Simple Tutorial
CS-562
Introduction to data analysis using Apache Spark

Instructor: Vassilis Christophides
T.A.: Kostas Solomos / Moustakas Serafeim

Outline
- What is Scala?
- Core Functionality
- Simple First Tour of Scala
- Crash Tests
What is Scala?

- Scala is a mixture. It is the place where 2 world connect:
  - Object oriented programming
  - Functional Programming

- This mixture is the strength of Scala.
- We can say that Scala is a better version of Java.
Core Functionality

var vs val

- Var -> Var - iable
- Val -> Variable + Final
Functions Vs Methods

- Function is a group of statements that perform a task

But what is the difference of a method and a function?

  **Method**: a function, which is defined as a member of some object

  **Function**: a group of statements that perform a task

Load our data

- `sc.parallelize()`

Parallelized collections are created by calling SparkContext’s `parallelize` method on an existing collection.
The elements of the collection are copied to form a distributed dataset that can be operated on in parallel.
Lists & Arrays

- Scala Lists are quite similar to arrays

- All the elements of a list have the same type but...

- First, lists are immutable, which means elements of a list cannot be changed by assignment.

- Second, lists represent a linked list whereas arrays are flat.

  ```scala
  val nums: List[Int] = List(1, 2, 3, 4)
  ```
Map

Spark RDD map function returns a new RDD by applying a function to all elements of source RDD.

Filter

Spark RDD filter function returns a new RDD containing only the elements that satisfy a predicate.
Reduce
Spark RDD reduce function reduces the elements of this RDD using the specified commutative and associative binary operator.

\[
\text{reduce ( } f(x) \text{ ) } \\
\text{ } f(x) = (x, n) \rightarrow (x + n) \\
\]

GroupBy
Spark RDD groupBy function returns an RDD of grouped items.

\[
\text{groupBy ( } f(x) \text{ ) } \\
\text{ } f(x) = x \rightarrow x.textA(x) \\
\]
Quick Tour of Scala Part 1

Scala functions (closures)

(x: Int) => x + 2 // full version
x => x + 2 // type inferred
_ + 2 // placeholder syntax (each argument must be used exactly once)
x => { // body is a block of code
  val numberToAdd = 2
  x + numberToAdd
}

// Regular functions
def addTwo(x: Int): Int = x + 2
Quick Tour of Scala Part 2

Processing collections with functional programming

```scala
val list = List(1, 2, 3)
list.foreach{ println(_) } // prints 1, 2, 3
list.foreach{ println(_) }  // same

list.map{ x => x + 2 }      // returns a new List(3, 4, 5)
list.map{ _ + 2 }           // same

list.filter{ x => x % 2 == 1 } // returns a new List(1, 3)
list.filter{ _ % 2 == 1 }     // same

list.reduce((x, y) => x + y) // => 6
list.reduce{ _ + _ }         // same
```

All of these leave the list unchanged as it is immutable.

Functional methods on collections

There are a lot of methods on Scala collections, just google Scala Seq or http://www.scala-lang.org/docss2.10/collections.html#seq

<table>
<thead>
<tr>
<th>Method on Seq[T]</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>map(T =&gt; U)</td>
<td>Each element is result of f</td>
</tr>
<tr>
<td>flatMap(T =&gt; Seq[U])</td>
<td>One to many map</td>
</tr>
<tr>
<td>filter(T =&gt; Boolean)</td>
<td>Keep elements passing f</td>
</tr>
<tr>
<td>exists(T =&gt; Boolean)</td>
<td>True if one element passes f</td>
</tr>
<tr>
<td>forall(T =&gt; Boolean)</td>
<td>True if all elements pass</td>
</tr>
<tr>
<td>reduce(T, T =&gt; T)</td>
<td>Merge elements using f</td>
</tr>
<tr>
<td>groupBy<a href="T">K</a> =&gt; Map[K, List[T]]</td>
<td>Group elements by f</td>
</tr>
<tr>
<td>sort</td>
<td>Sort elements</td>
</tr>
</tbody>
</table>
Let's Make our First Crash-Test

Our Data:

```scala
val data = 1 to 10000
create an RDD based on that data…
val distData = sc.parallelize(data)
then use a filter to select values less than 10…

distData.filter(_ < 10).collect()
```
Spark Deconstructed: Log Mining Example

// load error messages from a log into memory
// then interactively search for various patterns
// https://gist.github.com/ceteri/8e65b9509a08c08e1132

// base RDD
val lines = sc.textFile("hdfs:///...")

// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"")
messages.cache()

// action 1
messages.filter(_.contains("mysql")).count()

// action 2
messages.filter(_.contains("php")).count()

Spark Deconstructed: Log Mining Example

We start with Spark running on a cluster… submitting code to be evaluated on it:
Spark Deconstructed: Log Mining Example

```scala
// base RDD
val lines = sc.textFile("hdfs://...")

// transformed RDD
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t").map(_ => x{}))
messages.cache()
```

discussing the other part

```
// base RDD
val lines = sc.textFile("hdfs://...")

// transformed RDD
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t").map(_ => x{}))
messages.cache()
```

discussing the other part
Spark Deconstructed: Log Mining Example

```scala
// base RDD
val lines = sc.textFile("hdfs://...")

// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\n")).map(i => i(1))
messages.unpersist()
// action 1
messages.filter(_.contains("mysql")).count()
```
Spark Deconstructed: Log Mining Example

// base RDD
val lines = sc.textFile("hdfs://.../")

// transformed rdds
val errors = lines.filter(_._contains("ERROR")
val messages = errors.map(_.split("\t")).map(x => (x))
messages.cache()

// action 1
messages.filter(_._contains("mysql")), count()
Spark Deconstructed: Log Mining Example

```scala
// base RDD
val lines = sc.textFile("https://...")

// transform RDD
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.up(1)("\"\") -> r(1))
messages.collect()  // action 1
messages.filter(_.contains("\"php\") -> err()  // discussing the other part
```

Spark Deconstructed: Log Mining Example

```scala
// base RDD
val lines = sc.textFile("https://...")

// transform RDD
discussing the other part
```

```scala
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.up(1)("\"\") -> r(1))
messages.collect()  // action 1
messages.filter(_.contains("\"php\") -> err()  // discussing the other part
```
Spark Deconstructed: Log Mining Example

// action 2
messages.filter(_contains('php')).count()
Looking at the RDD transformations and actions from another perspective...

```scala
// load access messages from a log into memory
// then incrementally search for various patterns
// https://spark.apache.org/docs/2.4.4/index.html

// base RDD
val lines = sc.textFile("hdfs://...")

// transformed RDD
val access = lines.filter(_.contains("web"))
val messages = access.map(_.split(" "))

// action 1
messages.filter(_.contains("error"))

// action 2
messages.filter(_.contains("log") \n\ extras
```

Spark Deconstructed:

```
// base RDD
val lines = sc.textFile("hdfs://...")
```

Spark Deconstructed:
Spark Deconstructed:

```
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\n\t\n")).map(_(0))
messages.foreach()
```

Spark Deconstructed:

```
// action:
messages.filter(_.contains("mysql\"\")}.count()
```
Special Thanks

- CS543 Presentations
- Coursera Introduction to Apache Spark, University of California, Databricks
- https://backtobazics.com/big-data/spark