SparkLab May 2015
An Introduction to

Spark & Scala

Apostolos N. Papadopoulos
Assistant Professor

Data Engineering Lab, Department of Informatics,
Aristotle University of Thessaloniki

Abstract
Welcome to SparkLab ! In this seminar, we are going to learn the very basics in cluster computing using the **Scala** programming language and the exciting **Spark** engine. Spark steadily becomes the state-of-the-art in cluster computing and big data processing and analytics due to the excellent support it provides for several domains such as: **SQL processing**, **Streaming**, **Machine Learning** and **Graphs**. In addition, Spark supports three programming languages: **Java**, **Scala** and **Python**. In this seminar, we are going to use the Scala programming language.

To get the best out of the seminar, you are advised to bring your laptops in class. Also, to participate in the hands-on experience, you will need to either install the required software locally or logon to a remote server using ssh. However, in order to re-execute the examples and to write and test your own code in home, it is better to use the first option, and use the second option only as a fallback.

1. Installation for Local Access

We assume that your are working with Linux and more specifically with Ubuntu. The examples have been tested with Ubuntu, CentOS and Mint. Most probably, everything will run smoothly in any other Linux distribution. The same applies for MacOS. Window users can consult the Apache Spark website for installation instructions. For the examples to work we are going to need the following software:

java jdk (7 or 8 is recommended)
scala version 2.10.4
sbt version 0.13.8
spark version 1.3.1

Java Installation (if you already have Java version 1.6 or later skip this section)

To install Java (if it is not already in your system), you can either use any guideline you will find in the web, or use the following steps for Java 8.

```
sudo add-apt-repository ppa:webupd8team/java
sudo apt-get update
sudo apt-get install oracle-java8-installer
```

In case you are not working with `apt`, use the most convenient way for you to install Java in your system.

Sbt, Scala and Spark Installation

In fact, these packages do not need “installation”. All you need is to download a tarfile and extract it. For uniformity, it is better to create a directory named `sparklab` in your home directory and put everything there. For the seminar, I am going to assume, that these packages are extracted in `/usr/local`.

Download `sbt` from the following URL:

```
http://www.scala-sbt.org/download.html
```

Extract the tarfile in `/usr/local` and you should now have `sbt` stored under `/usr/local/sbt-0.13.8`

Download Scala version 2.10.4 using the following URL

```
http://www.scala-lang.org/download/2.10.4.html
```

Again, extract the tarfile inside `/usr/local` to get a directory `/usr/local/scala-2.10.4`

Finally, download Spark version 1.3.1 using the following URL

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

Be sure to download the file that refers to “Pre-built for Hadoop 2.6 and later” and the file should be

```
spark-1.3.1-bin-hadoop2.6.tgz
```

Again, extract the tarfile inside `/usr/local` to get `/usr/local/spark-1.3.1-bin-hadoop2.6`

Note that in order to extract tarfiles inside `/usr/local` we should either have `root` permissions or use `sudo`.

The next step is to put some exports in the `.bashrc` file to update the `PATH` environmental variable. Based on the previous discussion, you should copy-paste the following lines in your `.bashrc` file which located in your home directory.

```
export PATH=$PATH:/usr/local/scala-2.10.4/bin
export PATH=$PATH:/usr/local/sbt-0.13.8/bin
export PATH=$PATH:/usr/local/spark-1.3.1-bin-hadoop2.6/bin
export PATH=$PATH:/usr/local/spark-1.3.1-bin-hadoop2.6/sbin
```

Then run the command

```
source ~/.bashrc
```

in order to activate the updates performed into `.bashrc`

(*Note: You can also extract the required software anywhere in your filesystem. Just remember to set the PATH variable accordingly*.)

Installation of the Examples

The examples tarfile can be downloaded from

```
http://delab.csd.auth.gr/~apostol/sparklab.html
```

Create a folder with the name `sparklab` and extract the examples tarfile inside the `sparklab` folder.

Now, you are ready to go!

2. Remote Access

In case you have a problem with the previous process, you can still run the demos by using a remote installation of spark. To be able to run the demos presented in class, you will need to login to a remote server using **`ssh`** (secure shell). Please follow these steps:

1. Login to the remote server that will be given to you during the seminar. Assume, that this is **`server.csd.auth.gr`** using the following command:

```
ssh username@server.csd.auth.gr
```

The servername, the username and the password will be known to you during the seminar.

(Note: Windows users can use the PuTTY ssh client that can be downloaded from: http://www.putty.org/)

Attention: please do not delete or change any file since this account will be shared by all participants !!!

2. After login, go to the **`sparklab`** folder by executing:

```
cd sparklab
```

3. Check the contents of the folder by executing:

```
ls -las
```

You should see the folder **`examples`**

4. Enter the **`examples`** folder by executing

```
cd examples
```

There you should see two folders, **`scala`** and **`spark.`**

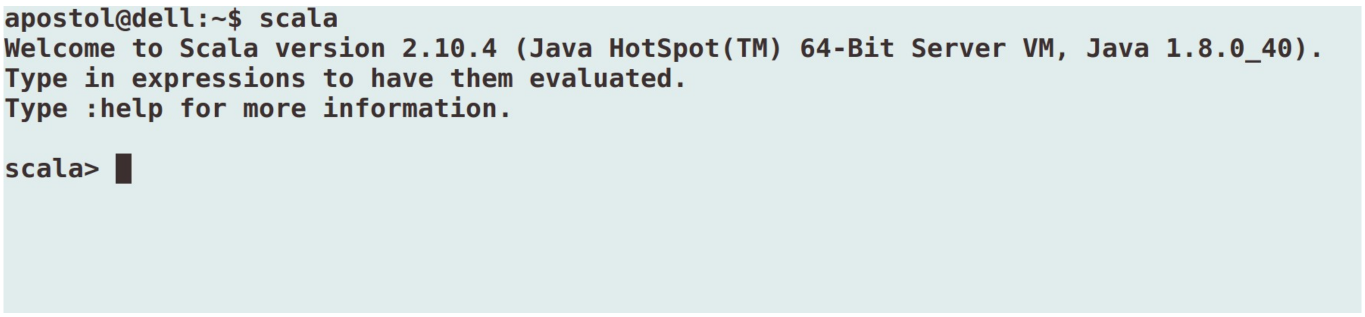
3. Running the Scala REPL

If you want to test some code in the Scala programming language, the easiest way is to use the Scala REPL (Read-Evaluate-Print Loop). This is an interactive tool which accepts Scala code and provides the result in an interactive fashion. To start the REPL issue the command:

```
scala
```

You will see the scala prompt waiting for commands. We are going to demonstrate the use of the REPL in class. You can exit the REPL by giving the command **:q**

The REPL looks like the following screenshot.



```
apostol@dell:~$ scala
Welcome to Scala version 2.10.4 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_40).
Type in expressions to have them evaluated.
Type :help for more information.

scala> 
```

4. Running Scala Examples

Compilation and code running will be performed by using the **Simple Build Tool** (sbt). First, we will talk about **Scala** so, write **`cd scala`** and inspect the contents of the folder by executing **`ls -las`**. You should see 12 folders: `01-HelloWorld`, `02-LineCount`, etc. Assume that you want to run the hello world example. Enter the folder `01-HelloWorld` by executing

```
cd 01-HelloWorld
```

(Note: use command completion using the TAB key for faster typing.)

Lets see the source code. Type the following command:

```
cat ./src/main/scala/HelloWorld.scala
```

*(Note: if you are familiar with **vi** editor then of course you can use it.)*

The result should be:

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

To compile the code

```
sbt compile
```

Run the code by typing:

```
sbt run
```

Use the same procedure to run the rest of Scala examples.

5. Running Spark Examples

Now we are going to test some **Spark** examples. The code is written in Scala but this time we are going to use the libraries provided by Spark in order to facilitate cluster computations. Assuming that you are in the `examples` folder, go into the `spark` folder by typing:

```
cd spark
```

You should see 7 folders named `01-HelloSpark` through `07-TriangleCounting`. Enter the hello spark folder by executing:

```
cd 01-HelloSpark
```

First, check the source code by typing:

```
cat ./src/main/scala/HelloSpark.scala
```

You should see the following Scala code:

```
import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._
import org.apache.spark.SparkConf

object HelloSpark {
  def main(args: Array[String]): Unit = {
    println("*************************************************")
    println("Hello, Spark!")
    println("*************************************************")
  }
}
```

To build the project execute the command:

```
    sbt package
```

Execute the code by calling the `spark-submit` command as follows:

```
spark-submit --class "HelloSpark" ./target/scala-2.10/hellospark_2.10-1.0.jar
```

Check the output on your screen. Use the same procedure to run the rest of Spark examples.

(*Note: For interactive use of Spark, you may execute the Spark Shell by issuing the command* ***`spark-shell`****. This is an interactive tool (just like the Scala REPL) with additional capabilities to use RDDs and write code for Spark. This is a nice option to test Spark, try some code and write small programs. For larger programs, it is better to write your code in an IDE, package the code and then execute it using* ***`spark-submit`****.*)

In case of problems, please contact me: papadopo@csd.auth.gr

Happy Sparking!