College of Professional Studies

Northeastern University San Jose

MPS Analytics

Course: ALY6110 – Data Management and Big Data

Assignment:

Module 5 Lab 2

Submitted on:

July 1st, 2023

Submitted to:

Professor: BEHZAD AHMADI

Submitted by:

NIKSHITA RANGANATHAN

INTRODUCTION

Apache Spark is an open-source unified analytics engine for large-scale data processing. It provides high-level APIs in Java, Scala, Python, and R, and supports a wide range of workloads, including batch processing, interactive queries, streaming, machine learning, and graph processing.

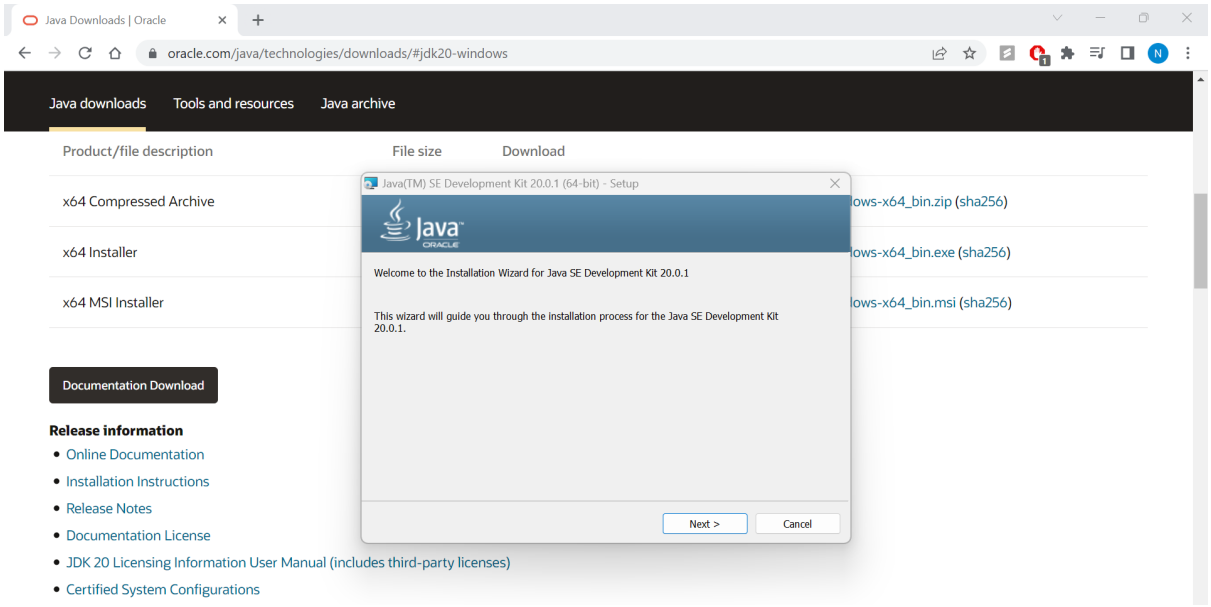
Spark is known for its speed and scalability. It can process data up to 100x faster than Hadoop MapReduce, and it can scale to thousands of nodes. Spark is also memory-efficient, which allows it to process data in memory, which can further improve performance.

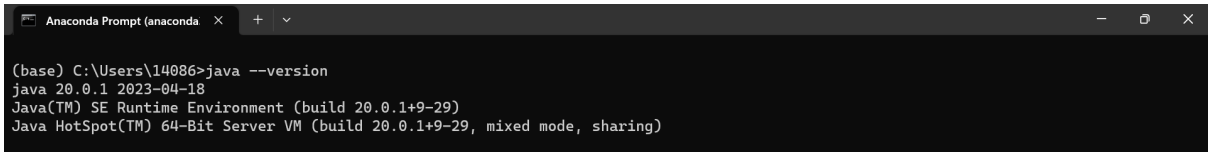
Spark is a popular choice for big data processing. It is used by companies such as Netflix, Yahoo, and eBay. It is also used by many academic institutions and research organizations.

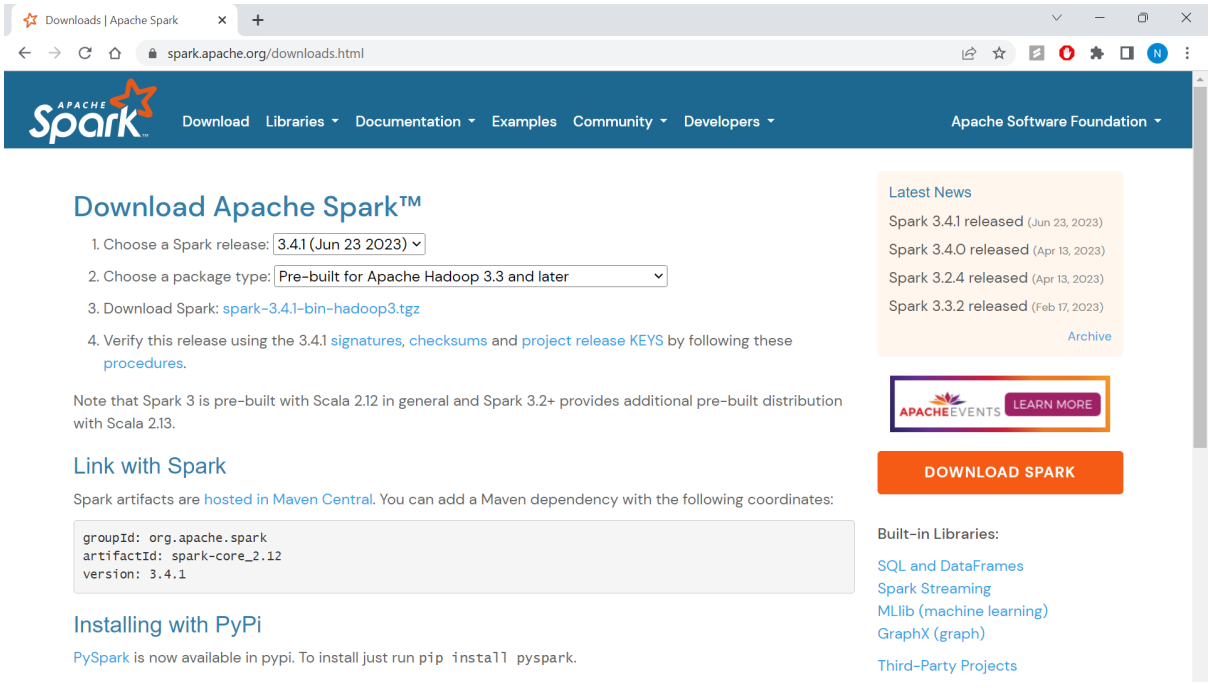
In this assignment, we will explore the basics of Big Data Processing with Apache Spark, specifically focusing on the Word Count application. The Word Count application is a popular example used to demonstrate the capabilities of Spark and serves as a starting point for many Spark developers.

ANALYSIS & RESULTS

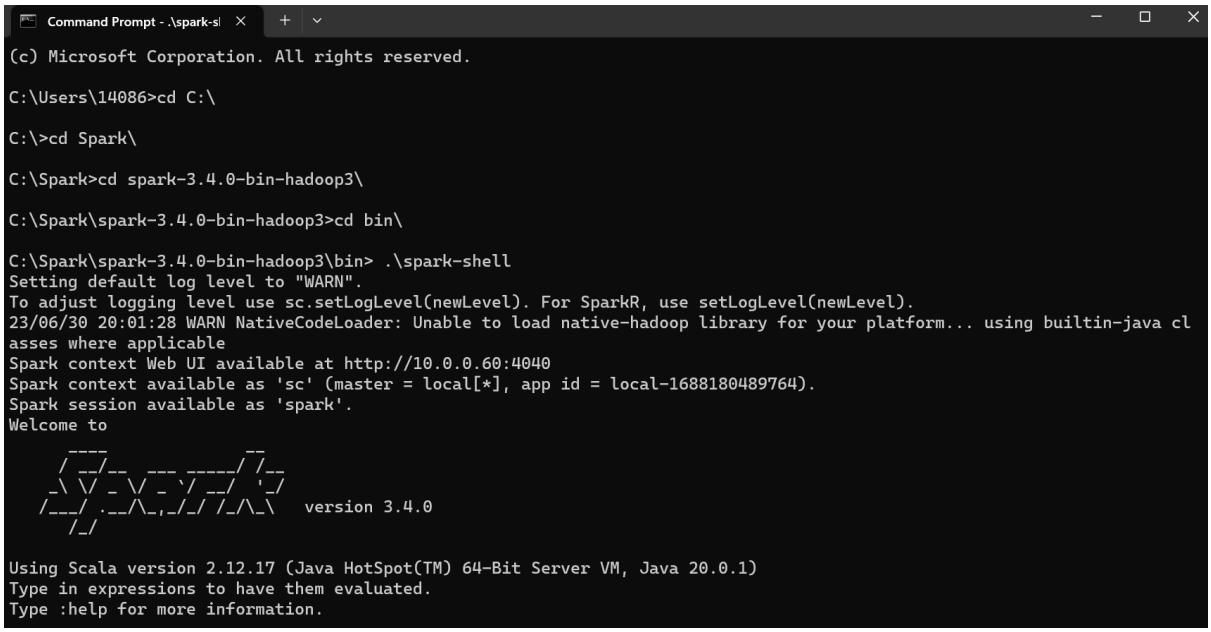
1. Installation of Java, Anaconda and Apache Spark







2. Verifying Spark Installation




```
scala> sc.version
res0: String = 3.4.0

scala> sc.appName
res1: String = Spark shell
```

3. Importing SparkContext and SparkContext Methods

The import statements allow us to access the functionalities provided by Apache Spark's SparkContext class and related functions. It provides methods for creating RDDs, performing transformations and actions on data, and managing the Spark cluster.

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

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

4. Loading Text File Data

```
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[1] at textFile at <console>:28
```

5. Caching the data

Caching an RDD means persisting it in memory to optimize data processing. Caching is particularly useful when we need to reuse the RDD multiple times in our Spark application.

```
scala> txtData.cache()
res2: txtData.type = README.md MapPartitionsRDD[1] at textFile at <console>:28
```

6. Counting the Number of Lines

The result of txtData.count() shows that there are 125 lines in the "README.md" file. Each line represents a separate element in the RDD, and this count indicates the total number of lines present in the text file.

```
scala> txtData.count()
res3: Long = 125
```

7. Word count analysis

The resulting wcData RDD contains a collection of key-value pairs, where each key represents a unique word from the text data, and the corresponding value denotes the count of that word in the text. For example, (package, 1) indicates that the word "package" appears once in the text.

The contents of the wcData are printed, and this output provides valuable insights into the frequency distribution of words in the text, enabling further analysis and understanding of the data.


Command Prompt - .\spark-sl
scala> val wcData = txtData.flatMap(l => l.split(" ")).map(word => (word, 1)).reduceByKey(_ + _)
wcData: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[4] at reduceByKey at <console>:27
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)
(graph,1)
(Hive,2)
(info,1)
(["Specifying,1)
("yarn",1)
([params]`.,1)
(Downloads](https://static.pepy.tech/personalized-badge/pyspark?period=month&units=international_system&left_color=black&right_color=

Questions

- **How many times is the word "Hadoop" counted when the tutorial has printed out all the word counts?**

  The result highlights that the word "Hadoop" occurs three times

```
scala> val hadoopCount = wcData.lookup("Hadoop").sum
hadoopCount: Int = 3

scala> println(s"The word 'Hadoop' is counted $hadoopCount times.")
The word 'Hadoop' is counted 3 times.
```

- **Which is the most common word used in the file? How many times does the word occur?**

  In the first step, we can see that the most common word is represented by an empty string "" with a frequency of 41 occurrences.

  However, in the second step, we excluded blank spaces and in this case, the most common word is "the" with a count of 23.

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

scala> val CommonWordnoblanks = wcData.filter{ case (word, count) => word.trim.nonEmpty }.reduce((a, b) => if (a._2 > b._2) a else b)
CommonWordnoblanks: (String, Int) = (the,23)
```

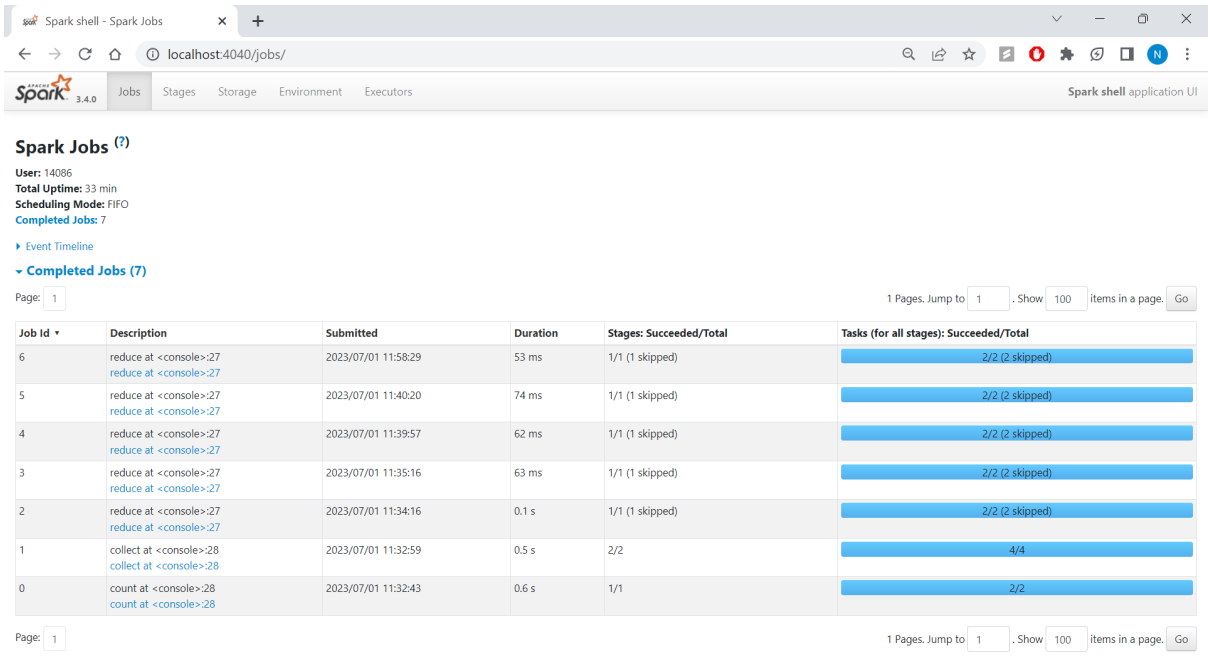
- **Which word occurs the fewest times? How many times does the word occur?**

  I could notice that there are many words that occur only 1 time like sample, find, package etc. But when we run the code, the result showed the word "must".

```
scala> val fewestOccurrenceWord = wcData.filter { case (word, count) => word.trim.nonEmpty } .reduce((a, b) => if (a._2 < b._2) a else b)
fewestOccurrenceWord: (String, Int) = (must,1)
```

- **Then look at the web console (http://<>:4040/jobs). How many seconds did it take to complete the word count job?**

We can connect to http://localhost:4040/ to examine the total time taken by the command to calculate the frequency of all words. The execution time is 0.5s. This allows us to assess the efficiency of the computation and monitor the performance of Spark.



REFERENCES

Penchikala, S. (2015, January 30). Big Data Processing with Apache Spark – Part 1: Introduction. *InfoQ*. https://www.infoq.com/articles/apache-spark-introduction/

RDD Programming Guide - Spark 3.4.1 Documentation. (n.d.). https://spark.apache.org/docs/latest/rdd-programming-guide.html

Downloads | Apache Spark. (n.d.-b). https://spark.apache.org/downloads.html

Download the Latest Java LTS Free. (n.d.). https://www.oracle.com/java/technologies/downloads/#jdk20-windows