

BIG DATA HADOOP AND SPARK DEVELOPMENT

CASE STUDY V

Table of Contents:

1. Introduction	2
2. Objective	2
3. Problem Statement	2
4. Expected Output	
• Task 1	3
○ Objective 1	3
○ Objective 2	5

BIG DATA HADOOP AND SPARK DEVELOPMENT

1. Introduction

In this case study, the given tasks are performed and Output of the tasks are recorded in the form of Screenshots.

2. Objective

This case study consolidates the deeper understanding of the Sessions

3. Problem Statement

- Task 1

There are two parts this case study

- Objective 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 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.

- Objective 2

- **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

4. Expected Output

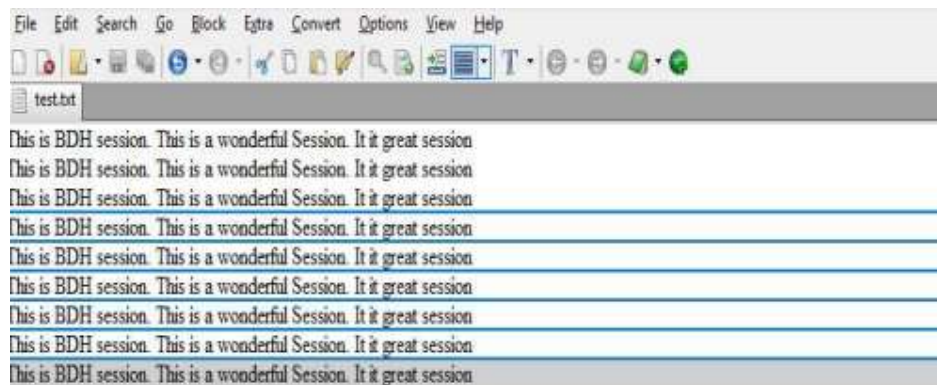
- Task 1

There are two parts this case study

- Objective 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 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.

Input file for the below Tasks 1:



Program to perform the above Objective 1:

```
package SparkstreamingAssignments

import org.apache.spark.SparkConf
import org.apache.spark.streaming._
object Wordcount {

  def main(args: Array[String]) {
    // create a spark configuration with Appname and set the local master
    val sparkConf = new SparkConf().setAppName("TextFileStream").setMaster("local[*]")
    // create a streaming context with a 10 seconds batch
    val ssc = new StreamingContext(sparkConf, Seconds(10))
    // create a text file stream by giving the directory path
    val lines = ssc.textFileStream("e:\\sparkstreaming")
    // word count function
    val wordcounts = lines.flatMap(line => line.split(" ").map(word => (word, 1))).reduceByKey(_ + _)
    // print the output in the console
    wordcounts.print()
    ssc.start()
    ssc.awaitTermination()
  }
}
```

Executing the program:

```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
NewProj uc main scala SparkstreamingAssignments Wordcount.scala

Run Wordcount()
20/06/17 17:11:54 INFO BlockManagerMasterEndpoint: Registering block manager (192.168.56.1:49436) with 902.7 MB RAM, BlockManagerId(driver, 192.168.56.1, 49436, None)
20/06/17 17:11:54 INFO BlockManagerMaster: Registered BlockManager BlockManagerId(driver, 192.168.56.1, 49436, None)
20/06/17 17:11:54 INFO BlockManager: Initialized BlockManager: BlockManagerId(driver, 192.168.56.1, 49436, None)
20/06/17 17:11:54 INFO FileInputOutputStream: Duration for assembling RDD set to 4000 ms for org.apache.spark.streaming.dstream.FileInputStream$$Inflator
20/06/17 17:11:54 INFO FileInputOutputStream: Slide time = 10000 ms
20/06/17 17:11:54 INFO FileInputOutputStream: Storage level = Serialized in Replicated
20/06/17 17:11:54 INFO FileInputOutputStream: Checkpoint interval = null
20/06/17 17:11:54 INFO FileInputOutputStream: Retention interval = 4000 ms
20/06/17 17:11:54 INFO FileInputOutputStream: Initialized and validated org.apache.spark.streaming.dstream.FileInputStream$$Inflator
20/06/17 17:11:54 INFO HadoopStream: Slide time = 10000 ms
20/06/17 17:11:54 INFO HadoopStream: Storage level = Serialized in Replicated
20/06/17 17:11:54 INFO HadoopStream: Checkpoint interval = null
20/06/17 17:11:54 INFO HadoopStream: Retention interval = 10000 ms
20/06/17 17:11:54 INFO HadoopStream: Initialized and validated org.apache.spark.streaming.dstream.HadoopStream$$Inflator
20/06/17 17:11:54 INFO FileHadoopStream: Slide time = 10000 ms
20/06/17 17:11:54 INFO FileHadoopStream: Storage level = Serialized in Replicated
20/06/17 17:11:54 INFO FileHadoopStream: Checkpoint interval = null
20/06/17 17:11:54 INFO FileHadoopStream: Retention interval = 10000 ms
20/06/17 17:11:54 INFO FileHadoopStream: Initialized and validated org.apache.spark.streaming.dstream.FileHadoopStream$$Inflator
20/06/17 17:11:54 INFO ShuffleStream: Slide time = 10000 ms
20/06/17 17:11:54 INFO ShuffleStream: Storage level = Serialized in Replicated
20/06/17 17:11:54 INFO ShuffleStream: Checkpoint interval = null
20/06/17 17:11:54 INFO ShuffleStream: Retention interval = 10000 ms
20/06/17 17:11:54 INFO ShuffleStream: Initialized and validated org.apache.spark.streaming.dstream.ShuffleStream$$Inflator
20/06/17 17:11:54 INFO ForwardStream: Slide time = 10000 ms
20/06/17 17:11:54 INFO ForwardStream: Storage level = Serialized in Replicated
20/06/17 17:11:54 INFO ForwardStream: Checkpoint interval = null
20/06/17 17:11:54 INFO ForwardStream: Retention interval = 10000 ms
20/06/17 17:11:54 INFO ForwardStream: Initialized and validated org.apache.spark.streaming.dstream.ForwardStream$$Inflator
20/06/17 17:11:54 INFO MonitoringTimer: Started timer for Subsequent at time 1529237700000
20/06/17 17:11:54 INFO JobGenerator: Started JobGenerator at 1529237710000 ms
20/06/17 17:12:04 INFO JobScheduler: Started JobScheduler
```

```
NewProj [C:\Users\Laptop\IdeaProjects\NewProj] - ..\src\main\scala\SparkstreamingAssignments\Wordcount.scala (newproj) - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
NewProj uc main scala SparkstreamingAssignments Wordcount.scala

Run Wordcount()
20/06/17 17:13:57 INFO JobScheduler: Validated RDD set as stage 51.0 (RDD 51) in 16 ms on localhost (executor driver) (1/3)
20/06/17 17:14:00 INFO TaskSetManager: Finished task 2.0 in stage 51.0 (TID 51) in 16 ms on localhost (executor driver) (1/3)
20/06/17 17:14:00 INFO Executor: Finished task 1.0 in stage 51.0 (TID 51), 1974 bytes result sent to driver
20/06/17 17:14:00 INFO TaskSetManager: Finished task 0.0 in stage 51.0 (TID 51) in 20 ms on localhost (executor driver) (2/3)
20/06/17 17:14:00 INFO TaskSetManager: Finished task 1.0 in stage 51.0 (TID 51) in 18 ms on localhost (executor driver) (3/3)
20/06/17 17:14:00 INFO TaskSchedulerImpl: Removed TaskSet 51.0, whose tasks have all completed, from pool
20/06/17 17:14:00 INFO DAGScheduler: ResultStage 51 (print at Wordcount.scala:12) finished in 0.026 s
20/06/17 17:14:00 INFO DAGScheduler: Job 25 finished: print at Wordcount.scala:12, task 0.036274 s
-----
Time: 152923840000 ms
-----
(session,,9)
(session,,9)
(session@This,6)
(x,9)
(great,9)
(is,10)
(RDD,9)
(wonderful,9)
(session,,3)
(This,12)
...

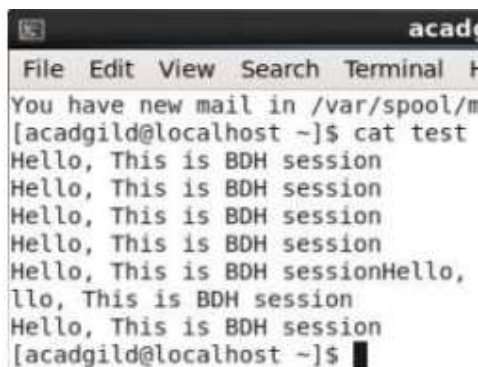
20/06/17 17:14:00 INFO DAGScheduler: Finished job streaming job 152923840000 ms.0 from job set of time 152923840000 ms
20/06/17 17:14:00 INFO DAGScheduler: Total delay: 0.210 s for time 152923840000 ms (execution: 0.163 s)
20/06/17 17:14:00 INFO ShuffleRDD: Removing RDD 46 from persistence list
20/06/17 17:14:00 INFO BlockManager: Removing RDD 46
20/06/17 17:14:00 INFO MapPartitionsRDD: Removing RDD 47 from persistence list
20/06/17 17:14:00 INFO MapPartitionsRDD: Removing RDD 46 from persistence list
20/06/17 17:14:00 INFO BlockManager: Removing RDD 47
```

Objective 2

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

Input file for the below Objective 2:



The screenshot shows a terminal window with a menu bar (File, Edit, View, Search, Terminal) and a title bar (acad). The terminal output shows the command `cat test` being executed, displaying the contents of a file named 'test'. The file contains the text 'Hello, This is BDH session' repeated multiple times.

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 ")
    }
  }
}
```

```
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/hadoop2.6.5/etc/hadoop/core-site.xml"))  
hadoopConf.addResource(new  
Path("/home/acadgild/install/hadoop/hadoop2.6.5/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/part00000").getLines().toArray  
val hdfsWCfile= Source.fromFile("/home/acadgild/wordcount2/part00000").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 }}
```


We can see the output as below:

```
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 cla
18/06/15 16:56:59 WARN Utils: Your hostname, localhost.localdomain resolves to a loopback address: 127.0.0.1; using 10.0.
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 perm
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 topol
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
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 po
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, BlockMana
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.
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.
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.
```

```

<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), whi
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
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
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 WordCou
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,
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 4.0, 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 Wor
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), whi
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
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
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 WordCou
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,
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)(BDH,9)(session,6)(This,9)(Hello,,6)(sessionHello,,3) 18/06/15 16:57:15 INFO SparkUI: Stopped Spark web UI at http://10.0.3.1
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

```

- 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.


```
[acagild@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/acagild
[acagild@localhost ~]$ cd wordcount1
[acagild@localhost wordcount1]$ ls
part-000000 SUCCESS
[acagild@localhost wordcount1]$ cat part-000000
(is,9)
(BDH,9)
(session,6)
(This,9)
>Hello,,6)
(sessionHello,,3)
[acagild@localhost wordcount1]$ cd ..
[acagild@localhost ~]$ cd wordcount2
[acagild@localhost wordcount2]$ cat part-000000
(is,9)
(BDH,9)
(session,6)
(This,9)
>Hello,,6)
(sessionHello,,3)
[acagild@localhost wordcount2]$
```

Output files

word counts of the file
in local file system

word counts of the file
in HDFS