Lab 7: Introduction to Scala

Objectives

- Introduce Scala as a programming language
- Familiarize students with basic Scala syntax and concepts
- Apply Scala in the context of big data processing using Spark

1. Introduction to Scala

Scala is a modern multi-paradigm programming language designed to express common programming patterns in a concise, elegant, and type-safe way. It combines features of both object-oriented and functional programming.

Features:

- Statically typed
- Interoperable with Java
- Supports both object-oriented and functional programming
- Concise syntax

2. Setting Up the Environment

Installing Scala

Follow the official installation guide to install Scala on your machine.

Download from archives:

https://www.scala-lang.org/files/archive/

Then

```
sudo apt install ./scala-x.x.x.deb
```

The IntelliJ IDEA is another option to setup scala.

Running a Simple Scala Script

Create a file named `HelloWorld.scala`:

```scala
object HelloWorld {
  def main(args: Array[String]): Unit = {
    println("Hello, world!")
  }
}
```

Compile and run:

```
scalac HelloWorld.scala
scala HelloWorld
```

3. Basic Syntax and Data Structures

Variables and Data Types

```
val immutableVar: Int = 10 // Immutable variable
var mutableVar: String = "Hello" // Mutable variable
mutableVar = "World"
```

Control Structures

```
// if-else
val x = 10
if (x > 0) {
  println("Positive")
} else {
  println("Non-positive")
}

// for loop
for (i <- 1 to 5) {
  println(i)
}

// while loop
var i = 5
while (i > 0) {
  println(i)
  i -= 1
}
```

Functions and Methods

```
def add(a: Int, b: Int): Int = {
  a + b
}

println(add(2, 3))
```

Collections

```
val numbers = List(1, 2, 3, 4, 5)
val fruits = Set("Apple", "Banana", "Orange")
val map = Map("Alice" -> 25, "Bob" -> 30)

println(numbers)
println(fruits)
println(map)
```

4. Object-Oriented Programming in Scala

Classes and Objects

```
class Person(val name: String, var age: Int) {
  def greet(): String = s"Hello, my name is $name and I am $age years old."
}

val john = new Person("John", 30)
println(john.greet())
```

Traits and Mixins

```
trait Greeter {
  def greet(name: String): Unit = println(s"Hello, $name!")
}

class Person(name: String) extends Greeter {
  def sayHello(): Unit = greet(name)
}

val alice = new Person("Alice")
alice.sayHello()
```

Case Classes

```
case class Point(x: Int, y: Int)

val p1 = Point(1, 2)
val p2 = Point(1, 2)
println(p1 == p2) // true
```

5. Functional Programming in Scala

Immutability and Pure Functions

```
val list = List(1, 2, 3, 4, 5)
val doubledList = list.map(_ * 2)
println(doubledList)
```

Higher-Order Functions

```
def applyFunction(f: Int => Int, x: Int): Int = f(x)
val result = applyFunction(x => x * 2, 5)
println(result)
```

Anonymous Functions (Lambdas)

```
val add = (a: Int, b: Int) => a + b
println(add(3, 4))
```

Pattern Matching

```
val number = 10
number match {
  case 1 => println("One")
  case 2 => println("Two")
  case _ => println("Other number")
}
```

Spark + Scala!

To launch the scala spark shell:

```
# path to your spark intallation
cd spark-without-hadoop-xx/bin
./spark-shell.sh
```

- Scala Documentation
- Scala Tutorials
- Spark with Scala

Example 1: Word Count

The classic Word Count example demonstrates the power of parallel processing in Spark.

```
// Create an RDD from a text file
val textFile = spark.sparkContext.textFile("path/to/your/textfile.txt")

// Count the occurrences of each word
val wordCounts = textFile.flatMap(line => line.split(" ")).map(word => (word,
1)).reduceByKey(_ + _)

wordCounts.collect().foreach(println)
```

Example 2: Running a Machine Learning Algorithm in Parallel

Using Spark's MLlib to perform linear regression on a dataset.

```
import org.apache.spark.ml.regression.LinearRegression

// Load training data
val training =
spark.read.format("libsvm").load("data/mllib/sample_linear_regression_data.txt")

// Create a linear regression model
val lr = new LinearRegression()
```

```
  .setMaxIter(10)
  .setRegParam(0.3)
  .setElasticNetParam(0.8)

// Fit the model to the training data
val lrModel = lr.fit(training)

// Print the coefficients and intercept for linear regression
println(s"Coefficients: ${lrModel.coefficients} Intercept:
${lrModel.intercept}")

// Summarize the model over the training set and print out some metrics
val trainingSummary = lrModel.summary
println(s"numIterations: ${trainingSummary.totalIterations}")
println(s"objectiveHistory: ${trainingSummary.objectiveHistory.toList}")
trainingSummary.residuals.show()
println(s"RMSE: ${trainingSummary.rootMeanSquaredError}")
println(s"r2: ${trainingSummary.r2}")
```

Example 3: BigDL-2.x DLlib

> https://bigdl.readthedocs.io/en/latest/doc/DLlib/QuickStart/scala-getting-started.html