Introduction to Spark

October 11, 2017

1 Exercises

1. Change to the following folder

```
$ cd example-codes/spark-examples/
```

1.1 Word count

1. Check out the wordcount example in `src/main/scala/spark/examples/WordCount.scala` which should be like as follows,

```
object SimpleApp {
  def main(args: Array[String]) = {

    val conf = new SparkConf().setAppName("Wordcount Application")
    val sc = new SparkContext(conf)
    val textFile = sc.textFile("c:/tmp/input/wordcount/")
    val counts = textFile.flatMap(line => line.split(" "))
    .map((word:String) => (word, 1))
    .reduceByKey(_ + _)
    counts.saveAsTextFile("c:/tmp/output/wordcount/")
  }
}
```

   which loads the text files from the `input` folder. For each line found in the text files, we split the line by spaces into words. That would give us RDD of lists of words. The `flatMap` collapse the inner lists. For each word in the RDD, we associate it with the number `1`. Then we shuffle and group them by words. Finally, the `1`s are aggregated by `+`.

2. Before running the code, we need to make sure the folders are created and the data are copied into data folder.

```
$ mkdir c:\tmp\input\wordcount\
$ copy data\wordcount\TheCompleteSherlockHolmes.txt  c:\tmp\input\wordcount\
$ rmdir /s c:\tmp\output\wordcount\
```

3. To compile the code

   ```
   $ sbt package
   ```

   If you are behind a proxy firewall,

4. To execute the code

   ```
   $ spark-submit  --class SimpleApp target/scala-2.11/spark-examples_2.11-0.1.0.jar
   ```

   We migtht encounter some error complaining

   ```
   17/10/10 14:31:11 WARN SparkEnv: Exception while deleting Spark temp
   dir: C:\Users\Kenny Lu\AppData\Local\Temp\spark-da6f0428...
   ```

   This is because spark is not having write access to your user account temp folder, we can ignore it, because the error shows up after the spark job terminates successfully.

5. To observe the output

   ```
   $ dir c:\tmp\output\wordcount\
   ```

1.2 Transformation

We consider an example of transforming data using Spark

1. Check out the example in `src/main/scala/spark/examples/Transform.scala`

   ```
   object Transform {
     def main(args: Array[String]) = {
       val conf = new SparkConf().setAppName("ETL (Transform) Example")
       val sc   = new SparkContext(conf)
       // load the file
       val input:RDD[String] = sc.textFile("c:/tmp/input/transform/")
       // split by spaces
       val tokenizeds:RDD[Array[String]] = input.map(line => line.split(" "))
       tokenizeds.cache()

       // process all the ones
       val ones = tokenizeds
       .filter(tokenized => tokenized(0) == "1")
       .map(tokenized => {
         val x = (tokenized(1).split(":"))(1)
         val y = (tokenized(2).split(":"))(1)
         List(x,y).mkString("\t")
       })
       ones.saveAsTextFile("c:/tmp/output/transform/ones")
   ```

```
      val zeros = tokenizeds
      .filter(tokenized => tokenized(0) == "0")
      .map(tokenized => {
        val x = (tokenized(1).split(":"))(1)
        val y = (tokenized(2).split(":"))(1)
        List(x,y).mkString("\t")
      })
      zeros.saveAsTextFile("c:/tmp/output/transform/zeros")
    }
  }
```

The above program parses and transform a data file in the format of

```
<label> 0:<x-value> 1:<y-value>
...
<label> 0:<x-value> 1:<y-value>
```

into two output files where in the `ones` we find all the rows with label equals to `1` and in the `zeros` we find all the rows with label `0`. The output files are in the format of

```
<x-value>    <y-value>
...
<x-value>    <y-value>
```

For instance, given the input file as

```
1 0:102 1:230
0 0:123 1:56
0 0:22  1:2
1 0:74 1:102
```

The output files in `ones` will be as

```
102     230
74      102
```

and those in `zeros` will be as

```
123     56
22      2
```

2. Before running the code, we need to make sure the folders are created and the data are copied into data folder

```
$ mkdir c:\tmp\input\transform\
$ copy data\transform\input.txt  c:\tmp\input\transform\
$ rmdir /s c:\tmp\output\transform\
```

3. To compile

```
$ sbt package
```

4. To execute the code

```
$ spark-submit  --class Transform target/scala-2.11/spark-examples_2.11-0.1.0.jar
```

5. To observe the output

```
$ dir c:\tmp\output\transform\
```

1.3 Extraction

In this section, we consider an example of extract the US addresses from an input file. The input file contains lines of text which could be addresses. If a US address is found in the line, the line and a "Y" is appended to the output, otherise the line and an "N" is appended to the output. To extract the US address we use a regular expression.

1. Consider the source code in `src/main/scala/spark/examples/Extract.scala`

```
object Extract {
  val opat = compile("^(.*) ([A-Za-z]{2}) ([0-9]{5})(-[0-9]{4})?$")
  def main(args: Array[String]) = {

    opat match
    {
      case None    => println("Pattern compilation error." )
      case Some(p) =>
      {
        val conf = new SparkConf().setAppName("ETL (Extract) Example")
        val sc   = new SparkContext(conf)
        // load the file
        val input:RDD[String] = sc.textFile("c:/tmp/input/extract/")

        val extracted = input.map(l => {
          exec(p,l.trim) match
          {
            case Some(env) => List(l,"Y").mkString("\t")
            case None => List(l,"N").mkString("\t")
          }
        })
```

```
          extracted.saveAsTextFile(s"c:/tmp/output/extract/")

        }
      }
    }
```

2. Before running the code, we need to make sure the folders are created and the data are copied into data folder

   ```
   $ mkdir c:\tmp\input\extract\
   $ copy data\extract\input.txt  c:\tmp\input\extract\
   $ rmdir /s c:\tmp\output\extract\
   ```

3. To compile

   ```
   $ sbt assembly
   ```

   Note that in case you are behind firewall, you need to modify `sbtconfig.txt` (In Windows, `C:/Program Files (x86)/sbt/config/sbtconfig.txt` with the following

   ```
   -Dhttp.proxyHost=proxy.hs-karlsruhe.de
   -Dhttp.proxyPort=8888
   -Dhttp.proxyUser=username
   -Dhttp.proxyPassword=password
   -Dhttps.proxyHost=proxy.hs-karlsruhe.de
   -Dhttps.proxyPort=8888
   -Dhttps.proxyUser=username
   -Dhttps.proxyPassword=password
   ```

   To minize the network traffic, you can use the `sbt-cache.zip` that I pass around in the USB drive. Unzip the folder and overwrite the folders `.ivy2` and `.sbt` in `C:/Users/<User Name>/`

   Note that we use `assembly` instead of `package` due to the need of external libraries such as `scala-pderiv` which is an efficient regular expression matching library based on partial derivative. To ensure that the library is distributed to all the workers, we have to bundle all the depended jars into the main jar.

   For some reason, the `assembly` command failed in windows due to the default JVM heap size.

   Hence we need to adjust the heap size in `sbtconfig.txt`

   ```
   -Xmx1024M
   ```

   Alterantively, we could use the pre-compiled jar found here

```
https://mega.nz/#!OgBWUCJI!Th9cAxQviw4VzSLB5bTsV1Z7W55cqI4DndchAYnjoVY
```

4. To execute

```
$ spark-submit --class Extract target/scala-2.11/spark-examples-assembly-0.1.0.jar
```

5. To observe the output

```
$ dir c:\tmp\output\extract\
```