Kafka+SparkStreaming集成

1. 引入KafkaUtils，创建一个input DStream

```
SparkConf sparkConf = new SparkConf().setAppName("JavaKafkaWordCount");
 //设置批间隔时间为2秒
 JavaStreamingContext jssc = new JavaStreamingContext(sparkConf, new Duration(2000));
 int numThreads = Integer.parseInt(args[3]);
 String[] topics = args[2].split(",");
 //键为kafka主题，值为kafka消费者使用的线程数
 Map<String, Integer> topicMap = new HashMap<>();
 for (String topic: topics) {
     topicMap.put(topic, numThreads);
 }
 //从Kafka的topic中接收消息，args[0]代表Zookeeper集群，arg[1]代表消费者所属的组号
 JavaPairReceiverInputDStream<String, String> messages =  KafkaUtils.createStream(jssc,
   args[0], args[1], topicMap);
```

2. 在SparkStream中编程（以WordCount程序为例）

```
//将接受的消息转换为行
JavaDStream<String> lines = messages.map(new Function<Tuple2<String, String>, String>() {
               @Override
               public String call(Tuple2<String, String> tuple2) {
                 return tuple2._2();
               }
             });
```

```
//将行转换成单词
JavaDStream<String> words = lines.flatMap(new FlatMapFunction<String, String>() {
               @Override
               public Iterator<String> call(String x) {
                 return Arrays.asList(SPACE.split(x)).iterator();
               }
             });
//执行map-reduce操作将单词转换成键-值的形式，值用来对单词计数
JavaPairDStream<String, Integer> wordCounts = words.mapToPair(
               new PairFunction<String, String, Integer>() {
                 @Override
                 public Tuple2<String, Integer> call(String s) {
                   return new Tuple2<>(s, 1);
                 }
               }).reduceByKey(new Function2<Integer, Integer, Integer>() {
                 @Override
                 public Integer call(Integer i1, Integer i2) {
                   return i1 + i2;
                 }
               });
```

3. 将程序打成jar包，ljg-jar-with-dependencies.jar
4. kafka生产消息

Ip地址	name
192.168.125.171	master-cent7-1
192.168.125.172	master-cent7-2
192.168.125.173	master-cent7-3

(1)创建topic

```
bin/kafka-topics.sh --create   --zookeeper master-cent7-1:2181,master-cent7-2:2181,master-
cent7-3:2181     --replication-factor 3 --partitions 1 --topic   lig_test
```

(2)查看状态

```
bin/kafka-console-producer.sh  --broker-list   master-cent7-1:9092, master-cent7-2:9092,
master-cent7-3:9092  --topic   lig_test
```

(3) 启动一个生产者，从终端输入消息

```
bin/kafka-console-producer.sh  --broker-list   master-cent7-1:92, master-cent7-2:9092,
master-cent7-3:9092  --topic   lig_test
```

5. 以yarn-cluster 模式向Spark提交程序

```
./bin/spark-submit --class com.ljg.KafkaWordCount  --master yarn  --deploy-mode  cluster
/home/bigdata/ljg-jar-with-dependencies.jar       master-cent7-1:2181,master-cent7-
2:2181,master-cent7-3:2181    spark_group    lig_test   2
```

--class代表main程序类名

--master yarn --deploy-mode cluster 表示以yarn-cluster模式启动

/home/bigdata/ljg-jar-with-dependencies.jar为SparkStream程序jar包

程序参数1：Zookeeper 集群信息

程序参数2：指定消费者的groupId，可有用户自定义

程序参数3：指定topic名称为lig_test

程序参数4：对消息进行消费的线程数

提交：

```
^C[bigdata@master-cent7-1 spark-2.0.2-bin-hadoop2.7]$ ./bin/spark-submit --class com.ljg.KafkaWordCount  --mas
r yarn  --deploy-mode  cluster  /home/bigdata/ljg-jar-with-dependencies.jar       master-cent7-1:2181,master-c
ent7-2:2181,master-cent7-3:2181    spark_group    lig_test   2
```

程序开始运行：

```
16/12/30 14:49:39 INFO yarn.Client:
        client token: N/A
        diagnostics: N/A
        ApplicationMaster host: 192.168.125.172
        ApplicationMaster RPC port: 0
        queue: default
        start time: 1483080568361
        final status: UNDEFINED
        tracking URL: http://master-cent7-1:8088/proxy/application_1482975932708_0033/
        user: bigdata
16/12/30 14:49:40 INFO yarn.Client: Application report for application_1482975932708_0033 (state: RUNNING)
16/12/30 14:49:41 INFO yarn.Client: Application report for application_1482975932708_0033 (state: RUNNING)
16/12/30 14:49:42 INFO yarn.Client: Application report for application_1482975932708_0033 (state: RUNNING)
16/12/30 14:49:43 INFO yarn.Client: Application report for application_1482975932708_0033 (state: RUNNING)
16/12/30 14:49:44 INFO yarn.Client: Application report for application_1482975932708_0033 (state: RUNNING)
16/12/30 14:49:45 INFO yarn.Client: Application report for application_1482975932708_0033 (state: RUNNING)
16/12/30 14:49:46 INFO yarn.Client: Application report for application_1482975932708_0033 (state: RUNNING)
16/12/30 14:49:47 INFO yarn.Client: Application report for application_1482975932708_0033 (state: RUNNING)
```

6. 程序演示

(1)在Kafka终端生产者中输入单词

```
[bigdata@master-cent7-1 kafka_2.11-0.9.0.1]$ bin/kafka-console-producer.sh  --broker-list   master-cent7-1:909
2, master-cent7-2:9092, master-cent7-3:9092  --topic   lig_test
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/bigdata/work/hadoop-2.7.3/share/hadoop/common/lib/slf4j-log4j12-1.7.10
.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/bigdata/work/kafka_2.11-0.9.0.1/libs/slf4j-log4j12-1.7.6.jar!/org/slf4
j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
hello world hello world good boy good man my life man good hello man
```

(2)在Hadoop的userlogs中对应的application目录下，查看输出信息

```
-------------------------------------------
Time: 1483080908000 ms
-------------------------------------------


-------------------------------------------
Time: 1483080910000 ms
-------------------------------------------
(life,1)
(hello,3)
(boy,1)
(my,1)
(man,3)
(world,2)
(good,3)

-------------------------------------------
```