

# **Spark Streaming – Case Study**

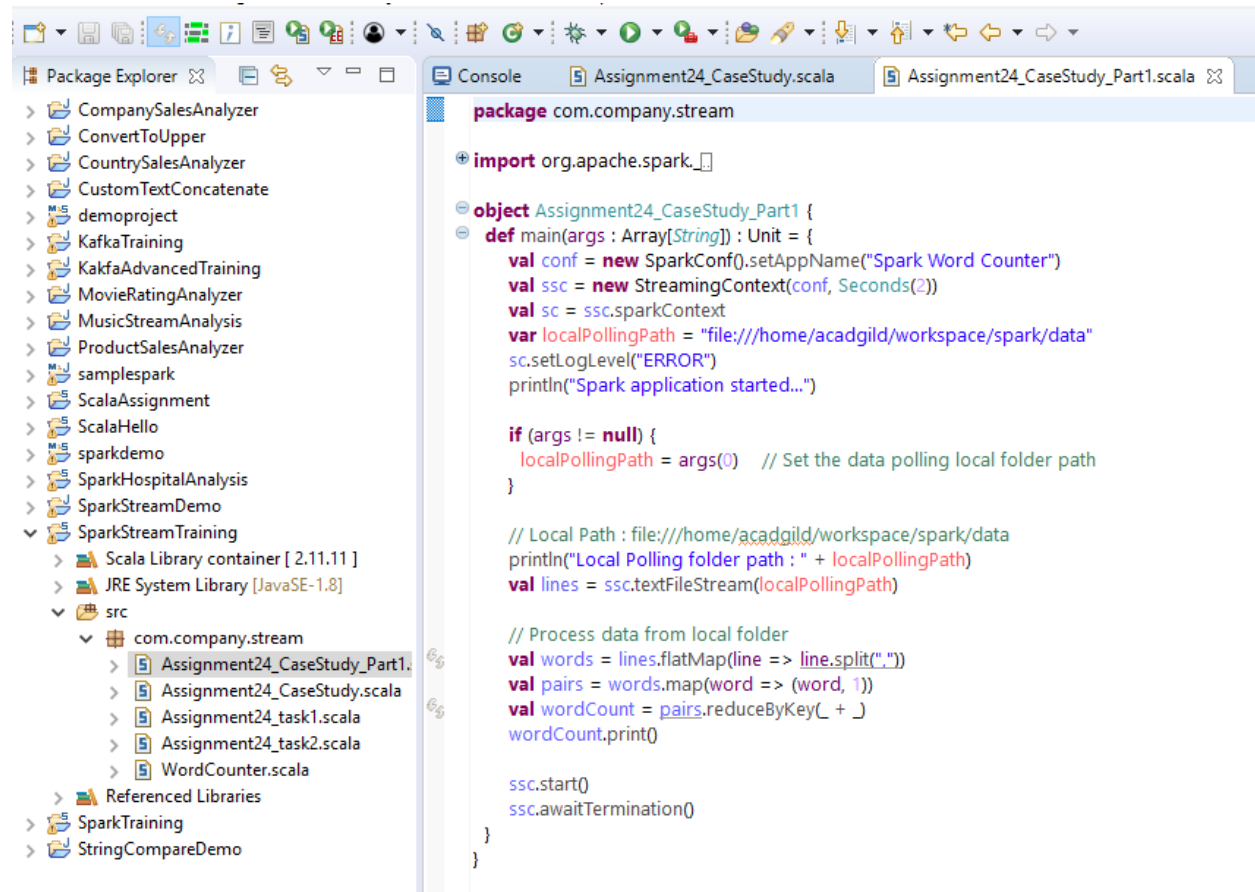
Date: 18-Mar-2019

Author: Saravanan Ponnaiah

## Case Study – Part 1:

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.

Program:



```
package com.company.stream

import org.apache.spark._

object Assignment24_CaseStudy_Part1 {
  def main(args : Array[String]) : Unit = {
    val conf = new SparkConf().setAppName("Spark Word Counter")
    val ssc = new StreamingContext(conf, Seconds(2))
    val sc = ssc.sparkContext
    var localPollingPath = "file:///home/acadgild/workspace/spark/data"
    sc.setLogLevel("ERROR")
    println("Spark application started...")

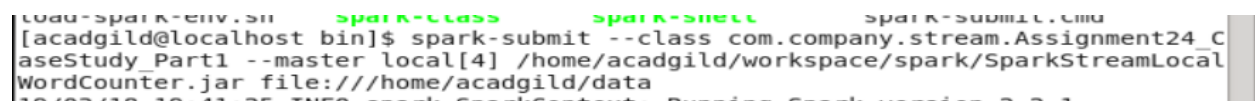
    if (args != null) {
      localPollingPath = args(0) // Set the data polling local folder path
    }

    // Local Path : file:///home/acadgild/workspace/spark/data
    println("Local Polling folder path : " + localPollingPath)
    val lines = ssc.textFileStream(localPollingPath)

    // Process data from local folder
    val words = lines.flatMap(line => line.split(" "))
    val pairs = words.map(word => (word, 1))
    val wordCount = pairs.reduceByKey(_ + _)
    wordCount.print()

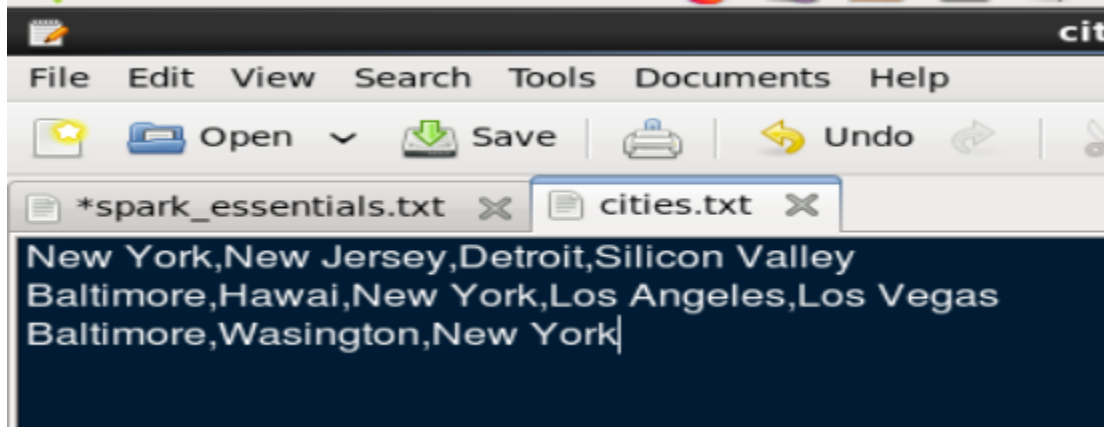
    ssc.start()
    ssc.awaitTermination()
  }
}
```

Execution:

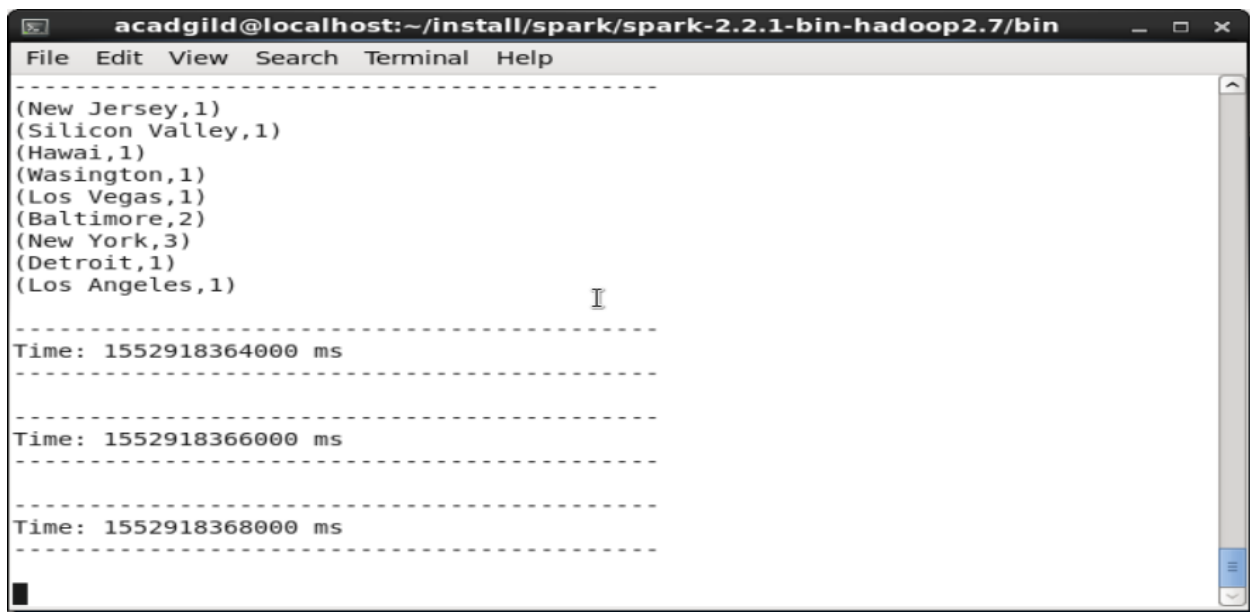


```
[acadgild@localhost bin]$ spark-submit --class com.company.stream.Assignment24_CaseStudy_Part1 --master local[4] /home/acadgild/workspace/spark/SparkStreamLocalWordCounter.jar file:///home/acadgild/data
```

At runtime, new file cities.txt is created with below data,



Spark streaming application that polls local folder path "/home/acadgild/data" picked up the file and started processing. The word count results are as below,

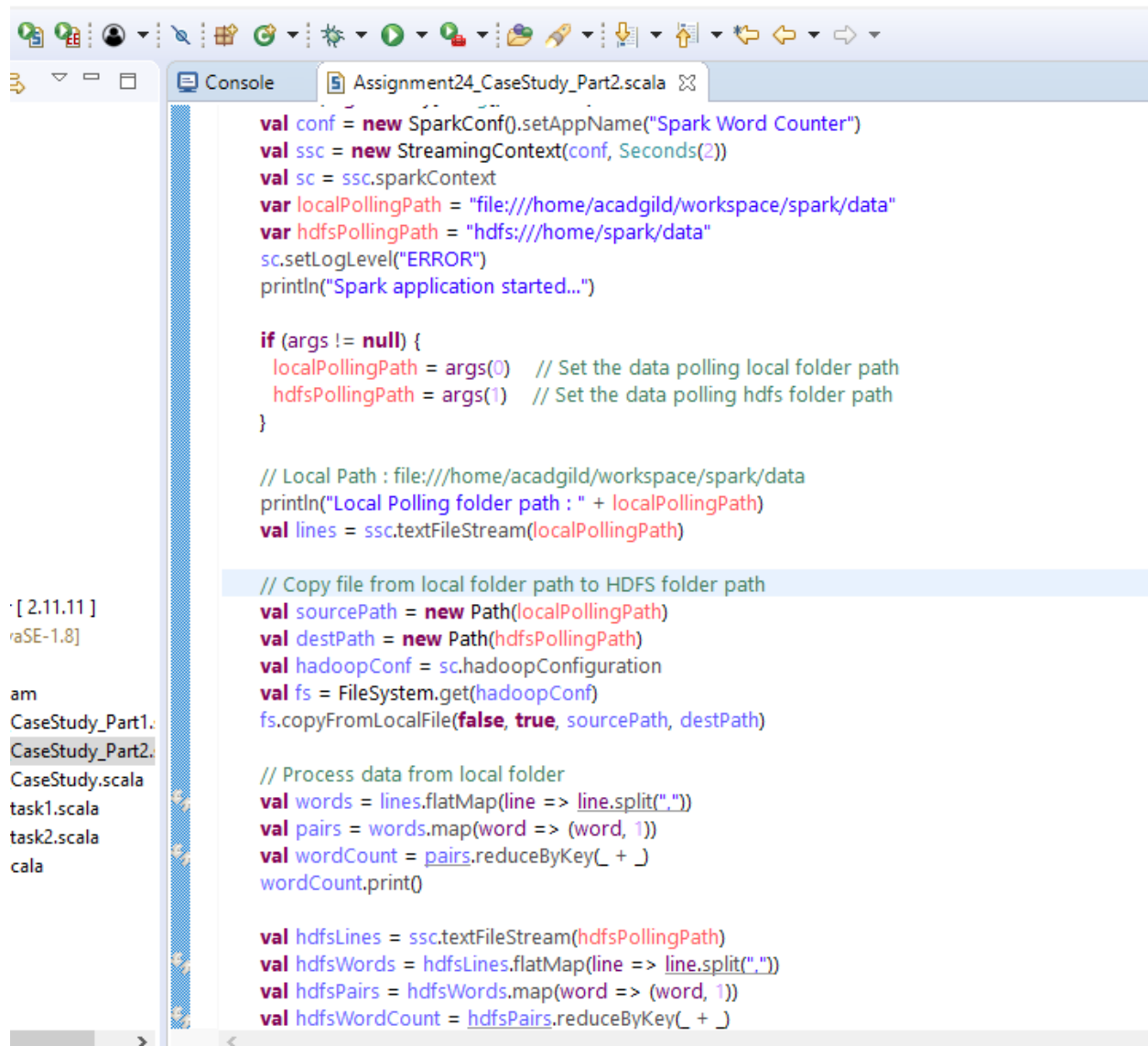


## Case Study – Part 2:

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

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

Program:



```
val conf = new SparkConf().setAppName("Spark Word Counter")
val ssc = new StreamingContext(conf, Seconds(2))
val sc = ssc.sparkContext
val localPollingPath = "file:///home/acadgild/workspace/spark/data"
val hdfsPollingPath = "hdfs:///home/spark/data"
sc.setLogLevel("ERROR")
println("Spark application started...")

if (args != null) {
  localPollingPath = args(0) // Set the data polling local folder path
  hdfsPollingPath = args(1) // Set the data polling hdfs folder path
}

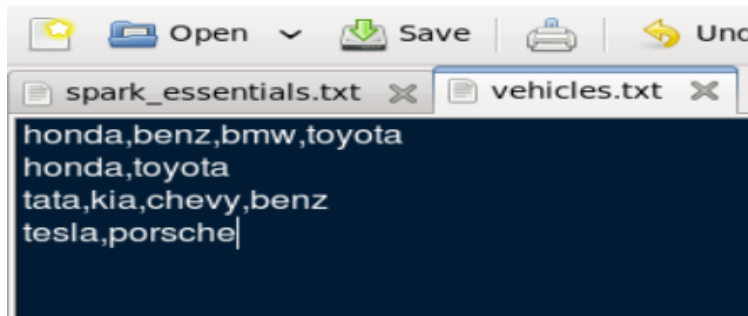
// Local Path : file:///home/acadgild/workspace/spark/data
println("Local Polling folder path : " + localPollingPath)
val lines = ssc.textFileStream(localPollingPath)

// Copy file from local folder path to HDFS folder path
val sourcePath = new Path(localPollingPath)
val destPath = new Path(hdfsPollingPath)
val hadoopConf = sc.hadoopConfiguration
val fs = FileSystem.get(hadoopConf)
fs.copyFromLocalFile(false, true, sourcePath, destPath)

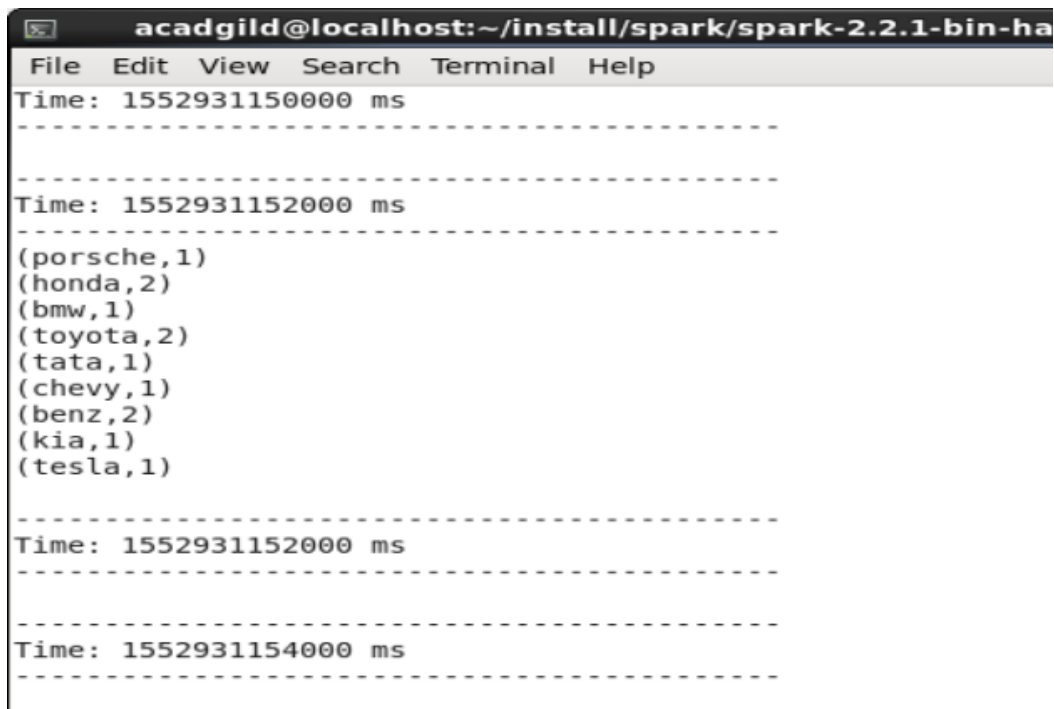
// Process data from local folder
val words = lines.flatMap(line => line.split(" "))
val pairs = words.map(word => (word, 1))
val wordCount = pairs.reduceByKey(_ + _)
wordCount.print()

val hdfsLines = ssc.textFileStream(hdfsPollingPath)
val hdfsWords = hdfsLines.flatMap(line => line.split(" "))
val hdfsPairs = hdfsWords.map(word => (word, 1))
val hdfsWordCount = hdfsPairs.reduceByKey(_ + _)
```

Execution:



A screenshot of a text editor window with two tabs: 'spark\_essentials.txt' and 'vehicles.txt'. The 'vehicles.txt' tab is active, showing a list of car brands and their counts: 'honda,benz,bmw,toyota', 'honda,toyota', 'tata,kia,chevy,benz', and 'tesla,porsche'.



A screenshot of a terminal window titled 'acadgild@localhost: ~/install/spark/spark-2.2.1-bin-ha'. The terminal shows the output of a Spark Streaming application. It displays the time in milliseconds and the contents of the 'vehicles.txt' file being processed. The output is as follows:

```
Time: 1552931150000 ms
-----
Time: 1552931152000 ms
-----
(porsche,1)
(honda,2)
(bmw,1)
(toyota,2)
(tata,1)
(chevy,1)
(benz,2)
(kia,1)
(tesla,1)
-----
Time: 1552931152000 ms
-----
Time: 1552931154000 ms
-----
```

**Note:** Though we have code to copy file from local folder path to HDFS folder path, during run time, it is not copying the file to HDFS path. I tried by specifically polling for a file instead of a folder (ex: /home/acadgild/spark/data/vehicles.txt) for the program to pick up vehicles.txt file whenever I update the file. But it didn't work. As per the hortonworks forum, it was given that Spark Streaming will not work picking the recently updated file and it will work only for newly created file. So, I am not sure how to copy file from local folder to HDFS path through spark streaming. It will be great if you could share the workable code snippet of this case study for my reference.