透彻讲解使用Scala和Java两种方式实战Spark Streaming开发

第一步:配置SparkConf

1. 设置本地Master,如果指定local的话至少2条线程，因为Spark Streaming应用程序在运行的时候，至少有一条线程用于不断的循环接收数据，并且至少有一条线程用于处理接受的数据（否则的话无法有线程用于处理数据随着时间的推移，内存和磁盘都会不堪重负)

温馨提示:

对于集群而言，每个executor一般肯定不只一个Thread,那对于处理Spark Streaming应用程序而言，每个executor一般分配多少core比较合适？根据我们过去的经验，5个左右的core是最佳的（段子：分配为奇数个core的表现最佳，例如：分配3个，5个，7个core等

1. 对于集群而言，每个Executor一般肯定不止一个Thread，那对于处理Spark Streaming的应用程序而言，每个Executor一般分配多少core比较合适？根据我们过去的经验，5个左右的Core是最佳的（一个段子分配为奇数个Core表现最佳，例如3个，5个，7个Core等)

第二步:

创建SparkStreamingContext，这个是SparkStreaming应用程序所有功能的起始点和程序调度的核心，SparkStreamingContext的构建可以基于SparkConf参数，也可以基于持久化的SparkStreamingContext的内容来恢复过来（典型的场景是Driver崩溃后重新启动，由于SparkStreaming是具有连续7\*24小时不间断运行的特诊，所有需要在Driver重新启动后继续上一次的状态，此时的状态恢复需要基于曾经的Checkpoint)

在一个Spark Streaming应用程序可以创建若干个SparkStreamingContext对象，使用下一个SparkStreaming之前需要把前面正在运行的SparkStreamingContext对象关闭掉，由此，我们获得一个重大的启发，Spark Streaming框架也只是Spark core上的一个应用程序而已，只不过Spark Streaming框架运行的话需要Spark工程师写业务逻辑处理代码

第三步:创建Spark Streaming输入数据来源input Streaming;

1. 数据输入来源可以基于File,HDFS,FLUME,kafka，Socket等
2. 在这里我们指定数据来源于网络Socket端口，Spark Streaming链接上盖端口并运行的时候，一直监听该端口的数据，当然改端口服务首先必须存在，并且在后续会根据业务需要不断的有数据产生（当然对于Spark Streaming的应用程序来说，有无数据其处理流程都是一样的);
3. 如果经常在每间隔5秒钟没有数据的话不断的启动空的Job,其实会造成调度资源的浪费，因为没有数据需要发生计算，所以实际的企业级生产环境的代码在具体提交Job前会判断是否有数据，如果没有的话就不再提交job
4. 此处没有Receiver,SparkStreaming应用程序只是按照时间间隔监控目录下的每个batch新增的内容作为RDD的数据来源生产原始的RDD

第四步：

/\*接下来就像对于RDD编程一样基于Dstream进行编程，原因是Dstream是RDD产生的模板或者说类\*/，在Spark发生计算前，其实质是把每个batch的Dstream的操作翻译成为对RDD的操作

第五步:此处的print并不会直接触犯job的执行，因为现在的一切都是在Spark Streaming框架的控制之下的，对于Spark而言具体是否触发真正的Job运行是基于设置的duration时间间隔的

Spark Streaming执行引擎也就是Driver开始运行，Driver启动的时候是位于一条新的线程中的，当然其内部有消息循环体来接收应用程序本身或者Executor中的消息

诸位一定要注意的是Spark Streaming应用程序要想执行具体的job,对Dstream就必须有output stream操作，outputStream有很多类型的函数触发，类print,saveAsTextFile，saveHadoopFiles等，最为重要的一个方法是foreachRDD，因为Spark Streaming处理的结果一般都会放在Redis,DB,DashBoard等上面，foreachRDD主要就是用来完成这些功能的，而且可以随意的自定义具体数据到底放在哪里