
Group B

Assignment No: 3

Title of the Assignment: Write a simple program in SCALA using Apache Spark framework.

Objective of the Assignment: Students should be able to Write a simple program in SCALA using Apache Spark framework.

Prerequisites:

1. basic knowledge of Scala
2. basic knowledge of Java syntax
3. installation of Java
4. Operating System recommended: - 64-bit Open source Linux or its derivative/Windows

---

Contents for Theory:

1. **What is Scala?**
2. **Apache Spark**
3. **Steps to Install Scala & Apache Spark Framework Installation (Windows)**
4. **Steps to Install Scala & Apache Spark Framework Installation (Ubuntu)**
5. **Write a simple program in SCALA using Apache Spark framework.**

---

1. What is Scala

Scala is an acronym for "**Scalable Language**". It is a general-purpose programming language designed for the programmers who want to write programs in a concise, elegant, and type-safe way. Scala enables programmers to be more productive. Scala is developed as an object oriented and functional programming language.

If you write a code in Scala, you will see that the style is similar to a scripting language. Even though Scala is a new language, it has gained enough users and has a wide community support. It is one of the most user-friendly languages.

The design of Scala started in 2001 in the programming methods laboratory at EPFL (École Polytechnique Fédérale de Lausanne). Scala made its first public appearance in January 2004 on the JVM platform and a few months later in June 2004, it was released on the .(dot)NET platform. The .(dot)NET support of Scala was officially dropped in 2012.

A few more characteristics of Scala are:

1. Scala is pure Object-Oriented programming language
2. Scala is a functional language
3. Scala is a compiler based language (and not interpreted)

2. What is Apache Spark?

Apache Spark is an open source data processing framework for performing Big data analytics on distributed computing cluster.

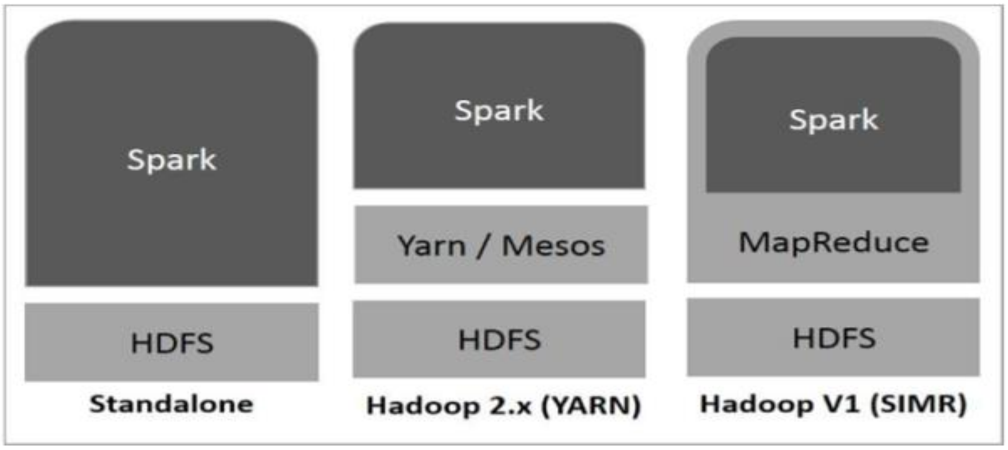
Spark was initially started by Matei Zaharia at UC Berkeley's AMPLab in 2009. It was an academic project in UC Berkley. Initially the idea was to build a cluster management tool, which can support different kind of cluster computing systems. The cluster management tool which was built as a result is Mesos. After Mesos was created, developers built a cluster computing framework on top of it, resulting in the creating of Spark.

Spark was meant to target interactive iterative computations like machine learning. In the year 2013, the spark project was passed on to the Apache Software Foundation.

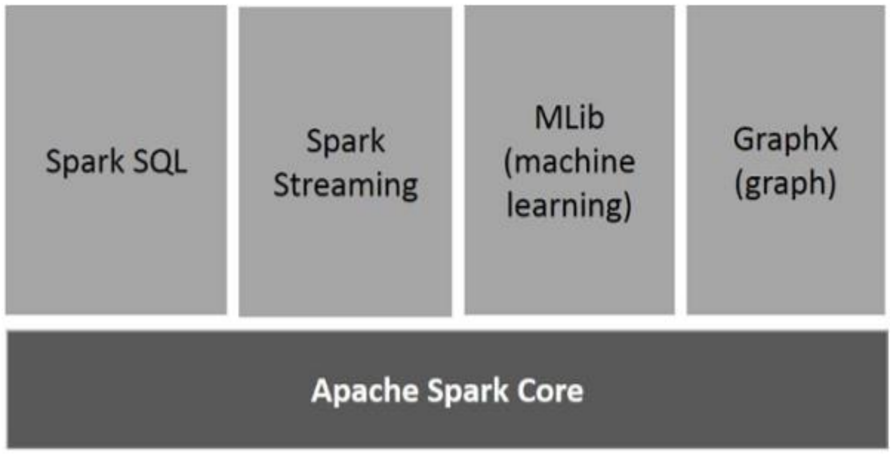
Apache Spark has other features, such as:

1. Supports wide variety of operations, compared to Map and Reduce functions.
2. Provides concise and consistent APIs in Scala, Java and Python.
3. Spark is written in Scala Programming Language and runs in JVM.
4. It currently has support in the following programming languages to develop applications in Spark: Scala, Java, Python, R
5. Features interactive shell for Scala and Python. This is not available in Java yet.
6. It leverages the distributed cluster memory for doing computations for increased speed and data processing.
7. Spark enables applications in Hadoop clusters to run up to as much as 100 times faster in memory and 10 times faster even when running in disk.
8. It is most suitable for real time decision making with big data.
9. It runs on top of existing Hadoop cluster and access Hadoop data store (HDFS), it can also process data stored by HBase structure. It can also run without Hadoop with apache Mesos or alone in standalone mode.
10. Apache Spark can be integrated with various data sources like SQL, NoSQL, S3, HDFS, local file system etc.
11. Good fit for iterative tasks like Machine Learning (ML) algorithms.
12. In addition to Map and Reduce operations, it supports SQL like queries, streaming data, machine learning and data processing in terms of graph.

Spark Built on Hadoop



Components of Spark



Apache Spark Core

Spark Core is the underlying general execution engine for spark platform that all other functionality is built upon. It provides In-Memory computing and referencing datasets in external storage systems.

Spark SQL

Spark SQL is a component on top of Spark Core that introduces a new data abstraction called SchemaRDD, which provides support for structured and semi-structured data.

Spark Streaming

Spark Streaming leverages Spark Core's fast scheduling capability to perform streaming analytics. It ingests data in mini-batches and performs RDD (Resilient Distributed Datasets) transformations on those mini-batches of data.

MLlib (Machine Learning Library)

MLlib is a distributed machine learning framework above Spark because of the distributed memory-based Spark architecture. It is, according to benchmarks, done by the MLlib developers against the

Alternating Least Squares (ALS) implementations. Spark MLlib is nine times as fast as the Hadoop disk-based version of Apache Mahout (before Mahout gained a Spark interface).

GraphX

GraphX is a distributed graph-processing framework on top of Spark. It provides an API for expressing graph computation that can model the user-defined graphs by using Pregel abstraction API. It also provides an optimized runtime for this abstraction.

3. Steps to Install Scala & Apache Spark Framework Installation (Windows)

Step 1: Java Installation

Java installation is one of the mandatory things in installing Spark. Try the following command to verify the JAVA version.

```
java -version
```

If Java is already, installed on your system, you get to see the following response −

```
java version "1.7.0_71"
Java(TM) SE Runtime Environment (build 1.7.0_71-b13)
Java HotSpot(TM) Client VM (build 25.0-b02, mixed mode)
```

In case you do not have Java installed on your system, then Install Java before proceeding to next step.

Step 2: Scala installation

You should Scala language to implement Spark. So let us verify Scala installation using following command.

```
scala –version
```

if Scala is already installed on your system, you get to see the following response –

```
Scala code runner version 2.11.6 -- Copyright 2002-2013, LAMP/EPFL
```

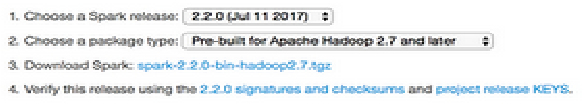
In case you don’t have Scala installed on your system, then proceed to next step for Scala installation.

Step 3: Apache Spark Download & Installation

Download the latest version of Scala by visit the following link:

https://spark.apache.org/downloads.html

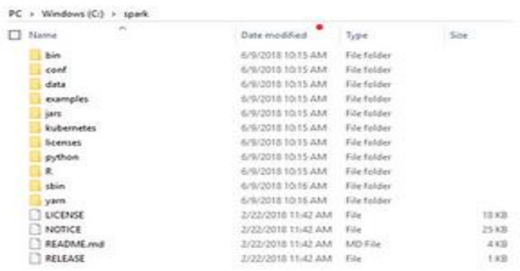
1. Download a pre-built version of Apache Spark from this link. Again, don't worry about the version, it might be different for you. Choose latest Spark release from drop down menu and package type as pre-built for Apache Hadoop.



1. Choose a Spark release: 2.2.0 (Jul 11 2017)
2. Choose a package type: Pre-built for Apache Hadoop 2.7 and later
3. Download Spark: spark-2.2.0-bin-hadoop2.7.tgz
4. Verify this release using the 2.2.0 signatures and checksums and project release KEYS.

2. If necessary, download and install WinRAR so that you can extract the .tgz file that you just downloaded.

3. Create a separate directory spark in C drive. Now extract Spark files using WinRAR, and copy its contents from downloads folder => C:\spark.



PC > Windows (C:) > spark

Name	Date modified	Type	Size
bin	6/9/2018 10:15 AM	File folder	
conf	6/9/2018 10:15 AM	File folder	
data	6/9/2018 10:15 AM	File folder	
examples	6/9/2018 10:15 AM	File folder	
jars	6/9/2018 10:15 AM	File folder	
kubernetes	6/9/2018 10:15 AM	File folder	
licenses	6/9/2018 10:15 AM	File folder	
python	6/9/2018 10:15 AM	File folder	
R	6/9/2018 10:15 AM	File folder	
sbin	6/9/2018 10:16 AM	File folder	
yarn	6/9/2018 10:16 AM	File folder	
LICENSE	2/22/2018 11:42 AM	File	18 KB
NOTICE	2/22/2018 11:42 AM	File	25 KB
README.md	2/22/2018 11:42 AM	MD File	4 KB
RELEASE	2/22/2018 11:42 AM	File	1 KB

Step 4: Configuring Windows Environment for Apache Spark

4. Make sure you "Hide file extension properties" in your file explorer (view tab) is unchecked. Now go to C:\spark\conf folder and rename log4j.properties.template file to log4j.properties.

You should see filename as log4j.properties and not just log4j.

5. Now open log4j.properties with word pad and change the statement log4j.rootCategory=INFO, console --> log4j.rootCategory=ERROR, console.

Save the file and exit, we did this change to capture ERROR messages only when we run Apache Spark, instead of capturing all INFO.

6. Now create C:\winutils\bin directory.

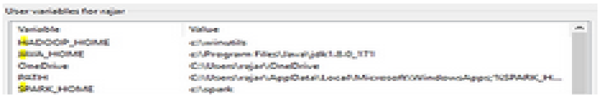
Download winutils.exe from GitHub and extract all the files. You will find multiple versions of Hadoop inside it, you just need to focus on Hadoop version which you selected while downloading package type pre-built Hadoop 2.x/3.x in **Step 1**.

Copy all the underlying files (all .dll, .exe etc) from Hadoop version folder and move it into C:\winutils\bin folder. This step is needed to make windows fool as we are running Hadoop. This location (C:\winutils\bin) will act as Hadoop home.

7. Now right-click your Windows menu, Select Control Panel --> System and Security --> System --> "Advanced System Settings" --> then click "Environment Variables" button.

Click on "New" button in User variables and add 3 variables:

1. SPARK_HOME c:\spark
2. JAVA_HOME (path you noted while JDK Installation Step 3, for example C:\Program Files\Java\jdk1.8.0_171)
3. HADOOP_HOME c:\winutils

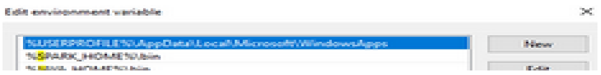


User variables for rajar

Variable	Value
HADOOP_HOME	c:\winutils
JAVA_HOME	c:\Program Files\Java\jdk1.8.0_171
OneDrive	C:\Users\rajar\OneDrive
PATH	C:\Users\rajar\AppData\Local\Microsoft\WindowsApps;%SPARK_H...
SPARK_HOME	c:\spark

8. Add the following 2 paths to your PATH user variable. Select "PATH" user variable and edit, if not present create new.

1. %SPARK_HOME%\bin
2. %JAVA_HOME%\bin



Edit environment variable

```
%USERPROFILE%\AppData\Local\Microsoft\WindowsApps
%SPARK_HOME%\bin
%JAVA_HOME%\bin
```

Step 5: Download & Install Scala IDE

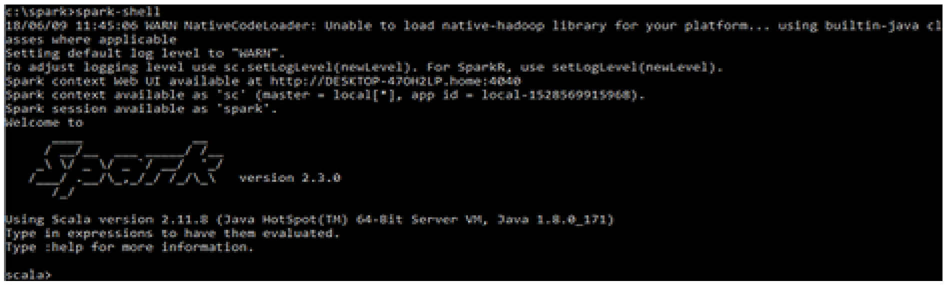
1. Now install the latest Scala IDE from here. I have installed Scala-SDK-4.7 on my machine. Download the zipped file and extract it. That's it.

2. Under Scala-SDK folder you will find eclipse folder, extract it to c:\eclipse.

Run eclipse.exe and it will open the IDE (we will use this later).

Step 6: Test the Environment

1. Open up a Windows command prompt in administrator mode. Right click on command prompt in search menu and run as admin.
2. Type java -version and hit Enter to check if Java is properly installed. If you see the Java version that means Java is installed properly.
3. Type cd c:\spark and hit Enter. Then type dir and hit Enter to get a directory listing.
4. Look for any text file, like README.md or CHANGES.txt.
5. Type spark-shell and hit Enter.
6. At this point you should have a scala> prompt as shown below. **If not,** double check the steps above, check the environment variables and after making change close the command prompt and retry again.
7. Type val rdd = sc.textFile("README.md") and hit Enter. Now type rdd.count() and hit Enter.
8. You should get a count of the number of lines from readme file! Congratulations, you just ran your first Spark program! We just created a rdd with readme text file and ran count action on it. Don't worry we will be going through this in detail in next sections.
9. Hit control-D to exit the spark shell, and close the console window.
10. You've got everything set up! Hooray!



```
c:\spark>spark-shell
18/06/09 11:45:06 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://DESKTOP-47OH2LP.home:4040
Spark context available as 'sc' (master = local[*], app id = local-1528569915968).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.3.0
      /_/

Using Scala version 2.11.8 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_171)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

Step 7: Choosing a development environment

Once you have installed Scala, there are various options for choosing an environment. Here are the 3 most common options:

- Terminal / Shell based
- Notepad / Editor based
- IDE (Integrated development environment)

Choosing right environment depends on your preference and use case. I personally prefer writing a program on shell because it provides a lot of good features like suggestions for method call and you can also run your code while writing line by line.

Step 8: Warming up: Running your first Scala program in Shell:

Write a simple program in SCALA using Apache Spark framework.

Let's write a first program which adds two numbers.

```
scala> var Var4 = 2
Output: Var4: Int = 2
scala> var Var5 = 3
Output: Var5: Int = 3
```

Now, let us apply some operations using operators in Scala.

Apply '+' operator

```
Var4+Var5
Output:
res1: Int = 5
```

Step 9: Writing & Running a program in Scala using an editor

Let us start with a "Hello World!" program. It is a good simple way to understand how to write, compile and run codes in Scala. No prizes for telling the outcome of this code!

```
object HelloWorld {
 def main(args: Array[String]) {
 println("Hello, world!")
 }
 }
```

As mentioned before, if you are familiar with Java, it will be easier for you to understand Scala. If you know Java, you can easily see that the structure of above "HelloWorld" program is very similar to Java program.

This program contains a method "main" (not returning any value) which takes an argument – a string array through command line. Next, it calls a predefined method called "Println" and passes the argument "Hello, world!".

You can define the main method as static in Java but in Scala, the static method is no longer available.

Step 10: Compile a Scala Program

To run any Scala program, you first need to compile it. “Scalac” is the compiler which takes source program as an argument and generates object files as output.

Let’s start compiling your “HelloWorld” program using the following steps:

1. For compiling it, you first need to paste this program into a text file then you need to save this program as HelloWorld.scala.
2. Now you need change your working directory to the directory where your program is saved.
3. After changing the directory you can compile the program by issuing the command.

   ```
   scalac HelloWorld.scala
   ```
4. After compiling, you will get Helloworld.class as an output in the same directory. If you can see the file, you have successfully compiled the above program.

Step 11: Compile a Scala Program

After compiling, you can now run the program using following command:

```
scala HelloWorld
```

You will get an output if the above command runs successfully.

The program will print “Hello, world!”

4. Steps to Install Scala & Apache Spark Framework Installation (Ubuntu)

1) Install Scala

Step 1) java -version

Step 2) Install **Scala** from the apt repository by running the following commands to search for scala and install it.

sudo apt search scala ⇒ Search for the package

sudo apt install scala ⇒ Install the package

Step 3) To verify the installation of **Scala**, run the following command.

scala -version

2) Apache Spark Framework Installation

Apache Spark is an open-source, distributed processing system used for **big data workloads**. It utilizes in-memory caching, and optimized query execution for fast analytic queries against data of any size.

Step 1) Now go to the official Apache Spark download page and grab the latest version (i.e. 3.2.1) at the time of writing this article. Alternatively, you can use the wget command to download the file directly in the terminal.

wget https://apachemirror.wuchna.com/spark/spark-3.2.1/spark-3.2.1-bin-hadoop2.7.tgz

Step 2) Extract the Apache Spark tar file.

tar -xvzf spark-3.1.1-bin-hadoop2.7.tgz

Step 3) Move the extracted **Spark** directory to **/opt** directory. sudo mv spark-3.1.1-bin-hadoop2.7 /opt/spark

3) Configure Environmental Variables for Spark

Step 4) Now you have to set a few environmental variables in **.profile** file before starting up the spark.

echo "export SPARK_HOME=/opt/spark" >> ~/.profile

echo "export PATH=$PATH:/opt/spark/bin:/opt/spark/sbin" >> ~/.profile

echo "export PYSPARK_PYTHON=/usr/bin/python3" >> ~/.profile

Step 5) To make sure that these new environment variables are reachable within the shell and available to Apache Spark, it is also mandatory to run the following command to take recent changes into effect. source ~/.profile

Step 6) ls -l /opt/spark

4) Start Apache Spark in Ubuntu

Step 7) Run the following command to start the **Spark** master service and slave service.

start-master.sh

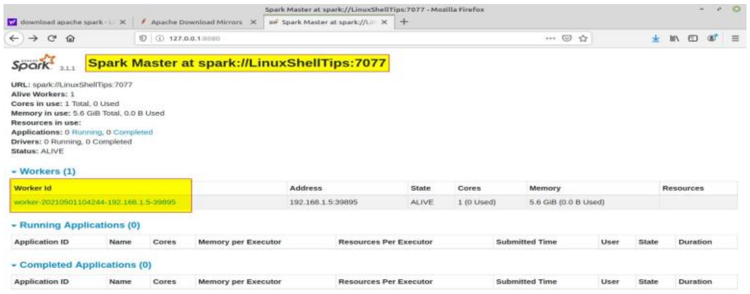
start-workers.sh spark://localhost:7077

(if workers not starting then remove and install openssh:

sudo apt-get remove openssh-client openssh-server

sudo apt-get install openssh-client openssh-server)

Step 8) Once the service is started go to the browser and type the following URL access spark page. From the page, you can see my master and slave service is started. http://localhost:8080/


Spark Master at spark://LinuxShellTips:7077 - Mozilla Firefox

127.0.0.1:8080

Spark 3.1.1 Spark Master at spark://LinuxShellTips:7077

URL: spark://LinuxShellTips:7077
Alive Workers: 1
Cores in use: 1 Total, 0 Used
Memory in use: 5.6 GiB Total, 0.0 B Used
Resources in use:
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (1)

Worker Id	Address	State	Cores	Memory	Resources
worker-20210501104244-192.168.1.5-39895	192.168.1.5:39895	ALIVE	1 (0 Used)	5.6 GiB (0.0 B Used)	

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

Step 9) You can also check if **spark-shell** works fine by launching the **spark-shell** command.

```
Spark-shell
```

Source Code:

/* Sample Code to print Statement */

```
object ExampleString {

def main(args: Array[String]) {

        //declare and assign string variable "text"
        val text : String = "You are reading SCALA programming language.";

        //print the value of string variable "text"

        println("Value of text is: " + text);
    }
}
```

/*The if-else expression in Scala*/

In Scala, if-else expression is used for conditional statements. You can write one or more conditions inside "if". Let's declare a variable called "Var3" with a value 1 and then compare "Var3" using if-else expression.

```
        var Var3 =1
        if (Var3 ==1){
         println("True")}else{
         println("False")}
        Output: True
```

/*Iteration in Scala*/

Like most languages, Scala also has a FOR-loop which is the most widely used method for iteration. It has a simple syntax too.

```
for( a <- 1 to 10){
 println( "Value of a: " + a );
 }
```

Output:

```
Value of a: 1
Value of a: 2
Value of a: 3
Value of a: 4
Value of a: 5
Value of a: 6
Value of a: 7
Value of a: 8
Value of a: 9
Value of a: 10
```

/Scala program to find a number is positive, negative or positive. */**

```
        object ExCheckNumber {
        def main(args: Array[String]) {
                /**declare a variable*/ var
                number= (-100);
                if(number==0){ println("number is
                zero"); }
                else if(number>0){
                println("number is positive"); } else{
                println("number is negative"); }
        }
        }
```

/*Scala program to print your name*/

```
object ExPrintName {
        def main(args: Array[String]) {
                println("My name is Mike!")
        }
        }
```

/Scala Program to find largest number among two numbers. */**

```
        object ExFindLargest {
        def main(args: Array[String]) {
        var number1=20; var number2=30; var x10;
        if( number1>number2){ println("Largest number is:" + number1);
        } else{ println("Largest number is:" + number2);
        }
        }
        }
```

Conclusion- In this way we have Written a simple program in SCALA using Apache Spark framework.

Assignment Questions

1. **Write down steps to install scala**.