspark-workspace/streaming/kafka$ spark-submit --jars /apps/spark/jars/spark-streaming-kafka-0-8-assembly_2.11-2.4.3.jar StreamingKafkaToMysql.py
20/12/08 23:55:20 WARN Utils: Your hostname, ubuntu resolves to a loopback address: 127.0.1.1; using 10.0.0.135 instead (on interface ens33)
20/12/08 23:55:20 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
20/12/08 23:55:21 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
-------------------------------------------
Time: 2020-12-08 23:55:30
-------------------------------------------
('python', 1)
('hello', 1)

-------------------------------------------
Time: 2020-12-08 23:55:35
-------------------------------------------
('streaming', 1)
('world', 1)
('hello', 3)
('flume', 1)
('spark', 1)

-------------------------------------------
Time: 2020-12-08 23:55:40
-------------------------------------------
```

9.查看Mysql中的结果

```
select * from wordcount;
```

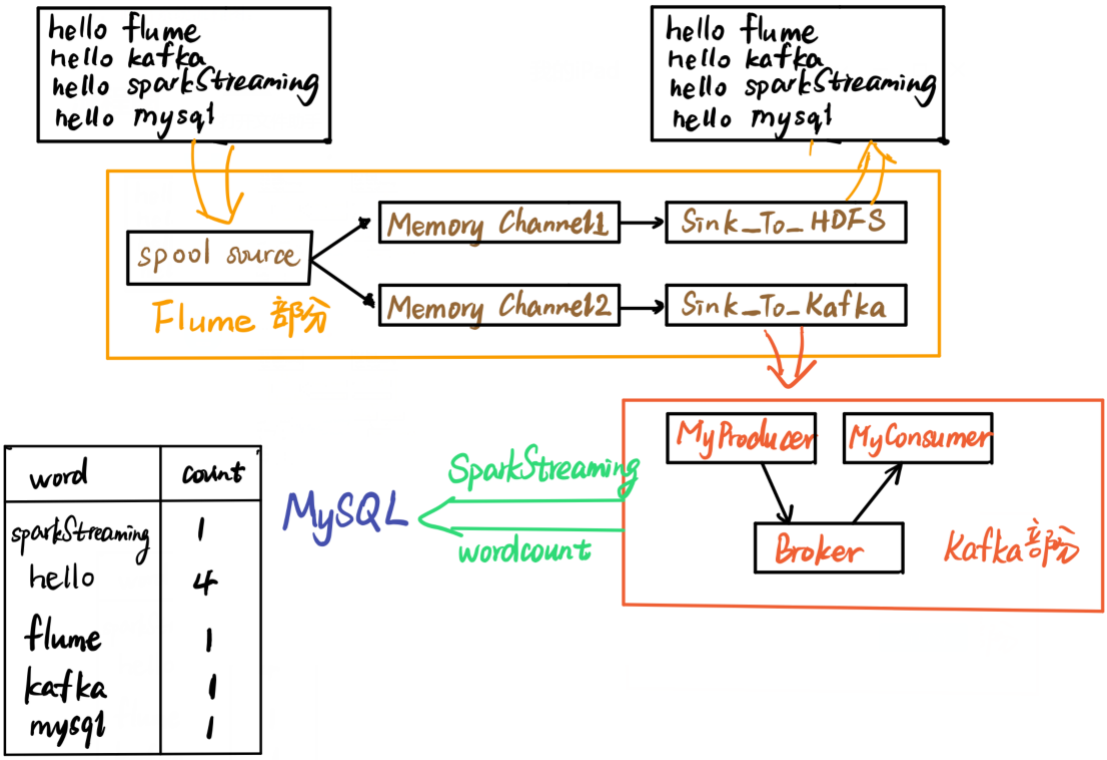
```
Database changed
mysql> select * from wordcount;
+-----------+-------+
| word      | count |
+-----------+-------+
| python    |     1 |
| hello     |     1 |
| streaming |     1 |
| world     |     1 |
| hello     |     3 |
| flume     |     1 |
| spark     |     1 |
+-----------+-------+
7 rows in set (0.00 sec)
```

# 综合实验

使用 Flume，Kafka 和 Spark streaming 构建一个完整的流数据处理系统，要求

```
1. Flume 的源类型为 Spooling Directory Source，
2. Flume 将数据存入 HDFS 和传给 Kafka，
3. Kafka 将数据传给 Spark Streaming 进行流式处理，
4. Spark Streaming 处理完以后将数据存入 Mysql。
5. 画出系统流程图。
```

## 流程图




在/apps/flume/conf/case_spool_com.conf文件写入如下内容

```
#定义各个组件
agent1.sources = src
agent1.channels = ch_hdfs ch_kafka
agent1.sinks = des_hdfs des_kafka
#把组件关联起来
agent1.sources.src.channels = ch_hdfs ch_kafka
agent1.sinks.des_hdfs.channel = ch_hdfs
agent1.sinks.des_kafka.channel = ch_kafka
#配置 source
agent1.sources.src.type = spooldir
agent1.sources.src.spoolDir = /home/chen/flume/spool/logs #下面放待测试的文件
wordcount.txt
agent1.sources.src.fileHeader = true
```

```
#配置 channel
agent1.channels.ch_hdfs.type = memory
agent1.channels.ch_kafka.type = memory
#配置 hdfs sink
agent1.sinks.des_hdfs.type = hdfs
agent1.sinks.des_hdfs.hdfs.path = hdfs://localhost:9000/myflume_com/
agent1.sinks.des_hdfs.hdfs.useLocalTimeStamp = true
#设置 flume 临时文件的前缀为_
agent1.sinks.des_hdfs.hdfs.inUsePrefix=_
#设置 flume 写入文件的前缀
agent1.sinks.des_hdfs.hdfs.filePrefix = q7
agent1.sinks.des_hdfs.hdfs.fileType = DataStream
agent1.sinks.des_hdfs.hdfs.writeFormat = Text
#配置 kafka sink
agent1.sinks.des_kafka.type = org.apache.flume.sink.kafka.KafkaSink
agent1.sinks.des_kafka.brokerList = localhost:9092
agent1.sinks.des_kafka.topic = flume_kafka_streaming_mysql
agent1.sinks.des_kafka.batchSize=100
agent1.sinks.des_kafka.requiredAcks=1
```

**启动hadoop**

```
/apps/hadoop/sbin/start-all.sh
```

```
chen@ubuntu:~$ /apps/hadoop/sbin/start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as chen in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [ubuntu]
Starting resourcemanager
Starting nodemanagers
```

**启动zookeeper(如果还没有开启)**

```
/apps/kafka/bin/zookeeper-server-start.sh -daemon
/apps/kafka/config/zookeeper.properties
```

**启动kafka-server(如果还没有开启)**

```
/apps/kafka/bin/kafka-server-start.sh /apps/kafka/config/server.properties
```

**创建topic**

再打开一个终端，在 Kafka 中创建名为 flume_kafka_streaming_mysql 的 topic

```
/apps/kafka/bin/kafka-topics.sh --create \
--bootstrap-server localhost:9092 \
--replication-factor 1 \
--partitions 1 \
--topic flume_kafka_streaming_mysql
```

---

## Spark Streaming 处理完以后将数据存入 Mysql

1.进入数据库

```
mysql -u root -p
```

2.创建数据库spark(存在不用创建)

```
create database spark;
use spark;
```

3.新建表 wordcount 用于存储词频统计结果

```
create table wordcount2 (word char(20), count int(4));
```

4.查看表结构，

```
desc wordcount2;
```

```
mysql> desc wordcount2;
+-------+----------+------+-----+---------+-------+
| Field | Type     | Null | Key | Default | Extra |
+-------+----------+------+-----+---------+-------+
| word  | char(20) | YES  |     | NULL    |       |
| count | int(4)   | YES  |     | NULL    |       |
+-------+----------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

5.编写程序

在~/pyspark-workspace/streaming/kafka 中创建文件 StreamingKafkaToMysql.py，写入以下内容

```
#!/usr/bin/env python
# coding=utf-8
from pyspark import SparkContext,SparkConf
from pyspark.streaming import StreamingContext
from pyspark.streaming.kafka import KafkaUtils
import pymysql

################################################################################
############
def dbfunc(records):
    db=pymysql.connect("localhost","root","123456","spark")
    cursor=db.cursor()

    def doinsert(p):
        sql="insert into wordcount2(word,count) values ('%s','%s')" %
(str(p[0]),str(p[1]))

        try:
            cursor.execute(sql)
            db.commit()
        except:
            db.rollback()

    for item in records:
        doinsert(item)

def func(rdd):
    repartitionedRDD=rdd.repartition(3)
    repartitionedRDD.foreachPartition(dbfunc)
```

```
##############################################################################
##############
conf=SparkConf()
conf.setAppName('StreamingKafka').set('spark.io.compression.codec','snappy')
conf.setMaster('local[2]')
sc=SparkContext(conf=conf)
ssc=StreamingContext(sc,5)
brokers='localhost:9092'
topic='flume_kafka_streaming_mysql'
# 使用streaming直接模式消费kafka
kafka_streaming_rdd=KafkaUtils.createDirectStream(ssc,[topic],
{"metadata.broker.list":brokers})
lines_rdd=kafka_streaming_rdd.map(lambda x:x[1])

counts=lines_rdd.flatMap(lambda line:line.strip().split(" ")).map(lambda word:
(word,1)).reduceByKey(lambda a,b:a+b)

counts.pprint()
# 与StreamingKafka.py不同的地方
# 写入mysql数据库
counts.foreachRDD(func)
ssc.start()
ssc.awaitTermination()
```

**启动spark streaming**

```
spark-submit --jars /apps/spark/jars/spark-streaming-kafka-0-8-assembly_2.11-
2.4.3.jar StreamingKafkaToMysql.py
```

---

**启动flume**

```
flume-ng agent --conf conf \
--conf-file /apps/flume/conf/case_spool_com.conf \
--name agent1 \
-Dflume.root.logger=DEBUG,console
```

再开一个终端，在flume的检测目录~flume/spool/logs下创建wordcount2.txt

```
hello flume
hello kafka
hello sparkStreaming
hello mysql
```

**查看结果**

在flume-ng终端看到输出到Hdfs如下：

```
2020-12-09 05:58:42,583 INFO instrumentation.MonitoredCounterGroup: Component type: SINK, name: des
_kafka started
2020-12-09 05:58:42,687 INFO avro.ReliableSpoolingFileEventReader: Last read took us just up to a f
ile boundary. Rolling to the next file, if there is one.
2020-12-09 05:58:42,687 INFO avro.ReliableSpoolingFileEventReader: Preparing to move file /home/che
n/flume/spool/logs/wordcount.txt to /home/chen/flume/spool/logs/wordcount.txt.COMPLETED
2020-12-09 05:58:42,698 INFO hdfs.HDFSDataStream: Serializer = TEXT, UseRawLocalFileSystem = false
2020-12-09 05:58:42,894 INFO hdfs.BucketWriter: Creating hdfs://localhost:9000/myflume_com//_q7.160
7522322699.tmp
2020-12-09 05:58:43,047 INFO clients.Metadata: Cluster ID: uaC5Ssk-RDqXD8CE7smmSA
2020-12-09 05:59:14,155 INFO hdfs.HDFSEventSink: Writer callback called.
2020-12-09 05:59:14,155 INFO hdfs.BucketWriter: Closing hdfs://localhost:9000/myflume_com// q7.1607
522322699.tmp
2020-12-09 05:59:14,199 INFO hdfs.BucketWriter: Renaming hdfs://localhost:9000/myflume_com/_q7.1607
522322699.tmp to hdfs://localhost:9000/myflume_com/q7.1607522322699
```

```
hadoop fs -cat /myflume_com/q7.1607522322699
```

```
chen@ubuntu:~$ hadoop fs -cat /myflume_com/q7.1607522322699
hello flume
hello kafka
hello sparkStreaming
hello mysql
```

在StreamingKafkaToMysql.py的终端看到如下：

```
-------------------------------------------
Time: 2020-12-09 05:58:45
-------------------------------------------
('sparkStreaming', 1)
('hello', 4)
('flume', 1)
('kafka', 1)
('mysql', 1)

-------------------------------------------
Time: 2020-12-09 05:58:50
-------------------------------------------
```

查看Mysql中的结果

```
select * from wordcount2;
```

```
mysql> select * from wordcount2;
+----------------+-------+
| word           | count |
+----------------+-------+
| sparkStreaming |     1 |
| hello          |     4 |
| flume          |     1 |
| kafka          |     1 |
| mysql          |     1 |
+----------------+-------+
5 rows in set (0.00 sec)
```