总结、分享、收获
实验室主页 (//dblab.xmu.edu.cn)

首页 (http://dblab.xmu.edu.cn/blog/)
大数据 (https://dblab.xmu.edu.cn/blog/category/big-data/)
数据库 (https://dblab.xmu.edu.cn/blog/category/databases/)
数据挖掘 (https://dblab.xmu.edu.cn/blog/category/data-mining/)
其他 (https://dblab.xmu.edu.cn/blog/category/others/)

搜索…
搜索

Spark+Kafka构建实时分析Dashboard案例（2022年9月V2.0版）——步骤三：Structured Streaming实时处理数据（python版本）

刘, 浩然 2022年9月14日 (updated: 2022年9月24日) 18031



(//dblab.xmu.edu.cn/post/bigdataroadmap/)

返回本案例首页 (http://dblab.xmu.edu.cn/post/spark-kafka-dashboard/)

该版本是原先教程的python版本。

查看前一步骤操作步骤二：数据处理和Python操作Kafka
(https://dblab.xmu.edu.cn/blog/2022090001-4/)
查看scala版本scala版本：Structured Streaming实时处理数据
(https://dblab.xmu.edu.cn/blog/2022090001-5/)

《Spark+Kafka构建实时分析Dashboard案例——步骤三：Structured Streaming实时处理数据(python版本)》

开发团队：厦门大学数据库实验室 联系人：林子雨老师ziyulin@xmu.edu.cn

版权声明：版权归厦门大学数据库实验室所有，请勿用于商业用途；未经授权，其他网站请勿转载

本教程介绍大数据课程实验案例“Spark+Kafka构建实时分析Dashboard”的第三个步骤，Structured Streaming实时处理数据。在本篇博客中，将介绍如何利用Structed Streaming实时接收处理Kafka数据以及将处理后的结果发给的Kafka。

所需知识储备

会使用python编写Structured Streaming程序，Kafka原理。

训练技能

编写Structured Streaming程序，熟悉pySpark操作Kafka。

任务清单

1. Structured Streaming实时处理Kafka数据；
2. 将处理后的结果发送给Kafka；

编程思想

本案例在于实时统计每秒中男女生购物人数，而Structured Streaming接收的数据为1,1,0,2...，其中0代表女性，1代表男性，所以对于2或者null值，则不考虑。其实通过分析，可以发现这个就是典型的wordcount问题，而且是基于Spark流计算。女生的数量，即为0的个数，男生的数量，即为

1的个数。

因此利用Structured Streaming的groupBy接口，设置窗口大小为1，滑动步长为1，这样统计出的0和1的个数即为每秒男生女生的人数。

编程实现

配置Spark开发Kafka环境

kafka的安装可以参考Kafka的安装和简单实例测试 (https://dblab.xmu.edu.cn/blog/1096-2/)。下面主要介绍配置Spark开发Kafka环境。首先点击下载 (https://pan.baidu.com/s/12JCw8L0kt4x0rZFoTWIGAw?pwd=b5wl)，下载Spark连接Kafka的代码库。然后把下载的代码库放到目录/usr/local/spark/jars目录下，命令如下：

Shell

```
1. sudo mv ~/下载/spark-streaming_2.12-3.2.0.jar /usr/local/spark/jars
2. sudo mv ~/下载/spark-streaming-kafka-0-10_2.12-3.2.0.jar /usr/local/spark/jars
```

然后在/usr/local/spark/jars目录下新建kafka目录，把/usr/local/kafka/libs下所有函数库复制到/usr/local/spark/jars/kafka目录下，命令如下

Shell

```
1. cd /usr/local/spark/jars
2. mkdir kafka
3. cd kafka
4. cp /usr/local/kafka/libs/* .
```

然后,修改 Spark 配置文件,命令如下

Shell

```
1. cd /usr/local/spark/conf
2. sudo vim spark-env.sh
```

把 Kafka 相关 jar 包的路径信息增加到 spark-env.sh,修改后的 spark-env.sh 类似如下:

```
export SPARK_DIST_CLASSPATH=$(/usr/local/hadoop/bin/hadoopclasspath):/usr/local/spark/jars/kafka/*:/usr/local/kafka/libs/*
```

因为我使用的是anaconda中创建的python环境，所以介绍一下，怎么为spark设置python环境。

需要修改conf目录下的spark_env.sh：在这个文件的开头添加：

```
export PYSPARK_PYTHON=/home/hadoop/anaconda3/envs/py37/bin/python
#其中，py37应该为自行创建的环境名称
```

这里的py37是我在anaconda中自行创建的环境，读者在自己实验的时候，要找到你自己创建的环境。

如果后续运行程序时候还出现缺少模块的，请继续参照kafka安装：

Shell

```
1. conda activate env_name #这里env_name应替换为创建的conda环境名称
2. conda install kafka
```

执行上述步骤之后，Spark开发Kafka环境即已配置好，下面介绍如何编码实现。

建立pySpark项目

首先在/usr/local/spark/mycode新建项目目录

Shell

```
1. cd /usr/local/spark/mycode
2. mkdir kafka
```

然后在kafka这个目录下创建一个kafka_test.py文件。

Python

```python
from kafka import KafkaProducer
from pyspark.streaming import StreamingContext
#from pyspark.streaming.kafka import KafkaUtils
from pyspark import SparkConf, SparkContext
import json
import sys
from pyspark.sql import DataFrame
from pyspark.sql import SparkSession
from pyspark.sql.functions import window
from pyspark.sql.types import StructType, StructField
from pyspark.sql.types import TimestampType, StringType
from pyspark.sql.functions import col, column, expr

def KafkaWordCount(zkQuorum, group, topics, numThreads):

    spark = SparkSession \
    .builder \
    .appName("KafkaWordCount") \
    .getOrCreate()

    spark.sparkContext.setLogLevel("ERROR")

    topicAry = topics.split(",")
        # 将topic转换为hashmap形式，而python中字典就是一种hashmap
    topicMap = {}
    for topic in topicAry:
        topicMap[topic] = numThreads
    #lines = KafkaUtils.createStream(ssc, zkQuorum, group, topicMap).map(lambda x : x[1])

    df = spark \
      .readStream \
      .format("kafka") \
      .option("kafka.bootstrap.servers", "localhost:9092") \
      .option("subscribe", "sex") \
      .load()
    df.selectExpr( "CAST(timestamp AS timestamp)","CAST(value AS STRING)")
    #lines = df.selectExpr("CAST(value AS STRING)")

    windowedCounts = df \
        .withWatermark("timestamp", "1 seconds") \
        .groupBy(
            window(col("timestamp"), "1 seconds" ,"1 seconds"),
            col("value")) \
        .count()
    wind = windowedCounts.selectExpr( "CAST(value AS STRING)","CAST(count AS STRING)")

```

```
    query = wind.writeStream.option("checkpointLocation", "/check").outputMode("append").foreach(sendmsg).start()

    query.awaitTermination()
    query.stop()


# 格式转化，将格式变为[{1: 3}]
def Get_dic(row):
    res = []
    #for elm in row:
    tmp = {row[0]: row[1]}
    res.append(tmp)
    print(res)
    return json.dumps(res)


def sendmsg(row):
    print(row)
    if row.count != 0:
        msg = Get_dic(row)
        # 实例化一个KafkaProducer示例，用于向Kafka投递消息
        producer = KafkaProducer(bootstrap_servers='localhost:9092')
        producer.send("result", msg.encode('utf8'))
        # 很重要，不然不会更新
        producer.flush()


if __name__ == '__main__':
    # 输入的四个参数分别代表着
    # 1.zkQuorum为zookeeper地址
    # 2.group为消费者所在的组
    # 3.topics该消费者所消费的topics
    # 4.numThreads开启消费topic线程的个数
    if (len(sys.argv) < 5):
        print("Usage: KafkaWordCount <zkQuorum> <group> <topics> <numThreads>")
        exit(1)
    zkQuorum = sys.argv[1]
    group = sys.argv[2]
    topics = sys.argv[3]
    numThreads = int(sys.argv[4])
    print(group, topics)
    KafkaWordCount(zkQuorum, group, topics, numThreads)


```

上述代码注释已经也很清楚了，下面在简要说明下：

1. 首先按每秒的频率读取Kafka消息；

2. 然后对每秒的数据执行wordcount算法，统计出0的个数，1的个数，2的个数；

3. 最后将上述结果封装成json发送给Kafka。

另外，需要注意，上面代码中有一段如下代码：

```
.option("checkpointLocation", "/check")
```

这行代码表示把检查点文件写入分布式文件系统HDFS，所以一定要事先启动Hadoop。如果没有启动Hadoop，则后面运行时会出现“拒绝连接”的错误提示。如果你还没有启动Hadoop，则可以现在在Ubuntu终端中，使用如下Shell命令启动Hadoop：

Shell 命令

```
$ cd /usr/local/hadoop  #这是hadoop的安装目录
$ ./sbin/start-dfs.sh
```

运行项目

编写好程序之后，接下来编写运行脚本，在/usr/local/spark/mycode/kafka目录下新建startup.sh文件，输入如下内容：

Shell

```
1. /usr/local/spark/bin/spark-submit  --packages org.apache.spark:spark-
   sql-kafka-0-10_2.12:3.2.0 /usr/local/spark/mycode/kafka/kafka_test.py
   127.0.0.1:2181 1 sex 1
```

其中最后四个为输入参数，含义如下

1. 127.0.0.1:2181为Zookeeper地址
2. 1 为consumer group标签
3. sex为消费者接收的topic
4. 1 为消费者线程数

最后在/usr/local/spark/mycode/kafka目录下，运行如下命令即可执行刚编写好的Structured Streaming程序

Shell

```
1. sh startup.sh
```

程序运行成功之后，下面通过步骤二的KafkaProducer和KafkaConsumer来检测程序。

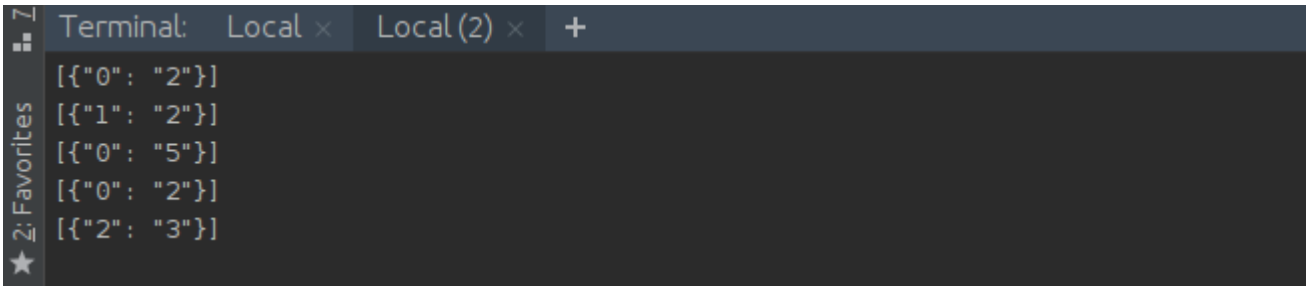
测试程序

下面开启之前编写的KafkaProducer投递消息，然后将KafkaConsumer中接收的topic改为result，验证是否能接收topic为result的消息，更改之后的KafkaConsumer为

Python

```
1. from kafka import KafkaConsumer
2.
3. consumer = KafkaConsumer('result')
4. for msg in consumer:
5.     print((msg.value).decode('utf8'))
```

在同时开启Structured Streaming项目，KafkaProducer以及KafkaConsumer之后，可以在KafkaConsumer运行窗口看到如下输出：



到此为止，Structured Streaming程序编写完成，下篇文章将分析如何处理得到的最终结果。

下篇文章链接为Spark+Kafka构建实时分析Dashboard案例——步骤四：结果展示 (https://dblab.xmu.edu.cn/blog/2022090001-6/)

本文作者

刘, 浩然 (//dblab.xmu.edu.cn/blog/author/liuhaoran/)

1609126475@qq.com

https://dblab.xmu.edu.cn/blog/3242/ (https://dblab.xmu.edu.cn/blog/3242/)

其他 (https://dblab.xmu.edu.cn/blog/category/others/)

© 2014 厦大数据库实验室 (//dblab.xmu.edu.cn)