整合 Spark Streaming 与 Kafka

刘磊

2020 年 12 月

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
```

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 服务

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

4. 创建一个主题

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
```

5. 确认一下主题是否创建成功

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

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

__consumer_offsets
first_topic
sparkapp
```

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

```
from pyspark import SparkContext,SparkConf
from pyspark.streaming import StreamingContext
from pyspark.streaming.kafka import KafkaUtils

conf = SparkConf()
conf.setAppName('StreamingKafka')
conf.setMaster('local[2]')
sc = SparkContext(conf = conf)
ssc = StreamingContext(sc, 5)
brokers = 'localhost:9092'
topic = 'sparkapp'
# 使用 streaming 直连模式消费 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()
ssc.start()
ssc.awaitTermination()
```

7. 提交任务

注意提交任务时要指定依赖的库

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

启动成功后，Streaming 进入循环监听状态

```
-------------------------------------------
Time: 2020-12-08 19:47:20
-------------------------------------------

-------------------------------------------
Time: 2020-12-08 19:47:25
-------------------------------------------
```

8. 生成数据

再打开一个终端，运行 Kafka 生产者，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
```

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

9. 查看结果

将会在 StreamingKafka.py 运行窗口看到处理后的数据

```
-------------------------------------------
Time: 2020-12-02 17:27:10
-------------------------------------------
('hello', 1)
('kafka', 1)

-------------------------------------------
Time: 2020-12-02 17:27:15
-------------------------------------------
('streaming', 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~/pyspark-workspace/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
```

```
lei@ubuntu:~/pyspark-workspace/streaming/kafka$ python3 kafka_producer.py
hello world

hello hdfs

hello big data
```

12. 再次查看结果

在 StreamingKafka.py 运行窗口就可以看到词频统计结果

```
-------------------------------------------
Time: 2020-12-08 20:25:50
-------------------------------------------
('world', 1)
('hdfs', 1)
('hello', 3)
('big', 1)
('data', 1)
```

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。

将 Streaming 结果存储到 Mysql

1. 打开一个终端启动 MySQL

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

2. 打开数据库 spark，如果不存在先进行创建

```
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，不要关闭 MySQL，后面还要查看运行结果。

```
desc wordcount;
```

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

5. 编写程序

在~/pyspark-workspace/streaming/kafka 中创建文件

StreamingKafkaToMysql.py，写入以下内容

```
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')
conf.setMaster('local[2]')
sc = SparkContext(conf = conf)
ssc = StreamingContext(sc, 5)
brokers = 'localhost:9092'
topic = 'sparkapp'
# 使用 streaming 直连模式消费 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()
counts.foreachRDD(func)
ssc.start()
ssc.awaitTermination()
```

给代码添加注释与 StreamingKafka.py 的不同。

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

```
-------------------------------------------
Time: 2020-12-08 20:21:45
-------------------------------------------
('hdfs\n', 1)
('data\n', 1)
('hello', 3)
('world\n', 1)
('big', 1)
```

9. 查看 mysql 中的结果

```
select * from wordcount;
```

```
mysql> select * from wordcount;
+-------+-------+
| word  | count |
+-------+-------+
| world |     1 |
| hdfs  |     1 |
| hello |     3 |
| big   |     1 |
| data  |     1 |
+-------+-------+
```

综合实验

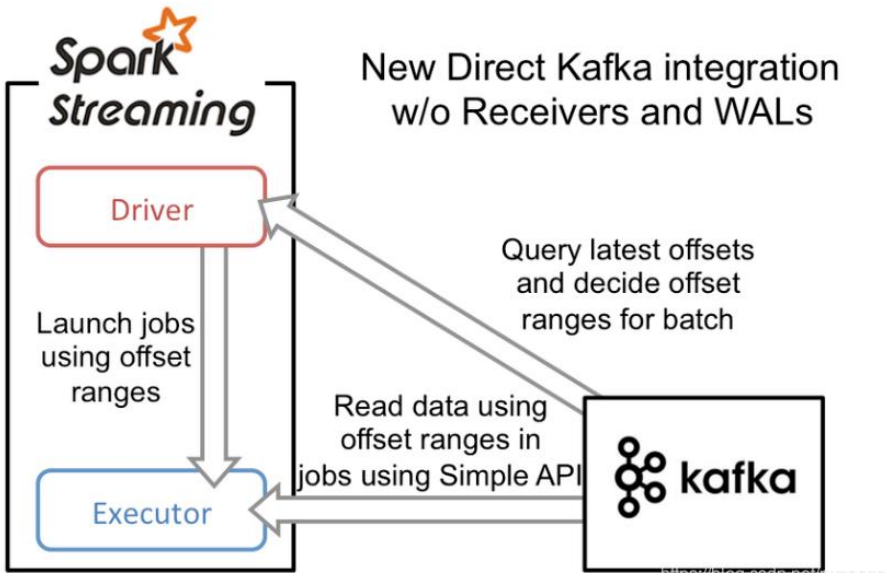
使用 Flume，Kafka 和 Spark streaming 构建一个完整的流数据处理系统，要求

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。

扩展阅读

在上面的代码中用的 streaming 直连 Kafka 进行消费消息。目前 Spark Streaming 与 Kafka 的结合主要有两种方式：Receiver Dstream 和 Direct Dstream。

基于 Direct 消费消息方式



Driect 模式下，Streaming 定时主动查询 Kafka，以获得指定 topic 的所有 partition

的最新 offset，结合上一批次已保存的 offset 位置，Streaming 就可以确定出每个批次拉取消息 offset 的范围，例如第 1 批次的消息（offset 范围 0-100）正在处理过程中，streaming 指定特定的线程定时去 Kafka 查询第 2 批次最新的 offset，发现最新值为 300，那么如果 streaming 没有限制每批次的最大消费速率，在第 2 批次取消息时，会一次性取回 offset=101 到 300 的消息记录，这个就是所谓的 offset ranges。若 streaming 设置的限制每批次的最大消费速率为每批次 100 条，那么即使查询到 Kafka topic 最新 offset 位置为 300，streaming 在第 2 批次消费的 offset 访问也只能是 101~200 共计 100 条消费记录。

当处理数据的 job 启动时，就会使用 Kafka 的简单 Consumer API 来获取 Kafka 中指定 offset 范围的数据。此外，Streaming 已消费的 offset 不再交由 Zookeeper 来管理，而是手动采用外部存储数据库如 MySQL、Redis 等存放和管理已消费的 offset。