Spark Tutorial (Local Machine)

Junghoon Kang

1 Overview

In this tutorial, you will learn how to install Spark and run a simple Spark application on your local machine. As this tutorial simply concatenates parts of documents provided in Spark's main website (http://spark.apache.org), please refer to it for more information. Also, note that I am writing this tutorial based on Ubuntu, a Linux distribution. Other Linux distros and Mac OS users can follow this tutorial as the procedures are very similar. However, if you are a Windows user, I highly recommend creating a Ubuntu virtual machine using VMWare.

2 Installation

2.1 Ubuntu

1. Install Java (jdk-8u221).
   - If you already have Java 8 installed on your machine, you can skip this.
   - Download Java Development Kit of version jdk-8u221 (https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html). Spark runs on Java 8, so other versions of java may cause issues.
   - Follow the command lines below to untar the file.

     ```
     $ cd ~
     $ mkdir -p application/java
     $ mv jdk-8u221-linux-x64.tar.gz application/java
     $ tar -xvzf application/java/jdk-8u221-linux-x64.tar.gz
     ```

   ~/.zshrc

   - Add the following lines in `~/.bashrc`

     ```
     export APPLICATION_HOME=~/application
     export JAVA_HOME=$APPLICATION_HOME/java/jdk1.8.0_221
     export PATH=$JAVA_HOME/bin:$PATH
     ```

   - Test whether the Java installation is successful.

     ```
     $ source ~/.bashrc
     $ java -version
     java version "1.8.0_221"
     Java(TM) SE Runtime Environment (build 1.8.0_221-b02)
     Java HotSpot(TM) 64-Bit Server VM (build 25.73-b02, mixed mode)
     ```

2. Install SBT (v.1.3.2).

- Download the latest version of SBT (http://www.scala-sbt.org/).
- Follow the command lines below to untar the file.

```
$ cd ~
$ mkdir application/sbt
$ mv sbt-1.3.2.tgz application/sbt
$ tar -xvzf application/sbt/sbt-1.3.2.tgz
$ mv application/sbt/sbt application/sbt/sbt-1.3.2
```

- Add the following lines in `~/.bashrc`

```
export SBT_HOME=$APPLICATION_HOME/sbt/sbt-1.3.2
export PATH=$SBT_HOME/bin:$PATH
```

3. Install Spark (v.2.4.4).

- Download spark-2.4.4-bin-hadoop2.7.tgz, which is a prebuilt Spark for Hadoop 2.7 or later (http://spark.apache.org/).
- Follow the command lines below to untar the file.

```
$ cd ~
$ mkdir application/spark
$ mv spark-2.4.4-bin-hadoop2.7.tgz application/spark
$ tar -xvzf application/spark/spark-2.4.4-bin-hadoop2.7.tgz
```

- Add the following lines in `~/.bashrc`

```
export SPARK_HOME=$APPLICATION_HOME/spark/spark-2.4.4-bin-hadoop2.7
export PATH=$SPARK_HOME/bin:$PATH
```

- Try running Spark interactive shell, which is inside the `spark-2.4.4-bin-hadoop2.7/bin` directory, by typing:

```
$ spark-shell
```

2.2 Mac OS

Mac OS users can follow the same procedure above.
If you do not have `~/.bashrc` file, then you will need to create one.

3 Spark Application

3.1 Write

Open up a text editor and copy-paste the following code into the `WordCount.scala` file. This application simply counts the number of words in an input textfile.

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

object WordCount {
    def main(args: Array[String]) {
        val conf = new SparkConf()
            .setMaster("local[*]")
            .setAppName("WordCount")

        val sc = new SparkContext(conf)

        // Read in the input text file.
        // Then, for each line in the text file, apply remove_punctuation() function.
        val lines_rdd = sc
            .textFile("YOUR_INPUT_FILE.txt")  // pass in your own input file
            .map(remove_punctuation)

        // For each data (line string) in lines_rdd, split it into words.
        // Then, filter out empty strings.
        val words_rdd = lines_rdd
            .flatMap( line => line.split("\\s+") )
            .filter( word => word != "" )

        // For each data (word string) in words_rdd, create a (word,1) tuple.
        // Then, count the number of occurrences for each word.
        val wordcounts_rdd = words_rdd
            .map( word => (word, 1) )
            .reduceByKey( (a, b) => (a + b) )

        // Print the top 15 words which occurs the most.
        wordcounts_rdd
            .takeOrdered(10)(Ordering[Int].reverse.on(x => x._2))
            .foreach(println)
    }

    def remove_punctuation(line: String): String = {
        line.toLowerCase
            .replaceAll("""[\p{Punct}]""", " ")
            .replaceAll("""[^a-zA-Z]""", " ")
    }
}
```

You can find the details of each function under API Docs tab in the following website:
http://spark.apache.org/docs/latest/

3.2 Compile

As our application depends on the Spark API, we will include an sbt configuration file, `build.sbt`, which describes the dependencies of the application. Open up a text editor and copy-paste the following lines into the `build.sbt` file. Feel free to change scalaVersion, but Spark 2.4.4 may fail with other scala versions. Run spark-shell to see which scala it is using, and copy paste the number here. Scala 2.11.12 should work with Spark 2.4.4, so you don't need to change.

```
/* build.sbt */
name := "SparkApp"
version := "1.0"
scalaVersion := "2.11.12"
libraryDependencies += "org.apache.spark" %% "spark-core" % "2.4.4"
```

For `sbt` to work correctly, we will need to layout `WordCount.scala` and `build.sbt` files according to the typical directory structure. Your directory layout should look something like below when you type `find` command inside your application directory. Suppose your application directory is `~/sparkapp`. Then,

```
$ cd ~/sparkapp
$ find .
.
./build.sbt
./src
./src/main
./src/main/scala
./src/main/scala/WordCount.scala
```

Once that is in place, we can create a JAR package containing the application code.

build

```
$ cd ~/sparkapp
$ sbt package
...
[info] Packaging {..}/{..}/target/scala-2.11/sparkapp_2.11-1.0.jar ...
[info] Done packaging.
[success] Total time: ...
```

If you are a Mac user and having a compilation error, please read:
http://stackoverflow.com/questions/5748451/why-do-i-need-semicolons-after-these-imports
In short, do not use the native Mac text editor but use third-party text editor, such as eclipse, vim or emacs, in order to create/edit scala programs.

3.3 Run

Finally, we can run the application using `spark-submit` script inside `spark-2.4.4-bin-hadoop2.7/bin` directory.

```
$ spark-submit \
--class WordCount \
target/scala-2.11/sparkapp_2.11-1.0.jar
```

There are two things to note following the above command line. First, `YOUR_INPUT_FILE.txt` is a textfile you pass into the application. You should create one under the app root folder (e.g. `~/sparkapp`). Feel free to change the name. Second, backslashes are used to split a single line command into multiple lines. You can remove them and concatenate multiple lines into one.