Case Study 4 - Spark Streaming

Task 1

In this case, study, we create two spark applications:

First we create Spark Application which streams data from a local directory on our machine and will do a word count.

The word count should done by spark application in such a way that as soon as we drop a file in that local directory, our spark-application should immediately do the word count of that file.

Add the spark external jars through build path -> configure build path

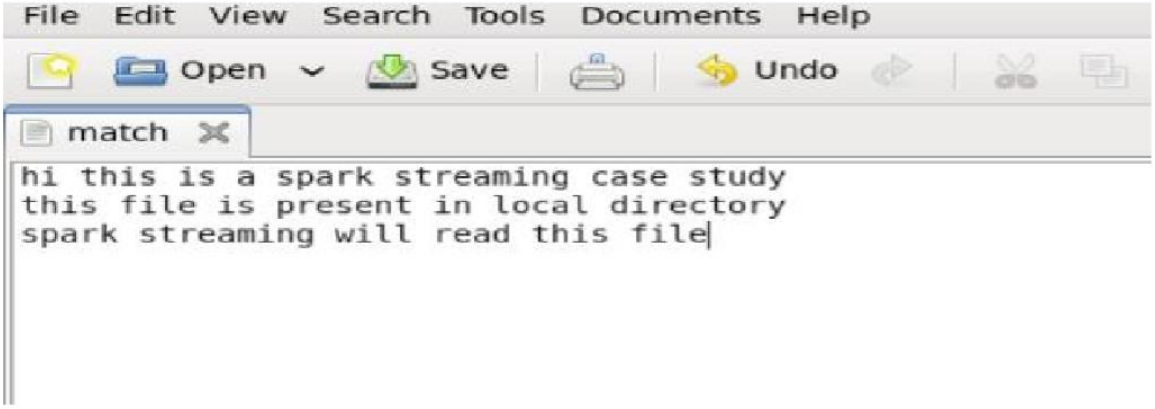
Code screenshot , I have attached in the code in the link as well.

Below is the code for task 1

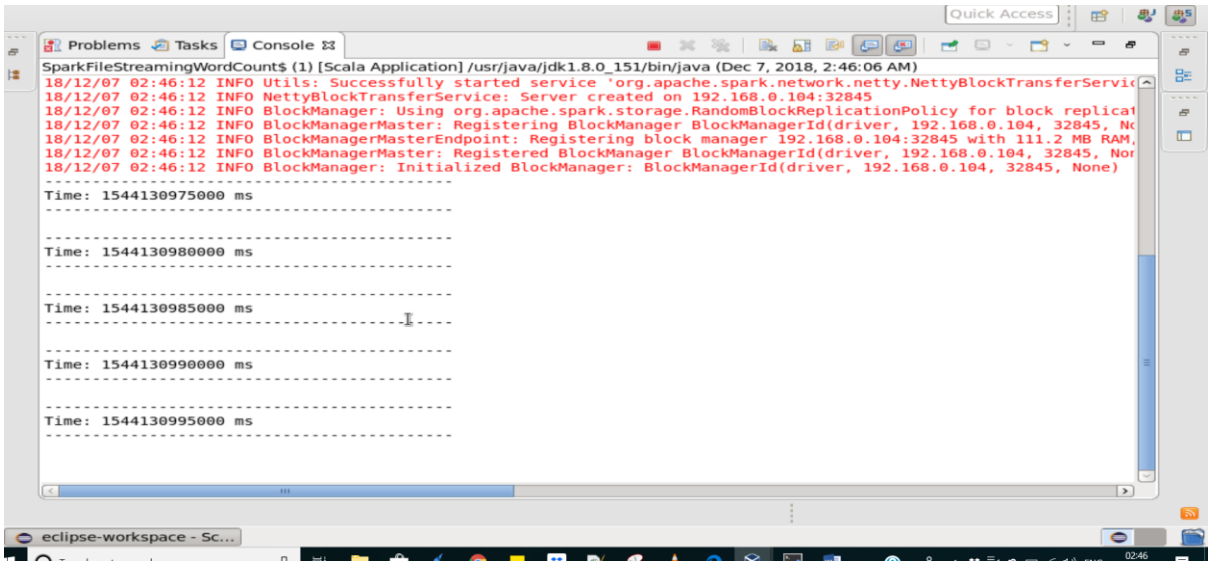
NetworkWordCount.scala | SparkFileStreamingWordCount.scala | *SparkHDFSWordCountComparison.scala

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

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)

   // Create Streaming context  to set batch duration 5 seconds
   val ssc = new StreamingContext(sc, Seconds(5))

//Create RDD for text file streaming by
val lines = ssc.textFileStream("/home/acadgild/Desktop/Spark_Streaming")
//Split each line into words
val words = lines.flatMap(_.split(" "))
//Count each word in each batch
val wordCounts = words.map(x => (x, 1)).reduceByKey(_ + _)
wordCounts.print()
//Start the computation
ssc.start()
//wait for the computation to terminate
ssc.awaitTermination()

     }
```

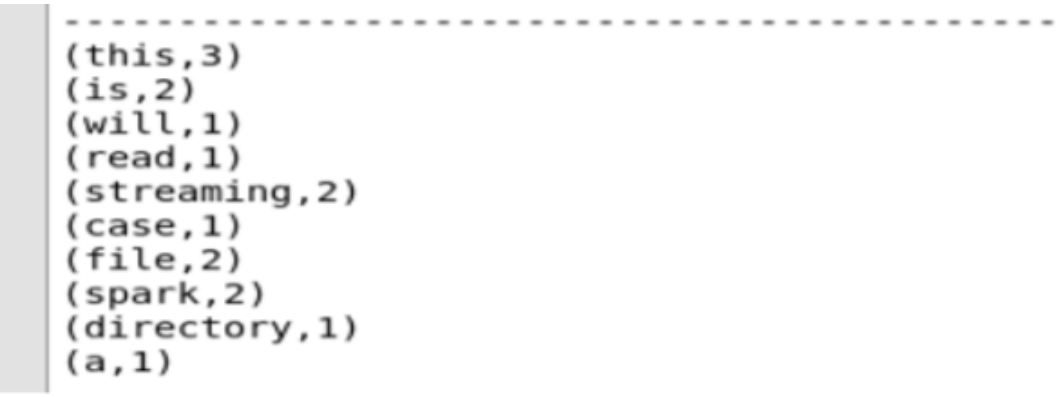
File in spark streaming directory



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



Output of console



Task 2

Second, we create Spark Application that pick up a file from local directory and do the word count, and then put the same file on HDFS, then same application will word count from this copied file on HDFS.

Lastly compare the word count results of first and second step, both should match otherwise throws an error.

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

```
[acadgild@localhost Spark_Streaming]$ ll
total 12
-rw-rw-r--. 1 acadgild acadgild 16 Dec  7 06:55 match
-rw-rw-r--. 1 acadgild acadgild 13 Dec  7 03:43 text
-rw-rw-r--. 1 acadgild acadgild  9 Dec  7 03:42 text~
[acadgild@localhost Spark_Streaming]$ cat text
hi hello
hi

[acadgild@localhost Spark_Streaming]$
```

Below is code for the second case

```
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")

//read the file which is present in local directory  and convert into string
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 ->>" +localWord
```

```
33
34      println("Performing local word count Completed !!")
35
36      println("Creating Spark Context")
37
38  //Create spark context
39
40      val conf = new SparkConf().setMaster("local[2]").setAppName("SparkHDFSWordCountComparisonApp")
41    val sc = new SparkContext(conf)
42
43  // Setting log level to [WARN] for streaming executions  and to override add a custom log4j.properties to t
44    val rootLogger =Logger.getRootLogger()
45    rootLogger.setLevel(Level.ERROR)
46
47            println("Spark Context Created")
48
49      println("Writing local file to DFS")
50    val dfsFilename = dfsDirPath + "/dfs_read_write_test"
51
52      val fileRDD = sc.parallelize(fileContents)
53    fileRDD.saveAsTextFile(dfsFilename)
54
55      println("Writing local file to DFS Completed")
56
57      println("Reading file from DFS and running Word Count")
58
59      val readFileRDD = sc.textFile(dfsFilename)
60
61      val dfsWordCount = readFileRDD
62    .flatMap(_.split(" "))
```

```
63    .flatMap(_.split("\t"))
64    .filter(_.nonEmpty)
65    .map(w => (w, 1))
66    .countByKey()
67    .values
68    .sum
69
70      sc.stop()
71
72
73
74  //apply if condition to check word count result from both the directories
75
76      if (localWordCount == dfsWordCount)
77    {
78    println(s"Success! Local Word Count ($localWordCount) "
79    +s"and DFS Word Count ($dfsWordCount) agree.")
80    }
81    else
82    {
83    println(s"Failure! Local Word Count ($localWordCount) " +
84    s"and DFS Word Count ($dfsWordCount) disagree.")
85    }
86
87    }
88
89    /***private def parseArgs(args: Array[String]): Unit = {
90    if (args.length != NPARAMS) {
91    printUsage()
92    System.exit(1)
```

Writable | Smart Insert | 88 : 2

```
97    private def printUsage(): Unit = {
98    val usage: String = "DFS Read-Write Test\n" +
99    "\n" +
100   "Usage: localFile dfsDir\n" +
101   "\n" +
102   "localFile - (string) local file to use in test\n" +
103   "dfsDir - (string) DFS directory for read/write tests\n"
104
105     println(usage)
106   }
107
108
109
110   private def readFile(filename: String): List[String] = {
111
112     val lineIter: Iterator[String] = fromFile(filename).getLines()
113   val lineList: List[String] = lineIter.toList
114   lineList
115   }
116
117   def runLocalWordCount(fileContents: List[String]): Int = {
118   fileContents.flatMap(_.split(" "))
119   .flatMap(_.split("\t"))
120   .filter(_.nonEmpty)
121   .groupBy(w => w)
122   .mapValues(_.size)
123   .values
124   .sum
125   }
126 }
```

Output console

```
Console
<terminated> SparkHDFSWordCountComparison$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Dec 7, 2018, 7:29:59 AM)
SparkHDFSWordCountComparison : Main Called Successfully
Performing local word count
Performing local word count - File Content ->>List(hi hello, hi, )
SparkHDFSWordCountComparison : Main Called Successfully -> Local Word Count is ->>3
Performing local word count Completed !!
Creating Spark Context
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
18/12/07 07:30:03 INFO SparkContext: Running Spark version 2.2.1
18/12/07 07:30:04 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-ja
18/12/07 07:30:05 WARN Utils: Your hostname, localhost.localdomain resolves to a loopback address: 127.0.0.1; using
18/12/07 07:30:05 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
18/12/07 07:30:05 INFO SparkContext: Submitted application: SparkHDFSWordCountComparisonApp
18/12/07 07:30:05 INFO SecurityManager: Changing view acls to: acadgild
18/12/07 07:30:05 INFO SecurityManager: Changing modify acls to: acadgild
18/12/07 07:30:05 INFO SecurityManager: Changing view acls groups to:
18/12/07 07:30:05 INFO SecurityManager: Changing modify acls groups to:
18/12/07 07:30:05 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users  with vie
18/12/07 07:30:06 INFO Utils: Successfully started service 'sparkDriver' on port 40220.
18/12/07 07:30:06 INFO SparkEnv: Registering MapOutputTracker
18/12/07 07:30:06 INFO SparkEnv: Registering BlockManagerMaster
18/12/07 07:30:06 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting
18/12/07 07:30:06 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
18/12/07 07:30:06 INFO DiskBlockManager: Created local directory at /tmp/blockmgr-d76259b8-088f-4df8-97d7-082ed5743
18/12/07 07:30:06 INFO MemoryStore: MemoryStore started with capacity 111.2 MB
18/12/07 07:30:06 INFO SparkEnv: Registering OutputCommitCoordinator
18/12/07 07:30:07 INFO Utils: Successfully started service 'SparkUI' on port 4040.
18/12/07 07:30:08 INFO SparkUI: Bound SparkUI to 0.0.0.0, and started at http://192.168.0.104:4040
18/12/07 07:30:08 INFO Executor: Starting executor ID driver on host localhost
18/12/07 07:30:08 INFO Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferServic
```

```
18/12/07 07:30:08 INFO NettyBlockTransferService: Server created on 192.168.0.104:34398
18/12/07 07:30:08 INFO BlockManager: Using org.apache.spark.storage.RandomBlockReplicationPolicy for block replicat
18/12/07 07:30:08 INFO BlockManagerMaster: Registering BlockManager BlockManagerId(driver, 192.168.0.104, 34398, No
18/12/07 07:30:08 INFO BlockManagerMasterEndpoint: Registering block manager 192.168.0.104:34398 with 111.2 MB RAM,
18/12/07 07:30:08 INFO BlockManagerMaster: Registered BlockManager BlockManagerId(driver, 192.168.0.104, 34398, Nor
18/12/07 07:30:08 INFO BlockManager: Initialized BlockManager: BlockManagerId(driver, 192.168.0.104, 34398, None)
Spark Context Created
Writing local file to DFS
Writing local file to DFS Completed
Reading file from DFS and running Word Count
Success! Local Word Count (3) and DFS Word Count (3) agree.
```

Below is the screen shot where the data is saved in HDFS under /user directory

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

```
[acadgild@localhost Spark_Streaming]$ hadoop dfs -ls /user/dfs_read_write_test
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

18/12/07 08:12:15 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
Found 3 items
-rw-r--r--   3 acadgild supergroup          0 2018-12-07 07:30 /user/dfs_read_write_test/_SUCCESS
-rw-r--r--   3 acadgild supergroup          9 2018-12-07 07:30 /user/dfs_read_write_test/part-00000
-rw-r--r--   3 acadgild supergroup          4 2018-12-07 07:30 /user/dfs_read_write_test/part-00001
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost Spark_Streaming]$ hadoop dfs -cat /user/dfs_read_write_test/part-00000
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

18/12/07 08:12:23 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
hi hello
[acadgild@localhost Spark_Streaming]$ hadoop dfs -cat /user/dfs_read_write_test/part-00001
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

18/12/07 08:12:32 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
hi

[acadgild@localhost Spark_Streaming]$
```