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: 8

1. Pick up a file from the local directory and do the word count 8

2. Then in the same Spark Application, write the code to put the same file on HDFS. 8

3. Then in same Spark Application, do the word count of the file copied on HDFS in step 2 8

4. Lastly, compare the word count of step 1 and 2. Both should match, other throw an error 8

Input file for all the below tasks:

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.

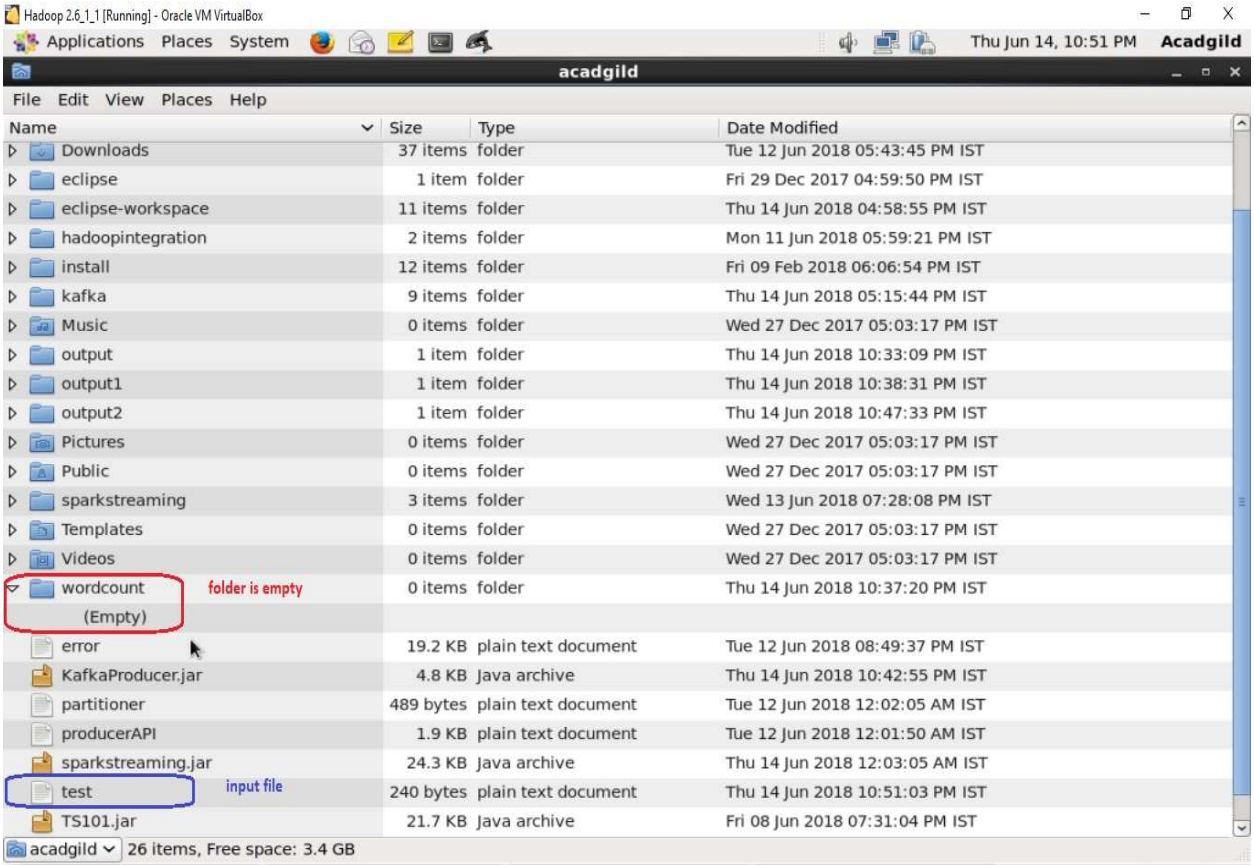
Program to perform the above task:

Required packages and imports: -

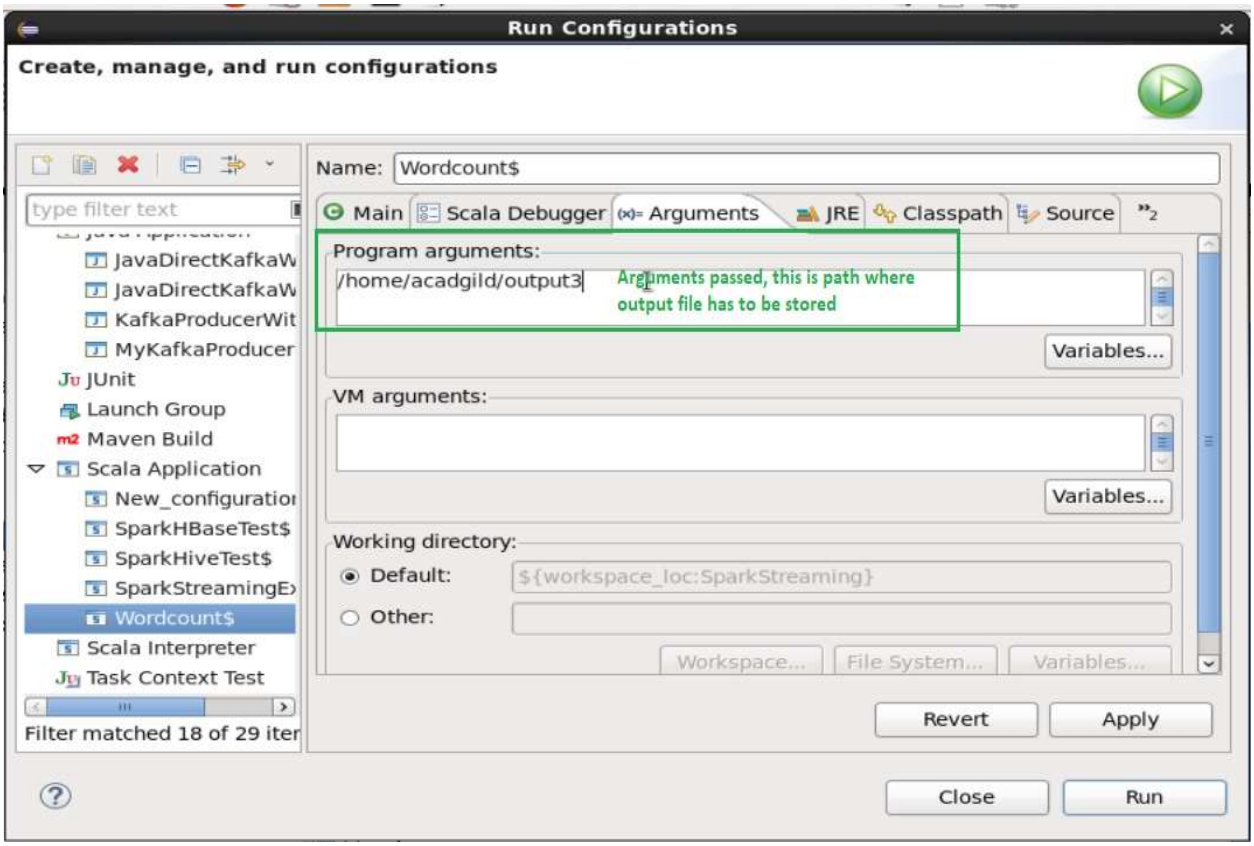
```
package com.acadgild.spark

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 object
val sc = new SparkContext(conf)
//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 = sc.textFile("/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.saveAsTextFile(args(0))
//stop the spark context
sc.stop }}
```

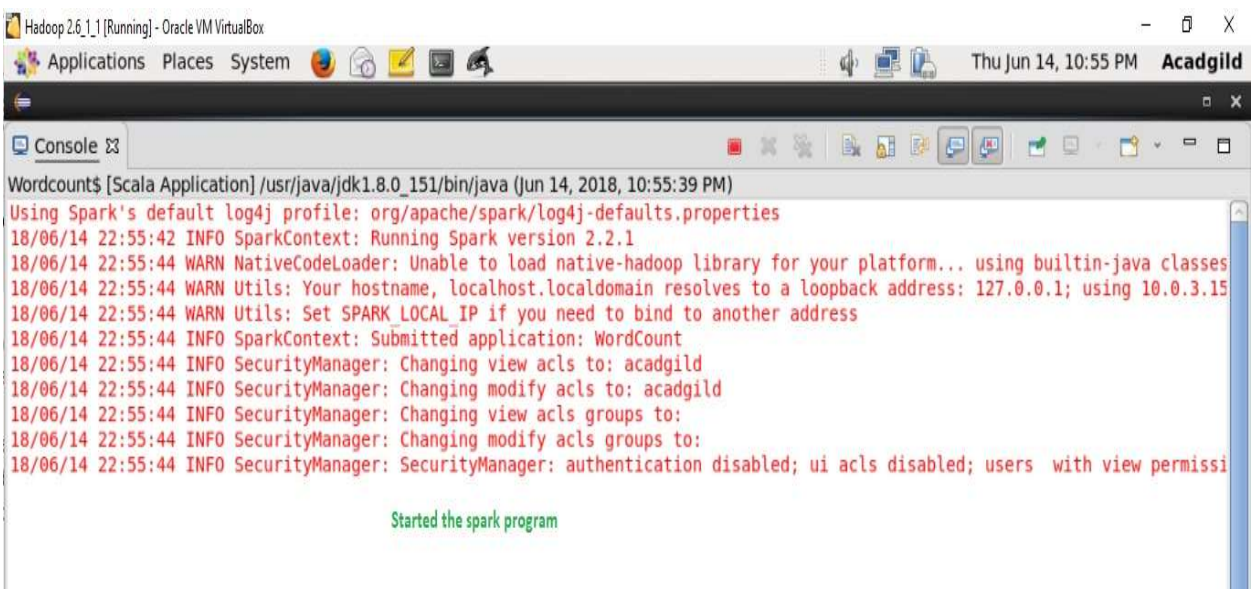
- Below screenshot shows that the directory is empty before the program is executed in eclipse.
- Input file “test” is outside the directory, which we will be adding it to wordcount directory during the program execution



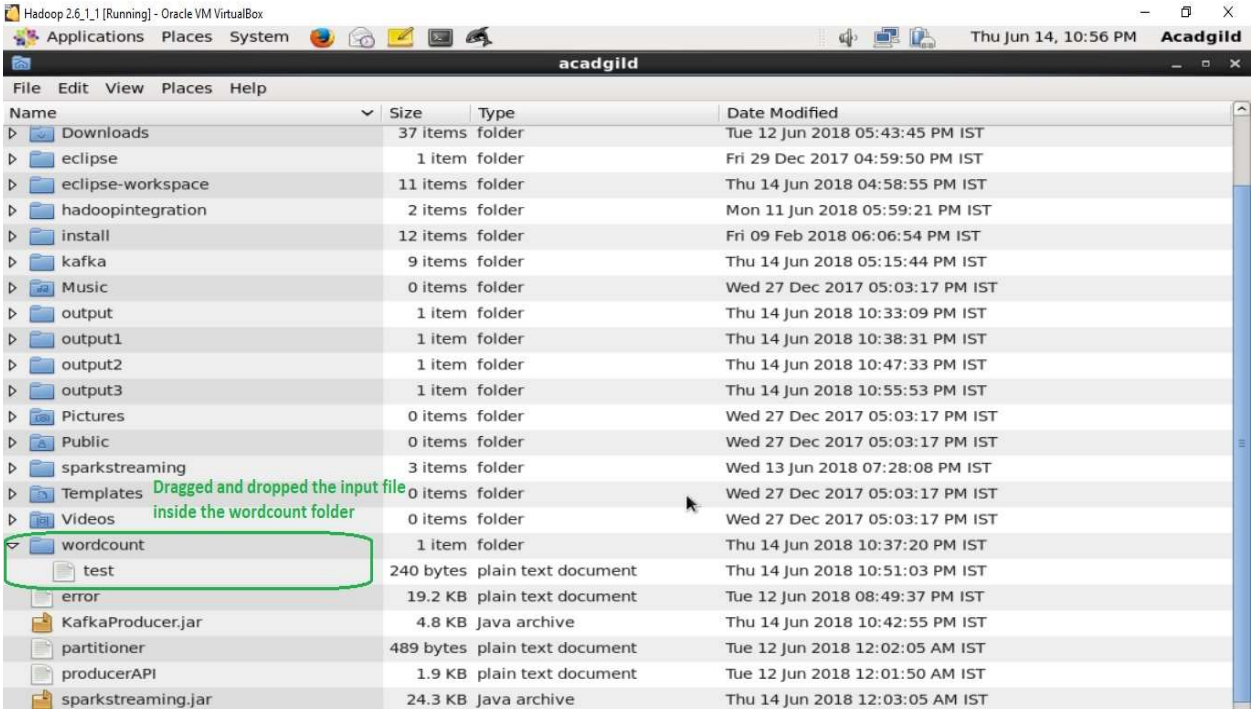
We will execute the program by passing the arguments in run configurations, as shown below:



Now we execute the program:



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:

Hadoop 2.6_1_1 [Running] - Oracle VM VirtualBox

Applications Places System Thu Jun 14, 11:03 PM Acadgil

Console

```
<terminated> Wordcount$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Jun 14, 2018, 10:57:41 PM)
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
18/06/14 22:57:44 INFO SparkContext: Running Spark version 2.2.1
18/06/14 22:57:46 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes
18/06/14 22:57:46 WARN Utils: Your hostname, localhost.localdomain resolves to a loopback address: 127.0.0.1; using 10.0.3.15
18/06/14 22:57:46 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
18/06/14 22:57:46 INFO SparkContext: Submitted application: WordCount
18/06/14 22:57:47 INFO SecurityManager: Changing view acls to: acadgild
18/06/14 22:57:47 INFO SecurityManager: Changing modify acls to: acadgild
18/06/14 22:57:47 INFO SecurityManager: Changing view acls groups to:
18/06/14 22:57:47 INFO SecurityManager: Changing modify acls groups to:
18/06/14 22:57:47 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users  with view permissi
18/06/14 22:57:48 INFO Utils: Successfully started service 'sparkDriver' on port 42888.
18/06/14 22:57:48 INFO SparkEnv: Registering MapOutputTracker
18/06/14 22:57:48 INFO SparkEnv: Registering BlockManagerMaster
18/06/14 22:57:48 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology
18/06/14 22:57:48 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
18/06/14 22:57:48 INFO DiskBlockManager: Created local directory at /tmp/blockmgr-737d6dc7-e286-4995-af88-9de9742fd83d
18/06/14 22:57:48 INFO MemoryStore: MemoryStore started with capacity 309.5 MB
18/06/14 22:57:49 INFO SparkEnv: Registering OutputCommitCoordinator
18/06/14 22:57:50 INFO Utils: Successfully started service 'SparkUI' on port 4040.
18/06/14 22:57:50 INFO SparkUI: Bound SparkUI to 0.0.0.0, and started at http://10.0.3.15:4040
18/06/14 22:57:51 INFO Executor: Starting executor ID driver on host localhost
18/06/14 22:57:51 INFO Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' on port
18/06/14 22:57:51 INFO NettyBlockTransferService: Server created on 10.0.3.15:42183
18/06/14 22:57:51 INFO BlockManager: Using org.apache.spark.storage.RandomBlockReplicationPolicy for block replication policy
18/06/14 22:57:51 INFO BlockManagerMaster: Registering BlockManager BlockManagerId(driver, 10.0.3.15, 42183, None)
18/06/14 22:57:51 INFO BlockManagerMasterEndpoint: Registering block manager 10.0.3.15:42183 with 309.5 MB RAM, BlockManagerI
18/06/14 22:57:51 INFO BlockManagerMaster: Registered BlockManager BlockManagerId(driver, 10.0.3.15, 42183, None)
18/06/14 22:57:51 INFO BlockManager: Initialized BlockManager: BlockManagerId(driver, 10.0.3.15, 42183, None)
18/06/14 22:57:54 INFO MemoryStore: Block broadcast_0 stored as values in memory (estimated size 236.5 KB, free 309.3 MB)
18/06/14 22:57:55 INFO MemoryStore: Block broadcast_0_piece0 stored as bytes in memory (estimated size 22.9 KB, free 309.3 MB
18/06/14 22:57:55 INFO BlockManagerInfo: Added broadcast_0_piece0 in memory on 10.0.3.15:42183 (size: 22.9 KB, free: 309.5 MB
18/06/14 22:57:55 INFO SparkContext: Created broadcast 0 from textFile at Wordcount.scala:22
18/06/14 22:57:55 INFO FileInputFormat: Total input paths to process : 1
18/06/14 22:57:56 INFO FileOutputCommitter: File Output Committer Algorithm version is 1
18/06/14 22:57:56 INFO SparkContext: Starting job: saveAsTextFile at Wordcount.scala:31
18/06/14 22:57:57 INFO DAGScheduler: Final stage: ResultStage 1 (saveAsTextFile at Wordcount.scala:31)
18/06/14 22:57:57 INFO DAGScheduler: Parents of final stage: List(ShuffleMapStage 0)
18/06/14 22:57:57 INFO DAGScheduler: Missing parents: List(ShuffleMapStage 0)
18/06/14 22:57:57 INFO DAGScheduler: Submitting ShuffleMapStage 0 (MapPartitionsRDD[3] at map at Wordcount.scala:28), which h
18/06/14 22:57:58 INFO MemoryStore: Block broadcast_1 stored as values in memory (estimated size 4.8 KB, free 309.3 MB)
18/06/14 22:57:58 INFO MemoryStore: Block broadcast_1_piece0 stored as bytes in memory (estimated size 2.8 KB, free 309.3 MB)
18/06/14 22:57:58 INFO BlockManagerInfo: Added broadcast_1_piece0 in memory on 10.0.3.15:42183 (size: 2.8 KB, free: 309.5 MB)
18/06/14 22:57:58 INFO SparkContext: Created broadcast 1 from broadcast at DAGScheduler.scala:1006
18/06/14 22:57:58 INFO DAGScheduler: Submitting 1 missing tasks from ShuffleMapStage 0 (MapPartitionsRDD[3] at map at Wordcou
18/06/14 22:57:58 INFO TaskSchedulerImpl: Adding task set 0.0 with 1 tasks
18/06/14 22:57:58 INFO TaskSetManager: Starting task 0.0 in stage 0.0 (TID 0, localhost, executor driver, partition 0, PROCES
18/06/14 22:57:58 INFO Executor: Running task 0.0 in stage 0.0 (TID 0)
18/06/14 22:57:59 INFO HadoopRDD: Input split: file:/home/acadgild/wordcount/test:0+240
18/06/14 22:57:59 INFO Executor: Finished task 0.0 in stage 0.0 (TID 0). 1193 bytes result sent to driver
18/06/14 22:57:59 INFO TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 1102 ms on localhost (executor driver) (1/1)
18/06/14 22:57:59 INFO TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool
18/06/14 22:57:59 INFO DAGScheduler: ShuffleMapStage 0 (map at Wordcount.scala:28) finished in 1.222 s
18/06/14 22:57:59 INFO DAGScheduler: looking for newly runnable stages
18/06/14 22:57:59 INFO DAGScheduler: running: Set()
18/06/14 22:57:59 INFO DAGScheduler: waiting: Set(ResultStage 1)
18/06/14 22:57:59 INFO DAGScheduler: failed: Set()
18/06/14 22:57:59 INFO DAGScheduler: Submitting ResultStage 1 (MapPartitionsRDD[5] at saveAsTextFile at Wordcount.scala:31),
18/06/14 22:58:00 INFO MemoryStore: Block broadcast_2 stored as values in memory (estimated size 72.0 KB, free 309.2 MB)
18/06/14 22:58:00 INFO MemoryStore: Block broadcast_2_piece0 stored as bytes in memory (estimated size 25.8 KB, free 309.2 MB
18/06/14 22:58:00 INFO BlockManagerInfo: Added broadcast_2_piece0 in memory on 10.0.3.15:42183 (size: 25.8 KB, free: 309.5 MB
18/06/14 22:58:00 INFO SparkContext: Created broadcast 2 from broadcast at DAGScheduler.scala:1006
18/06/14 22:58:00 INFO DAGScheduler: Submitting 1 missing tasks from ResultStage 1 (MapPartitionsRDD[5] at saveAsTextFile at
18/06/14 22:58:00 INFO TaskSchedulerImpl: Adding task set 1.0 with 1 tasks
18/06/14 22:58:00 INFO TaskSetManager: Starting task 0.0 in stage 1.0 (TID 1, localhost, executor driver, partition 0, ANY, 4
18/06/14 22:58:00 INFO Executor: Running task 0.0 in stage 1.0 (TID 1)
18/06/14 22:58:00 INFO ShuffleBlockFetcherIterator: Getting 1 non-empty blocks out of 1 blocks
18/06/14 22:58:00 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 11 ms
18/06/14 22:58:00 INFO FileOutputCommitter: File Output Committer Algorithm version is 1
18/06/14 22:58:00 INFO FileOutputCommitter: Saved output of task 'attempt_20180614225756_0001_m_000000_1' to file:/home/acadg
18/06/14 22:58:00 INFO SparkHadoopMapRedUtil: attempt_20180614225756_0001_m_000000_1: Committed
18/06/14 22:58:00 INFO Executor: Finished task 0.0 in stage 1.0 (TID 1). 1267 bytes result sent to driver
18/06/14 22:58:00 INFO TaskSetManager: Finished task 0.0 in stage 1.0 (TID 1) in 532 ms on localhost (executor driver) (1/1)
18/06/14 22:58:00 INFO DAGScheduler: ResultStage 1 (saveAsTextFile at Wordcount.scala:31) finished in 0.533 s
18/06/14 22:58:00 INFO TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have all completed, from pool
18/06/14 22:58:00 INFO DAGScheduler: Job 0 finished: saveAsTextFile at Wordcount.scala:31, took 4.468466 s
18/06/14 22:58:01 INFO SparkUI: Stopped Spark web UI at http://10.0.3.15:4040
18/06/14 22:58:01 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
18/06/14 22:58:01 INFO MemoryStore: MemoryStore cleared
18/06/14 22:58:01 INFO BlockManager: BlockManager stopped
18/06/14 22:58:01 INFO BlockManagerMaster: BlockManagerMaster stopped
18/06/14 22:58:01 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
18/06/14 22:58:01 INFO SparkContext: Successfully stopped SparkContext
18/06/14 22:58:01 INFO ShutdownHookManager: Shutdown hook called
18/06/14 22:58:01 INFO ShutdownHookManager: Deleting directory /tmp/spark-05bfcde5-d814-4cf5-a210-cafb6f1a447b
```

Now the program is successfully executed, so we will view the output in terminal as shown below:

```
acadgild@localhost:~/output3
File  Edit  View  Search  Terminal  Help
[acadgild@localhost ~]$ ls output3
part-00000  _SUCCESS
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ cat part-00000
cat: part-00000: No such file or directory
[acadgild@localhost ~]$ cd output3
[acadgild@localhost output3]$ cat part-00000
(is,9)
(BDH,9)
(session,6)
(This,9)
(Hello,,6)
(sessionHello,,3)
[acadgild@localhost output3]$
```

output: wordcount of the input file on the fly

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 hdfsfile = sc.textFile("hdfs://localhost:8020/wordcount/test")
     val hdfswords = hdfsfile.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-00000").getLines().toArray

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

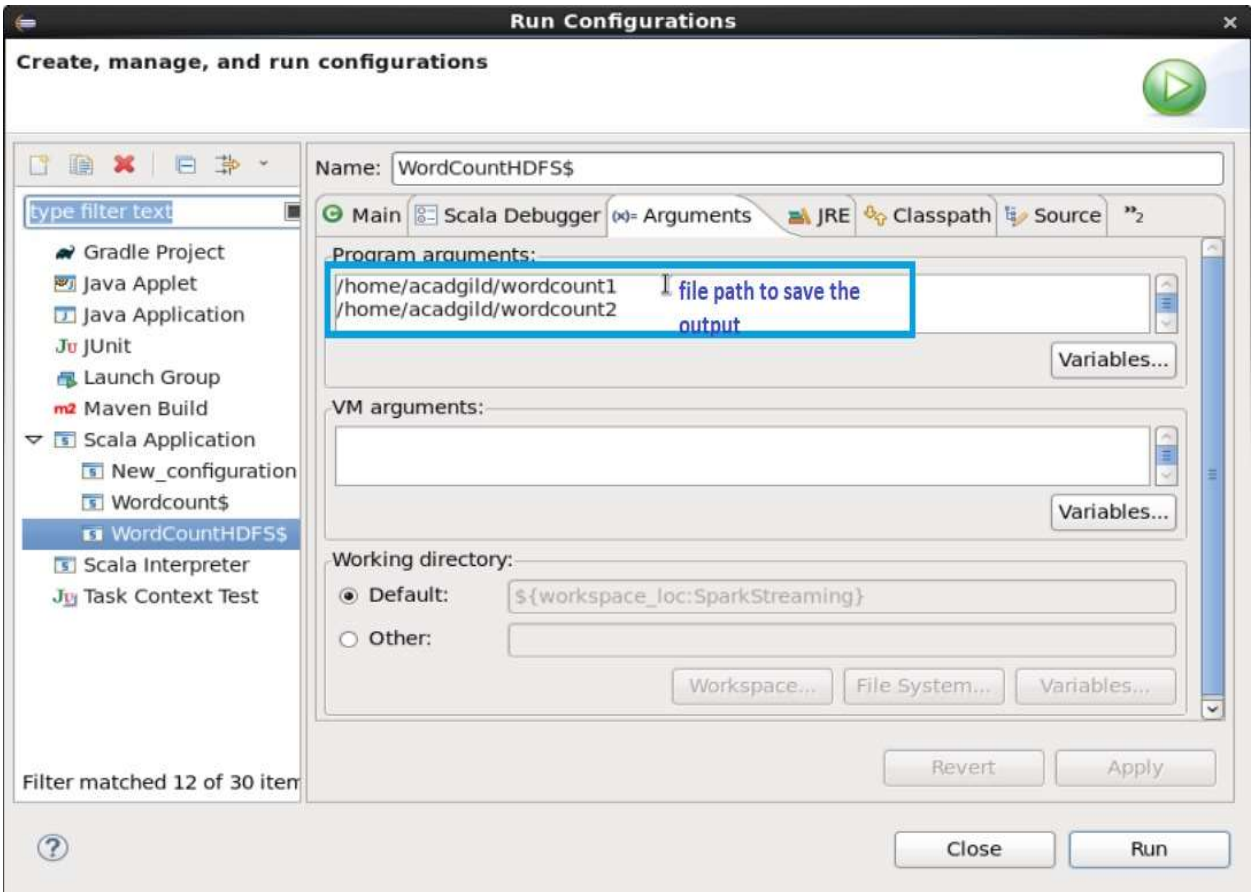
//now we save the Boolean value in variable “elem” and check if it is true or false, if it is //false it will print and error saying contents mismatch if not it will print contents match!

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

if(elem == false){

  println("Error!: Contents mismatch")

}else

  println("Contents match!")
```

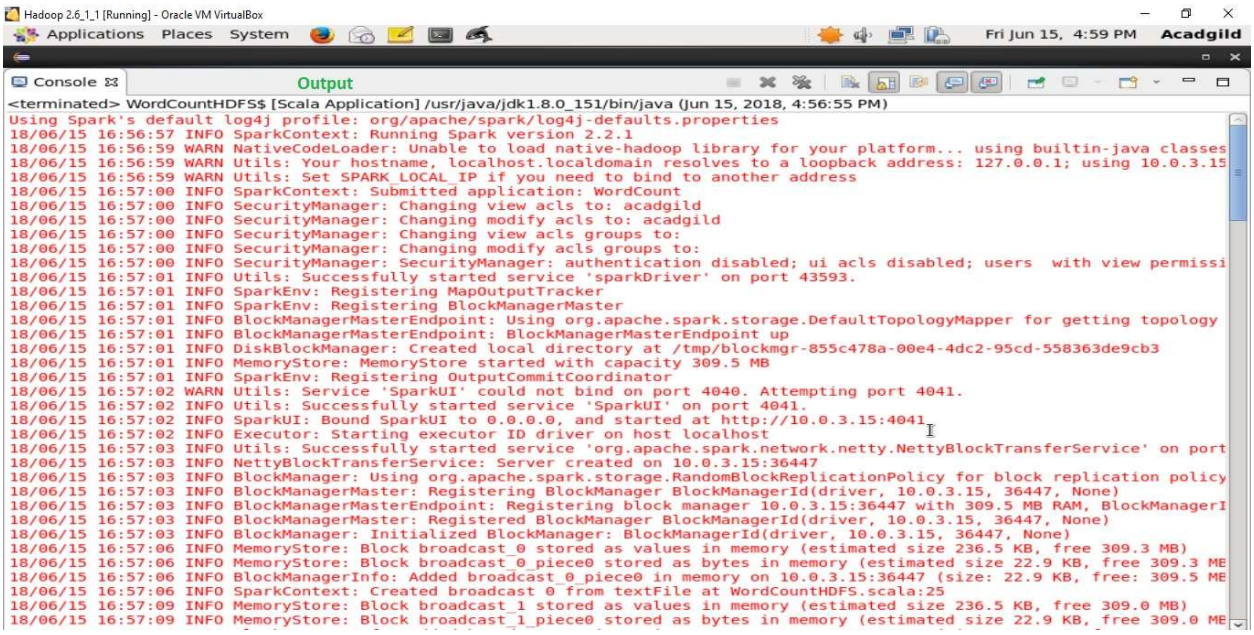
// we will print the output to console as well.

```
        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:



```
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/idk1.8.0_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), whic
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 M
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 M
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 WordCoun
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) 8/06/15 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), whic
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 M
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 M
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 WordCoun
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) 8/06/15 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
```

Wordcount of file in local file system

Wordcount of file in HDFS

- 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]$
```

Output files

word counts of the file in local file system

word counts of the file in HDFS