Case Study 4
Spark Streaming

There are two parts in 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.

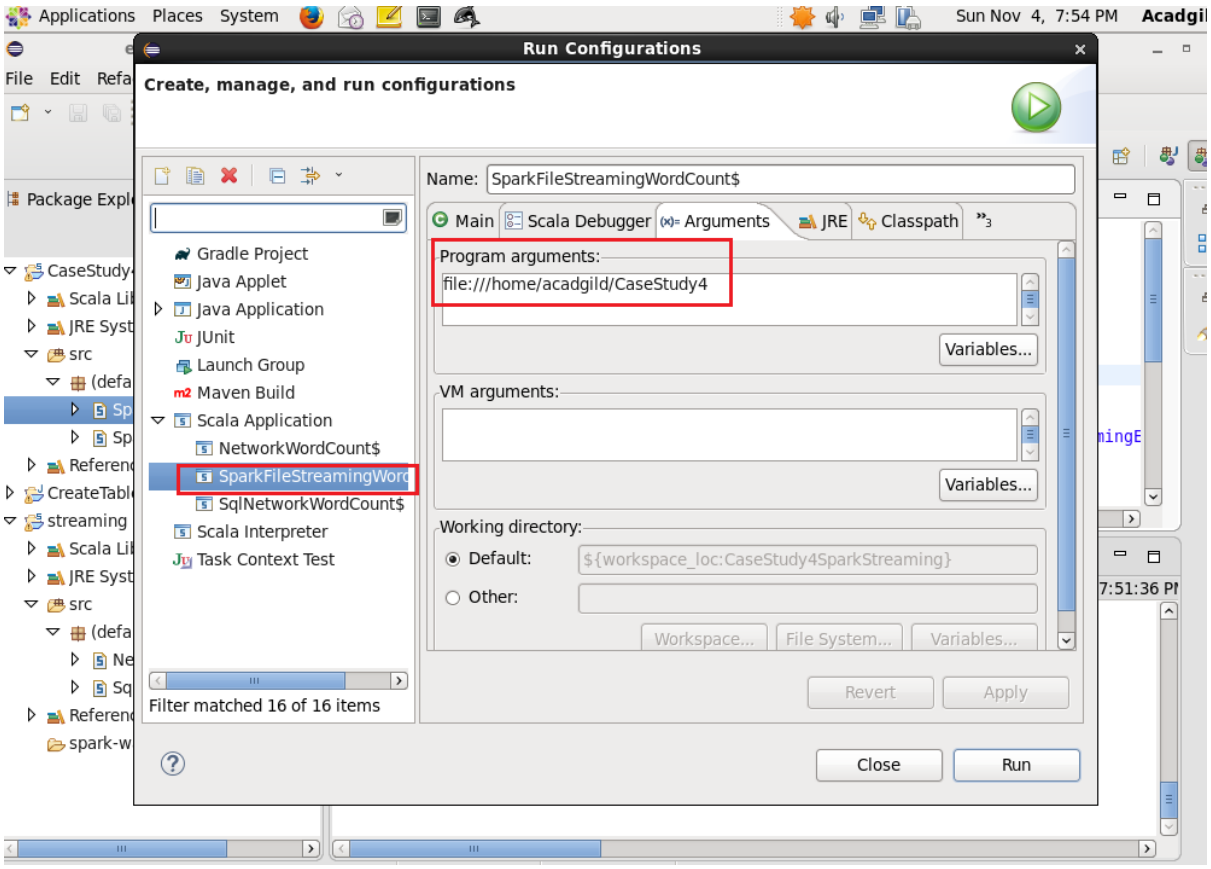
We have created local directory **CaseStudy4** for streaming under /home/acadgild by using below command:

mkdir CaseStudy4

```
[acadgild@localhost ~]$ mkdir CaseStudy4
[acadgild@localhost ~]$ cd CaseStudy4
[acadgild@localhost CaseStudy4]$ ls -l
total 0
```

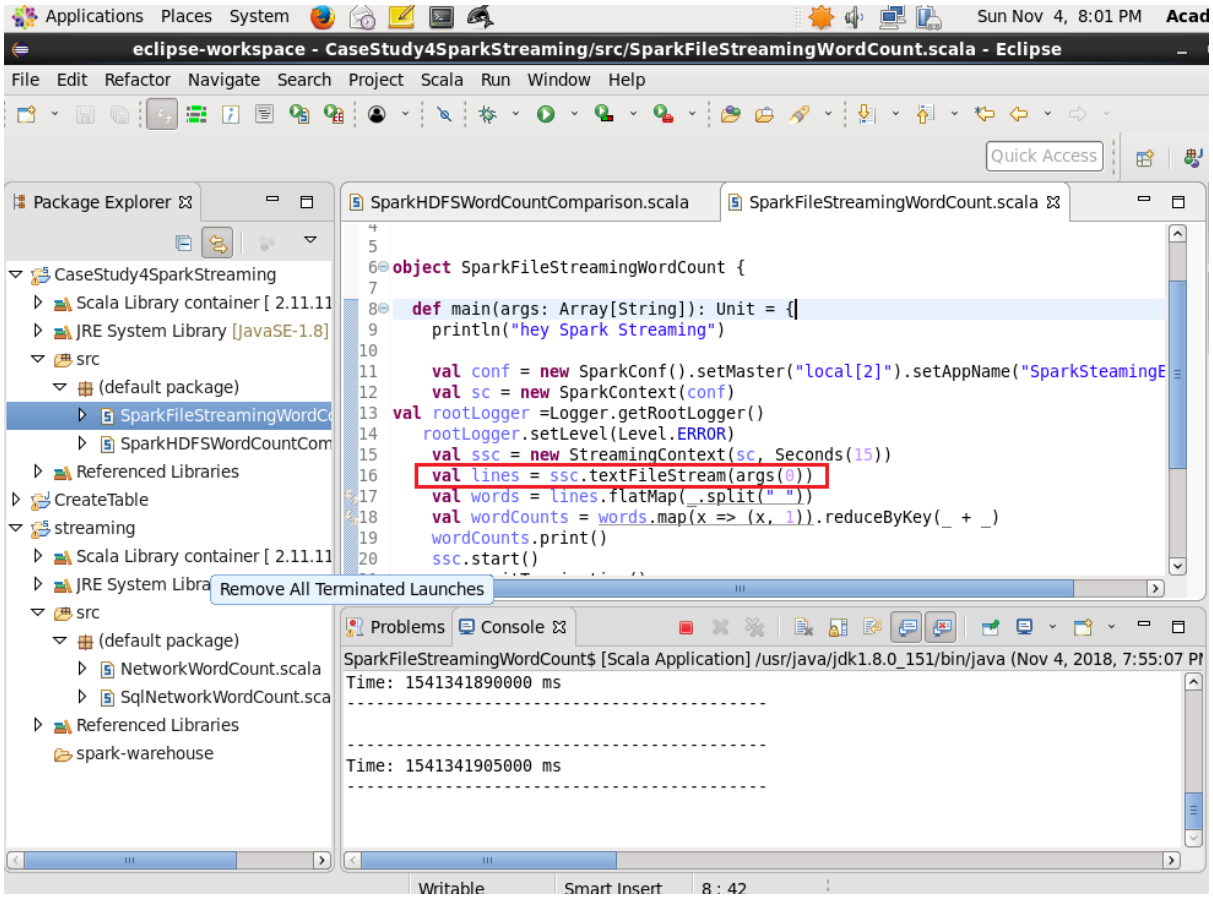
There is no file present in CaseStudy4 folder.

We have specified above directory as input to Program arguments for code.



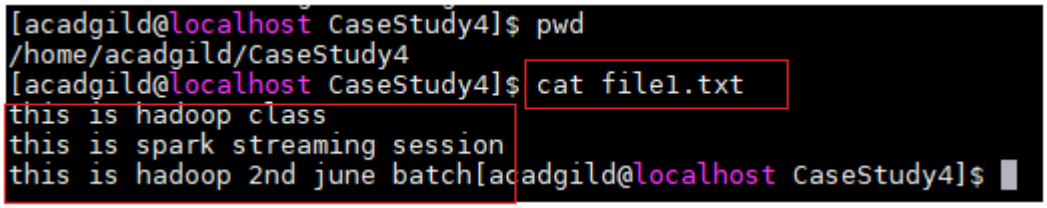
Case Study 4
Spark Streaming

Argument **args(0)** specified within code reads file that is added in local directory **CaseStudy4** and performs wordcount operation on the fly. Please refer below code:



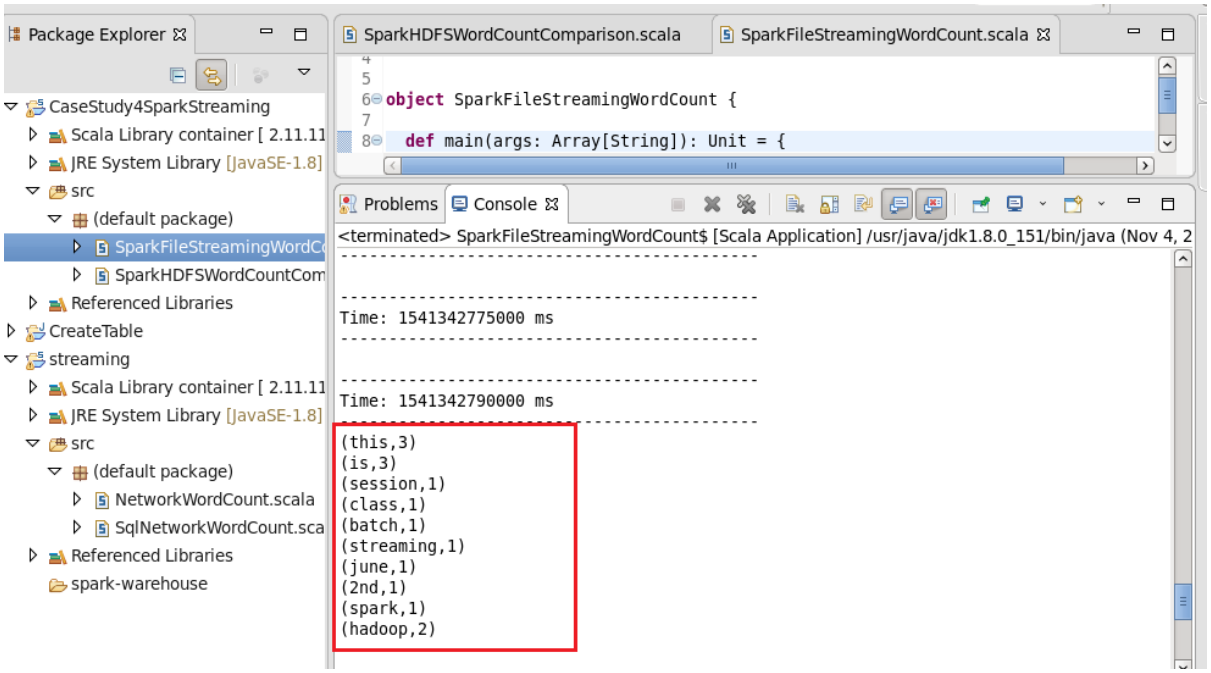
We are running this code as Scala Application.
Then as streaming has been started, we are creating file within input directory **CaseStudy4.**

We have added some content in **file1.txt** inside **CaseStudy4** folder.

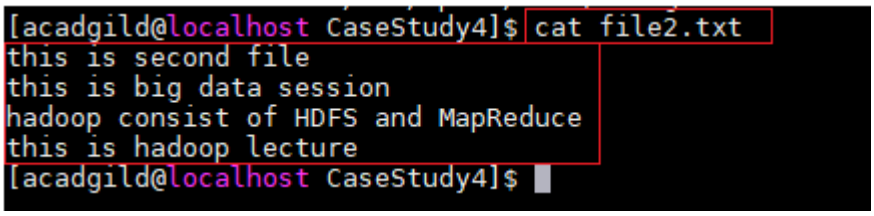


The content of file1.txt is being read by Spark Streaming application and computes word count on the fly as shown below:

Case Study 4
Spark Streaming

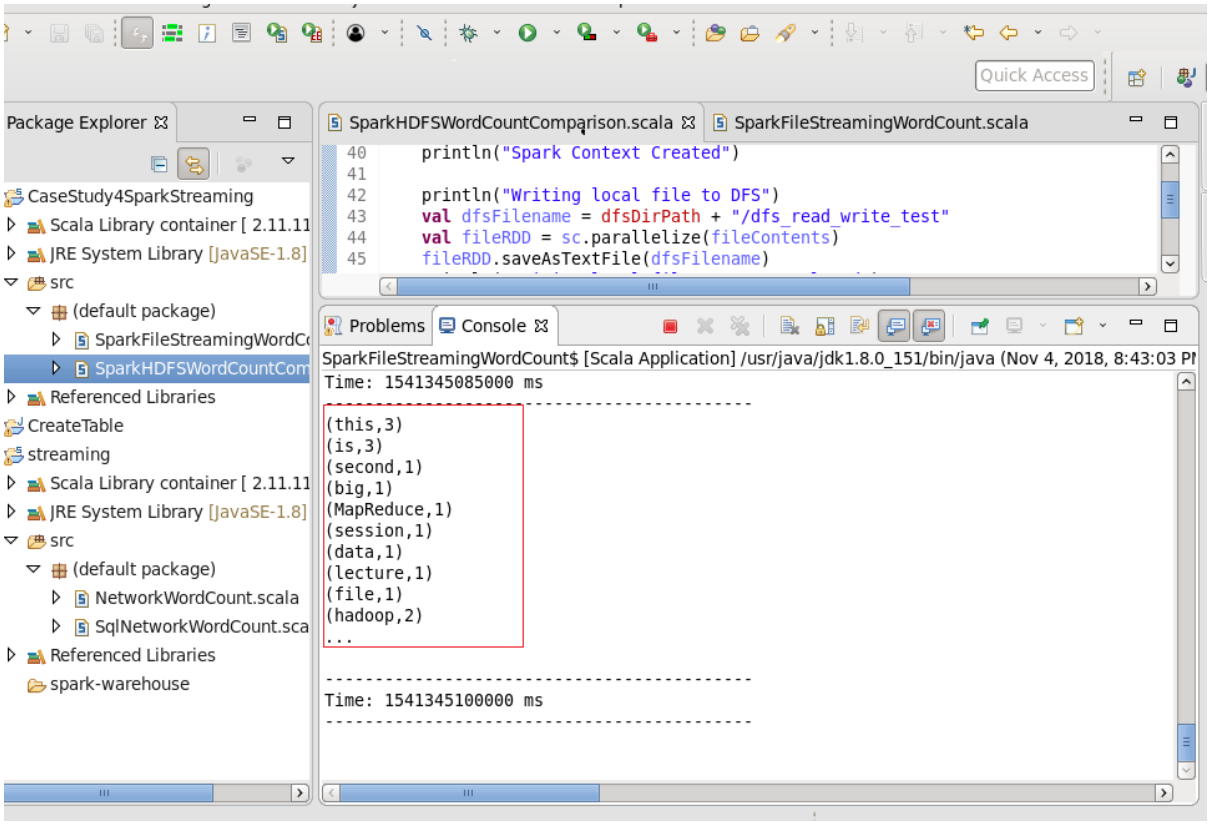


Similarly we have added some content in **file2.txt** inside **CaseStudy4** folder.

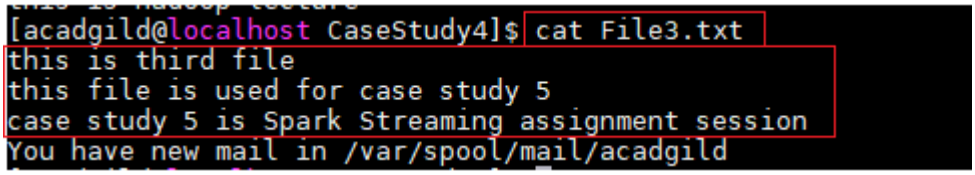


The content of file2.txt is being read by Spark Streaming application and computes word count on the fly as shown below:

Case Study 4
Spark Streaming

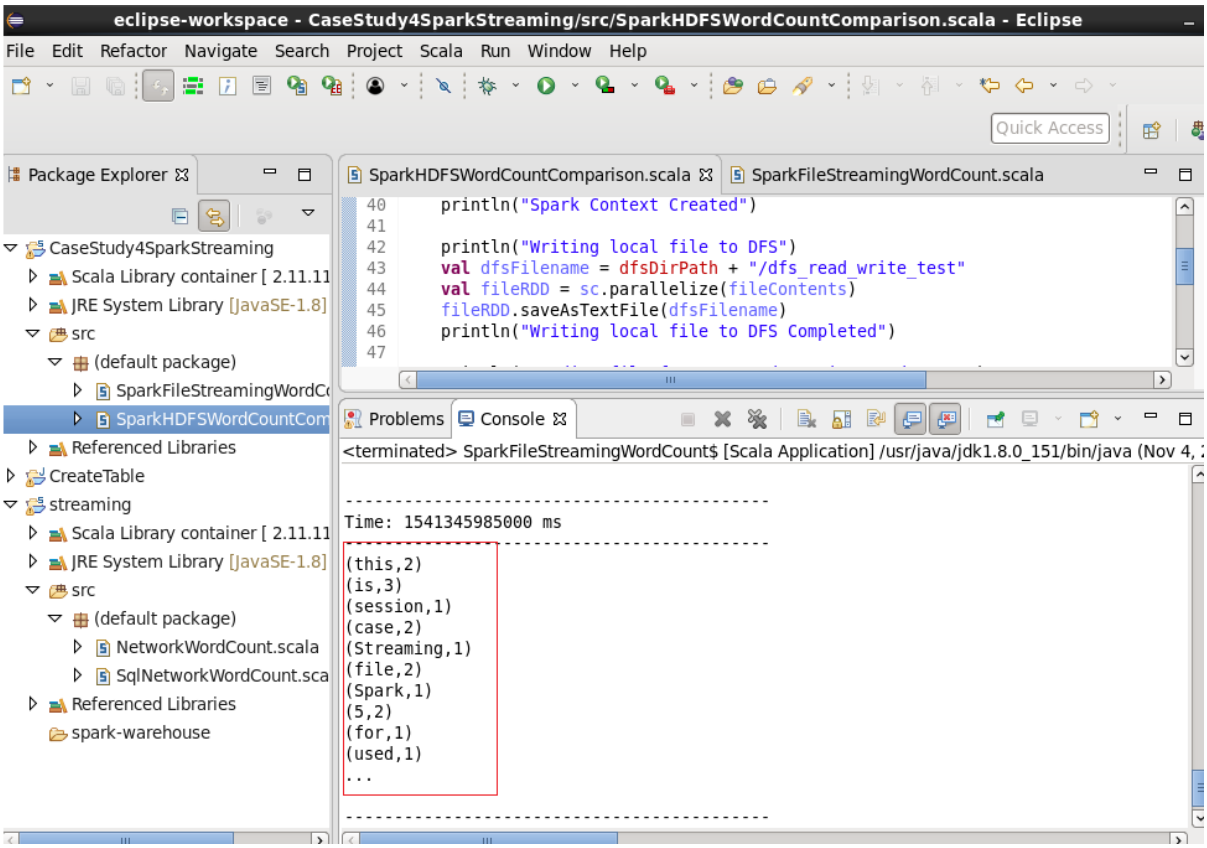


Then we have added some content in **File3.txt** inside **CaseStudy4** folder.



The content of File3.txt is being read by Spark Streaming application and computes word count on the fly as shown below:

Case Study 4
Spark Streaming



Scala code:
SparkFileStreamingWordCount.scala

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

object SparkFileStreamingWordCount {

  def main(args: Array[String]): Unit = {
    println("hey Spark Streaming")

    val conf = new
SparkConf().setMaster("local[2]").setAppName("SparkSteamingExample")
    val sc = new SparkContext(conf)
    val rootLogger =Logger.getRootLogger()
    rootLogger.setLevel(Level.ERROR)
    val ssc = new StreamingContext(sc, Seconds(15))
    val lines = ssc.textFileStream(args(0))
    val words = lines.flatMap(_.split(" "))
    val wordCounts = words.map(x => (x, 1)).reduceByKey(_ + _)
    wordCounts.print()
    ssc.start()
    ssc.awaitTermination()
  }
}
```

Case Study 4
Spark Streaming

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.

HDFS does not contain streaming directory before running the application.

We have given local file path as **/home/acadgild/CaseStudy4/part2/test.txt** and we have added test.txt to perform word count operation for local directory.

Below is content in **test.txt** file present in local directory :

```
[acadgild@localhost part2]$ cat /home/acadgild/CaseStudy4/part2/test.txt
this is hadoop class
this is comparison file
```

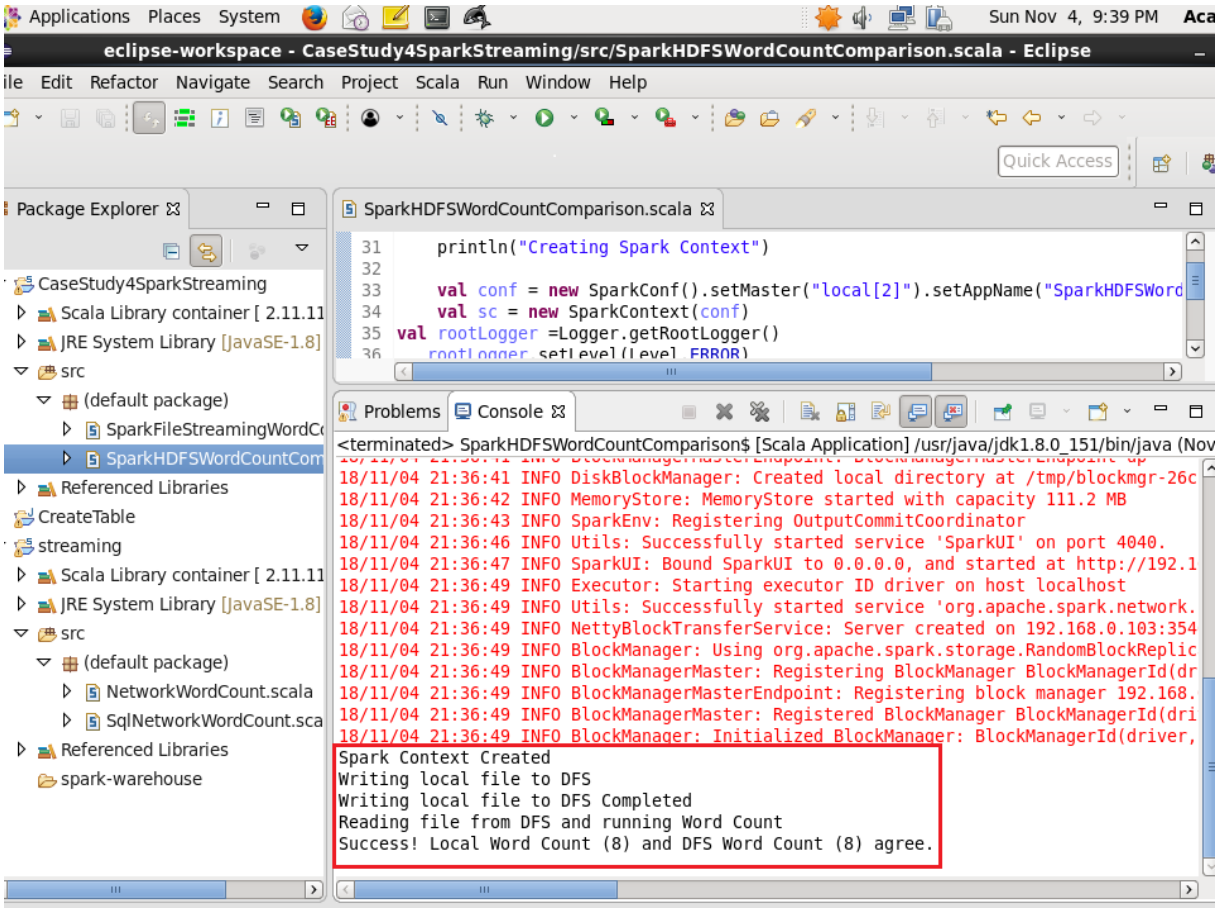
Then in the same Spark Application, do the word count for file created in HDFS.

Here we have created output folder 'dfs_read_write_test' in **streaming**.
Then you could see that content which is present in test.txt in local is also present in HDFS files **part-00000** and **part-00001.**

```
[acadgild@localhost ~]$ hadoop fs -ls /user
18/11/04 21:47:52 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where
applicable
Found 3 items
drwxr-xr-x   - acadgild supergroup          0 2018-08-22 09:34 /user/flume
drwxr-xr-x   - acadgild supergroup          0 2018-02-09 14:50 /user/hive
drwxr-xr-x   - acadgild supergroup          0 2018-11-04 21:37 /user/streaming
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ hadoop fs -ls /user/streaming
18/11/04 21:49:20 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where
applicable
Found 1 items
drwxr-xr-x   - acadgild supergroup          0 2018-11-04 21:37 /user/streaming/dfs_read_write_test
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ hadoop fs -ls /user/streaming/dfs_read_write_test
18/11/04 21:50:11 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where
applicable
Found 3 items
-rw-r--r--   3 acadgild supergroup          0 2018-11-04 21:37 /user/streaming/dfs_read_write_test/_SUCCESS
-rw-r--r--   3 acadgild supergroup         21 2018-11-04 21:37 /user/streaming/dfs_read_write_test/part-00000
-rw-r--r--   3 acadgild supergroup         24 2018-11-04 21:37 /user/streaming/dfs_read_write_test/part-00001
[acadgild@localhost ~]$ hadoop fs -cat /user/streaming/dfs_read_write_test/part-00000
18/11/04 21:50:54 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where
applicable
this is hadoop class
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ hadoop fs -cat /user/streaming/dfs_read_write_test/part-00001
18/11/04 21:52:37 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where
applicable
this is comparison file
You have new mail in /var/spool/mail/acadgild
```

Then compare the word count in local directory and HDFS. Both counts are matching.

Case Study 4
Spark Streaming



Here you could see that both in local directory and HDFS, word count is showing as 8 which means both are matching.

Scala code :

SparkHDFSWordCountComparison.scala

```
import java.io.File

import org.apache.spark.{SparkConf, SparkContext}

import scala.io.Source._
import org.apache.log4j.{Level,Logger}


object SparkHDFSWordCountComparison {

  private var localFilePath: File = new
File("/home/acadgild/Documents/s26/usecase/inputs/test.txt")
  private var dfsDirPath: String = "hdfs://localhost:8020/user/streaming"
  private val NPARAMS = 2


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

```
    println("Performing local word count")
    val fileContents = readFile(localFilePath.toString())

    println("Performing local word count - File Content ->>"+fileContents)
    val localWordCount = runLocalWordCount(fileContents)

    println("SparkHDFSWordCountComparison : Main Called Successfully -> Local Word
Count is ->>"+localWordCount)

    println("Performing local word count Completed !!")

    println("Creating Spark Context")

    val conf = new
SparkConf().setMaster("local[2]").setAppName("SparkHDFSWordCountComparisonApp")
    val sc = new SparkContext(conf)
    val rootLogger =Logger.getRootLogger()
    rootLogger.setLevel(Level.ERROR)


    println("Spark Context Created")

    println("Writing local file to DFS")
    val dfsFilename = dfsDirPath + "/dfs_read_write_test"
    val fileRDD = sc.parallelize(fileContents)

    fileRDD.saveAsTextFile(dfsFilename)
    println("Writing local file to DFS Completed")

    println("Reading file from DFS and running Word Count")
    val readFileRDD = sc.textFile(dfsFilename)

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

    sc.stop()

    if (localWordCount == dfsWordCount) {
      println(s"Success! Local Word Count ($localWordCount) " +
        s"and DFS Word Count ($dfsWordCount) agree.")
    } else {
      println(s"Failure! Local Word Count ($localWordCount) " +
        s"and DFS Word Count ($dfsWordCount) disagree.")
    }



  }

  /***private def parseArgs(args: Array[String]): Unit = {
    if (args.length != NPARAMS) {
      printUsage()
      System.exit(1)
    }
  }***/

  private def printUsage(): Unit = {
    val usage: String = "DFS Read-Write Test\n" +
      "\n" +
      "Usage: localFile dfsDir\n" +
      "\n" +
      "localFile - (string) local file to use in test\n" +
```

Case Study 4
Spark Streaming

```
      "dfsDir - (string) DFS directory for read/write tests\n"

    println(usage)
  }

  private def readFile(filename: String): List[String] = {
    val lineIter: Iterator[String] = fromFile(filename).getLines()
    val lineList: List[String] = lineIter.toList
    lineList
  }




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


}
```