Lab 13: First Steps with Apache Spark

Instructor: Vivek Sarkar, Eric Allen

Course wiki : `https://wiki.rice.edu/confluence/display/PARPROG/COMP322`

Staff Email : comp322-staff@mailman.rice.edu

Importants tips and links

edX site : `https://edge.edx.org/courses/RiceX/COMP322/1T2014R`

Piazza site : `https://piazza.com/rice/spring2015/comp322/home`

Java 8 Download : `https://jdk8.java.net/download.html`

Maven Download : `http://maven.apache.org/download.cgi`

IntelliJ IDEA : `http://www.jetbrains.com/idea/download/`

HJ-lib Jar File : `http://www.cs.rice.edu/~vs3/hjlib/code/maven-repo/edu/rice/hjlib-cooperative/0.1.5-SNAPSHOT/hjlib-cooperative-0.1.5-SNAPSHOT.jar`

HJ-lib API Documentation : `https://wiki.rice.edu/confluence/display/PARPROG/API+Documentation`

HelloWorld Project : `https://wiki.rice.edu/confluence/pages/viewpage.action?pageId=14433124`

1 Acknowledgements

This lab presents the Spark implementation of word count described at `http://spark.apache.org`.

2 Overview

The purpose of this lab is to walk through an installation of Apache Spark, execute a simple word count operation on your machine, and then modify word count so that only a selected set of words are counted.

Although Apache Spark is best applied to distributed computation, it can be be run in "local mode," where it will simply make use of the available cores on your computer. Using local mode, we can view Spark as another viable model of multicore parallel computing.

For the purposes of this lab, we will run in local mode. However, the commands you will execute are identical to what you would use to run a distributed computation on a cluster with Spark installed.

3 Installing Spark

To install Spark on your computer, perform the following steps:

- First ensure you have a JVM installed on your machine by typing `java -version` at a command line
- Go to `http://spark.apache.org` and click on "Download"
- Select release 1.3.0

- Select "Prebuilt for Hadoop 2.4 and later"
- Select "Direct download"
- Click on the Download Spark link

4 Unpacking Spark

- In your home directory, create a subdirectory `spark` in which to store Spark
- Move the downloaded file into this directory
- Unpack the file. At the command line, type `tar xvf spark-1.3.0-bin-hadoop2.4.tar`
- (Note, on Windows, you might have to install tar in order to unpack.)

5 Running Spark

- After unpacking the tar file, cd into the new directory you just created: `cd spark/spark-1.3.0-bin-hadoop2.4`
- You can now start an interactive session with Spark by calling the Spark Shell `./bin/spark-shell`

This will start a read-eval-print loop for Scala, similar to what you might have seen for many scripting languages, such as Python. You can try it out by evaluating some simple Scala expressions:

```
scala> 1 + 2
res0: Int = 3
```

As you type expressions at the prompt, the resulting values are displayed, along with new variables that they are bound to (in this case, `res0`), which you can use in subsequent expressions.

6 Interacting with Spark

In your interactive session, you can interact with Spark via the global "spark context" variable `sc`. Try this out by creating a simple RDD from the text in Spark's own README file:

```
scala> val textFile = sc.textFile("README.md")
```

You now have a handle on an RDD! The elements in the RDD correspond to the lines in the README file. We can find out the number of lines using `count`:

```
scala> textFile.count()
```

As shown in class, we can also use the map/reduce pattern to count the number of occurences of each word in the RDD:

```
scala> val wordCounts = textFile.flatMap(line => line.split(" ")).
                                 map(word => (word, 1)).
                                 reduceByKey((a, b) => a + b)
```

We can then view the result of running our word count via the `collect` operation:

```
scala> wordCounts.collect()
```

7 Selective Wordcount

Your task is to alter our map/reduce operation on `textFile` so that only words of length 5 are counted, and then display the counts of all (and only) words of length 5.

Hints:

- Scala syntax for `if` expressions is:

  ```
  if testExpr thenExpr else elseExpr
  ```

  An `if` expression can be used in any context that an expression can be used, and returns the value returned by whichever branch of the `if` expression is executed.

- The length of a string `s` in Scala can be found by using the method `s.length`
- The elements of a pair can be retrieved using the accessors `_0` and `_1`. For example:

  ```
  scala> (1,2)._1
  res2: Int = 1
  ```

8 Turning in your lab work

For each lab, you will need to turn in your work before leaving, as follows.

1. Show your work to an instructor or TA to get credit for this lab (as in COMP 215). Be prepared to explain the lab at a high level.
2. Unlike with previous labs, you are not required to submit code to the repository.