

DISTRIBUTED COMPUTING: APACHE SPARK WITH SCALA API

Lab Work

Spark Console and the Basic Operations with the Scala API to Process Datasets Stored in a HDFS Repository

February 3rd, 2018

Jeremy Williams

Problem Statement

The goal of this lab work is to find out the minimum temperature in cloudera environment using spark-shell based on the dataset provided. Once you load the scala file into memory, scala class is available for execution. Now you can invoke the main method with null parameter to execute the processing and the output will finally be displayed in the console.

Approach to Solution

The source code was written in Scala. A spark-shell was used to load and run the code. The code has the business logics that is used in spark processing method like map, reduce, reduceByKey etc.

Solution Description

As the code is being run from spark-shell, it is built in spark context via spark shell; that's available. The Execution logic is to load the data file into spark memory first.

Then find-out and filter-out only the valid data which are passing the validation criteria.

Once the valid and filter dataset is available, the dataset will consist of year and temperature tuples.

Now the reduction mechanism is applied through 'reduceByKey' mechanism.

This basically calculates the minimum value of each year entry.

Data File

weather.txt

Sample data

```
0029029070999991901010106004+64333+023450FM-  
12+000599999V0202701N015919999999N0000001N9-  
00781+99999102001ADDGF1089919999999999999999
```

00290290709999991901010113004+64333+023450FM-
12+000599999V0202901N008219999999N0000001N9-
00721+99999102001ADDGF104991999999999999999999

00290290709999991901010120004+64333+023450FM-
12+000599999V0209991C000019999999N0000001N9-
00941+99999102001ADDGF108991999999999999999999

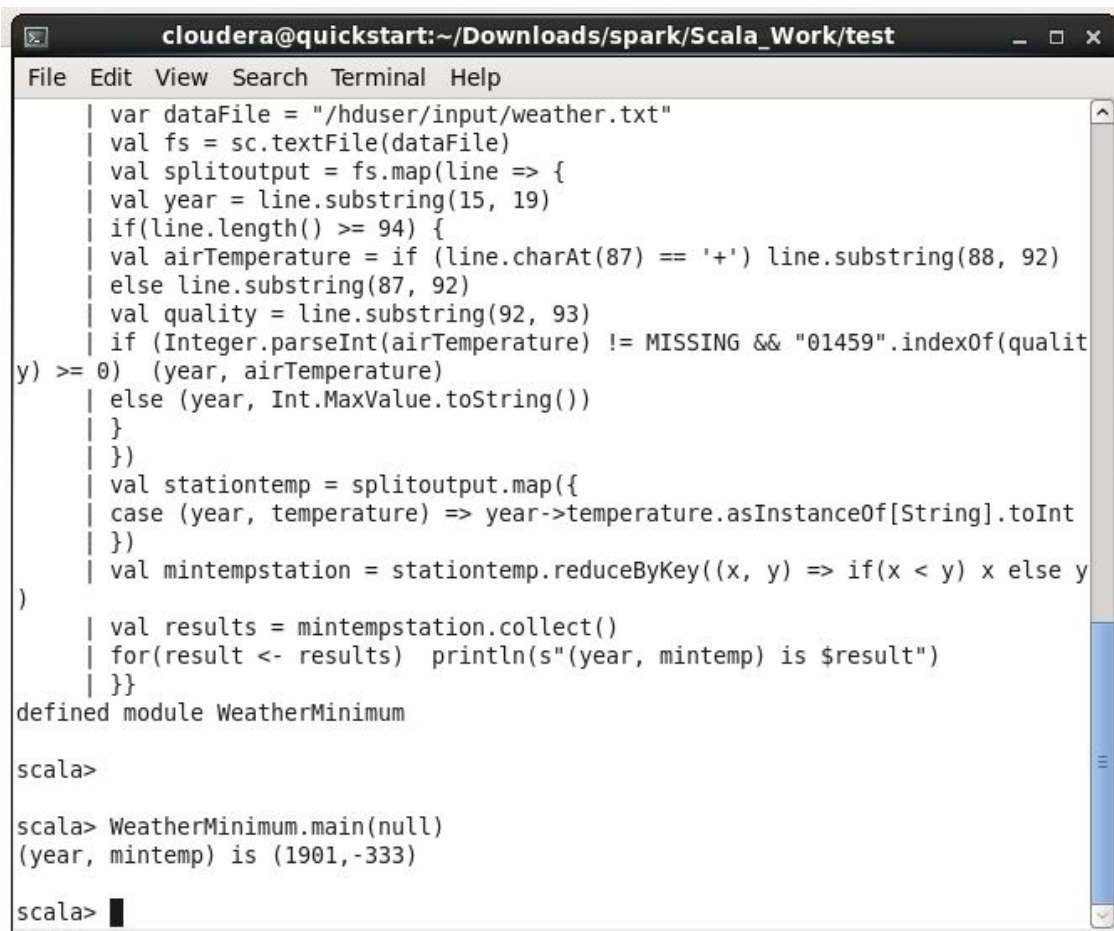
00290290709999991901010206004+64333+023450FM-
12+000599999V0201801N008219999999N0000001N9-
00611+99999101831ADDGF108991999999999999999999

00290290709999991901010213004+64333+023450FM-
12+000599999V0201801N009819999999N0000001N9-
00561+99999101761ADDGF108991999999999999999999

Final Code (via to-do list)

```
2 spark-shell
3
4 import org.apache.spark._
5 import org.apache.spark.SparkContext._
6 import org.apache.log4j._
7 import scala.math.min
8 import org.apache.spark.rdd.RDD.rddToPairRDDFunctions
9
10 object WeatherMinimum extends Serializable {
11   val MISSING = 9999
12   def main(args:Array[String]) {
13     var dataFile = "/hduser/input/weather.txt"
14     val fs = sc.textFile(dataFile)
15     val splitoutput = fs.map(line => {
16       val year = line.substring(15, 19)
17       if(line.length() >= 94) {
18         val airTemperature = if (line.charAt(87) == '+') line.substring(88, 92)
19         else line.substring(87, 92)
20         val quality = line.substring(92, 93)
21         if (Integer.parseInt(airTemperature) != MISSING && "01459".indexOf(quality) >= 0)
22           (year, airTemperature)
23         else (year, Int.MaxValue.toString())
24       }
25     })
26     val stationtemp = splitoutput.map({
27       case (year, temperature) => year->temperature.asInstanceOf[String].toInt
28     })
29     val mintempstation = stationtemp.reduceByKey((x, y) => if(x < y) x else y)
30     val results = mintempstation.collect()
31     for(result <- results) println(s"(year, mintemp) is $result")
32   }
33
34   WeatherMinimum.main(null)
```

Results/Output

A screenshot of a terminal window titled "cloudera@quickstart:~/Downloads/spark/Scala_Work/test". The terminal shows a Scala script being executed. The script defines a module "WeatherMinimum" with a "main" method. It reads a file "/hduser/input/weather.txt", processes its content to find the minimum temperature for each year, and prints the result. The output shows that for the year 1901, the minimum temperature is -333.

```
cloudera@quickstart:~/Downloads/spark/Scala_Work/test
File Edit View Search Terminal Help
| var dataFile = "/hduser/input/weather.txt"
| val fs = sc.textFile(dataFile)
| val splitoutput = fs.map(line => {
| val year = line.substring(15, 19)
| if(line.length() >= 94) {
| val airTemperature = if (line.charAt(87) == '+') line.substring(88, 92)
| else line.substring(87, 92)
| val quality = line.substring(92, 93)
| if (Integer.parseInt(airTemperature) != MISSING && "01459".indexOf(qualit
y) >= 0) (year, airTemperature)
| else (year, Int.MaxValue.toString())
| }
| })
| val stationtemp = splitoutput.map({
| case (year, temperature) => year->temperature.asInstanceOf[String].toInt
| })
| val mintempstation = stationtemp.reduceByKey((x, y) => if(x < y) x else y
)
| val results = mintempstation.collect()
| for(result <- results) println(s"(year, mintemp) is $result")
| }}
defined module WeatherMinimum

scala>

scala> WeatherMinimum.main(null)
(year, mintemp) is (1901,-333)

scala> █
```

Year \implies 1901

Minimum Temperature \implies -333