如何优雅地终止正在运行的Spark Streaming程序

一直运行的Spark Streaming程序如何关闭呢？是直接使用kill命令强制关闭吗？这种手段是可以达到关闭的目的，但是带来的后果就是可能会导致数据的丢失，因为这时候如果程序正在处理接收到的数据，但是由于接收到kill命令，那它只能停止整个程序，而那些正在处理或者还没有处理的数据可能就会被丢失。那我们咋办？这里有两种方法。

等作业运行完再关闭

我们都知道，Spark Streaming每隔batchDuration的时间会把源源不断的流数据分割成一批有限数据集，然后计算这些数据，我们可以从Spark提供的监控页面看到当前batch是否执行完成，当作业执行完，我们就可以手动执行kill命令来强制关闭这个Streaming作业。这种方式的缺点就是得盯着监控页面，然后决定关不关闭，很不灵活。

通过Spark内置机制关闭

其实Spark内置就为我们提供了一种优雅的方法来关闭长期运行的Streaming作业，我们来看看 StreamingContext 类中定义的一个 stop 方法：

```
def stop(stopSparkContext: Boolean, stopGracefully: Boolean)
```

官方文档对其解释是：Stop the execution of the streams, with option of ensuring all received

data has been processed. 控制所有接收的数据是否被处理的参数就是 stopGracefully，如果我们将它设置为true，Spark则会等待所有接收的数据被处理完成，然后再关闭计算引擎，这样就可以避免数据的丢失。现在的问题是我们在哪里调用这个stop方法？

Spark 1.4版本之前

在Spark 1.4版本之前，我们需要手动调用这个 stop 方法，一种比较合适的方式是通过Runtime.getRuntime().addShutdownHook
来添加一个钩子，其会在JVM关闭的之前执行传递给他的函数，如下：

```
Runtime.getRuntime().addShutdownHook(new Thread() {
  override def run() {
    log("Gracefully stop Spark Streaming")
    streamingContext.stop(true, true)
  }
})
```

如果你使用的是Scala，我们还可以通过以下的方法实现类似的功能：

```
scala.sys.addShutdownHook({
  streamingContext.stop(true,true)
)})
```

通过上面的办法，我们客户确保程序退出之前会执行上面的函数，从而保证Streaming程序关闭的时候不丢失数据。

Spark 1.4版本之后

上面方式可以达到我们的需求，但是在每个程序里面都添加这样的重复代码也未免太过麻烦了！值得高兴的是，从Apache Spark 1.4版本开始，Spark内置提供了spark.streaming.stopGracefullyOnShutd
own参数来决定是
否需要以Gracefully方式来关闭Streaming程序（详情请参见SPARK-7776）。Spark会在启动StreamingContext 的时候注册这个钩子，如下：

```
shutdownHookRef = ShutdownHookManager.addShutdownHook(
        StreamingContext.SHUTDOWN_HOOK_PRIORITY)(stopOnShutdown)

private def stopOnShutdown(): Unit = {
```

```
    val stopGracefully = conf.getBoolean("spark.streaming.stopGracefullyOnShutdown", false)
    logInfo(s"Invoking stop(stopGracefully=$stopGracefully) from shutdown hook")
    // Do not stop SparkContext, let its own shutdown hook stop it
    stop(stopSparkContext = false, stopGracefully = stopGracefully)
}
```

从上面的代码可以看出，我们可以根据自己的需求来设置spark.streaming.stopGracefullyOnShutdown 的值，而不需要在每个Streaming程序里面手动调用StreamingContext的stop方法，确实方便多了。不过虽然这个参数在Spark 1.4开始引入，但是却是在Spark 1.6才开始才有文档正式介绍（可以参见https://github.com/apache/spark/pull/8898和http://spark.apache.org/docs/1.6.0/configuration.html）