**初始化 StreamingContext**



1. window()操作

window(windowLength,slideInterval)这个函数来表示上图的滑动窗口操作，假设批处理时间间隔为10秒，那么窗口时间为30秒，每隔20秒生成数据。那么函数参数设置为：

pairs.window(**Durations.**seconds(30), **Durations.**seconds(20))；

设置data = pairs.window(Durations.seconds(30))时，窗口的操作时间为30秒，处理间隔10秒(默认跟批处理时间一致)。

3）reduceByKeyAndWindow高效操作

reduceByKeyAndWindow该计算方法还有一种高效计算形式，通过只考虑新进入窗口的数据和离开窗口的数据，让spark增量计算聚合结果，使计算更加高效。该形式需要提供聚合函数的一个逆函数，比如聚合函数为+，则逆函数为-。

使用这种方式运行可以复用两个窗口共有的批次数据，计算增加进入窗口的数据，和使用逆函数减去离开窗口的数据。对于较大窗口，使用逆函数这种计算方式可以大大提高执行效率。

[**DStreams 上的 Transformations（转换）**](http://www.apache.wiki/pages/viewpage.action?pageId=2885990&src=contextnavpagetreemode):

countByValueAndWindow ( windowLength ,slideInterval ,[ numTasks ])

reduceByKeyAndWindow ( func, invFunc ,windowLength , slideInterval ,( numTasks ])

reduceByKeyAndWindow ( func,windowLength , slideInterval , ( numTasks ])

reduceByWindow ( func, windowLength ,slideInterval )

countByWindow ( windowLength ,slideInterval )

window ( windowLength , slideInterval )

## [DStreams 上的输出操作](http://www.apache.wiki/pages/viewpage.action?pageId=2886082)

foreachRDD通常用来把SparkStream运行得到的结果保存到外部系统比如HDFS、[MySQL](http://lib.csdn.net/base/14)等。

wordcountDS.foreachRDD( partitionOfRecords =>{

partitionOfRecords.foreachPartition( records =>{

**val** connect= MysqlPool.*getJdbcCoon*()

**while**(records.hasNext){

**val** tuple= records.next()

**val** sql="insert into wordcount values( now(),'"+tuple.\_1+"', "+tuple.\_2.toInt+")"

**val** statement=connect.createStatement()

statement.executeUpdate(sql)

*print*(sql)

}

MysqlPool.*releaseConn*(connect)

})

})

