Spark

*Use scala-shell for all questions except for the question 9. For the question 9, use spark-submit command to execute the Scala class. Provide code comments wherever applicable.*

**1.** Consider two RDD dataset: RDD1 has number 1 to 10, RDD2 has number 5 to 10

Combine the two dataset, remove duplicate and find maximum number.

**Hint:** Use distint

val input1= sc.parallelize(List(1, 2, 3, 4, 5, 6, 7, 8, 9, 10))

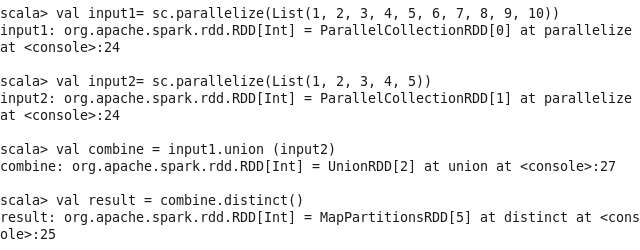
val input2= sc.parallelize(List(1, 2, 3, 4, 5)) // create datasets

val combine = input1.union (input2) //combine the two datasets

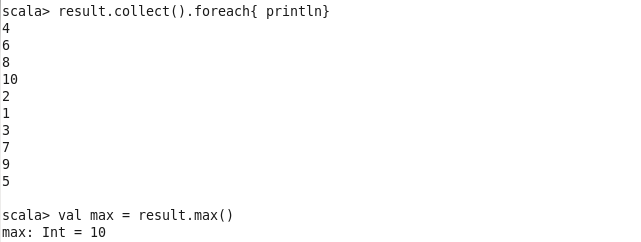
val result = combine.distinct() //remove duplicate

val max = result.max() //find maximun number

result: 10



Output:



**2.** Read a text file into a RDD. Filter and show all the sentences where the line **does not** contain a word as “ERROR”. Count number of lines which **does not** contain a word as “ERROR.

**Hint:** Use filter. For count see the Spark class notes “with Accumulator” example.

Input:

/tmp/task2.txt:

ERROR HUI

ERROR QI

CORRECT FEI

CORRECT QIAN

val input = sc.textFile("/tmp/task2.txt") // create dataset from txt file

val linesWithoutERROR = input.filter(line => !(line.contains("ERROR"))) // filter

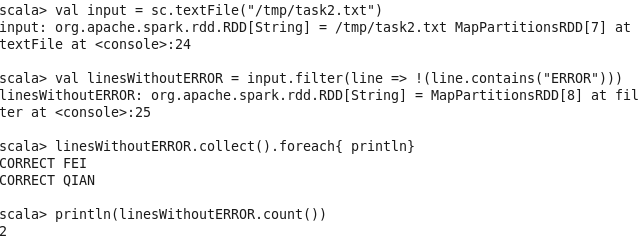
linesWithoutERROR.collect().foreach{ println} // print specific sentences

println(linesWithoutERROR.count()) // count numebr

Result:

CORRECT FEI

CORRECT QIAN



**3.** Use the RDD operation to break a word into letters. Each letter should be separated by comma as delimiter in the output. Print only the first 5 characters. **Hint:** use flatMap

Input:

/tmp/task3.txt:

SPARKCLASS

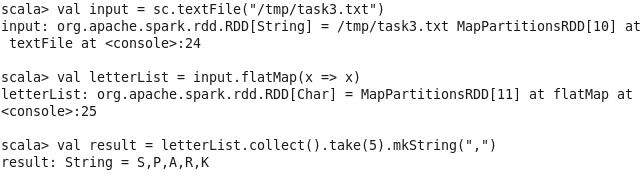
val input = sc.textFile("/tmp/task3.txt")

val letterList = input.flatMap(x => x) // map a word to letters

val result = letterList.collect().take(5).mkString(",") // make string with first 5 characters separated by //comma as delimiter

Result:

S,P,A,R,K



**4.** Read a file file /tmp/person.txt contains firstname lastname as showsn below: Xi Jinping

Hu Jintao

Xi Zemin

Hu Sangkun

Load the entire file into RDD. Count how many person with same first names. For example, how many Xi and how many Hu are there.

**Hint:** Load first name last name into key value pairs. Group by first name.

Input:

/tmp/person.txt

val person = sc.textFile("/tmp/person.txt")

val namePair = person.map(x => (x.split(" ")(0), x.split(" ")(1))) //make key-value pairs (firstname, lastname)

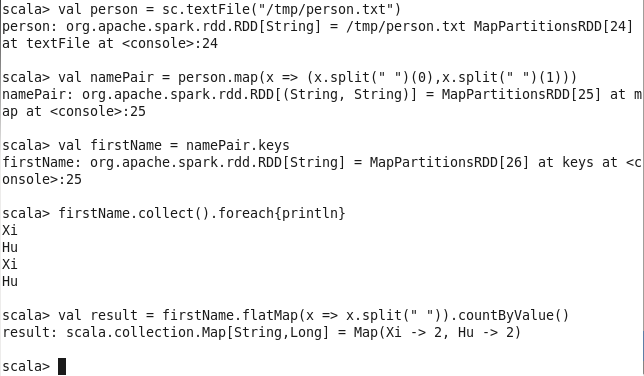
val firstName = namePair.keys // create RDD with all keys

val result = firstName.flatMap(x => x.split(" ")).countByValue() // group by key

Result:

Xi -> 2

Hu -> 2



**5.** Take the above input file /tmp/person.txt. Sort the file based on the first name. Print only the first record after sorting.

Input:

/tmp/person.txt

val person = sc.textFile("/tmp/person.txt")

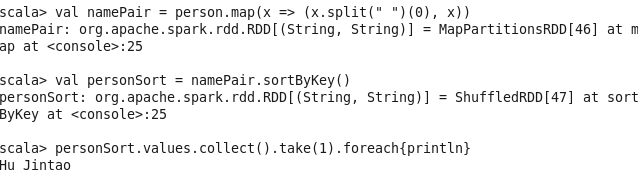
val namePair = person.map(x => (x.split(" ")(0), x)) // make key-value pairs (firstname, fullname)

val personSort = namePair.sortByKey() // Sort pairs by firstname

personSort.values.collect().take(1).foreach{println} // Print fullname of first one

Result:

Hu Jintao

Output:

**6.** Take the above input file /tmp/person.txt. Convert all the first names converted to uppercase and print only first names (do not print last names).

`

Input: /tmp/person.txt

val person = sc.textFile("/tmp/person.txt")

val upperPerson = person.map(x => (x.split(" ")(0).toUpperCase,x.split(" ")(1)))

// make key-value pairs (firstname in uppercase, fullname)

upperPerson.keys.collect().foreach{println}

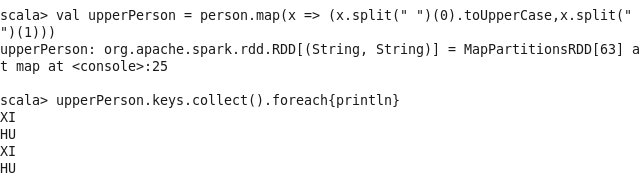
Output:

XI

HU

XI

HU



**7.** Take the above input file /tmp/person.txt. Search for first name Xi and print all people with same first name Xi and search result would be Xi Jinping and Xi Zemin.

**Hint:** Use lookup

Input: /tmp/person.txt

val person = sc.textFile("/tmp/person.txt")

val namePair = person.map(x => (x.split(" ")(0),x))

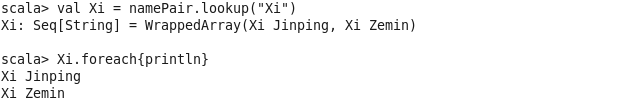
val Xi = namePair.lookup("Xi") // query the records with firstname == “Xi”

Xi.foreach{println}

Output:

Xi Jinping

Xi Zemin



**8.** Take the above input file /tmp/person.txt. Also, the second input file /tmp/person1.txt as follows:

Mao Zedong

Xi Zedong

Load two text files into two RDDs. Show the person with common first name in both the files

(i.e. result is **Xi**).

Input:

/tmp/person.txt, /tmp/person1.txt

val person = sc.textFile("/tmp/person.txt")

val person1 = sc.textFile("/tmp/person1.txt")

val namePair = person.map(x => (x.split(" ")(0),x))

val namePair1 = person1.map(x => (x.split(" ")(0),x))

val firstName = namePair.keys

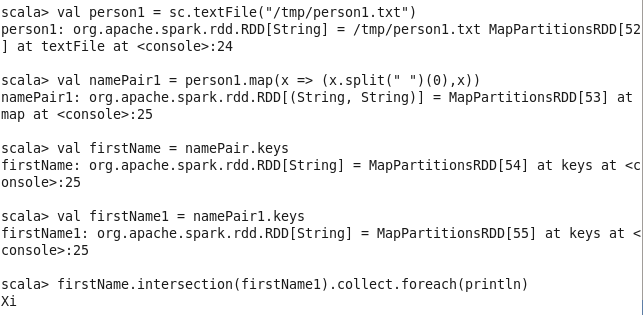
val firstName1 = namePair1.keys

firstName.intersection(firstName1).collect.foreach(println)

// Show the person with common first name in both RDD

Output:

Xi



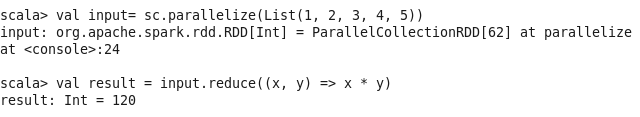
**9.** Take values from 1 to 5 into a RDD. Multiply all the numbers (i.e. 1×2×3×4×5). Persist the result

RDD into disk.

**Hint:** Use reduce() to multiply then convert the result into RDD then use persist() to store RDD

val input= sc.parallelize(List(1, 2, 3, 4, 5))

val result = input.reduce((x, y) => x \* y) // Multiply all the numbers in RDD



**10.** Write the above application (Question 8) into a Scala class file. Read the input from HDFS.

Provide spark submit command to execute the class in a Spark cluster. User should provide two

HDFS paths as two parameters in the spark-submit command line.

Code in scala Eclipse:

Task10.scala:

import org.apache.spark.\_

import org.apache.spark.SparkContext.\_

object task10 {

def main(args: Array[String]) {

val input = args(0)

val input1 = args(1)

val conf = new SparkConf().setAppName("reduce")

// Create a Scala Spark Context.

val sc = new SparkContext(conf)

val person = sc.textFile(input)

val person1 = sc.textFile(input1)

val namePair = person.map(x => (x.split(" ")(0),x))

val namePair1 = person1.map(x => (x.split(" ")(0),x))

val firstName = namePair.keys

val firstName1 = namePair1.keys

firstName.intersection(firstName1).collect.foreach(println)

}

}

Code in command terminal:

$cd /tmp/bigdata/

$sbt clean package

//

$spark-submit --class task10 --master local --deploy-mode client

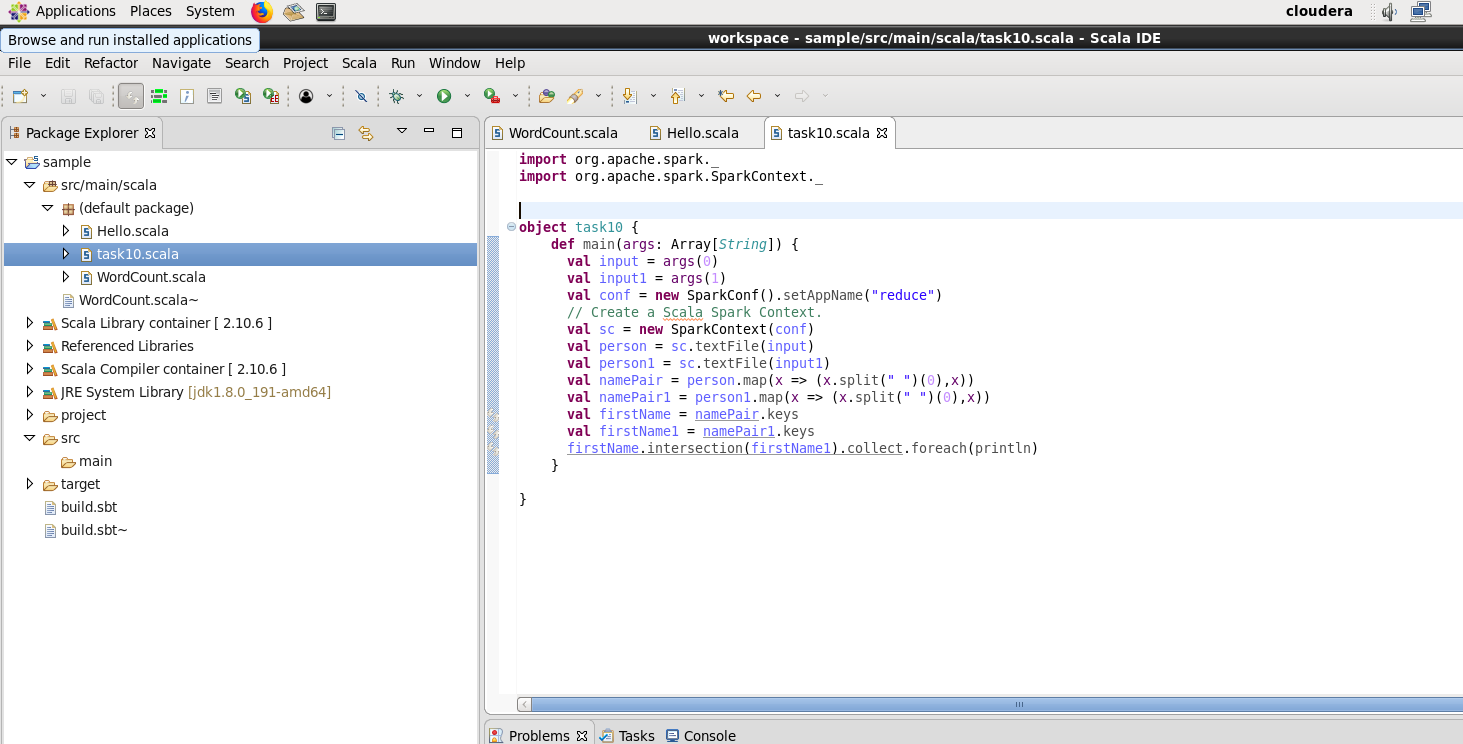
/tmp/bigdata/target/scala-2.10/sample\_2.10-0.1.0-SNAPSHOT.jar /tmp/person.txt /tmp/person1.txt

Output:

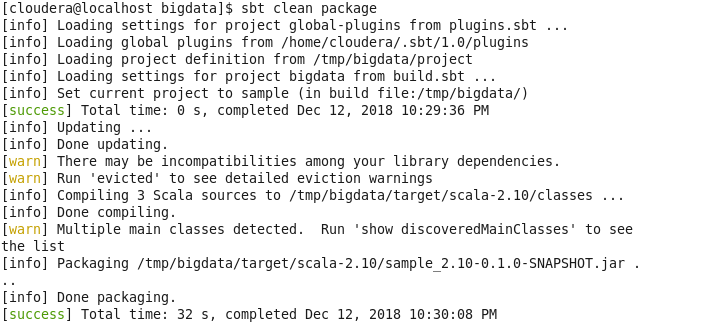
Xi

ScreenShot:

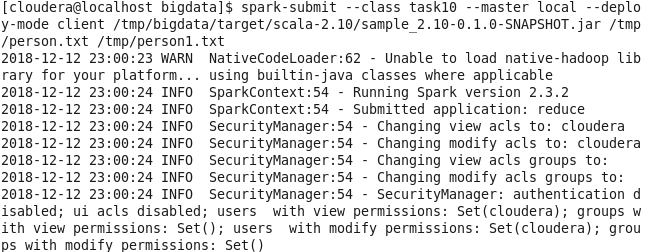
In scala eclipse:



In spark-shell, prepare the jar:



Start calculate:



…

…

