

Vijay's Assignment – Spark Streaming2

There are two parts this case study

▪ First Part - You have to create a Spark Application which streams data from a file on local directory on your machine and does the word count on the fly. The word should be done by the spark application in such a way that as soon as you drop the file in your local directory, your spark application should immediately do the word count for you.

What did I do?

- 1) Created a folder C:\Users\VIJAYLAKSHMANAN\spark\input
- 2) Spark streaming code given below monitors this folder for any "new" files copied to it.
- 3) Once a file is received, it runs a flatmap to remove all the hierarchical lines and then map to suffix 1 to each word. Using reduceByKey function each word (key) occurrence is counted
- 4) The file uploaded has data separated by ","



5)

Code:

```
package vksp1
import org.apache.spark.SparkConf
import org.apache.spark.SparkContext
import org.apache.spark.rdd.RDD.rddToPairRDDFunctions
import org.apache.spark.streaming._
import org.apache.spark.streaming.StreamingContext

object vjspstream2 {
  def main(args: Array[String]) = {
    val conf = new SparkConf()
      .setAppName("vjspstream2")
      .setMaster("local")

    val ssc = new StreamingContext(conf, Seconds(60))

    val lines1 = ssc.textFileStream("C:\\Users\\VIJAYLAKSHMANAN\\spark\\input")

    val words = lines1.flatMap(line=>line.split(","))

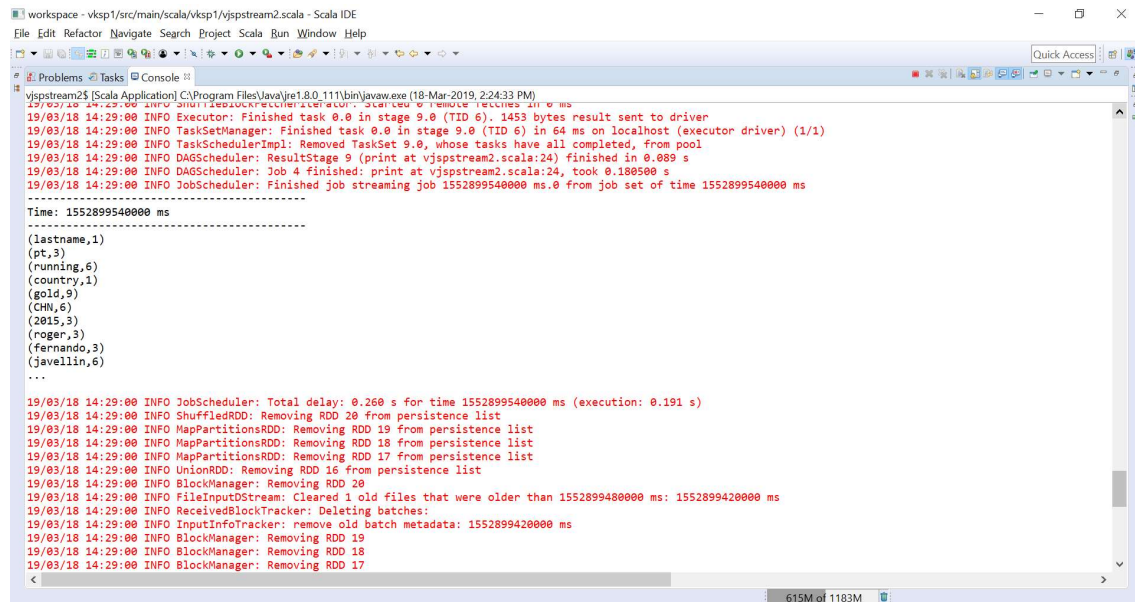
    val pairs = words.map(x=>(x,1))

    val wcnt = pairs.reduceByKey(_+_ )

    wcnt.print()

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

Output

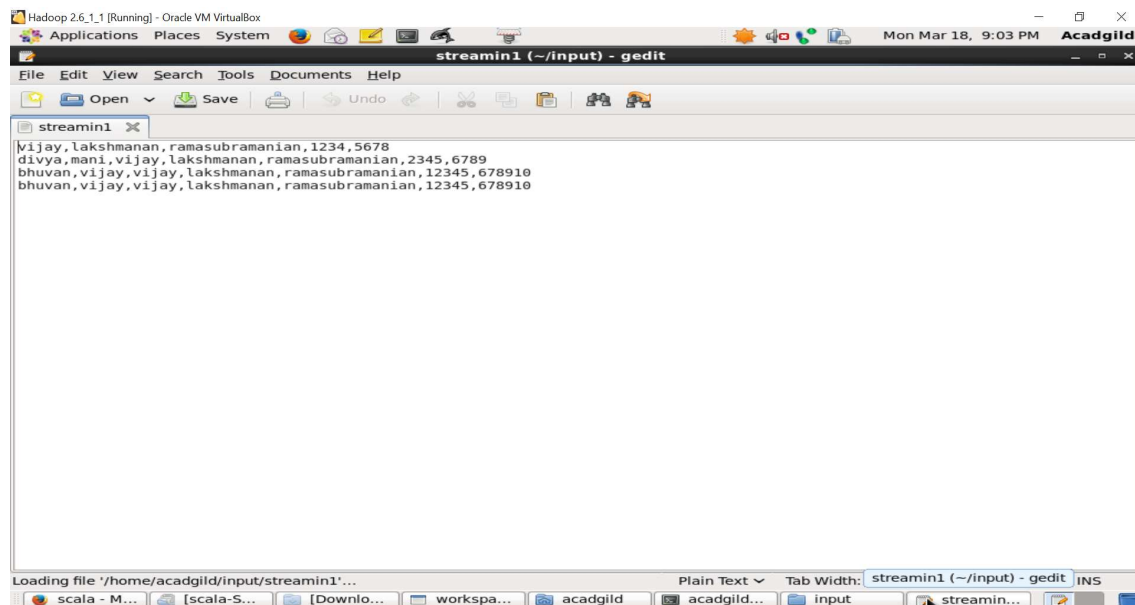


```
workspace - vksp1/src/main/scala/vksp1/vjspstream2.scala - Scala IDE
File Edit Refactor Navigate Search Project Scala Run Window Help

vjspstream2$ [Scala Application] C:\Program Files\Java\jre1.8.0_111\bin\javaw.exe (18-Mar-2019, 2:24:33 PM)
19/03/18 14:29:00 INFO TaskSchedulerImpl: Started to remove finished driver
19/03/18 14:29:00 INFO Executor: Finished task 0.0 in stage 9.0 (TID 6). 1453 bytes result sent to driver
19/03/18 14:29:00 INFO TaskSetManager: Finished task 0.0 in stage 9.0 (TID 6) in 64 ms on localhost (executor driver) (1/1)
19/03/18 14:29:00 INFO TaskSchedulerImpl: Removed TaskSet 9.0, whose tasks have all completed, from pool
19/03/18 14:29:00 INFO DAGScheduler: ResultStage 9 (print at vjspstream2.scala:24) finished in 0.089 s
19/03/18 14:29:00 INFO DAGScheduler: Job 4 finished: print at vjspstream2.scala:24, took 0.180500 s
19/03/18 14:29:00 INFO JobScheduler: Finished job streaming job 1552899540000 ms.0 from job set of time 1552899540000 ms
-----
Time: 1552899540000 ms
-----
(lastname,1)
(pt,3)
(running,6)
(country,1)
(gold,9)
(CHN,6)
(2015,3)
(roger,3)
(fernando,3)
(javellin,6)
...
19/03/18 14:29:00 INFO JobScheduler: Total delay: 0.260 s for time 1552899540000 ms (execution: 0.191 s)
19/03/18 14:29:00 INFO ShuffledRDD: Removing RDD 20 from persistence list
19/03/18 14:29:00 INFO MapPartitionsRDD: Removing RDD 19 from persistence list
19/03/18 14:29:00 INFO MapPartitionsRDD: Removing RDD 18 from persistence list
19/03/18 14:29:00 INFO MapPartitionsRDD: Removing RDD 17 from persistence list
19/03/18 14:29:00 INFO UnionRDD: Removing RDD 16 from persistence list
19/03/18 14:29:00 INFO BlockManager: Removing RDD 20
19/03/18 14:29:00 INFO FileInputFileStream: Cleared 1 old files that were older than 1552899480000 ms: 1552899420000 ms
19/03/18 14:29:00 INFO ReceivedBlockTracker: Deleting batches:
19/03/18 14:29:00 INFO InputInfoTracker: remove old batch metadata: 1552899420000 ms
19/03/18 14:29:00 INFO BlockManager: Removing RDD 19
19/03/18 14:29:00 INFO BlockManager: Removing RDD 18
19/03/18 14:29:00 INFO BlockManager: Removing RDD 17
```

▪ Second Part - In this part, you will have to create a Spark Application which should do the following

Text File used:



```
Hadoop 2.6.1_1 [Running] - Oracle VM VirtualBox
Applications Places System
streamin1 (~/.input) - gedit
File Edit View Search Tools Documents Help
Loading file '/home/acadgild/input/streamin1'...
Plain Text Tab Width: streamin1 (~/.input) - gedit INS
scala - M... [scala-S... [Downlo... workspa... acadgild acadgild... input streamin...
```

Vijay,lakshmanan,ramasubramanian,1234,5678
divya,mani,vijay,lakshmanan,ramasubramanian,2345,6789
bhuvan,vijay,vijay,lakshmanan,ramasubramanian,12345,678910
bhuvan,vijay,vijay,lakshmanan,ramasubramanian,12345,678910

1. Pick up a file from the local directory and do the word count

Code:

```
vjspstream2.scala 83
package com.devinline.spark
import org.apache.spark.SparkConf
import org.apache.spark.SparkContext
import org.apache.spark.streaming._
import org.apache.spark.streaming.StreamingContext

import org.apache.hadoop.conf.Configuration
import org.apache.hadoop.fs.FileSystem
import org.apache.hadoop.fs.Path

import org.apache.commons.io.IOUtils

object vjspstream2 {
  def main(args: Array[String]){
    val conf = new SparkConf().setAppName("vjspstream2").setMaster("local")
    val sc = new SparkContext(conf)

    val test = sc.textFile("file:///home//acadgild//input//streamin1")

    val words=test.flatMap(line=>line.split(","))
    val paris=words.map(x=>(x,1))
    val wcnt=paris.reduceByKey(_+_ )

    println(wcnt.collect().mkString(","))
  }
}
```

Output:

```
19/03/18 20:57:44 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 18 ms
19/03/18 20:57:44 INFO Executor: Finished task 0.0 in stage 1.0 (TID 1). 1580 bytes result sent to driver
19/03/18 20:57:44 INFO TaskSetManager: Finished task 0.0 in stage 1.0 (TID 1) in 330 ms on localhost (executor drive
19/03/18 20:57:44 INFO DAGScheduler: ResultStage 1 (collect at vjspstream2.scala:25) finished in 0.451 s
19/03/18 20:57:44 INFO TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have all completed, from pool
19/03/18 20:57:44 INFO DAGScheduler: Job 0 finished: collect at vjspstream2.scala:25, took 3.194586 s
(678910,2),(divya,1),(ramasubramanian,4),(12345,2),(lakshmanan,4),(mani,1),(6789,1),(,1),(2345,1),(1234,1),(5678,1),
```

2. Then in the same Spark Application, write the code to put the same file on HDFS.

3. Then in same Spark Application, do the word count of the file

copied on HDFS in step 2

4. Lastly, compare the word count of step 1 and 2. Both should

match, other throw an error