ALY 6110: DATA MANAGEMENT AND BIGDATA

Assignment 5 Lab: Word Count using Apache Spark

Submitted To:
Prof. Andy Chen, Faculty Lecturer
Mr. James Kong, Teaching Assistant

Submitted By:
Abhilash Dikshit

Academic Term: Spring 2023
Graduate Students at Northeastern University, Vancouver, BC, Canada
Master of Professional Studies in Analytics

June 24, 2023

Lab Title: Word Count using Apache Spark

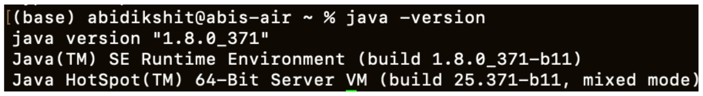
I. Introduction:

The purpose of this lab was to install Spark locally, cache a text file, perform word count analysis on the file, and answer a set of questions based on the word count results.

II. Lab Steps:

1. Installation:

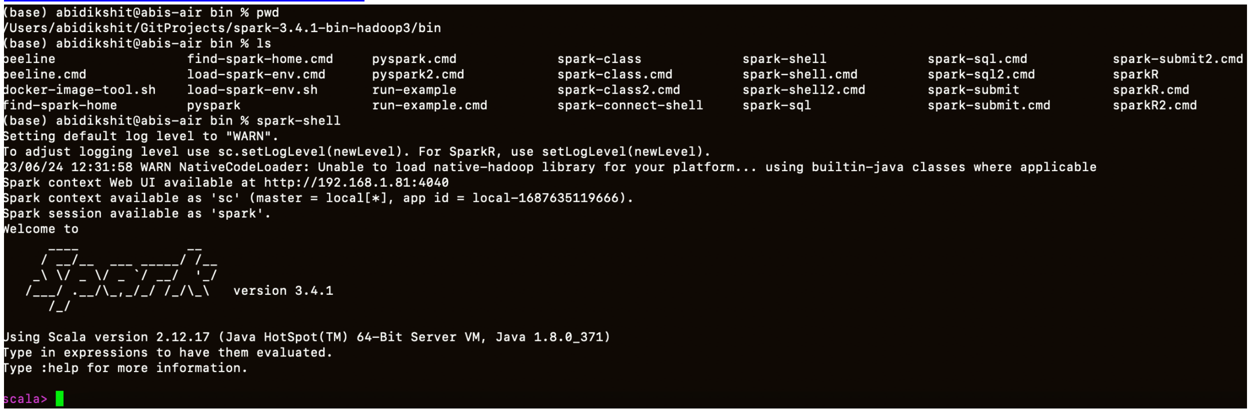
- Java Development Kit (JDK 8) was installed following the instructions provided by Oracle.



```
[(base) abidikshit@abis-air ~ % java -version
java version "1.8.0_371"
Java(TM) SE Runtime Environment (build 1.8.0_371-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.371-b11, mixed mode)
```

- Spark 3.4.1 was downloaded from the official Spark website and extracted to the desired location on the local machine.

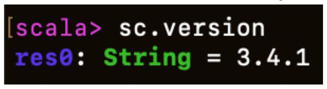
[Download Link for 3.4.1]



```
(base) abidikshit@abis-air bin % pwd
/Users/abidikshit/GitProjects/spark-3.4.1-bin-hadoop3/bin
(base) abidikshit@abis-air bin % ls
beeline                 find-spark-home.cmd     pyspark.cmd             spark-class             spark-shell             spark-sql.cmd           spark-submit2.cmd
beeline.cmd             load-spark-env.cmd      pyspark2.cmd            spark-class.cmd         spark-shell.cmd         spark-sql2.cmd          sparkR
docker-image-tool.sh    load-spark-env.sh       run-example             spark-class2.cmd        spark-shell2.cmd        spark-submit            sparkR.cmd
find-spark-home         pyspark                 run-example.cmd         spark-connect-shell     spark-sql               spark-submit.cmd        sparkR2.cmd
(base) abidikshit@abis-air bin % spark-shell
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
23/06/24 12:31:58 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Spark context Web UI available at http://192.168.1.81:4040
Spark context available as 'sc' (master = local[*], app id = local-1687635119666).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.4.1
      /_/

Using Scala version 2.12.17 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_371)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

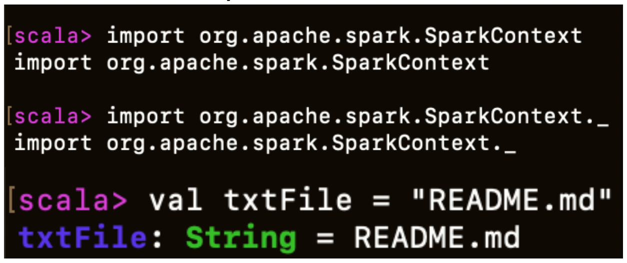
2. Caching the File:

- The README.md file was cached using the Spark shell.
- The Spark shell was launched from the Spark installation directory using the command `./bin/spark-shell`.



```
[scala> sc.version
res0: String = 3.4.1
```

- Inside the Spark shell, the README.md file was cached using the commands:

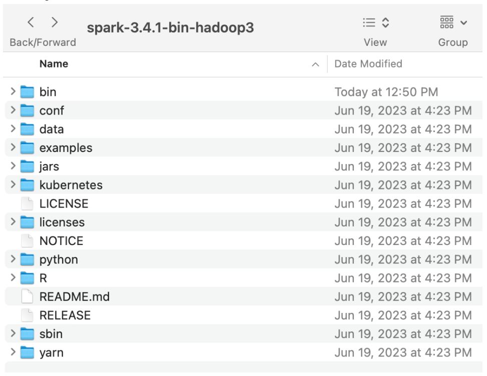


```
[scala> import org.apache.spark.SparkContext
import org.apache.spark.SparkContext

[scala> import org.apache.spark.SparkContext._
import org.apache.spark.SparkContext._

[scala> val txtFile = "README.md"
txtFile: String = README.md
```

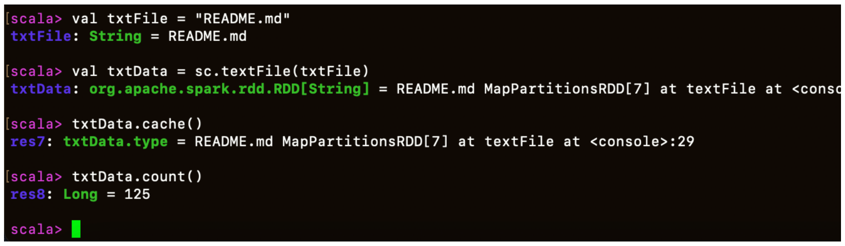
3. Word Count Analysis:

- The words in the README.md file was counted using Apache Spark.
- The following command was executed to perform the word count:

Copied Readme.md file to bin folder.

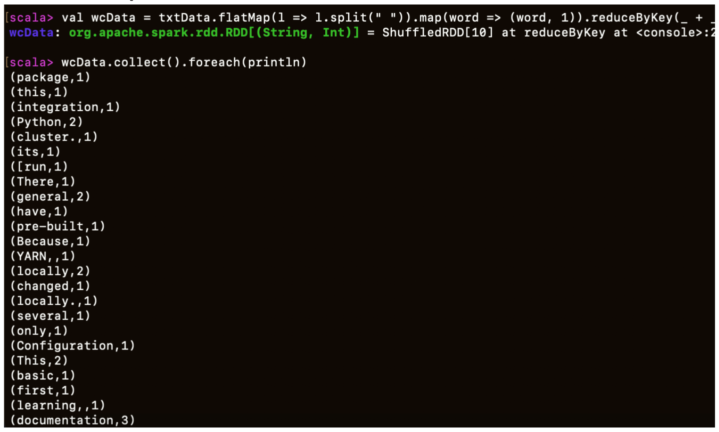


Name	Date Modified
bin	Today at 12:50 PM
conf	Jun 19, 2023 at 4:23 PM
data	Jun 19, 2023 at 4:23 PM
examples	Jun 19, 2023 at 4:23 PM
jars	Jun 19, 2023 at 4:23 PM
kubernetes	Jun 19, 2023 at 4:23 PM
LICENSE	Jun 19, 2023 at 4:23 PM
licenses	Jun 19, 2023 at 4:23 PM
NOTICE	Jun 19, 2023 at 4:23 PM
python	Jun 19, 2023 at 4:23 PM
R	Jun 19, 2023 at 4:23 PM
README.md	Jun 19, 2023 at 4:23 PM
RELEASE	Jun 19, 2023 at 4:23 PM
sbin	Jun 19, 2023 at 4:23 PM
yarn	Jun 19, 2023 at 4:23 PM

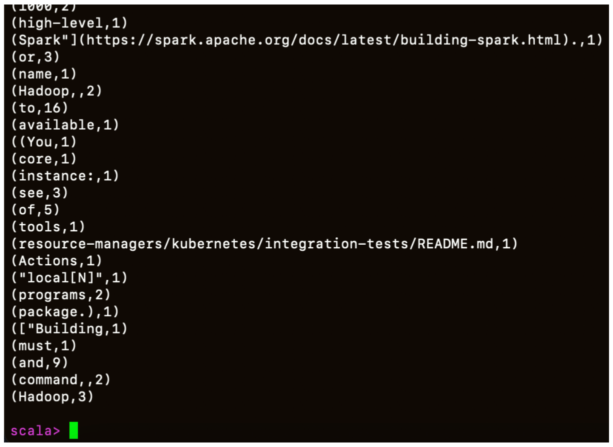


```
[scala> val txtFile = "README.md"
txtFile: String = README.md

[scala> val txtData = sc.textFile(txtFile)
txtData: org.apache.spark.rdd.RDD[String] = README.md MapPartitionsRDD[7] at textFile at <conso

[scala> txtData.cache()
res7: txtData.type = README.md MapPartitionsRDD[7] at textFile at <console>:29

[scala> txtData.count()
res8: Long = 125

scala>
```

Now, we can run the following commands to perform the word count. The count shows up next to each word in the text file.



```
[scala> val wcData = txtData.flatMap(l => l.split(" ")).map(word => (word, 1)).reduceByKey(_ + _
wcData: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[10] at reduceByKey at <console>:2

[scala> wcData.collect().foreach(println)
(package,1)
(this,1)
(integration,1)
(Python,2)
(cluster.,1)
(its,1)
([run,1)
(There,1)
(general,2)
(have,1)
(pre-built,1)
(Because,1)
(YARN,,1)
(locally,2)
(changed,1)
(locally.,1)
(several,1)
(only,1)
(Configuration,1)
(This,2)
(basic,1)
(first,1)
(learning,,1)
(documentation,3)
```

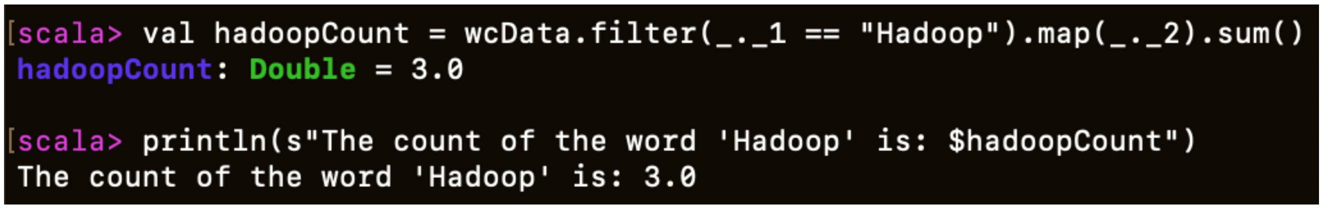


```
(1000,2)
(high-level,1)
(Spark"](https://spark.apache.org/docs/latest/building-spark.html).,1)
(or,3)
(name,1)
(Hadoop,,2)
(to,16)
(available,1)
((You,1)
(core,1)
(instance:,1)
(see,3)
(of,5)
(tools,1)
(resource-managers/kubernetes/integration-tests/README.md,1)
(Actions,1)
("local[N]",1)
(programs,2)
(package.),1)
(["Building,1)
(must,1)
(and,9)
(command,,2)
(Hadoop,3)

scala>
```

4. Answering the Questions:

- The questions were answered based on the word count results obtained from the previous step.
- The specific code snippets used to answer each question are as follows:

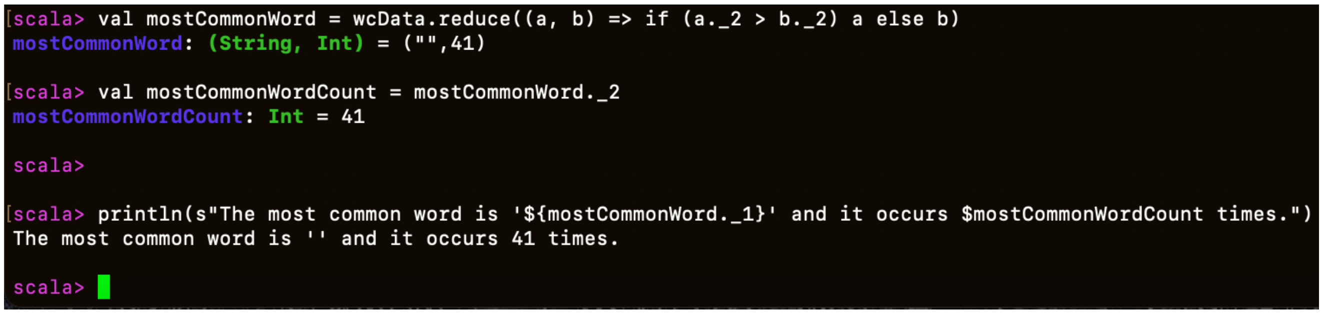
Question 1: How many times is the word "Hadoop" counted when the tutorial has printed out all the word counts?



```
[scala> val hadoopCount = wcData.filter(_._1 == "Hadoop").map(_._2).sum()
hadoopCount: Double = 3.0

[scala> println(s"The count of the word 'Hadoop' is: $hadoopCount")
The count of the word 'Hadoop' is: 3.0
```

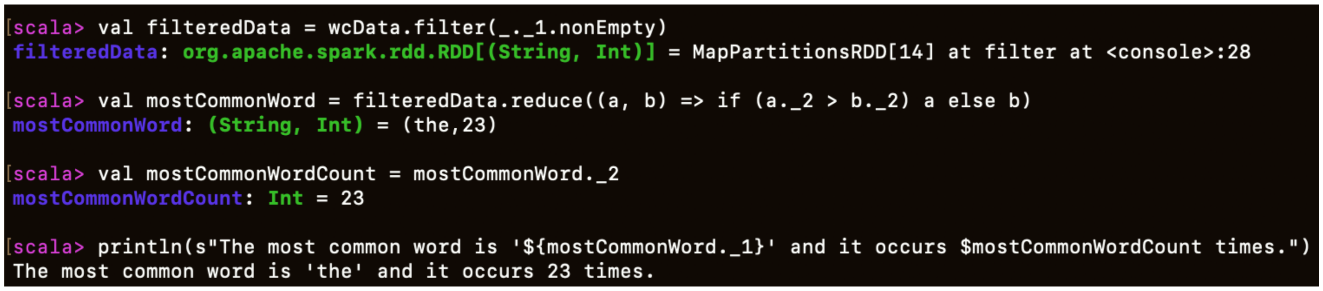
Answer: The count of the word 'Hadoop' is 3.

Question 2: Which is the most common word used in the file? How many times does the word occur?



```
scala> val mostCommonWord = wcData.reduce((a, b) => if (a._2 > b._2) a else b)
mostCommonWord: (String, Int) = ("",41)

scala> val mostCommonWordCount = mostCommonWord._2
mostCommonWordCount: Int = 41

scala>

scala> println(s"The most common word is '${mostCommonWord._1}' and it occurs $mostCommonWordCount times.")
The most common word is '' and it occurs 41 times.

scala>
```

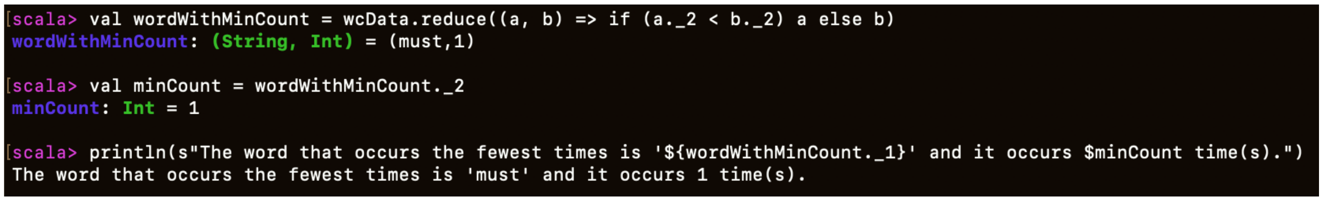
Based on above output we have empty values. Now, let's consider nonempty words.



```
scala> val filteredData = wcData.filter(_._1.nonEmpty)
filteredData: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[14] at filter at <console>:28

scala> val mostCommonWord = filteredData.reduce((a, b) => if (a._2 > b._2) a else b)
mostCommonWord: (String, Int) = (the,23)

scala> val mostCommonWordCount = mostCommonWord._2
mostCommonWordCount: Int = 23

scala> println(s"The most common word is '${mostCommonWord._1}' and it occurs $mostCommonWordCount times.")
The most common word is 'the' and it occurs 23 times.
```

Answer: The most common word is 'the' and it occurs 23 times.

Question 3: Which word occurs the fewest times (just pick one word)? How many times does the word occur?

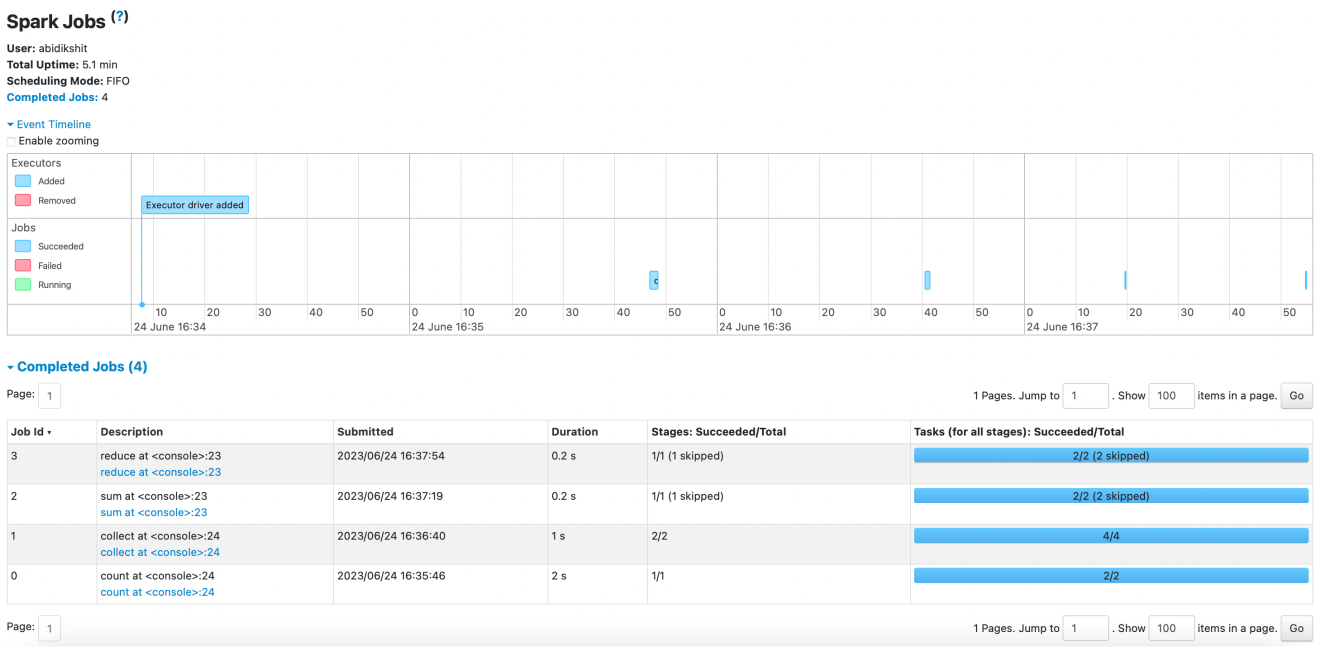


```
scala> val wordWithMinCount = wcData.reduce((a, b) => if (a._2 < b._2) a else b)
wordWithMinCount: (String, Int) = (must,1)

scala> val minCount = wordWithMinCount._2
minCount: Int = 1

scala> println(s"The word that occurs the fewest times is '${wordWithMinCount._1}' and it occurs $minCount time(s).")
The word that occurs the fewest times is 'must' and it occurs 1 time(s).
```

Answer: The word that occurs the fewest times is 'must' and it occurs 1 time(s).

5. Measurement of Execution Time:

- The execution time of the word count job was measured using the Spark web console.
- The web console was accessed by navigating to http://localhost:4040/jobs in a web browser.
- The completed job was identified in the list, and the execution time was noted.



Spark Jobs (?)

User: abidikshit
Total Uptime: 5.1 min
Scheduling Mode: FIFO
Completed Jobs: 4

Completed Jobs (4)

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
3	reduce at <console>:23 reduce at <console>:23	2023/06/24 16:37:54	0.2 s	1/1 (1 skipped)	2/2 (2 skipped)
2	sum at <console>:23 sum at <console>:23	2023/06/24 16:37:19	0.2 s	1/1 (1 skipped)	2/2 (2 skipped)
1	collect at <console>:24 collect at <console>:24	2023/06/24 16:36:40	1 s	2/2	4/4
0	count at <console>:24 count at <console>:24	2023/06/24 16:35:46	2 s	1/1	2/2

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

III. Conclusion:

In this lab, we successfully installed Spark locally, cached a text file, performed word count analysis, and answered a set of questions based on the word count results. The lab allowed us to gain hands-on experience with Spark and its functionalities. By following the provided instructions and completing the required tasks, we were able to accomplish the objectives of the lab and enhance our understanding of Apache Spark for big data processing.

IV. References:

1. Apache Spark. (n.d.). Apache Spark. Retrieved from https://spark.apache.org
2. Apache Spark. (n.d.). Examples. Retrieved from https://spark.apache.org/examples.html