Spark_Streaming_Case_Study

First Part:

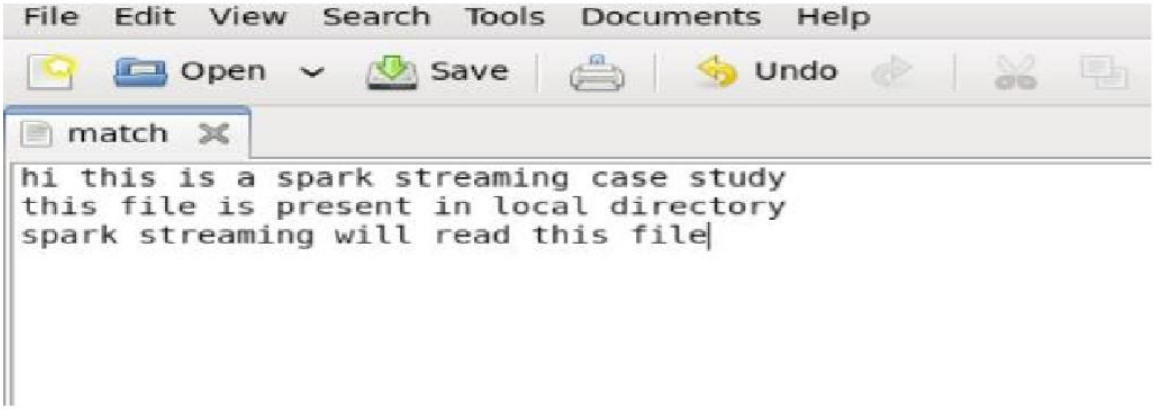
You have to create a Spark Application which streams data from a file on local directory on your machine and does the word count on the fly. The word should be done by the spark application in such a way that as soon as you drop the file in your local directory, your spark application should immediately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

Below is code for the first case.

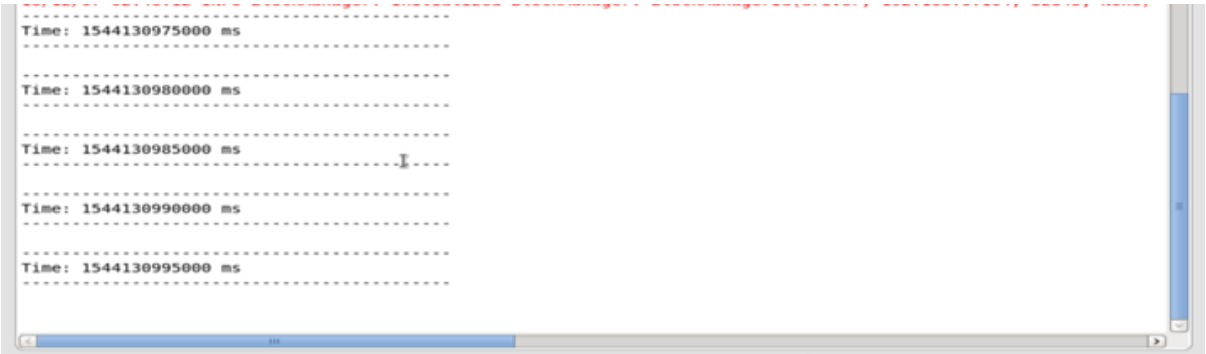
```
1  package com.casestudy
2
3  import org.apache.spark.{SparkConf, SparkContext}
6
7  object SparkFileStreamingWordCount
8   {
9
10     def main(args: Array[String]): Unit = {
11     println("hey Spark Streaming")
12   val conf = new SparkConf().setMaster("local[2]").setAppName("SparkSteamingExamp
13     val sc = new SparkContext(conf)
14     val rootLogger =Logger.getRootLogger()
15     rootLogger.setLevel(Level.ERROR)
16
17    // Create Streaming context  to set batch duration 10 seconds
18    val ssc = new StreamingContext(sc, Seconds(10))
19
20 //Create RDD for text file streaming by
21   //val lines = ssc.textFileStream("file:///home/acadgild/Documents/Match_data")
22   val lines = ssc.textFileStream("/home/acadgild/Spark_Streaming")
```

```
21   //val lines = ssc.textFileStream("file:///home/acadgild/Documents/Match_data")
22   val lines = ssc.textFileStream("/home/acadgild/Spark_Streaming")
23   println("file input data is: ")
24   lines.print()
25 //Split each line into words
26 val words = lines.flatMap(_.split(" "))
27 //Count each word in each batch
28 val wordCounts = words.map(x => (x, 1)).reduceByKey(_ + _)
29 wordCounts.print()
30 //Start the computation
31 ssc.start()
32 //wait for the computation to terminate
33 ssc.awaitTermination()
34
35      }
36
37 }
```

File in spark stream directory



In below screenshot we are able to see that spark streaming is running every 5 seconds



Output of console.

```
-------------------------------------------
(this,3)
(is,2)
(will,1)
(read,1)
(streaming,2)
(case,1)
(file,2)
(spark,2)
(directory,1)
(a,1)
```

Second Part:

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

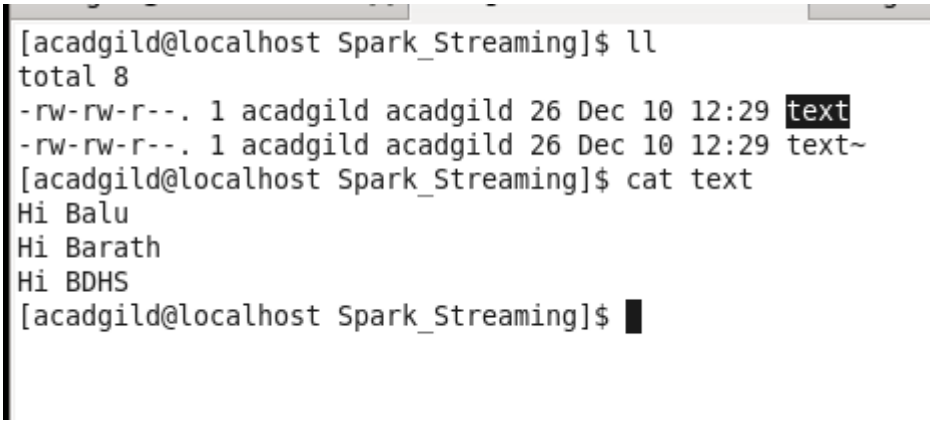
1. Pick up a file from the local directory and do the word count

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

Text file is on local machine under /home/acadgild/Desktop/Spark_Streaming

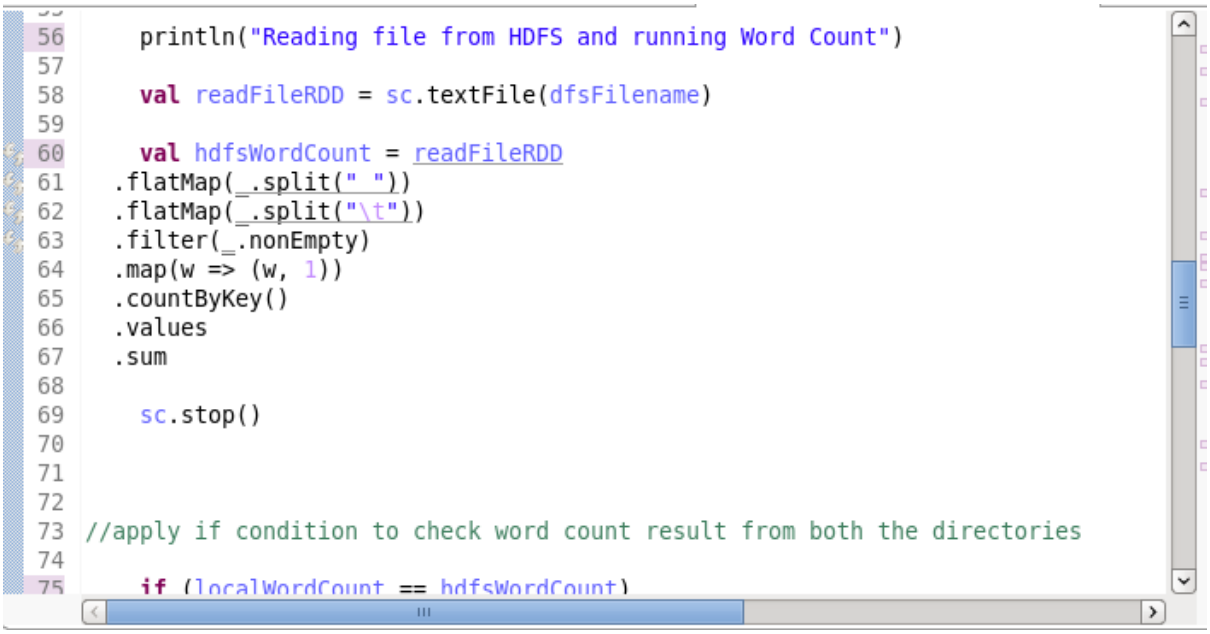


Below is code for the second case

```
package com.casestudy

import java.io.File

object SparkHDFSWordCountComparison
{

// defining the local file directory
private var localFilePath: File =
  new File("/home/acadgild/Desktop/Spark_Streaming/text")

//defining the directory in hdfs path
private var dfsDirPath: String = "hdfs://localhost:8020/user"
private val NPARAMS = 2

  def main(args: Array[String]): Unit = {
  //parseArgs(args)
    println("SparkHDFSWordCountComparison : Main Called Successfully")

    println("Performing local word count")
```

```
23      println("Performing local word count")
24
25  //read the file which is present in local directory  and convert into string
26  val fileContents = readFile(localFilePath.toString())
27
28      println("Performing local word count - File Content ->>"+fileContents)
29
30      val localWordCount = runLocalWordCount(fileContents)
31
32      println("SparkHDFSWordCountComparison : Main Called Successfully -> Local W
33
34      println("Performing local word count Completed !!")
35
36      println("Creating Spark Context")
37
38  //Create spark context
39    val conf = new SparkConf().setMaster("local[2]").setAppName("SparkHDFSWordCou
40    val sc = new SparkContext(conf)
41
42  // Setting log level to [WARN] for streaming executions  and to override add a
```

```
//create spark context
  val conf = new SparkConf().setMaster("local[2]").setAppName("SparkHDFSWordCou
  val sc = new SparkContext(conf)

// Setting log level to [WARN] for streaming executions  and to override add a
  val rootLogger =Logger.getRootLogger()
  rootLogger.setLevel(Level.ERROR)

        println("Spark Context Created")

    println("Writing local file to DFS")
  val dfsFilename = dfsDirPath + "/hdfs_read_write_test"

    val fileRDD = sc.parallelize(fileContents)
  fileRDD.saveAsTextFile(dfsFilename)

    println("Writing local file to HDFS Completed")

    println("Reading file from HDFS and running Word Count")

    val readFileRDD = sc.textFile(dfsFilename)
```



```
    println("Reading file from HDFS and running Word Count")

    val readFileRDD = sc.textFile(dfsFilename)

    val hdfsWordCount = readFileRDD
  .flatMap(_.split(" "))
  .flatMap(_.split("\t"))
  .filter(_.nonEmpty)
  .map(w => (w, 1))
  .countByKey()
  .values
  .sum

    sc.stop()



//apply if condition to check word count result from both the directories

    if (localWordCount == hdfsWordCount)
```

```
73 //apply if condition to check word count result from both the directories
74
75     if (localWordCount == hdfsWordCount)
76   {
77   println(s"Success! Local Word Count ($localWordCount) "
78   +s"and HDFS Word Count ($hdfsWordCount) agree.")
79   }
80   else
81   {
82   println(s"Failure! Local Word Count ($localWordCount) " +
83   s"and HDFS Word Count ($hdfsWordCount) disagree.")
84   }
85
86   }
87
88
89   private def printUsage(): Unit = {
90   val usage: String = "HDFS Read-Write Test\n" +
91   "\n" +
92   "Usage: localFile dfsDir\n" +
```

```
89   private def printUsage(): Unit = {
90   val usage: String = "HDFS Read-Write Test\n" +
91   "\n" +
92   "Usage: localFile dfsDir\n" +
93   "\n" +
94   "localFile - (string) local file to use in test\n" +
95   "dfsDir - (string) HDFS directory for read/write tests\n"
96
97     println(usage)
98   }
99
100  private def readFile(filename: String): List[String] = {
101
102    val lineIter: Iterator[String] = fromFile(filename).getLines()
103  val lineList: List[String] = lineIter.toList
104  lineList
105  }
106
107  def runLocalWordCount(fileContents: List[String]): Int = {
```

```
  def runLocalWordCount(fileContents: List[String]): Int = {
  fileContents.flatMap(_.split(" "))
  .flatMap(_.split("\t"))
  .filter(_.nonEmpty)
  .groupBy(w => w)
  .mapValues(_.size)
  .values
  .sum
  }
}
```

Output of Console:

```
<terminated> SparkHDFSWordCountComparison$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (De
SparkHDFSWordCountComparison : Main Called Successfully
Performing local word count
Performing local word count - File Content ->>List(Hi Balu, Hi Barath, Hi BDHS)
SparkHDFSWordCountComparison : Main Called Successfully -> Local Word Count is ->>6
Performing local word count Completed !!
Creating Spark Context
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
18/12/10 12:29:59 INFO SparkContext: Running Spark version 2.2.1
18/12/10 12:30:02 WARN NativeCodeLoader: Unable to load native-hadoop library for your
18/12/10 12:30:03 WARN Utils: Your hostname, localhost.localdomain resolves to a loopba
18/12/10 12:30:03 WARN Utils: Set SPARK LOCAL IP if you need to bind to another address
```

```
Spark Context Created
Writing local file to DFS
Writing local file to HDFS Completed
Reading file from HDFS and running Word Count
Success! Local Word Count (6) and HDFS Word Count (6) agree.
```

Below is the screen shot where the data is saved in HDFS.

The content of text file under local is saved in hdfs user folder.

```
[acadgild@localhost ~]$ hadoop fs -ls /user/hdfs_read_write_test
18/12/10 12:40:47 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platf
orm... using builtin-java classes where applicable
Found 3 items
-rw-r--r--   3 acadgild supergroup          0 2018-12-10 12:30 /user/hdfs_read_write_test/_SUCCES
S
-rw-r--r--   3 acadgild supergroup          8 2018-12-10 12:30 /user/hdfs_read_write_test/part-00
000
-rw-r--r--   3 acadgild supergroup         18 2018-12-10 12:30 /user/hdfs_read_write_test/part-00
001
[acadgild@localhost ~]$ hadoop fs -cat /user/hdfs_read_write_test/part-00000
18/12/10 12:41:06 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platf
orm... using builtin-java classes where applicable
Hi Balu
[acadgild@localhost ~]$ hadoop fs -cat /user/hdfs_read_write_test/part-00001
18/12/10 12:41:12 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platf
orm... using builtin-java classes where applicable
Hi Barath
Hi BDHS
[acadgild@localhost ~]$
```