

Spark入门实战系列--7.Spark Streaming（下）--实时流计算Spark Streaming实战

个人简介

郭景瞻，博客以家乡石山园为名，大数据图书作者，著《图解Spark：核心技术与案例实战》。

Spark核心技术与案例  
[《图解Spark：核心技术与案例实战》]

Visitors

|  |         |  |     |
|--|---------|--|-----|
|  | 317,929 |  | 759 |
|  | 6,776   |  | 422 |
|  | 5,911   |  | 328 |
|  | 2,138   |  | 282 |
|  | 791     |  | 237 |

FLAG counter

昵称：shishanyuan  
园龄：8年7个月  
荣誉：推荐博客  
粉丝：741  
关注：16  
+加关注

| 2018年8月 |    |    |    |    |    |    |
|---------|----|----|----|----|----|----|
| 日       | 一  | 二  | 三  | 四  | 五  | 六  |
| 29      | 30 | 31 | 1  | 2  | 3  | 4  |
| 5       | 6  | 7  | 8  | 9  | 10 | 11 |
| 12      | 13 | 14 | 15 | 16 | 17 | 18 |
| 19      | 20 | 21 | 22 | 23 | 24 | 25 |
| 26      | 27 | 28 | 29 | 30 | 31 | 1  |
| 2       | 3  | 4  | 5  | 6  | 7  | 8  |

搜索

找找看

- 随笔分类 (107)
- 10.《图解Spark:核心技术与案例实战》(10)

20.Spark入门实战系列(19)

27.Spark精彩文章(6)

28.Databricks博客翻译(7)

30.Hadoop入门进阶课程(13)

40.持续集成(8)

50.图解Oracle10g备份恢复系列(20)

51.基于OracleLogminer数据同步(4)

58.数据库优化(1)

90.Hadoop数据分析平台(12)

91.杂项(7)

【注】该系列文章以及使用到安装包/测试数据 可以在《倾情大奉送--Spark入门实战系列》获取

1、实例演示

1.1 流数据模拟器

1.1.1 流数据说明

在实例演示中模拟实际情况，需要源源不断地接入流数据，为了在演示过程中更接近真实环境将定义流数据模拟器。该模拟器主要功能：通过Socket方式监听指定的端口号，当外部程序通过该端口连接并请求数据时，模拟器将定时将指定的文件数据随机获取发送给外部程序。

1.1.2 模拟器代码

```
import java.io.{PrintWriter}
import java.net.ServerSocket
import scala.io.Source

object StreamingSimulation {
  // 定义随机获取整数的方法
  def index(length: Int) = {
    import java.util.Random
    val rdm = new Random
    rdm.nextInt(length)
  }

  def main(args: Array[String]) {
    // 调用该模拟器需要三个参数，分为为文件路径、端口号和间隔时间（单位：毫秒）
    if (args.length != 3) {
      System.err.println("Usage: <filename> <port> <millisecond>")
      System.exit(1)
    }

    // 获取指定文件总的行数
    val filename = args(0)
    val lines = Source.fromFile(filename).getLines.toList
    val filerow = lines.length

    // 指定监听某端口，当外部程序请求时建立连接
    val listener = new ServerSocket(args(1).toInt)
    while (true) {
      val socket = listener.accept()
      new Thread() {
        override def run = {
          println("Got client connected from: " + socket.getInetAddress)
          val out = new PrintWriter(socket.getOutputStream(), true)
          while (true) {
            Thread.sleep(args(2).toLong)
            // 当该端口接受请求时，随机获取某行数据发送给对方
            val content = lines(index(filerow))
            println(content)
            out.write(content + '\n')
            out.flush()
          }
          socket.close()
        }
      }.start()
    }
  }
}
```

## 随笔档案(106)

2018年2月 (12)  
 2017年3月 (2)  
 2016年12月 (8)  
 2016年7月 (1)  
 2015年9月 (7)  
 2015年8月 (13)  
 2015年7月 (14)  
 2015年2月 (1)  
 2015年1月 (5)  
 2014年12月 (6)  
 2013年6月 (4)  
 2011年11月 (2)  
 2011年9月 (4)  
 2011年8月 (5)  
 2010年12月 (6)  
 2010年3月 (5)  
 2010年2月 (3)  
 2010年1月 (8)

## 积分与排名

积分 - 188842  
 排名 - 1479

## 最新评论

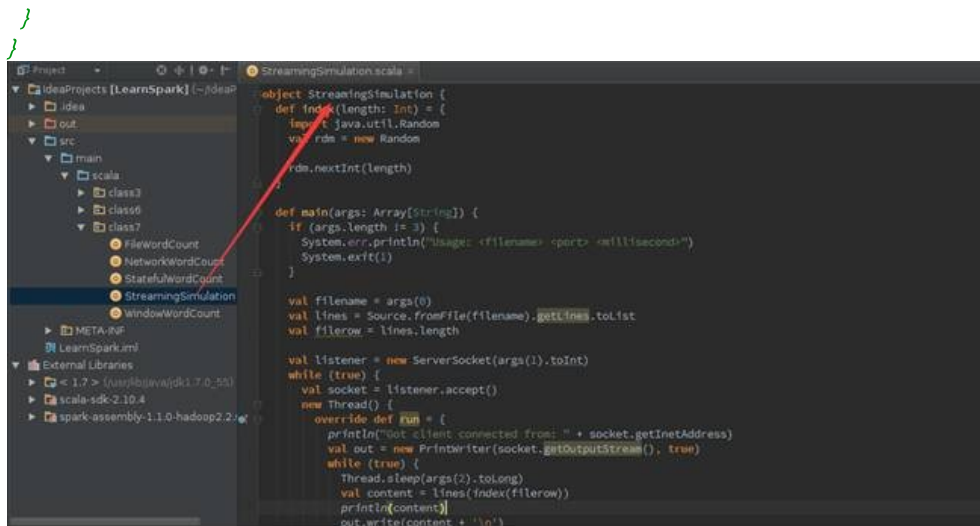
1. Re: Spark入门实战系列--6.SparkSQL (上) --SparkSQL简介  
写的不错，推荐  
--形色聪聪
2. Re: 《图解Spark：核心技术与案例实战》介绍及书附资源  
@超级飞侠飞飞已经修复...  
--shishanyuan
3. Re: 《图解Spark：核心技术与案例实战》介绍及书附资源  
百度云的 资源失效了  
--超级飞侠飞飞飞
4. Re: 倾情大奉送--Spark入门实战系列  
楼主好，百度云链接失效了，能麻烦更新下么  
--Elninom
5. Re: 倾情大奉送--Spark入门实战系列  
@九九学生哥老哥，解压密码是多少啊...  
--Elninom

## 阅读排行榜

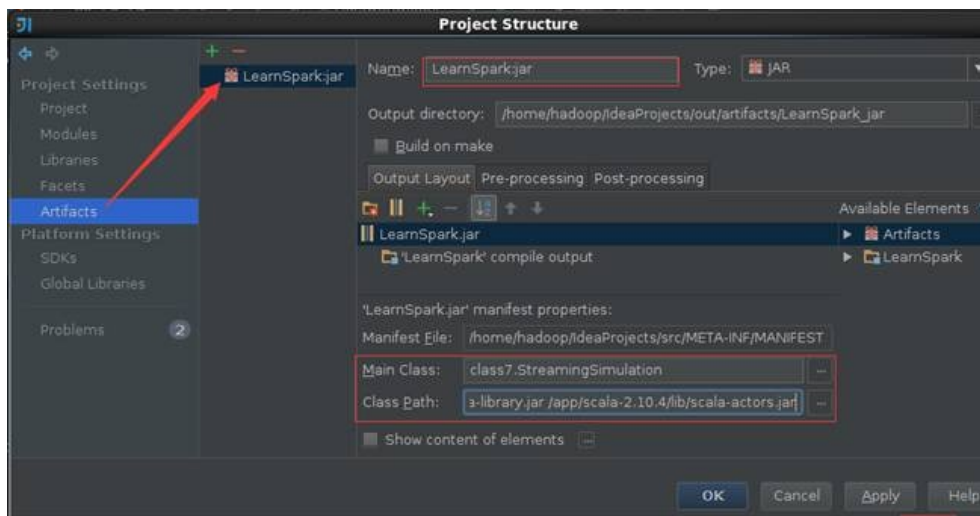
1. Spark入门实战系列--6.SparkSQL (上) --SparkSQL简介(117007)
2. 倾情大奉送--Spark入门实战系列(104583)
3. Spark入门实战系列--7.Spark Streaming (上) --实时流计算Spark Streaming原理介绍(69134)
4. Hadoop第4周练习--HDFS读写文件操作(59976)
5. Spark入门实战系列--8.Spark MLlib (下) --机器学习库SparkMLlib实战(52813)

## 评论排行榜

1. Spark入门实战系列--2.Spark编译与部署 (下) --Spark编译安装(59)
2. 倾情大奉送--Spark入门实战系列(56)
3. Spark入门实战系列--6.SparkSQL (上) --SparkSQL简介(29)
4. Spark入门实战系列--2.Spark编译与部署 (中) --Hadoop编译安装(20)
5. 《图解Spark：核心技术与案例实战》介绍及书附资源(20)

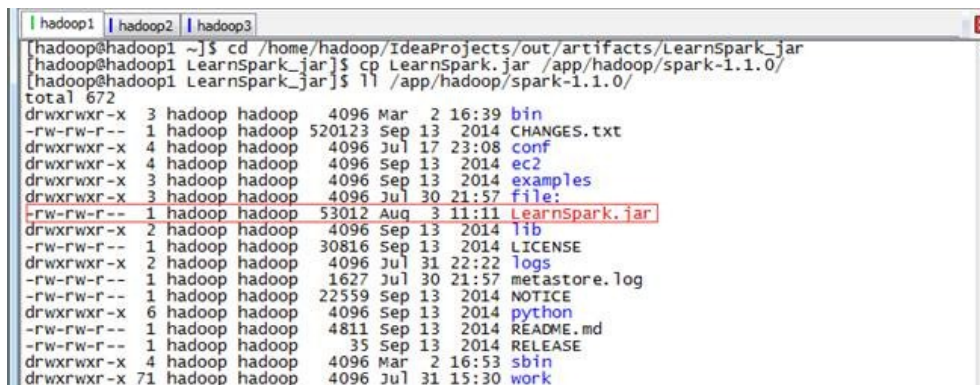


【注】可以参见第3课《Spark编程模型 (下) --IDEA搭建及实战》进行打包



在打包配置界面中，需要在Class Path加入：/app/scala-2.10.4/lib/scala-swing.jar /app/scala-2.10.4/lib/scala-library.jar /app/scala-2.10.4/lib/scala-actors.jar，各个jar包之间用空格分开，点击菜单Build->Build Artifacts，弹出选择动作，选择Build或者Rebuild动作，使用如下命令复制打包文件到Spark根目录下

```
cd /home/hadoop/IdeaProjects/out/artifacts/LearnSpark.jar
cp LearnSpark.jar /app/hadoop/spark-1.1.0/
ll /app/hadoop/spark-1.1.0/
```



### 1.2 实例1：读取文件演示

#### 1.2.1 演示说明

在该实例中Spark Streaming将监控某目录中的文件，获取在间隔时间段内变化的数据，然后通过Spark Streaming计算出改时间段内单词统计数。

#### 1.2.2 演示代码

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

```
import org.apache.spark.streaming.StreamingContext._

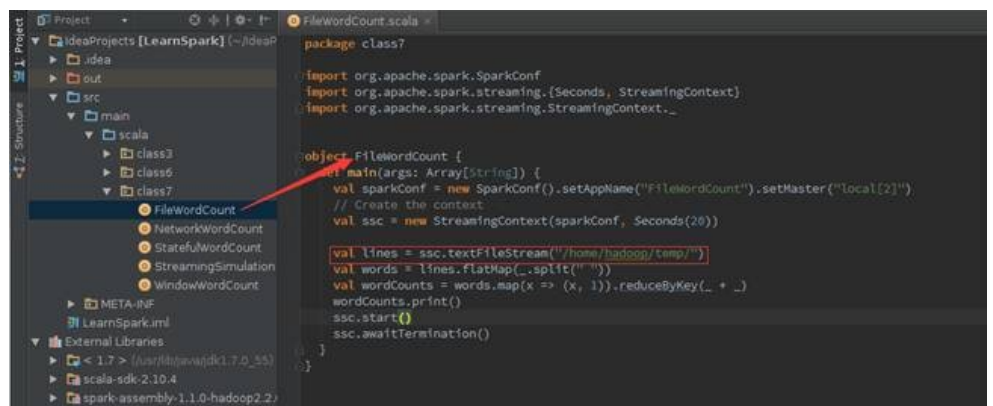
object FileWordCount {
  def main(args: Array[String]) {
    val sparkConf = new SparkConf().setAppName("FileWordCount").setMaster("local[2]")

    // 创建Streaming的上下文, 包括Spark的配置和时间间隔, 这里时间为间隔20秒
    val ssc = new StreamingContext(sparkConf, Seconds(20))

    // 指定监控的目录, 在这里为/home/hadoop/temp/
    val lines = ssc.textFileStream("/home/hadoop/temp/")

    // 对指定文件夹变化的数据进行单词统计并且打印
    val words = lines.flatMap(_.split(" "))
    val wordCounts = words.map(x => (x, 1)).reduceByKey(_ + _)
    wordCounts.print()

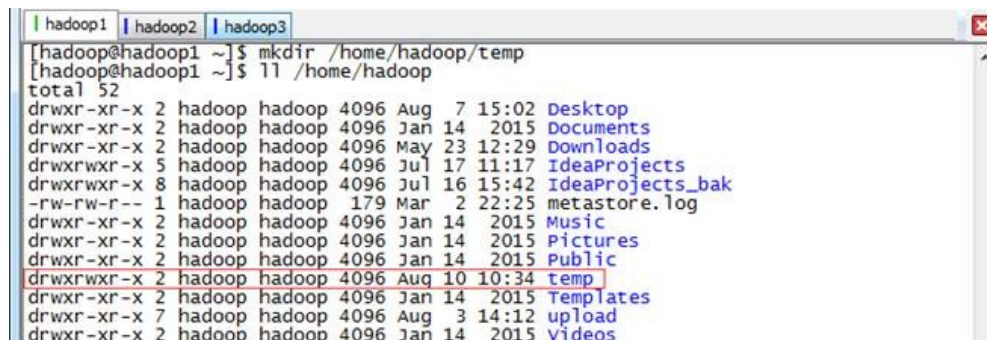
    // 启动Streaming
    ssc.start()
    ssc.awaitTermination()
  }
}
```



### 1.2.3 运行代码

#### 第一步 创建Streaming监控目录

创建/home/hadoop/temp为Spark Streaming监控的目录, 通过在该目录中定时添加文件内容, 然后由Spark Streaming统计出单词个数



#### 第二步 使用如下命令启动Spark集群

```
$cd /app/hadoop/spark-1.1.0
$sbin/start-all.sh
```

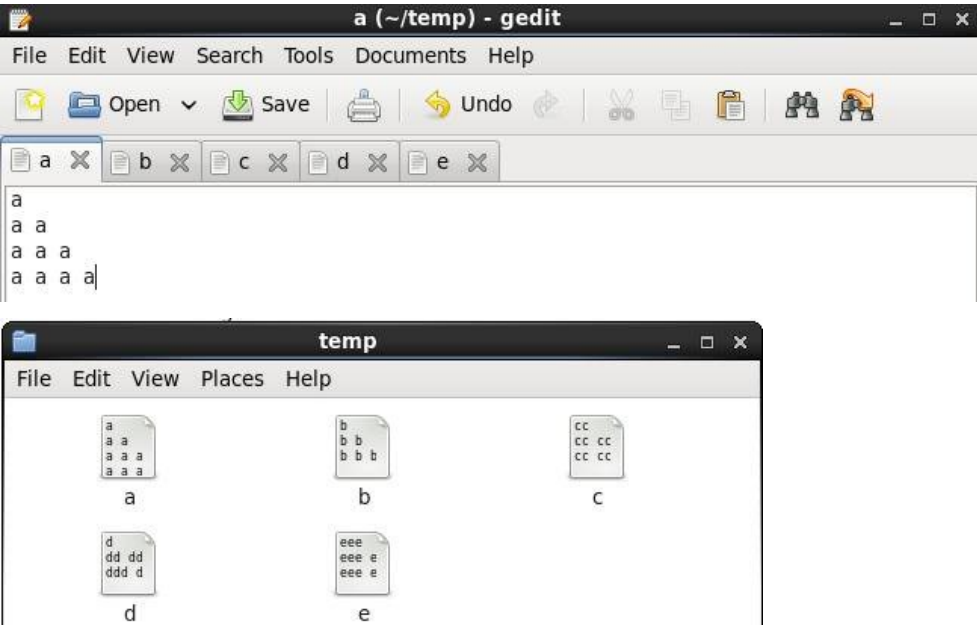
#### 第三步 在IDEA中运行Streaming程序

在IDEA中运行该实例, 由于该实例没有输入参数故不需要配置参数, 在运行日志中将定时打印时拦截。如果在监控目录中加入文件内容, 将输出时间戳的同时将输出单词统计个数。

clip\_image012

### 1.2.4 添加文本及内容

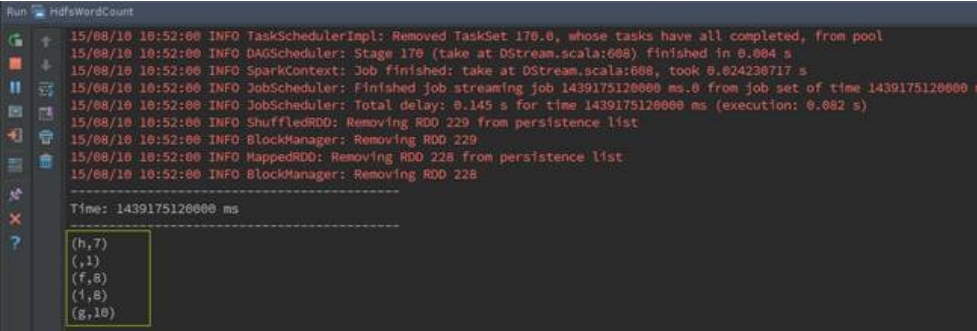




1.2.5 查看结果

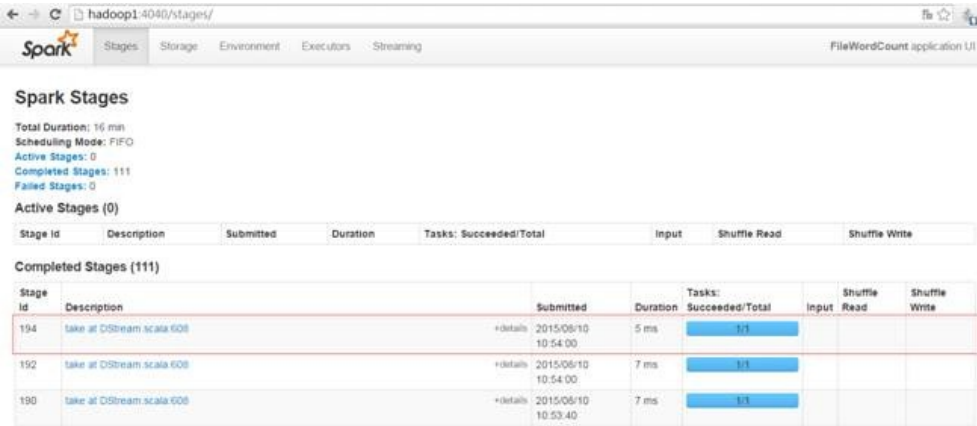
第一步 查看IDEA中运行情况

在IDEA的运行日志窗口中，可以观察到输出时间戳的同时将输出单词统计个数

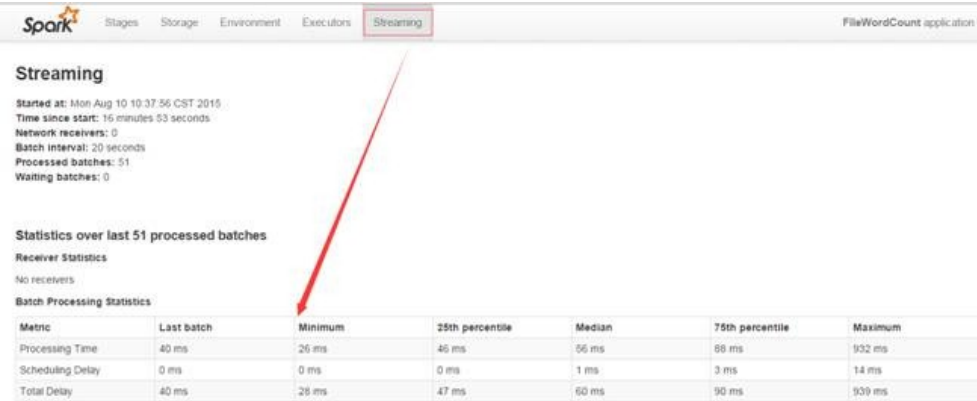


第二步 通过webUI监控运行情况

在http://hadoop1:4040监控Spark Streaming运行情况，可以观察到每20秒运行一次作业



并且与其他运行作业相比在监控菜单增加了"Streaming"项目，点击可以看到监控内容：



### 1.3 实例2：网络数据演示

#### 1.3.1 演示说明

在该实例中将由4.1流数据模拟以1秒的频率发送模拟数据，Spark Streaming通过Socket接收流数据并每20秒运行一次用来处理接收到数据，处理完毕后打印该时间段内数据出现的频度，即在各处理段时间之间状态并无关系。

#### 1.3.2 演示代码

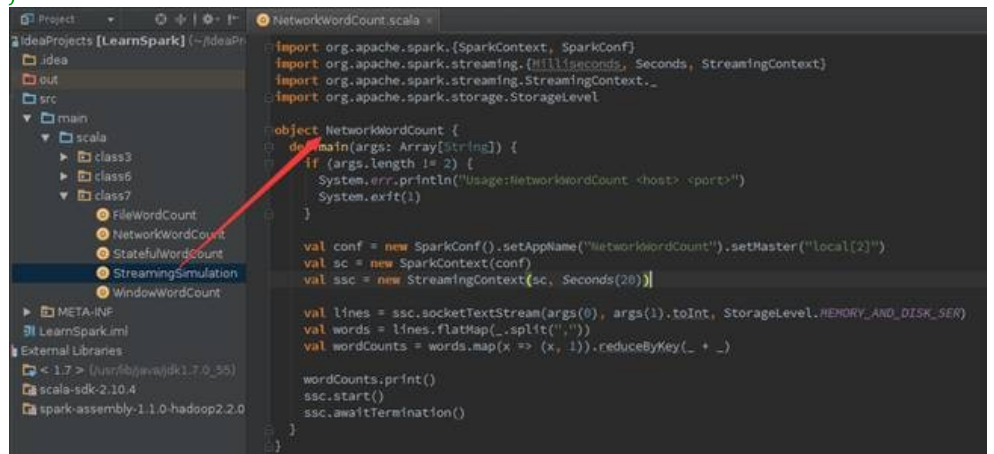
```
import org.apache.spark.{SparkContext, SparkConf}
import org.apache.spark.streaming.{Milliseconds, Seconds, StreamingContext}
import org.apache.spark.streaming.StreamingContext._
import org.apache.spark.storage.StorageLevel

object NetworkWordCount {
  def main(args: Array[String]) {
    val conf = new SparkConf().setAppName("NetworkWordCount").setMaster("local[2]")
    val sc = new SparkContext(conf)
    val ssc = new StreamingContext(sc, Seconds(20))

    // 通过Socket获取数据，该处需要提供Socket的主机名和端口号，数据保存在内存和硬盘中
    val lines = ssc.socketTextStream(args(0), args(1).toInt, StorageLevel.MEMORY_AND_DISK)

    // 对读入的数据进行分割、计数
    val words = lines.flatMap(_.split(" "))
    val wordCounts = words.map(x => (x, 1)).reduceByKey(_ + _)

    wordCounts.print()
    ssc.start()
    ssc.awaitTermination()
  }
}
```



#### 1.3.3 运行代码

第一步 启动流数据模拟器

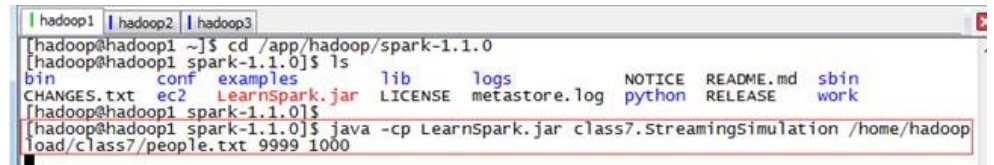
启动4.1打包好的流数据模拟器，在该实例中将定时发送/home/hadoop/upload/class7目录下的people.txt数据文件（该文件可以在本系列配套资源目录/data/class7中找到），其中people.txt数据内容如下：

```
1 Michael
2 Andy
3 Justin
4
```

模拟器Socket端口号为9999，频度为1秒，

```
$cd /app/hadoop/spark-1.1.0
```

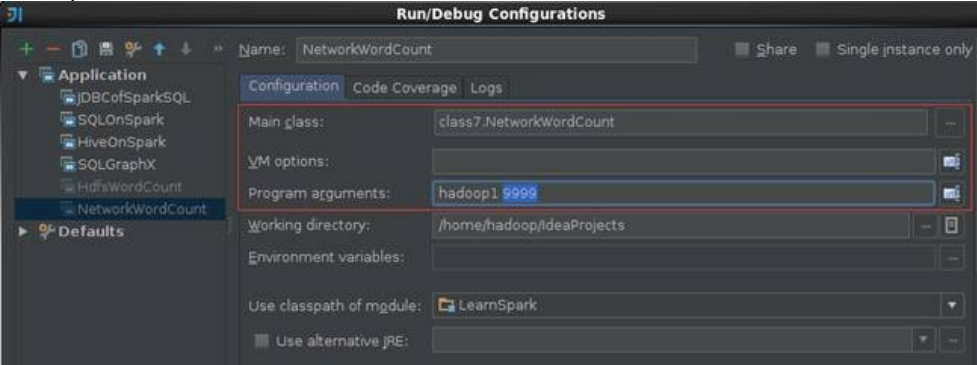
```
$java -cp LearnSpark.jar class7.StreamingSimulation /home/hadoop/upload/class7/people.txt 9999 1000
```



在没有程序连接时，该程序处于阻塞状态

第二步 在IDEA中运行Streaming程序

在IDEA中运行该实例，该实例需要配置连接Socket主机名和端口号，在这里配置参数机器名为hadoop1和端口号为9999



1.3.4 查看结果

第一步 观察模拟器发送情况

IDEA中的Spark Streaming程序运行与模拟器建立连接，当模拟器检测到外部连接时开始发送测试数据，数据是随机的在指定的文件中获取一行数据并发送，时间间隔为1秒



第二步 在监控页面观察执行情况

在webUI上监控作业运行情况，可以观察到每20秒运行一次作业

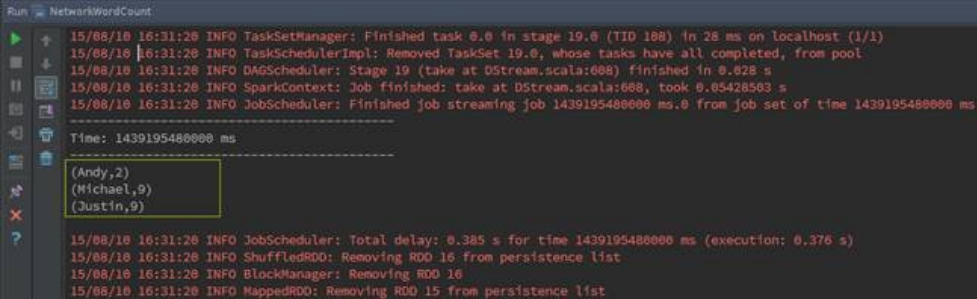
| Active Stages (1) |                                     |                     |          |                        |               |              |               |  |  |
|-------------------|-------------------------------------|---------------------|----------|------------------------|---------------|--------------|---------------|--|--|
| Stage Id          | Description                         | Submitted           | Duration | Tasks: Succeeded/Total | Shuffle Input | Shuffle Read | Shuffle Write |  |  |
| 0                 | runJob at ReceiverTracker.scala:275 | 2015/08/10 16:29:40 | 1.7 min  | 0/1                    |               |              |               |  |  |

| Completed Stages (15) |                               |                     |          |                        |               |              |               |  |  |
|-----------------------|-------------------------------|---------------------|----------|------------------------|---------------|--------------|---------------|--|--|
| Stage Id              | Description                   | Submitted           | Duration | Tasks: Succeeded/Total | Shuffle Input | Shuffle Read | Shuffle Write |  |  |
| 19                    | take at DStream.scala:608     | 2015/08/10 16:31:20 | 28 ms    | 1/1                    |               |              |               |  |  |
| 17                    | take at DStream.scala:608     | 2015/08/10 16:31:20 | 30 ms    | 1/1                    |               |              |               |  |  |
| 18                    | map at MappedDStream.scala:35 | 2015/08/10 16:31:20 | 0.2 s    | 20/20                  |               |              | 3.3 KB        |  |  |
| 15                    | take at DStream.scala:608     | 2015/08/10 16:31:30 | 8 ms     | 1/1                    |               |              |               |  |  |
| 13                    | take at DStream.scala:608     | 2015/08/10 16:31:30 | 32 ms    | 1/1                    |               |              |               |  |  |
| 14                    | map at MappedDStream.scala:35 | 2015/08/10 16:31:30 | 0.2 s    | 20/20                  |               |              | 3.3 KB        |  |  |
| 11                    | take at DStream.scala:608     | 2015/08/10 16:30:40 | 19 ms    | 1/1                    |               |              |               |  |  |

第三步 IDEA运行情况

在IDEA的运行窗口中，可以观测到的统计结果，通过分析在Spark Streaming每段时间内单词数为20，正好是20秒内每秒发送总数。



1.4 实例3：销售数据统计演示

1.4.1 演示说明

在该实例中将由4.1流数据模拟器以1秒的频度发送模拟数据（销售数据），Spark Streaming通过Socket接收流数据并每5秒运行一次用来处理接收到数据，处理完毕后打印该时间段内销售数据总和，需要注意的是各处理段时间之间状态并无关系。

1.4.2 演示代码

```
import org.apache.log4j.{Level, Logger}
```



```

import org.apache.spark.{SparkContext, SparkConf}
import org.apache.spark.streaming.{Milliseconds, Seconds, StreamingContext}
import org.apache.spark.streaming.StreamingContext._
import org.apache.spark.storage.StorageLevel

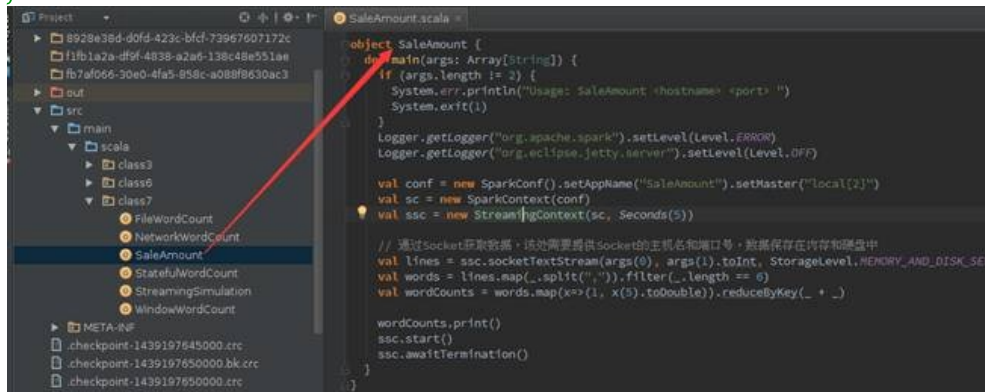
object SaleAmount {
  def main(args: Array[String]) {
    if (args.length != 2) {
      System.err.println("Usage: SaleAmount <hostname> <port> ")
      System.exit(1)
    }
    Logger.getLogger("org.apache.spark").setLevel(Level.ERROR)
    Logger.getLogger("org.eclipse.jetty.server").setLevel(Level.OFF)

    val conf = new SparkConf().setAppName("SaleAmount").setMaster("local[2]")
    val sc = new SparkContext(conf)
    val ssc = new StreamingContext(sc, Seconds(5))

    // 通过Socket获取数据，该处需要提供Socket的主机名和端口号，数据保存在内存和硬盘中
    val lines = ssc.socketTextStream(args(0), args(1).toInt,
      StorageLevel.MEMORY_AND_DISK_SER)
    val words = lines.map(_.split(",")).filter(_.length == 6)
    val wordCounts = words.map(x => (1, x(5).toDouble)).reduceByKey(_ + _)

    wordCounts.print()
    ssc.start()
    ssc.awaitTermination()
  }
}

```



### 1.4.3 运行代码

第一步 启动流数据模拟器

启动4.1打包好的流数据模拟器，在该实例中将定时发送第五课/home/hadoop/upload/class5/saledata目录下的tbStockDetail.txt数据文件（参见第五课《5.Hive（下）--Hive实战》中2.1.2数据描述，该文件可以在本系列配套资源目录/data/class5/saledata中找到），其中表tbStockDetail字段分别为订单号、行号、货品、数量、金额，数据内容如下：

```

代码演示: txt
tbStockDetail.txt
1 BYSL00000893,0,FS527258160501,-1,268,-268
2 BYSL00000893,1,FS527258169701,1,268,268
3 BYSL00000893,2,FS527230163001,1,198,198
4 BYSL00000893,3,24627209125406,1,298,298
5 BYSL00000893,4,K9527220210202,1,120,120
6 BYSL00000893,5,01527291670102,1,268,268
7 BYSL00000893,6,QY527271800242,1,158,158

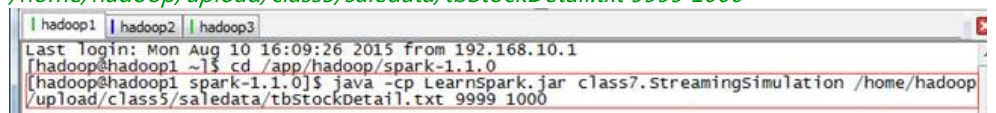
```

模拟器Socket端口号为9999，频度为1秒

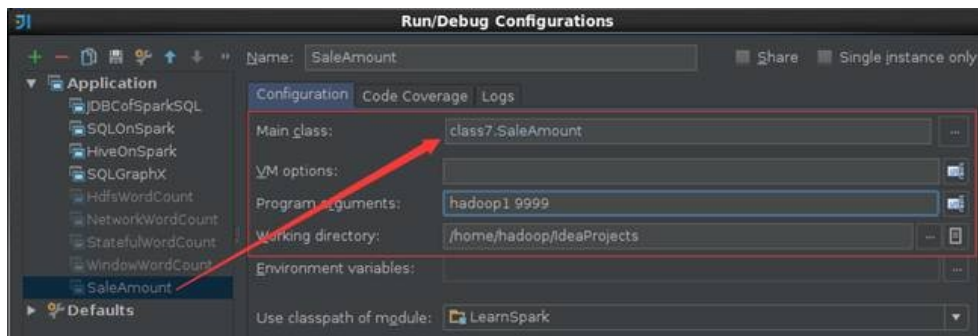
```

$cd /app/hadoop/spark-1.1.0
$java -cp LearnSpark.jar class7.StreamingSimulation
/home/hadoop/upload/class5/saledata/tbStockDetail.txt 9999 1000

```



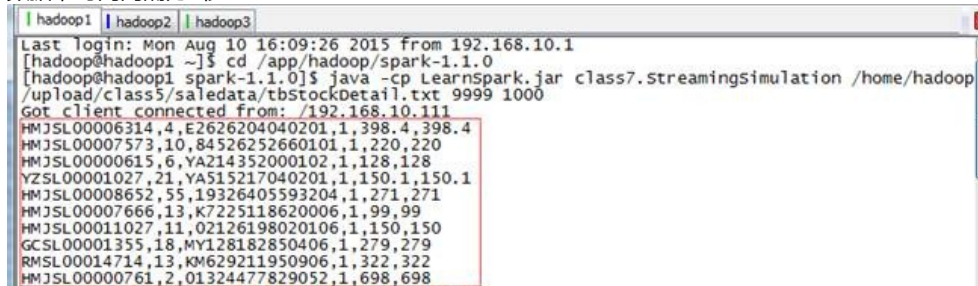
在IDEA中运行该实例，该实例需要配置连接Socket主机名和端口号，在这里配置参数机器名为hadoop1和端口号为9999



#### 1.4.4 查看结果

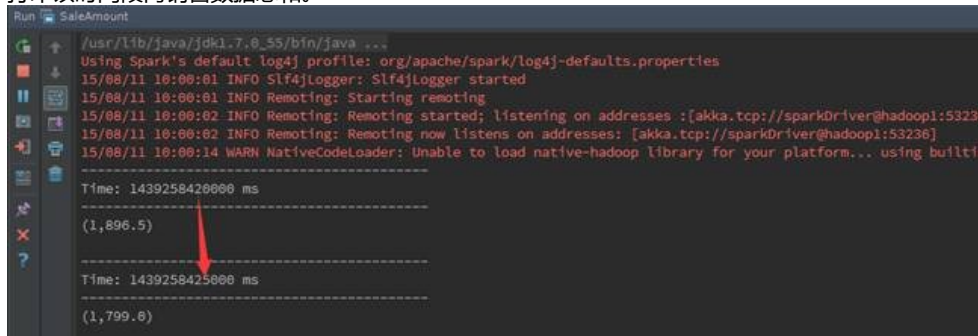
第一步 观察模拟器发送情况

IDEA中的Spark Streaming程序运行与模拟器建立连接，当模拟器检测到外部连接时开始发送销售数据，时间间隔为1秒



第二步 IDEA运行情况

在IDEA的运行窗口中，可以观察到每5秒运行一次作业（两次运行间隔为5000毫秒），运行完毕后打印该时间段内销售数据总和。



第三步 在监控页面观察执行情况

在webUI上监控作业运行情况，可以观察到每5秒运行一次作业

| Active Stages (1) |                                     |                     |          |                        |              |               |
|-------------------|-------------------------------------|---------------------|----------|------------------------|--------------|---------------|
| Stage Id          | Description                         | Submitted           | Duration | Tasks: Succeeded/Total | Shuffle Read | Shuffle Write |
| 0                 | runJob at ReceiverTracker.scala:275 | 2015/08/11 10:00:15 | 3.1 min  | 0/1                    |              |               |

| Completed Stages (111) |                               |                     |          |                        |              |               |
|------------------------|-------------------------------|---------------------|----------|------------------------|--------------|---------------|
| Stage Id               | Description                   | Submitted           | Duration | Tasks: Succeeded/Total | Shuffle Read | Shuffle Write |
| 147                    | take at DStream.scala:608     | 2015/08/11 10:03:20 | 44 ms    | 1/1                    |              |               |
| 145                    | take at DStream.scala:608     | 2015/08/11 10:03:20 | 5 ms     | 1/1                    |              |               |
| 146                    | map at MappedDStream.scala:35 | 2015/08/11 10:03:20 | 18 ms    | 5/5                    |              | 1000.0 B      |
| 143                    | take at DStream.scala:608     | 2015/08/11 10:03:11 | 17 ms    | 1/1                    |              |               |
| 141                    | take at DStream.scala:608     | 2015/08/11 10:03:11 | 4 ms     | 1/1                    |              |               |
| 142                    | map at MappedDStream.scala:35 | 2015/08/11 10:03:15 | 41 ms    | 5/5                    |              | 1000.0 B      |
| 139                    | take at DStream.scala:608     | 2015/08/11 10:03:10 | 7 ms     | 1/1                    |              |               |

### 1.5 实例4：Stateful演示

#### 1.5.1 演示说明

该实例为Spark Streaming状态操作，模拟数据由4.1流数据模拟以1秒的频度发送，Spark Streaming通过Socket接收流数据并每5秒运行一次用来处理接收到数据，处理完毕后打印程序启动后单词出现的频度，相比较前面4.3实例在该实例中各时间段之间状态是相关的。

#### 1.5.2 演示代码

```
import org.apache.log4j.{Level, Logger}
import org.apache.spark.{SparkContext, SparkConf}
import org.apache.spark.streaming.{Seconds, StreamingContext}
```



```
import org.apache.spark.streaming.StreamingContext._

object StatefulWordCount {
  def main(args: Array[String]) {
    if (args.length != 2) {
      System.err.println("Usage: StatefulWordCount <filename> <port> ")
      System.exit(1)
    }
    Logger.getLogger("org.apache.spark").setLevel(Level.ERROR)
    Logger.getLogger("org.eclipse.jetty.server").setLevel(Level.OFF)

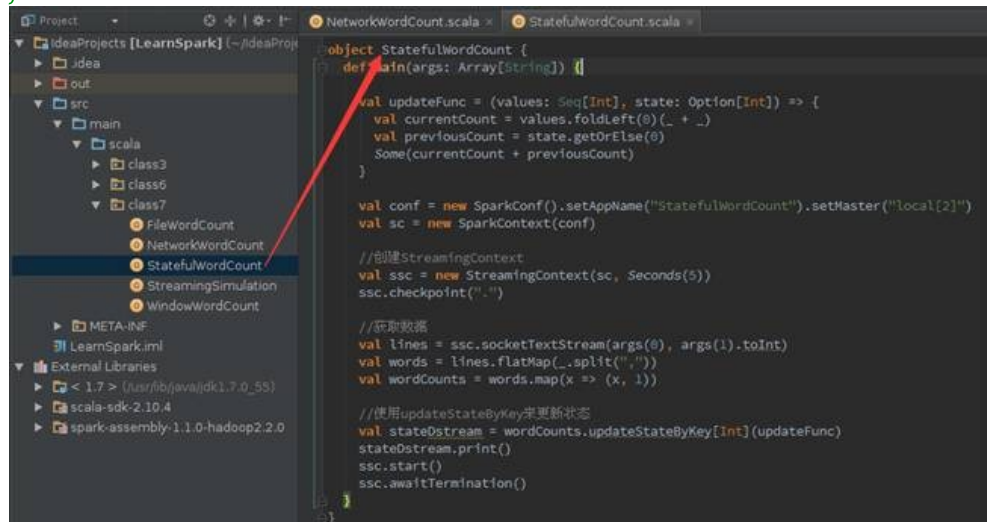
    // 定义更新状态方法, 参数values为当前批次单词频度, state为以往批次单词频度
    val updateFunc = (values: Seq[Int], state: Option[Int]) => {
      val currentCount = values.foldLeft(0)(_ + _)
      val previousCount = state.getOrElse(0)
      Some(currentCount + previousCount)
    }

    val conf = new SparkConf().setAppName("StatefulWordCount").setMaster("local[2]")
    val sc = new SparkContext(conf)

    // 创建StreamingContext, Spark Streaming运行时间间隔为5秒
    val ssc = new StreamingContext(sc, Seconds(5))
    // 定义checkpoint目录为当前目录
    ssc.checkpoint(".")

    // 获取从Socket发送过来数据
    val lines = ssc.socketTextStream(args(0), args(1).toInt)
    val words = lines.flatMap(_.split(" "))
    val wordCounts = words.map(x => (x, 1))

    // 使用updateStateByKey来更新状态, 统计从运行开始以来单词总的次数
    val stateDstream = wordCounts.updateStateByKey[Int](updateFunc)
    stateDstream.print()
    ssc.start()
    ssc.awaitTermination()
  }
}
```



### 1.5.3 运行代码

#### 第一步 启动流数据模拟器

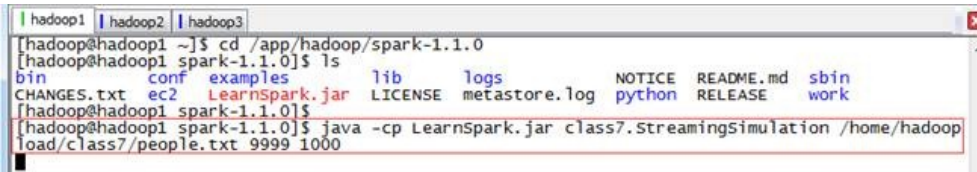
启动4.1打包好的流数据模拟器, 在该实例中将定时发送/home/hadoop/upload/class7目录下的people.txt数据文件(该文件可以在本系列配套资源目录/data/class7中找到), 其中people.txt数据内容如下:

```
1 Michael
2 Andy
3 Justin
4
```

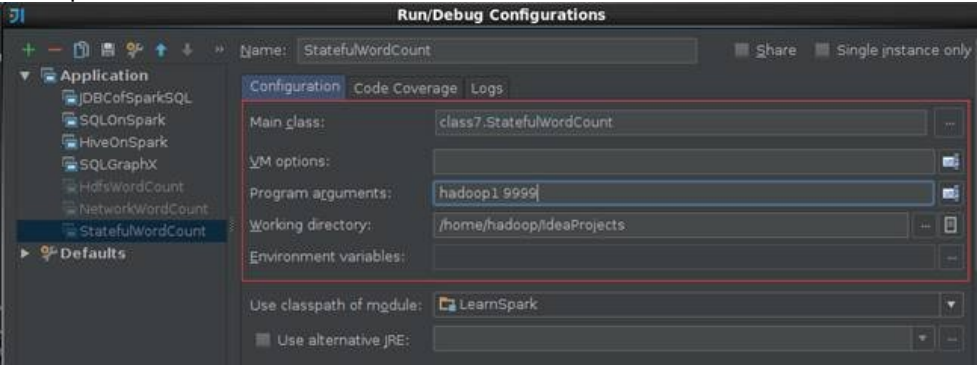
模拟器Socket端口号为9999, 频度为1秒

```
$cd /app/hadoop/spark-1.1.0
```

```
$java -cp LearnSpark.jar class7.StreamingSimulation /home/hadoop/upload/class7/people
1000
```

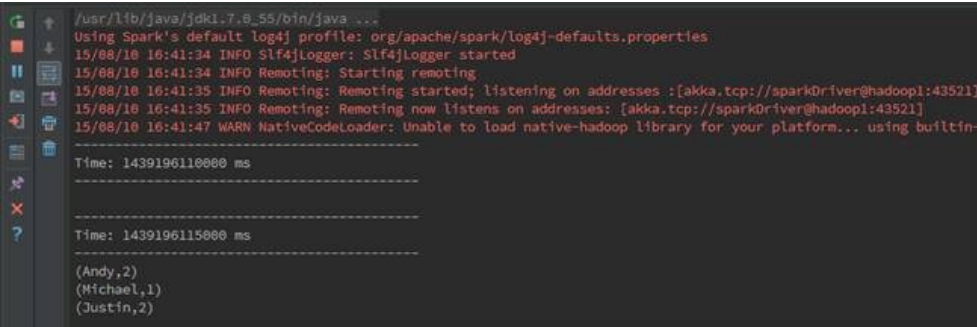


在没有程序连接时，该程序处于阻塞状态，在IDEA中运行Streaming程序  
在IDEA中运行该实例，该实例需要配置连接Socket主机名和端口号，在这里配置参数机器名为hadoop1和端口号为9999



1.5.4 查看结果

第一步 IDEA运行情况  
在IDEA的运行窗口中，可以观察到第一次运行统计单词总数为1，第二次为6，第N次为5(N-1)+1，即统计单词的总数为程序运行单词数总和。

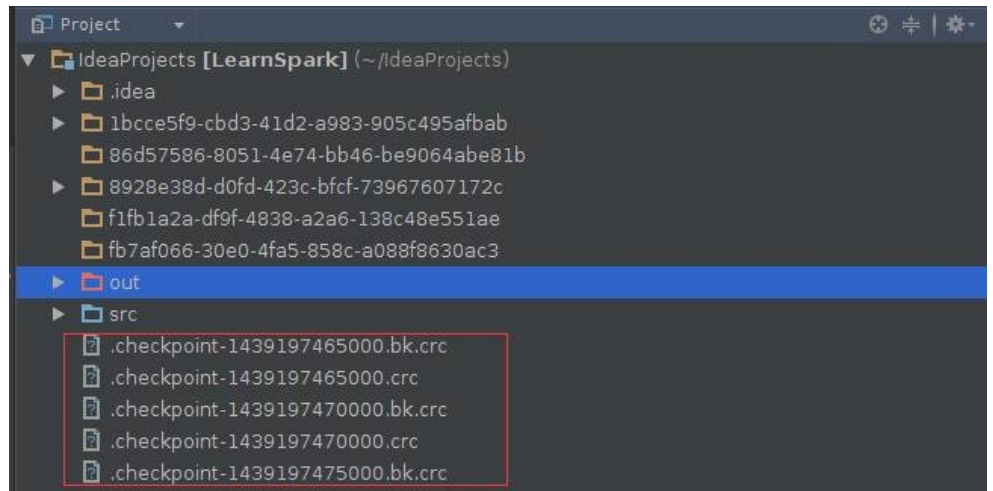


第二步 在监控页面观察执行情况  
在webUI上监控作业运行情况，可以观察到每5秒运行一次作业

| Active Stages (1) |                                     |                     |          | Tasks:          |       | Shuffle |       |
|-------------------|-------------------------------------|---------------------|----------|-----------------|-------|---------|-------|
| Stage Id          | Description                         | Submitted           | Duration | Succeeded/Total | Input | Read    | Write |
| 0                 | runJob at ReceiverTracker.scala:275 | 2015/08/10 16:41:49 | 40 s     | 0/1             |       |         |       |

| Completed Stages (27) |                               |                     |          | Tasks:          |         | Shuffle |         |
|-----------------------|-------------------------------|---------------------|----------|-----------------|---------|---------|---------|
| Stage Id              | Description                   | Submitted           | Duration | Succeeded/Total | Input   | Read    | Write   |
| 44                    | take at DStream.scala:608     | 2015/08/10 16:42:25 | 3 ms     | 1/1             | 181.0 B |         |         |
| 41                    | take at DStream.scala:608     | 2015/08/10 16:42:25 | 0.2 s    | 2/2             | 334.0 B |         |         |
| 38                    | take at DStream.scala:608     | 2015/08/10 16:42:25 | 21 ms    | 1/1             | 153.0 B |         |         |
| 39                    | map at MappedDStream.scala:35 | 2015/08/10 16:42:25 | 60 ms    | 5/5             |         |         | 853.0 B |
| 36                    | take at DStream.scala:608     | 2015/08/10 16:42:20 | 41 ms    | 1/1             | 181.0 B |         |         |
| 34                    | take at DStream.scala:608     | 2015/08/10 16:42:20 | 18 ms    | 1/1             | 153.0 B |         |         |
| 35                    | map at MappedDStream.scala:35 | 2015/08/10 16:42:20 | 57 ms    | 5/5             |         |         | 846.0 B |
| 33                    | take at DStream.scala:608     | 2015/08/10 16:42:15 | 15 ms    | 1/1             | 181.0 B |         |         |

第三步 查看CheckPoint情况  
在项目根目录下可以看到checkpoint文件



## 1.6 实例5：Window演示

### 1.6.1 演示说明

该实例为Spark Streaming窗口操作，模拟数据由4.1流数据模拟以1秒的频率发送，Spark Streaming通过Socket接收流数据并每10秒运行一次用来处理接收到数据，处理完毕后打印程序启动后单词出现的频率。相比前面的实例，Spark Streaming窗口统计是通过reduceByKeyAndWindow()方法实现的，在该方法中需要指定窗口时间长度和滑动时间间隔。

### 1.6.2 演示代码

```
import org.apache.log4j.{Level, Logger}
import org.apache.spark.{SparkContext, SparkConf}
import org.apache.spark.storage.StorageLevel
import org.apache.spark.streaming._
import org.apache.spark.streaming.StreamingContext._

object WindowWordCount {
  def main(args: Array[String]) {
    if (args.length != 4) {
      System.err.println("Usage: WindowWorldCount <filename> <port> <windowDuration> <slideDuration>")
      System.exit(1)
    }
    Logger.getLogger("org.apache.spark").setLevel(Level.ERROR)
    Logger.getLogger("org.eclipse.jetty.server").setLevel(Level.OFF)

    val conf = new SparkConf().setAppName("WindowWordCount").setMaster("local[2]")
    val sc = new SparkContext(conf)

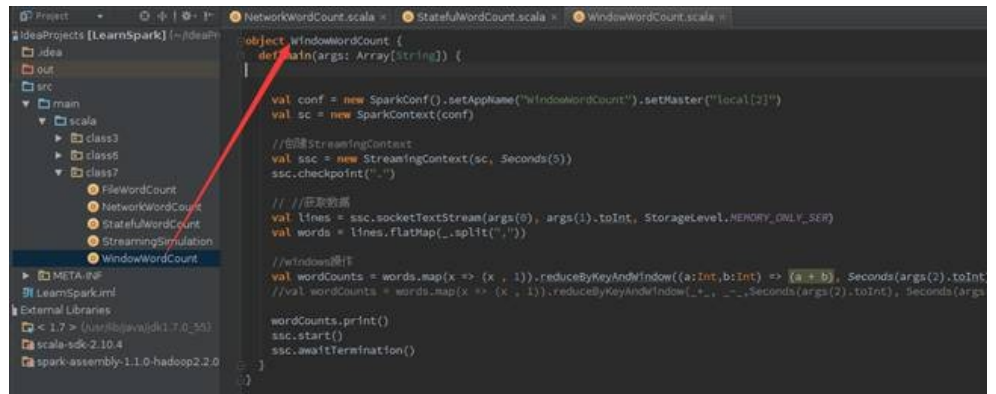
    // 创建StreamingContext
    val ssc = new StreamingContext(sc, Seconds(5))
    // 定义checkpoint目录为当前目录
    ssc.checkpoint(".")

    // 通过Socket获取数据，该处需要提供Socket的主机名和端口号，数据保存在内存和硬盘中
    val lines = ssc.socketTextStream(args(0), args(1).toInt, StorageLevel.MEMORY_ONLY_SE)
    val words = lines.flatMap(_.split(" "))

    // windows操作，第一种方式为叠加处理，第二种方式为增量处理
    val wordCounts = words.map(x => (x, 1)).reduceByKeyAndWindow((a: Int, b: Int) => (a + b), Seconds(args(2).toInt), Seconds(args(3).toInt))
    // val wordCounts = words.map(x => (x, 1)).reduceByKeyAndWindow(_+_ , Seconds(args(2).toInt), Seconds(args(3).toInt))

    wordCounts.print()
    ssc.start()
    ssc.awaitTermination()
  }
}
```





### 1.6.3 运行代码

第一步 启动流数据模拟器

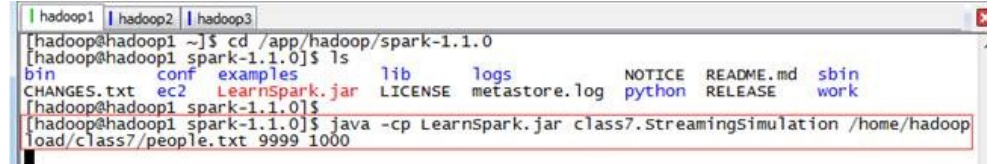
启动4.1打包好的流数据模拟器，在该实例中将定时发送/home/hadoop/upload/class7目录下的people.txt数据文件（该文件可以在本系列配套资源目录/data/class7中找到），其中people.txt数据内容如下：

```
1 Michael
2 Andy
3 Justin
4
```

模拟器Socket端口号为9999，频度为1秒

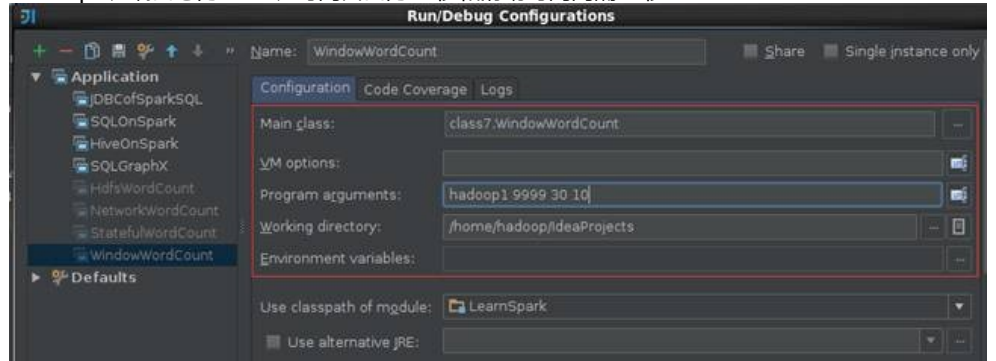
```
$cd /app/hadoop/spark-1.1.0
```

```
$java -cp LearnSpark.jar class7.StreamingSimulation /home/hadoop/upload/class7/people 1000
```



在没有程序连接时，该程序处于阻塞状态，在IDEA中运行Streaming程序

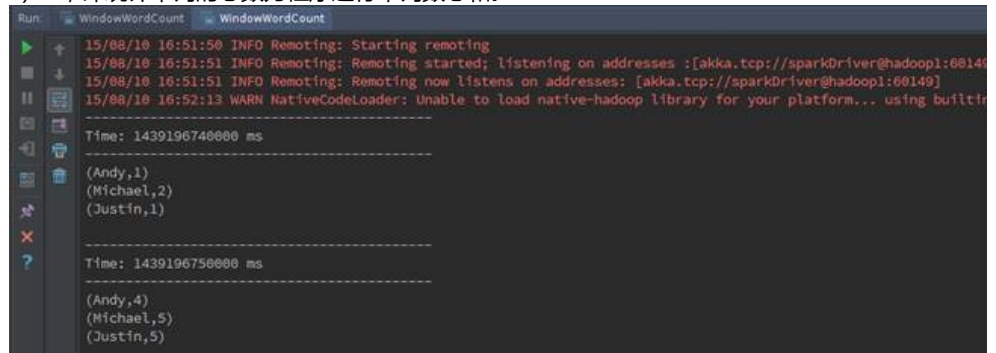
在IDEA中运行该实例，该实例需要配置连接Socket主机名和端口号，在这里配置参数机器名为hadoop1、端口号为9999、时间窗口为30秒和滑动时间间隔10秒



### 1.6.4 查看结果

第一步 IDEA运行情况

在IDEA的运行窗口中，可以观察到第一次运行统计单词总数为4，第二次为14，第N次为10(N-1)+4，即统计单词的总数为程序运行单词数总和。



第二步 在监控页面观察执行情况

在webUI上监控作业运行情况，可以观察到每10秒运行一次作业

| Active Stages (1) |                                     |                 |                     |          |                        |       |              |               |  |
|-------------------|-------------------------------------|-----------------|---------------------|----------|------------------------|-------|--------------|---------------|--|
| Stage Id          | Description                         |                 | Submitted           | Duration | Tasks: Succeeded/Total | Input | Shuffle Read | Shuffle Write |  |
| 0                 | runJob at ReceiverTracker.scala:275 | +details (kill) | 2015/09/10 16:46:03 | 18 ms    | 0/1                    |       |              |               |  |

| Completed Stages (338) |                               |          |                     |          |                        |       |              |               |  |
|------------------------|-------------------------------|----------|---------------------|----------|------------------------|-------|--------------|---------------|--|
| Stage Id               | Description                   |          | Submitted           | Duration | Tasks: Succeeded/Total | Input | Shuffle Read | Shuffle Write |  |
| 1494                   | take at DStream.scala:608     | +details | 2015/09/10 17:04:00 | 6 ms     | 1/1                    | 4.0 B |              |               |  |
| 1487                   | take at DStream.scala:608     | +details | 2015/09/10 17:04:00 | 11 ms    | 1/1                    | 4.0 B |              |               |  |
| 1489                   | map at MappedDStream.scala:35 | +details | 2015/09/10 17:04:00 | 53 ms    | 10/10                  |       |              | 1692.0 B      |  |
| 1480                   | take at DStream.scala:608     | +details | 2015/09/10 17:03:50 | 5 ms     | 1/1                    | 4.0 B |              |               |  |
| 1473                   | take at DStream.scala:608     | +details | 2015/09/10 17:03:50 | 10 ms    | 1/1                    | 4.0 B |              |               |  |
| 1475                   | map at MappedDStream.scala:35 | +details | 2015/09/10 17:03:50 | 52 ms    | 10/10                  |       |              | 1697.0 B      |  |

作者：[石山园](#) 出处：<http://www.cnblogs.com/shishanyuan/>  
本文版权归作者和博客园共有，欢迎转载，但未经作者同意必须保留此段声明，且在文章页面明显位置给出原文连接，否则保留追究法律责任的权利。如果觉得还有帮助的话，可以点一下右下角的【推荐】，希望能够持续的为大家带来好的技术文章！想跟我一起进步么？那就【关注】我吧。

分类: [20.Spark入门实战系列](#)

« 上一篇: [Spark入门实战系列--7.Spark Streaming（上）--实时流计算Spark Streaming原理介绍](#)  
» 下一篇: [Spark入门实战系列--8.Spark MLlib（上）--机器学习及SparkMLlib简介](#)

posted @ 2015-09-07 07:01 shishanyuan 阅读(21442) 评论(14) 编辑 收藏

发表评论

# 1楼 2015-12-21 15:40 | zookeepers

老师，问一下，您的idea是装在linux系统里的吗？

支持(0) 反对(0)

# 2楼[楼主] 2015-12-21 21:10 | shishanyuan

@ zookeepers  
对，该系列中演示的idea是装在桌面的linux中  
idea可以装在windows也可以装在linux系统中

支持(0) 反对(0)

# 3楼 2015-12-22 15:35 | zookeepers

@ shishanyuan  
老师，你好，我的idea是装在window中，在运行FileWordCount的程序时，总是找不到textFileStream路径，在window中可以直接引用spark集群中的路径吗？我在conf中已经设置好了Master,但是还是找不到

支持(0) 反对(0)

# 4楼[楼主] 2015-12-24 10:36 | shishanyuan

@ zookeepers  
我没有在windows系统中使用过，是不是textFileStream路径写法是不是需要处理一下，例如"file:\\\\D:\\..."这样形式？  
在window中应该可以直接引用spark集群中的路径，现在cmd命令工具中ping一下spark master是否能够连通，如果使用机器名需要在hosts文件进行映射

支持(0) 反对(0)

# 5楼 2016-07-26 15:10 | 开开心

老师你好，看了你的文章受益匪浅，不过中间也碰到了几个问题：  
1,4.1中模拟器的 val content = lines(index(filerow))这一句，报错说是类型不匹配。我也有疑问，这个lines是String类型的list，要是取某个下标的元素是不是要加上indexof，我加上之后程序就没有报错了。这中间是什么原因？  
2，第二个例子，我是在windows下跑的，但是无论是print还是保存到本地，都没有单词结果的统计。print只有时间信息，保存到本地的文件里面part-0000也是空的，什么都没有。刚接触spark streaming，有些问题理解的不是很彻，望老师能够给予解答。谢谢！！

支持(0) 反对(0)

---

#6楼[楼主] 2016-07-26 16:10 | shishanyuan

- 1、你理解lines是String类型的list这是没有问题，不过Scala中可以直接通过lines(index)来制定获取元素；
- 2、试一下wordCounts.collect().print()

支持(0) 反对(0)

---

#7楼 2016-08-26 17:23 | 小李飛菜刀

socket数据源的话，部署到yarn上，是一个节点接收据，还是其他节点也能接数据， 怎样一个处理过程？

支持(0) 反对(0)

---

#8楼 2017-01-06 12:00 | Z漫步

@ 开开好

"这些文件通过原子移动或重命名文件的方式在dataDirectory创建"

如果你是直接在目录下新建文件的话 是检测不到的，只能以上边的方式。我试过在其他目录新建文件然后复制进检测录是可以打印出来的。 重命名的方式 我没有实验成功

支持(0) 反对(0)

---

#9楼 2017-01-12 15:19 | superlouxuwei

老师，你好，一直在学习你的文章，想请教您一下，我在使用textFileStream时，一直获取不到数据，请问您一下是什么原因呢？我的文件是放在linux主机的文件系统中，没有使用hdfs，请问和这个有原因吗？希望能得到您的回复，谢谢您老师

支持(0) 反对(0)

---

#10楼 2017-03-29 11:28 | THHao

老师，您好，我用命令行提交.py文件，数据来自kinesis

```
lines = KinesisUtils.createStream(ssc, "testApp", "mySparkStream",  
"https://kinesis.cn-north-1.amazonaws.com", "cn-north-1", InitialPositionInStream.LATEST, 2)
```

运行报错：

Spark Streaming's Kinesis libraries not found in class path. Try one of the following.

1. Include the Kinesis library and its dependencies with in the spark-submit command as

```
$ bin/spark-submit --packages org.apache.spark:spark-streaming-kinesis-asl:2.1.0 ...
```

2. Download the JAR of the artifact from Maven Central <http://search.maven.org/>,  
Group Id = org.apache.spark, Artifact Id = spark-streaming-kinesis-asl-assembly, Version = 2.1.0.  
Then, include the jar in the spark-submit command as

```
$ bin/spark-submit --jars <spark-streaming-kinesis-asl-assembly.jar> ...
```

---

```
-----  
TypeError: Traceback (most recent call last)  
<ipython-input-12-aa6ce35534f6> in <module>()  
9  
10 lines = KinesisUtils.createStream(ssc, "testApp", "mySparkStream",  
--> 11 "https://kinesis.cn-north-1.amazonaws.com", "cn-north-1", InitialPositionInStream.LATEST,  
  
/usr/lib/spark/python/pyspark/streaming/kinesis.pyc in createStream(ssc, kinesisAppName, streamName,  
me, endpointUrl, regionName, initialPositionInStream, checkpointInterval, storageLevel, awsAccessKey  
Id, awsSecretKey, decoder)  
75 try:  
76 # Use KinesisUtilsPythonHelper to access Scala's KinesisUtils  
--> 77 helper = ssc._jvm.org.apache.spark.streaming.kinesis.KinesisUtilsPythonHelper()  
78 except TypeError as e:  
79 if str(e) == "'JavaPackage' object is not callable":  
  
TypeError: 'JavaPackage' object is not callable
```

怎么解决啊？谢谢了

支持(0) 反对(0)

---

#11楼[楼主] 2017-03-31 09:33 | shishanyuan



@ THHao

问题主要出在“Spark Streaming's Kinesis libraries not found in class path. Try one of the following.”根据示需要引入Kinesis 相关jar包

支持(1) 反对(0)

#12楼 2017-03-31 15:14 | THHao

@ shishanyuan

谢谢您，我再试试

支持(0) 反对(0)

#13楼 2017-08-30 11:23 | loverlself

老师你好，最近在学习您的文章，在学习到本节到时候，运行实例2的代码，出现如下报错

```
1 Exception in thread "main" java.lang.NoClassDefFoundError: org/apache/spark/streaming/St
2     at com.netbox.FileWordCount$.main(FileWordCount.scala:12)
3     at com.netbox.FileWordCount.main(FileWordCount.scala)
4 Caused by: java.lang.ClassNotFoundException: org.apache.spark.streaming.StreamingContext
5     at java.net.URLClassLoader.findClass(URLClassLoader.java:381)
6     at java.lang.ClassLoader.loadClass(ClassLoader.java:424)
7     at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:331)
8     at java.lang.ClassLoader.loadClass(ClassLoader.java:357)
9     ... 2 more
```

我到环境是jdk1.8.0\_111，scala2.10.5，hadoop2.6.1，spark1.6.2，idea是在centos上运行的，请问一下是什么原因？

支持(0) 反对(0)

#14楼 2018-03-06 22:59 | 小鸟爱吃醋

Spark Streaming实时流处理项目实战

网盘地址：<https://pan.baidu.com/s/1nwuLT3N> 密码: hrck

支持(0) 反对(0)

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。

【推荐】超50万VC++源码：大型组态工控、电力仿真CAD与GIS源码库！

【前端】SpreadJS表格控件，可嵌入应用开发的在线Excel

【免费】程序员21天搞定英文文档阅读

【推荐】如何快速搭建人工智能应用？



最新IT新闻：

- 进击的长租公寓：没有“情怀” 只有“套路”
- 腾讯云“老带新”拼团计划最后一天 最长可获40个月云服务器
- 亚马逊似已停产Kindle Voyage电子书阅读器
- Snapchat为Android版带来全新改造 但用户使用需root权限
- Tim Cook发文庆祝世界摄影日 南极洲企鹅出镜

» 更多新闻...

最新知识库文章：

- 一个故事看懂“区块链”
- 被踢出去的用户
- 成为一个有目标的学习者
- 历史转折中的“杭派工程师”
- 如何提高代码质量？
- » 更多知识库文章...

Copyright ©2018 shishanyuan