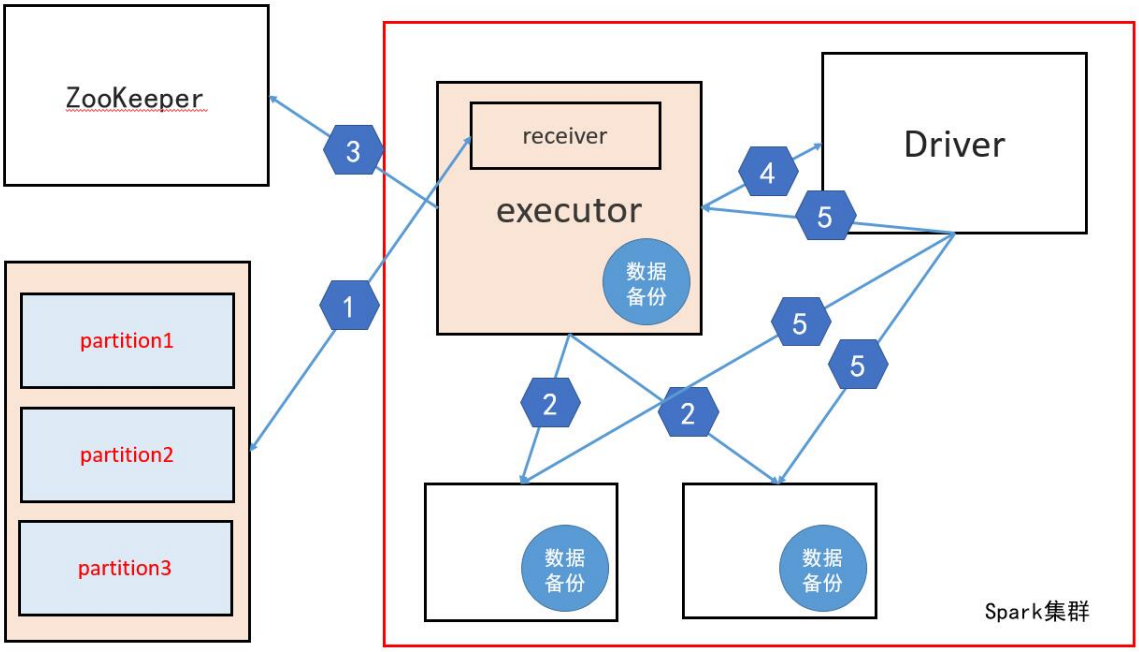
sparkStream从kafka获取数据方式一

1.KafkaUtils.createDstream方式



1、Spark集群中的某个executor中有一个receiver线程，这个线程负责从kafka中获取数据

注意：这里的获取数据并不是从kafka中拉(pull) 而是接收数据，具体原理是该receiver线程发送请求到kafka，这个请求包含对kafka中每个partition的消费偏移量(offset)，然后由kafka主动的推送数据到spark中，再有该receiver线程负责接收数据

2、当receiver线程接收到数据后会做备份处理，即把数据备份到其他的executor中，也可能会备份到这个receiver线程所在节点的executor中

3、当备份完毕后该线程会把每个partition的消费偏移量在zookeeper中修改，(新版本的kafka的offset保存在kafka集群中)

4、修改完offset后，该receiver线程会把"消费"的数据告诉Driver

5、Driver分发任务时会根据每个executor上的数据，根据数据本地性发送

问题:

当第三步执行完后，对于kafka来说这一批数据已经消费完成，那么如果此时Driver挂掉，那么这一批数据就会丢失，为了解决这个问题，有一个叫WAL逾写日志的概念，即把一部分数据存储在HDFS上，当Driver回复后可以从HDFS上获取这部分数据，但是开启WAL性能会受到很大的影响

原文链接：https://blog.csdn.net/lyzx_in_csdn/article/details/80602846

代码实战

pom文件依赖

```
<dependencies>
          <dependency>
            <groupId>org.apache.spark</groupId>
            <artifactId>spark-core_2.10</artifactId>
```

```
            <version>2.1.0</version>
        </dependency>

        <dependency>
            <groupId>org.apache.spark</groupId>
            <artifactId>spark-streaming_2.10</artifactId>
            <version>2.1.0</version>
        </dependency>

        <dependency>
            <groupId>org.apache.spark</groupId>
            <artifactId>spark-streaming-kafka-0-8_2.10</artifactId>
            <version>2.1.0</version>
        </dependency>
        <dependency>
            <groupId>org.apache.kafka</groupId>
            <artifactId>kafka-clients</artifactId>
            <version>0.11.0.0</version>
        </dependency>
    </dependencies>
```

在kafka集群（b0,b1,b2）中创建first主题，分区和副本数各为2。在Hadoop101机器上启动producer进程，sparkStreaming每五秒进行一次处理统计单词个数。

```
package com.bupt.sparkStreaming

import org.apache.spark.SparkConf
import org.apache.spark.streaming.kafka.KafkaUtils
import org.apache.spark.streaming.{Seconds, StreamingContext, kafka}

object KafkaSource {

  def main(args: Array[String]): Unit = {
    //使用SparkStreama完成WordCount
    //spark配置对象
    val conf = new SparkConf().setAppName("wc").setMaster("local[*]");
    //实时数据分析环境对象
    //采集周期，以指定的时间为周期采集实时数据
    val streamingContext = new StreamingContext(conf, Seconds(5));
    //从指定周期的端口采集数据
    //val socketLineStream =
streamingContext.socketTextStream("hadoop101",9999);
    val kafkaStream = KafkaUtils.createStream(
      streamingContext,
      "hadoop101:2181",
      "test",
      Map("first" -> 2)
    )

    //采集数据扁平化
    //val wordDStreaming = socketLineStream.flatMap(_.split(" "));
    //kafkaStrem 的返回值时(K,V)类型的数据
    val wordDStreaming =  kafkaStream.flatMap(t => t._2.split(" "));
    //转化结构
    val mapStream = wordDStreaming.map((_,1))
    //统计结果，聚合处理
    val wordToSumDStream = mapStream.reduceByKey(_+_)
```

```
    //将结果打印
    wordToSumDStream.print()
    //不能停止采集功能
    //streamingContext.stop()
    //启动采集器
    streamingContext.start()
    //Driver等待执行器的执行
    streamingContext.awaitTermination()

  }

}
```