1. Spark Streaming on HDFS实战
2. Spark Streaming on HDFS源码解密
3. hadoop dfs -mkdir /library/SparkStreaming/

hadoop dfs -mkdir /library/SparkStreaming/CheckPoint\_Data

Hadoop dfs -mkdir /library/SparkStreaming/Data

假如说数据流进来，remember：有时候进行机器学习或者图计算要算几分钟，但是按照流的batch来讲会定时的清楚RDD，所以remember可以延长RDD的清除

相同文件不让上传，重启之后要删除checkpoint下的文件才能运行，不然再次启动程序后就会在checkpoint的基础上进行运行，因为配置没有配置清楚所以才运行失败

从checkpoint中读取数据来处理

1. Checkpoint:

一方面：保持容错

一方面保持状态

1. remember:

流失处理中过一段时间就会被清理，但是可以通过remember可以延长数据在程序中的生命周期，另外延长RDD更长的时间。

应用场景:

假设数据流进来，进行ML或者Graphx的时候有时需要很长的时间，但是batch定时定条件的清除RDD，所以就可以通过remember使得数据可以延长更长时间

 \* Sets each DStreams in this context to remember RDDs it generated in the last given duration.  
 \* DStreams remember RDDs only for a limited duration of duration and releases them for garbage  
 \* collection. This method allows the developer to specify how long to remember the RDDs (  
 \* if the developer wishes to query old data outside the DStream computation).  
 \* **@param duration** Minimum duration that each DStream should remember its RDDs  
 \*/  
**def** remember(duration: Duration) {  
  ssc.remember(duration)  
}

如果设置了checkpoint，重启程序的时候，getOrCreate会重新从checkpoint目录中初始化出StreamingContext.

/\* \* Either recreate a StreamingContext from checkpoint data or create a new StreamingContext.

 \* If checkpoint data exists in the provided `checkpointPath`, then StreamingContext will be  
 \* recreated from the checkpoint data. If the data does not exist, then the provided factory  
 \* will be used to create a JavaStreamingContext.  
 \*  
 \* **@param checkpointPath** Checkpoint directory used in an earlier JavaStreamingContext program  
 \* **@param factory** JavaStreamingContextFactory object to create a new JavaStreamingContext  
 \* **@deprecated** As of 1.4.0, replaced by `getOrCreate` without JavaStreamingContextFactor.  
 \*/  
@deprecated("use getOrCreate without JavaStreamingContextFactor", "1.4.0")  
**def** getOrCreate(  
    checkpointPath: String,  
    factory: JavaStreamingContextFactory  
  ): JavaStreamingContext = {  
  **val** ssc = StreamingContext.getOrCreate(checkpointPath, () => {  
    factory.create.ssc  
  })  
  **new** JavaStreamingContext(ssc)  
}



为啥会报错了？

1. Streaming会定期的进行checkpoint
2. 重新启动程序的时候，他会从曾经checkpoint的目录中如果没有做额外配置的时候，所有的信息都会放在checkpoint的目录中（包括曾经应用程序信息），因此下次再次启动的时候就会报错，无法初始化shuffleDstream