



BIG DATA HADOOP & SPARK TRAINING

CASE STUDY V: Case study on
spark streaming

Rashmi Krishna

Contents

Input file for all the below tasks:.....	1
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 count 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.....	2
Second Part - In this part, you will have to create a Spark Application which should do the following:	6
1. Pick up a file from the local directory and do the word count.....	6
2. Then in the same Spark Application, write the code to put the same file on HDFS.....	6
3. Then in same Spark Application, do the word count of the file copied on HDFS in step 2	6
4. Lastly, compare the word count of step 1 and 2. Both should match, other throw an error	6

Input file for all the below tasks:



A terminal window titled 'acadgild@localhost:~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal output shows a notification 'You have new mail in /var/spool/mail/acadgild' followed by the command '[acadgild@localhost ~]\$ cat test'. The output of the command is a list of 'Hello, This is BDH session' repeated multiple times, with some lines being truncated. A blue arrow points to the output, with the text 'Input file contents' below it.

```
acadgild@localhost:~  
File Edit View Search Terminal Help  
You have new mail in /var/spool/mail/acadgild  
[acadgild@localhost ~]$ cat test  
Hello, This is BDH session  
Hello, This is BDH session  
Hello, This is BDH session  
Hello, This is BDH session  
Hello, This is BDH sessionHello, This is BDH sessionHello, This is BDH sessionHe  
llo, This is BDH session  
Hello, This is BDH session  
[acadgild@localhost ~]$
```

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 count 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 to perform the above task:

Required packages and imports: -

```
package com.acadgild.spark  
  
import org.apache.spark.SparkContext  
import org.apache.spark.SparkConf  
object Wordcount {
```

```

//main function with 1 parameter i.e. output file directory
def main(args: Array[String]) {
  //Create conf object and appname as WordCount
  val conf = new SparkConf().setMaster("local[*]")
  .setAppName("WordCount")

  //create spark context and streaming context object
  val sc = new SparkContext(conf)
  val ssc = new StreamingContext(sc,Seconds(60))

  //Check whether sufficient parameters are supplied or not
  if (args.length < 1) {
    println("Usage: ScalaWordCount <output>")
    System.exit(1)
  }

  //Read file and create RDD for the input file directory where we will drop the file on the fly
  val rawData = ssc.textFileStream("/home/acadgild/wordcount")

  //convert the lines into words using flatMap operation
  val words = rawData.flatMap(line => line.split(" "))

  //count the individual words using map and reduceByKey operation
  val wordCount = words.map(word => (word, 1)).reduceByKey(_ + _)

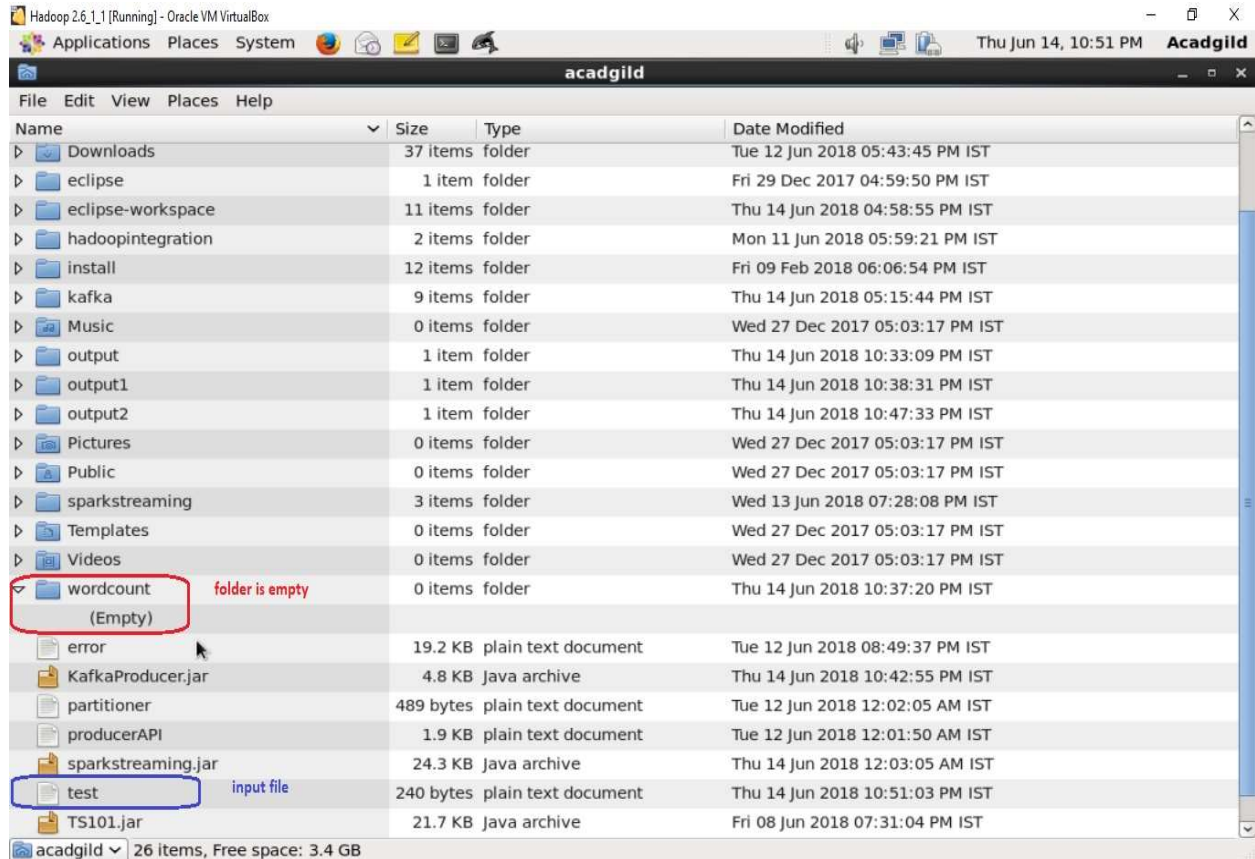
  //Save the result
  wordCount.print()

  //stop the spark context
  ssc.start
  ssc.awaitTermination()}}

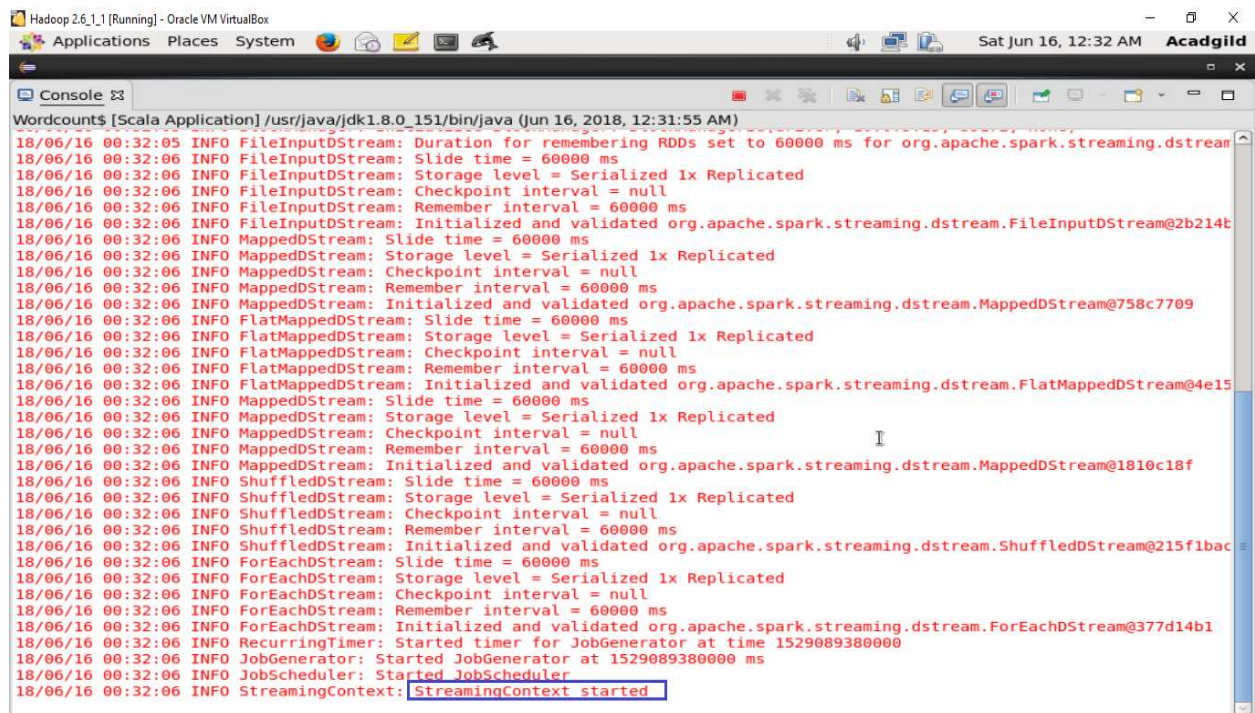
```

Below screenshot shows that the directory is empty before the program is executed in eclipse.

- Input file “test” is outside the directory, which we will be adding it to wordcount directory during the program execution



Now we execute the program:



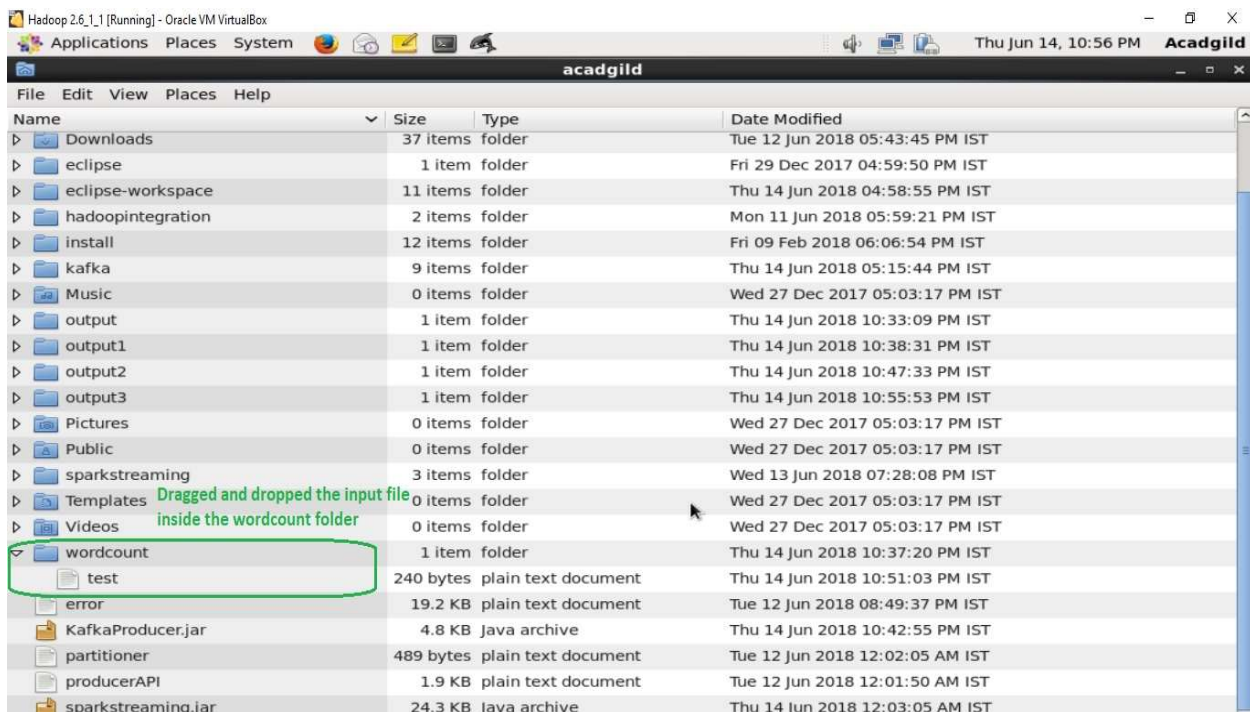

```
Wordcount$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Jun 16, 2018, 12:31:55 AM)
18/06/16 00:33:00 INFO FileInputDStream: New files at time 1529089380000 ms:

18/06/16 00:33:00 INFO JobScheduler: Added jobs for time 1529089380000 ms
18/06/16 00:33:00 INFO JobScheduler: Starting job streaming job 1529089380000 ms.0 from job set of time 1529089380000 ms
18/06/16 00:33:01 INFO SparkContext: Starting job: print at Wordcount.scala:32
18/06/16 00:33:01 INFO DAGScheduler: Registering RDD 3 (map at Wordcount.scala:28)
18/06/16 00:33:01 INFO DAGScheduler: Got job 0 (print at Wordcount.scala:32) with 1 output partitions
18/06/16 00:33:01 INFO DAGScheduler: Final stage: ResultStage 1 (print at Wordcount.scala:32)
18/06/16 00:33:01 INFO DAGScheduler: Parents of final stage: List(ShuffleMapStage 0)
18/06/16 00:33:01 INFO DAGScheduler: Missing parents: List()
18/06/16 00:33:01 INFO DAGScheduler: Submitting ResultStage 1 (ShuffledRDD[4] at reduceByKey at Wordcount.scala:28), which ha
18/06/16 00:33:02 INFO MemoryStore: Block broadcast 0 stored as values in memory (estimated size 2.8 KB, free 309.5 MB)
18/06/16 00:33:02 INFO MemoryStore: Block broadcast_0_piece0 stored as bytes in memory (estimated size 1709.0 B, free 309.5 M
18/06/16 00:33:03 INFO BlockManagerInfo: Added broadcast 0_piece0 in memory on 10.0.3.15:35172 (size: 1709.0 B, free: 309.5 M
18/06/16 00:33:03 INFO SparkContext: Created broadcast 0 from broadcast at DAGScheduler.scala:1006
18/06/16 00:33:03 INFO DAGScheduler: Submitting 1 missing tasks from ResultStage 1 (ShuffledRDD[4] at reduceByKey at Wordcoun
18/06/16 00:33:03 INFO TaskSchedulerImpl: Adding task set 1.0 with 1 tasks
18/06/16 00:33:03 INFO TaskSetManager: Starting task 0.0 in stage 1.0 (TID 0, localhost, executor driver, partition 0, PROCES
18/06/16 00:33:03 INFO Executor: Running task 0.0 in stage 1.0 (TID 0)
18/06/16 00:33:04 INFO ShuffleBlockFetcherIterator: Getting 0 non-empty blocks out of 0 blocks
18/06/16 00:33:04 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 27 ms
18/06/16 00:33:04 INFO Executor: Finished task 0.0 in stage 1.0 (TID 0). 1134 bytes result sent to driver
18/06/16 00:33:04 INFO TaskSetManager: Finished task 0.0 in stage 1.0 (TID 0) in 1018 ms on localhost (executor driver) (1/1)
18/06/16 00:33:04 INFO TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have all completed, from pool
18/06/16 00:33:04 INFO DAGScheduler: ResultStage 1 (print at Wordcount.scala:32) finished in 1.184 s
18/06/16 00:33:04 INFO DAGScheduler: Job 0 finished: print at Wordcount.scala:32, took 3.391655 s

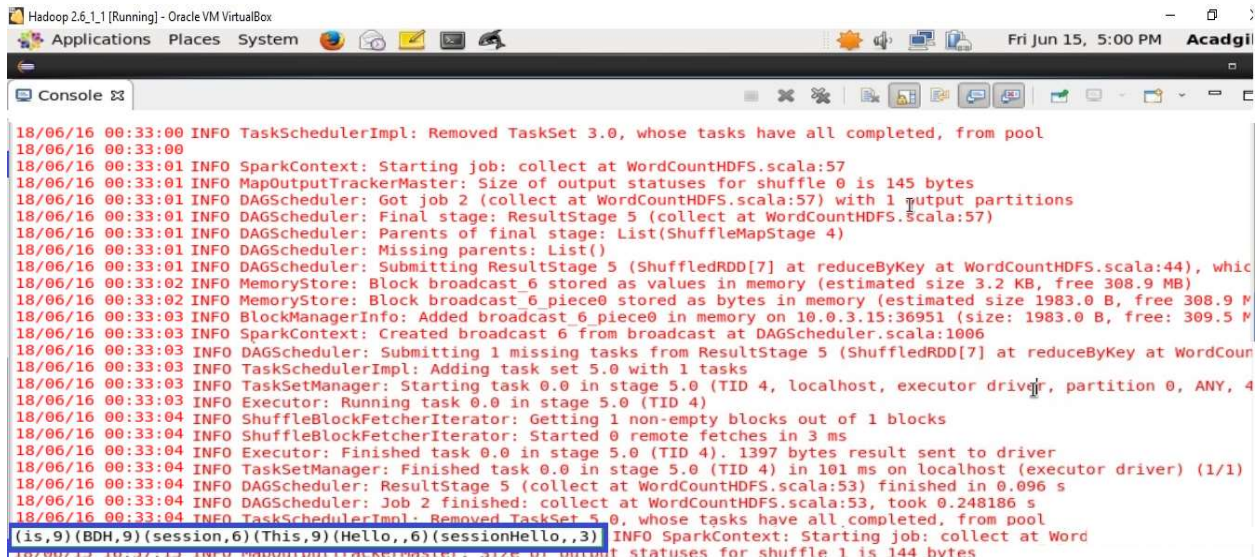
Time: 1529089380000 ms
```

After this we add the file

During the program execution, we add the file to the wordcount folder, as shown below:



As soon as we add the file in directory we can see that the word count is getting executed:



```
18/06/16 00:33:00 INFO TaskSchedulerImpl: Removed TaskSet 3.0, whose tasks have all completed, from pool
18/06/16 00:33:00 INFO SparkContext: Starting job: collect at WordCountHDFS.scala:57
18/06/16 00:33:01 INFO MapOutputTrackerMaster: Size of output statuses for shuffle 0 is 145 bytes
18/06/16 00:33:01 INFO DAGScheduler: Got job 2 (collect at WordCountHDFS.scala:57) with 1 output partitions
18/06/16 00:33:01 INFO DAGScheduler: Final stage: ResultStage 5 (collect at WordCountHDFS.scala:57)
18/06/16 00:33:01 INFO DAGScheduler: Parents of final stage: List(ShuffleMapStage 4)
18/06/16 00:33:01 INFO DAGScheduler: Missing parents: List()
18/06/16 00:33:01 INFO DAGScheduler: Submitting ResultStage 5 (ShuffledRDD[7] at reduceByKey at WordCountHDFS.scala:44), which
18/06/16 00:33:02 INFO MemoryStore: Block broadcast_6_piece0 stored as values in memory (estimated size 3.2 KB, free 308.9 MB)
18/06/16 00:33:02 INFO MemoryStore: Block broadcast_6_piece0 stored as bytes in memory (estimated size 1983.0 B, free 308.9 MB)
18/06/16 00:33:03 INFO BlockManagerInfo: Added broadcast_6_piece0 in memory on 10.0.3.15:36951 (size: 1983.0 B, free: 309.5 MB)
18/06/16 00:33:03 INFO SparkContext: Created broadcast 6 from broadcast at DAGScheduler.scala:1006
18/06/16 00:33:03 INFO DAGScheduler: Submitting 1 missing tasks from ResultStage 5 (ShuffledRDD[7] at reduceByKey at WordCount
18/06/16 00:33:03 INFO TaskSchedulerImpl: Adding task set 5.0 with 1 tasks
18/06/16 00:33:03 INFO TaskSetManager: Starting task 0.0 in stage 5.0 (TID 4, localhost, executor driver, partition 0, ANY, 4
18/06/16 00:33:03 INFO Executor: Running task 0.0 in stage 5.0 (TID 4)
18/06/16 00:33:04 INFO ShuffleBlockFetcherIterator: Getting 1 non-empty blocks out of 1 blocks
18/06/16 00:33:04 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 3 ms
18/06/16 00:33:04 INFO Executor: Finished task 0.0 in stage 5.0 (TID 4). 1397 bytes result sent to driver
18/06/16 00:33:04 INFO TaskSetManager: Finished task 0.0 in stage 5.0 (TID 4) in 101 ms on localhost (executor driver) (1/1)
18/06/16 00:33:04 INFO DAGScheduler: ResultStage 5 (collect at WordCountHDFS.scala:53) finished in 0.096 s
18/06/16 00:33:04 INFO DAGScheduler: Job 2 finished: collect at WordCountHDFS.scala:53, took 0.248186 s
18/06/16 00:33:04 INFO TaskSchedulerImpl: Removed TaskSet 5.0, whose tasks have all completed, from pool
18/06/16 00:33:04 INFO SparkContext: Starting job: collect at Word
(is,9)(BDH,9)(session,6)(This,9)(Hello,,6)(sessionHello,,3)
```

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

Program to do the above tasks:

Required packages and imports are as follows:-

```
package com.acadgild.spark

import org.apache.spark.SparkContext
import org.apache.spark.SparkConf
import org.apache.hadoop.conf.Configuration;
```

```

import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.fs.{FileAlreadyExistsException, FileSystem, FileUtil, Path}
import scala.io.Source

object WordCountHDFS {

  //main function which takes two arguments, both the arguments are file path to save the
  //output generated from word count

  def main(args: Array[String]) {

    //Create conf object

    val conf = new SparkConf().setMaster("local[*]").setAppName("WordCount")

    //create spark context object

    val sc = new SparkContext(conf)

    //create configuration configuration for Hadoop

    val hadoopConf = new Configuration()

    //Check whether sufficient parameters are supplied

    if (args.length < 2) {

      println("Usage: ScalaWordCount<output1> <output2>")

      System.exit(1)

    }

    //Read file and create RDD

    //Task1: Pick up a file from the local directory and do the word count

    val rawData = sc.textFile("/home/acadgild/wordcount/")

    // add core-site.xml and hdfs-site.xml for copying the file from local file system to HDFS

    //Task2: Then in the same Spark Application, write the code to put the same file on HDFS

    hadoopConf.addResource(new Path("/home/acadgild/install/hadoop/hadoop-
2.6.5/etc/hadoop/core-site.xml"))

    hadoopConf.addResource(new Path("/home/acadgild/install/hadoop/hadoop-
2.6.5/etc/hadoop/hdfs-site.xml"))
  }
}

```



```
//add Hadoop configuration to Filesystem, so that we can copy files from local file system  
//to HDFS
```

```
val fs = FileSystem.get(hadoopConf);  
    val sourcePath = new Path("/home/acadgild/wordcount/");  
    val destPath = new Path("hdfs://localhost:8020/");  
    if(!(fs.exists(destPath)))  
    { System.out.println("No Such destination exists :"+destPath);  
        return;    }  
  
    //lets copy file in sourcePath to destPath  
    fs.copyFromLocalFile(sourcePath, destPath);  
  
    //convert the lines into words using flatMap operation for both local files system file  
    //and HDFS file
```

```
    val words = rawData.flatMap(line => line.split(" "))
```

```
//Task3: Then in same Spark Application, do the word count of the file copied on HDFS in  
//step 2
```

```
    val hdfsfiler = sc.textFile("hdfs://localhost:8020/wordcount/test")  
    val hdfswords = hdfsfiler.flatMap(line => line.split(" "))  
  
    //count the individual words using map and reduceByKey operation for both the files  
    val wordCount = words.map(word => (word, 1)).reduceByKey(_ + _)  
    val hdfswc = hdfswords.map(word => (word,1)).reduceByKey(_ + _)  
  
    //Save the results in the path mentioned in the arguments  
    wordCount.saveAsTextFile(args(0))  
    hdfswc.saveAsTextFile(args(1))
```

```
// Task4: Lastly, compare the word count of step 1 and 2. Both should match, other throw an  
//error
```

```
// we will now convert both the files to an array and match the contents of them, to check if  
//the contents of both file match or not. If the contents match, "sameElements" function  
will return "True" if //not "false"
```

```
    val LFSWCfile = Source.fromFile("/home/acadgild/wordcount1/part-  
000000").getLines().toArray
```

```
val hdfsWCfile= Source.fromFile("/home/acadgild/wordcount2/part-00000").getLines().toArray
```

```
//now we save the Boolean value in variable "elem" and check if it is true or false, if it is  
//false it will print and error saying contents mismatch if not it will print contents match!
```

```
val elem = LFSWCfile.sameElements(hdfsWCfile)
```

```
if(elem == false){
```

```
    println("Error!: Contents mismatch")
```

```
}else
```

```
    println("Contents match!")
```

```
// we will print the output to console as well.
```

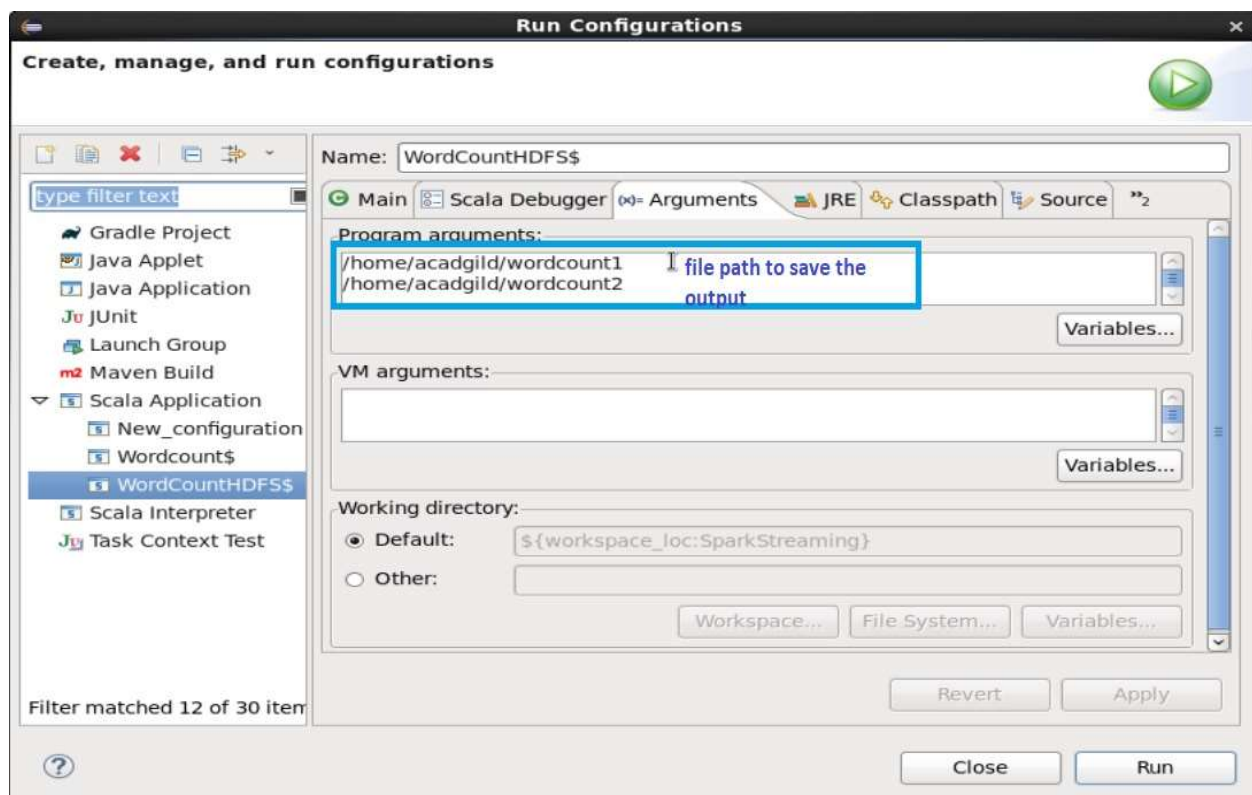
```
    wordCount.collect().foreach(print)
```

```
    hdfsWC.collect().foreach(print)
```

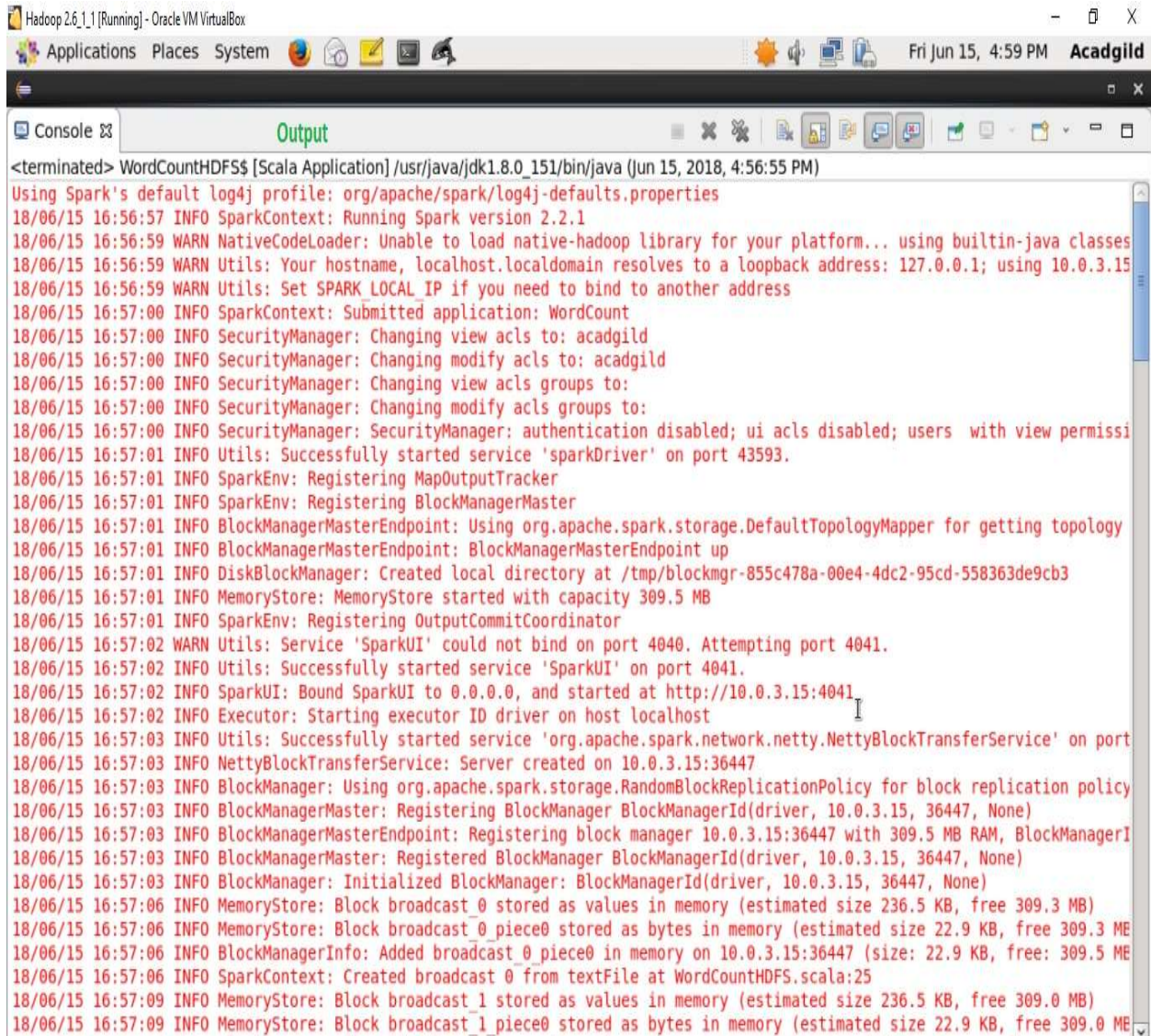
```
//stop the spark context
```

```
sc.stop }}
```

Now we will provide the run time arguments in run-configurations and execute the program as shown below:



We can see the output as below:



The screenshot shows a terminal window titled "Hadoop 2.6.1_1 [Running] - Oracle VM VirtualBox". The window has a menu bar with "Applications", "Places", and "System". The title bar includes the date and time "Fri Jun 15, 4:59 PM" and the username "Acadgild". The terminal output is as follows:

```
<terminated> WordCountHDFS$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Jun 15, 2018, 4:56:55 PM)
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
18/06/15 16:56:57 INFO SparkContext: Running Spark version 2.2.1
18/06/15 16:56:59 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes
18/06/15 16:56:59 WARN Utils: Your hostname, localhost.localdomain resolves to a loopback address: 127.0.0.1; using 10.0.3.15
18/06/15 16:56:59 WARN Utils: Set SPARK LOCAL IP if you need to bind to another address
18/06/15 16:57:00 INFO SparkContext: Submitted application: WordCount
18/06/15 16:57:00 INFO SecurityManager: Changing view acls to: acadgild
18/06/15 16:57:00 INFO SecurityManager: Changing modify acls to: acadgild
18/06/15 16:57:00 INFO SecurityManager: Changing view acls groups to:
18/06/15 16:57:00 INFO SecurityManager: Changing modify acls groups to:
18/06/15 16:57:00 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permission
18/06/15 16:57:01 INFO Utils: Successfully started service 'sparkDriver' on port 43593.
18/06/15 16:57:01 INFO SparkEnv: Registering MapOutputTracker
18/06/15 16:57:01 INFO SparkEnv: Registering BlockManagerMaster
18/06/15 16:57:01 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology
18/06/15 16:57:01 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
18/06/15 16:57:01 INFO DiskBlockManager: Created local directory at /tmp/blockmgr-855c478a-00e4-4dc2-95cd-558363de9cb3
18/06/15 16:57:01 INFO MemoryStore: MemoryStore started with capacity 309.5 MB
18/06/15 16:57:01 INFO SparkEnv: Registering OutputCommitCoordinator
18/06/15 16:57:02 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
18/06/15 16:57:02 INFO Utils: Successfully started service 'SparkUI' on port 4041.
18/06/15 16:57:02 INFO SparkUI: Bound SparkUI to 0.0.0.0, and started at http://10.0.3.15:4041
18/06/15 16:57:02 INFO Executor: Starting executor ID driver on host localhost
18/06/15 16:57:03 INFO Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' on port
18/06/15 16:57:03 INFO NettyBlockTransferService: Server created on 10.0.3.15:36447
18/06/15 16:57:03 INFO BlockManager: Using org.apache.spark.storage.RandomBlockReplicationPolicy for block replication policy
18/06/15 16:57:03 INFO BlockManagerMaster: Registering BlockManager BlockManagerId(driver, 10.0.3.15, 36447, None)
18/06/15 16:57:03 INFO BlockManagerMasterEndpoint: Registering block manager 10.0.3.15:36447 with 309.5 MB RAM, BlockManagerId
18/06/15 16:57:03 INFO BlockManagerMaster: Registered BlockManager BlockManagerId(driver, 10.0.3.15, 36447, None)
18/06/15 16:57:03 INFO BlockManager: Initialized BlockManager: BlockManagerId(driver, 10.0.3.15, 36447, None)
18/06/15 16:57:06 INFO MemoryStore: Block broadcast_0 stored as values in memory (estimated size 236.5 KB, free 309.3 MB)
18/06/15 16:57:06 INFO MemoryStore: Block broadcast_0 piece0 stored as bytes in memory (estimated size 22.9 KB, free 309.3 MB)
18/06/15 16:57:06 INFO BlockManagerInfo: Added broadcast_0_piece0 in memory on 10.0.3.15:36447 (size: 22.9 KB, free: 309.5 MB)
18/06/15 16:57:06 INFO SparkContext: Created broadcast 0 from textFile at WordCountHDFS.scala:25
18/06/15 16:57:09 INFO MemoryStore: Block broadcast_1 stored as values in memory (estimated size 236.5 KB, free 309.0 MB)
18/06/15 16:57:09 INFO MemoryStore: Block broadcast_1 piece0 stored as bytes in memory (estimated size 22.9 KB, free 309.0 MB)
```



```

Hadoop 2.6.1_1 [Running] - Oracle VM VirtualBox
Applications Places System
Fri Jun 15, 5:00 PM Acadgi

Console
<terminated> WordCountHDFS$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Jun 15, 2018, 4:56:55 PM)
18/06/15 19:23:05 INFO DAGScheduler: Job 1 finished: saveAsTextFile at WordCountHDFS.scala:48, took 0.869344 s
18/06/15 19:23:05 INFO TaskSchedulerImpl: Removed TaskSet 3.0, whose tasks have all completed, from pool
Contents match!
18/06/15 19:23:05 INFO SparkContext: Starting job: collect at WordCountHDFS.scala:57
18/06/15 19:23:05 INFO MapOutputTrackerMaster: Size of output statuses for shuffle 0 is 145 bytes
18/06/15 19:23:05 INFO DAGScheduler: Got job 2 (collect at WordCountHDFS.scala:57) with 1 output partitions
18/06/15 19:23:05 INFO DAGScheduler: Final stage: ResultStage 5 (collect at WordCountHDFS.scala:57)
18/06/15 19:23:05 INFO DAGScheduler: Parents of final stage: List(ShuffleMapStage 4)
18/06/15 19:23:05 INFO DAGScheduler: Missing parents: List()
18/06/15 19:23:05 INFO DAGScheduler: Submitting ResultStage 5 (ShuffledRDD[7] at reduceByKey at WordCountHDFS.scala:44), which
18/06/15 19:23:05 INFO MemoryStore: Block broadcast_6 stored as values in memory (estimated size 3.2 KB, free 308.9 MB)
18/06/15 19:23:05 INFO MemoryStore: Block broadcast_6_piece0 stored as bytes in memory (estimated size 1983.0 B, free 308.9 MB)
18/06/15 19:23:05 INFO BlockManagerInfo: Added broadcast_6_piece0 in memory on 10.0.3.15:36951 (size: 1983.0 B, free: 309.5 MB)
18/06/15 19:23:05 INFO SparkContext: Created broadcast 6 from broadcast at DAGScheduler.scala:1006
18/06/15 16:57:15 INFO DAGScheduler: Submitting 1 missing tasks from ResultStage 5 (ShuffledRDD[7] at reduceByKey at WordCount
18/06/15 16:57:15 INFO TaskSchedulerImpl: Adding task set 5.0 with 1 tasks
18/06/15 16:57:15 INFO TaskSetManager: Starting task 0.0 in stage 5.0 (TID 4, localhost, executor driver, partition 0, ANY, 4
18/06/15 16:57:15 INFO Executor: Running task 0.0 in stage 5.0 (TID 4)
18/06/15 16:57:15 INFO ShuffleBlockFetcherIterator: Getting 1 non-empty blocks out of 1 blocks
18/06/15 16:57:15 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 3 ms
18/06/15 16:57:15 INFO Executor: Finished task 0.0 in stage 5.0 (TID 4). 1397 bytes result sent to driver
18/06/15 16:57:15 INFO TaskSetManager: Finished task 0.0 in stage 5.0 (TID 4) in 101 ms on localhost (executor driver) (1/1)
18/06/15 16:57:15 INFO DAGScheduler: ResultStage 5 (collect at WordCountHDFS.scala:53) finished in 0.096 s
18/06/15 16:57:15 INFO DAGScheduler: Job 2 finished: collect at WordCountHDFS.scala:53, took 0.248186 s
18/06/15 16:57:15 INFO TaskSchedulerImpl: Removed TaskSet 5.0, whose tasks have all completed, from pool
(is,9)(BDH,9)(session,6)(This,9)(Hello,,6)(sessionHello,,3) 18/06/15 16:57:15 INFO SparkContext: Starting job: collect at Word
18/06/15 16:57:15 INFO MapOutputTrackerMaster: Size of output statuses for shuffle 1 is 144 bytes
18/06/15 16:57:15 INFO DAGScheduler: Got job 3 (collect at WordCountHDFS.scala:55) with 1 output partitions
18/06/15 16:57:15 INFO DAGScheduler: Final stage: ResultStage 7 (collect at WordCountHDFS.scala:55)
18/06/15 16:57:15 INFO DAGScheduler: Parents of final stage: List(ShuffleMapStage 6)
18/06/15 16:57:15 INFO DAGScheduler: Missing parents: List()
18/06/15 16:57:15 INFO DAGScheduler: Submitting ResultStage 7 (ShuffledRDD[9] at reduceByKey at WordCountHDFS.scala:45), which
18/06/15 16:57:15 INFO MemoryStore: Block broadcast_7_piece0 stored as bytes in memory (estimated size 1977.0 B, free 309.0 MB)
18/06/15 16:57:15 INFO BlockManagerInfo: Added broadcast_7_piece0 in memory on 10.0.3.15:36447 (size: 1977.0 B, free: 309.5 MB)
18/06/15 16:57:15 INFO SparkContext: Created broadcast 7 from broadcast at DAGScheduler.scala:1006
18/06/15 16:57:15 INFO DAGScheduler: Submitting 1 missing tasks from ResultStage 7 (ShuffledRDD[9] at reduceByKey at WordCount
18/06/15 16:57:15 INFO TaskSchedulerImpl: Adding task set 7.0 with 1 tasks
18/06/15 16:57:15 INFO TaskSetManager: Starting task 0.0 in stage 7.0 (TID 5, localhost, executor driver, partition 0, ANY, 4
18/06/15 16:57:15 INFO Executor: Running task 0.0 in stage 7.0 (TID 5)
18/06/15 16:57:15 INFO ShuffleBlockFetcherIterator: Getting 1 non-empty blocks out of 1 blocks
18/06/15 16:57:15 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 1 ms
18/06/15 16:57:15 INFO Executor: Finished task 0.0 in stage 7.0 (TID 5). 1354 bytes result sent to driver
18/06/15 16:57:15 INFO TaskSetManager: Finished task 0.0 in stage 7.0 (TID 5) in 137 ms on localhost (executor driver) (1/1)
18/06/15 16:57:15 INFO DAGScheduler: ResultStage 7 (collect at WordCountHDFS.scala:55) finished in 0.131 s
18/06/15 16:57:15 INFO TaskSchedulerImpl: Removed TaskSet 7.0, whose tasks have all completed, from pool
(is,9)18/06/15 16:57:15 INFO DAGScheduler: Job 3 finished: collect at WordCountHDFS.scala:55, took 0.252368 s
BDH,9)(session,6)(This,9)(Hello,,6)(sessionHello,,3) 18/06/15 16:57:15 INFO SparkUI: Stopped Spark web UI at http://10.0.3.15
18/06/15 16:57:15 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
18/06/15 16:57:16 INFO MemoryStore: MemoryStore cleared
18/06/15 16:57:16 INFO BlockManager: BlockManager stopped
18/06/15 16:57:16 INFO BlockManagerMaster: BlockManagerMaster stopped
18/06/15 16:57:16 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
18/06/15 16:57:16 INFO SparkContext: Successfully stopped SparkContext
18/06/15 16:57:16 INFO ShutdownHookManager: Shutdown hook called
18/06/15 16:57:16 INFO ShutdownHookManager: Deleting directory /tmp/spark-9827c31e-8d58-4b64-92fc-e299f4e1e0a7
```



- We can see that the sameElements function has returned true, which means that the contents of both files are matching.
- We can also compare the output shown in the console as well.

We will check the output in the terminal as well:

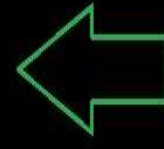
- We can see both the output files “Wordcount1” & “Wordcount2” in the local file system.
- We cat “part-00000” to view the contents of the output file.
- We can observe that both the contents are same.

```
[acadgild@localhost ~]$ ls
assignments  eclipse          hadoopintegration  Music  output3  Public  test  WC101.jar
Desktop      eclipse-workspace install           output  partitioner  sparkstreaming  test~  wordcount
Documents    error           kafka            output1 Pictures  sparkstreaming.jar TS101.jar wordcount1
Downloads    error~         KafkaProducer.jar output2  producerAPI  Templates  Videos wordcount2

You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ cd wordcount1
[acadgild@localhost wordcount1]$ ls
part-00000 SUCCESS
[acadgild@localhost wordcount1]$ cat part-00000
(is,9)
(BDH,9)
(session,6)
(This,9)
(Hello,,6)
(sessionHello,,3)
[acadgild@localhost wordcount1]$ cd ..
[acadgild@localhost ~]$ cd wordcount2
[acadgild@localhost wordcount2]$ cat part-00000
(is,9)
(BDH,9)
(session,6)
(This,9)
(Hello,,6)
(sessionHello,,3)
[acadgild@localhost wordcount2]$
```



word counts of the file
in local file system



word counts of the file
in HDFS