Big Data Analysis - Part II: Introduction to Scala for Spark

Zhao Zhang
zzhang@tacc.utexas.edu
Texas Advanced Computing Center

Apr 27, 2017

Big Data Training Series at TACC

Introduction to Hadoop and Spark On Wrangler - April 20, 2017 1:00-4:30

- 1:00-1:30 Overview of Big Data Processing
- 1:30-2:30 What's Hadoop
- 3:00-4:00 Programming with Hadoop and Spark
- 4:00-4:30 Hands on

Introduction to Scala/Spark - April 27, 2017 1:00- 4:30

- 1:00-2:20 Introduction to Scala for Spark
- 2:20-2:50 Break
- 2:50-4:00 Programming with Spark using Scala
- 4:10-4:30 Hands on

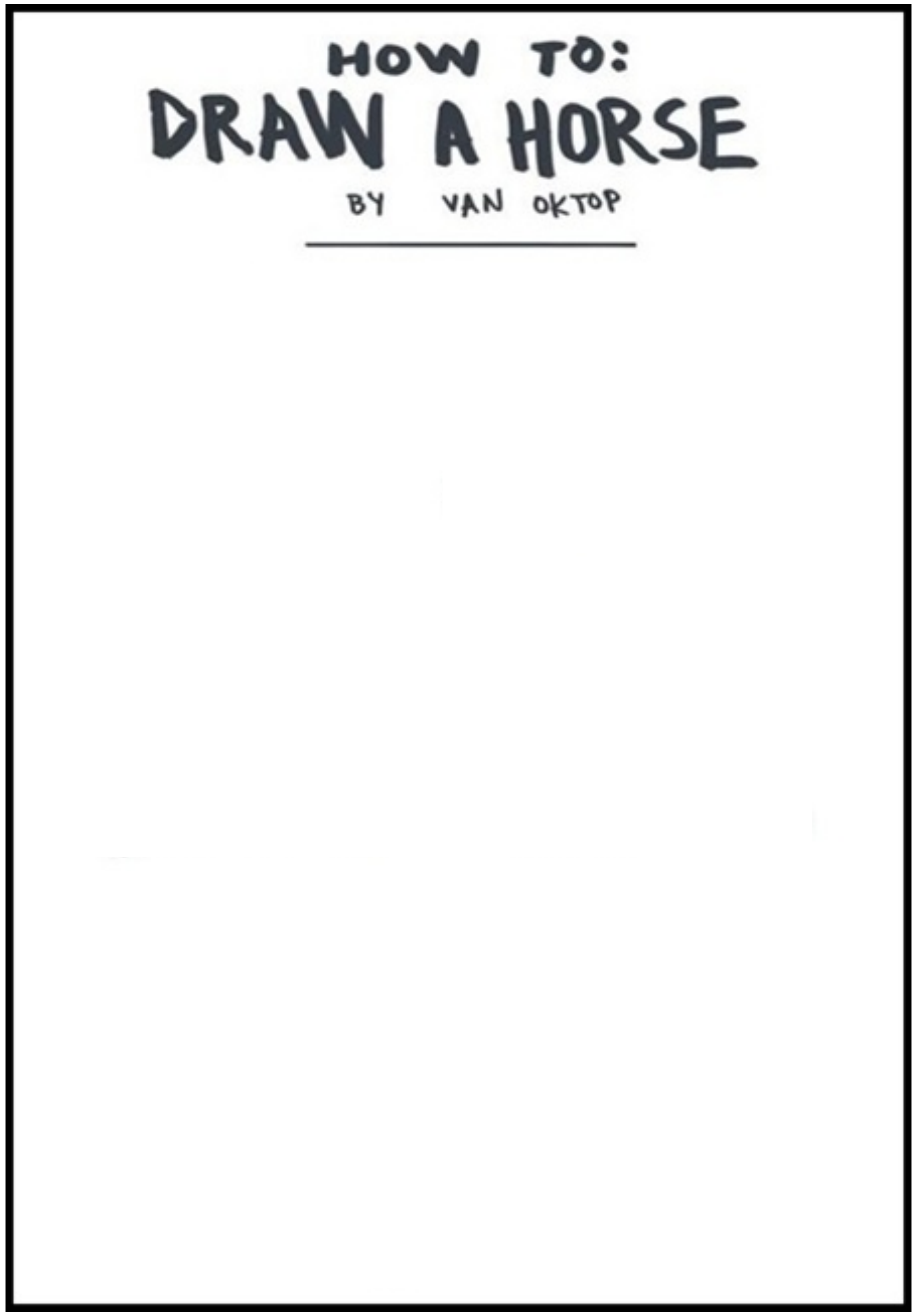
Data Analysis Using Hadoop/Spark - May 4, 2017

- 1:00-2:00 Dataframe, SparkSQL
- 2:00-3:00 Data analysis with MLlib and Graphx
- 3:00-4:00 Spark streaming, advanced topic configuration/optimization
- 4:00-4:30 Hands on

Self Introduction

- Zhao Zhang
- Research Associate in Data Intensive Group at Texas Advanced Computing Center
- Postdoc researcher and data science fellow in AMPLab and Berkeley Institute for Data Science, University of California, Berkeley. 2014-2016
- Ph.D Student in Department of Computer Science, University of Chicago

Interrupt Me When You Feel Like



Goal

- Learn just enough to use Spark with Scala

Scala

- A functional programming language
- JVM based
- Used in Apache Spark and Apache Kafka
- Tip: Always keep types in mind

Introduction to Scala

- Wrangler Setup
- Hello World!
- Variables
- Functions
- Control Flow

Introduction to Scala

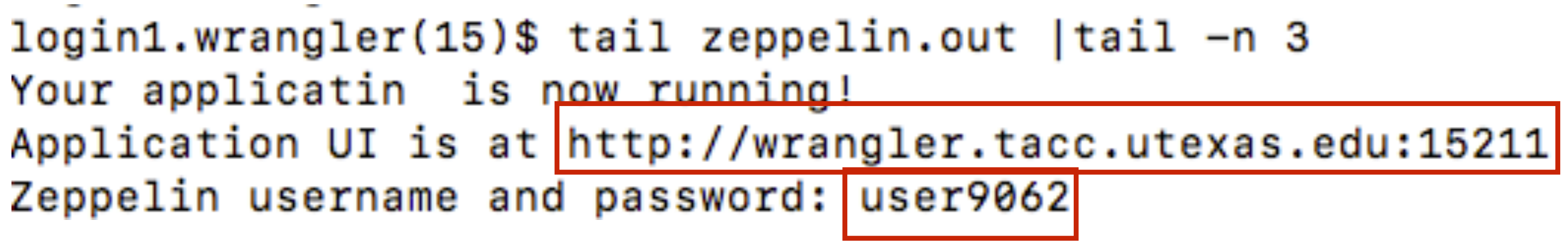
- Wrangler Setup
- Hello World!
- Variables
- Functions
- Control Flow

Wrangler Setup

- Shell
  - ssh $username wrangler.tacc.utexas.edu
  - idev -r hadoop+TRAINING-HPC+2187 -t 240
  - export PATH=/opt/apps/scala/scala-2.11.8/bin:$PATH

Wrangler Setup

- Zeppelin
  - ssh $username wrangler.tacc.utexas.edu
  - cd $WORK
  - cp /data/apps/.zeppelin/job.zeppelin
  - sbatch --reservation=hadoop+TRAINING-HPC+2187 job.zeppelin
  - tail zeppelin.out | tail -n 3



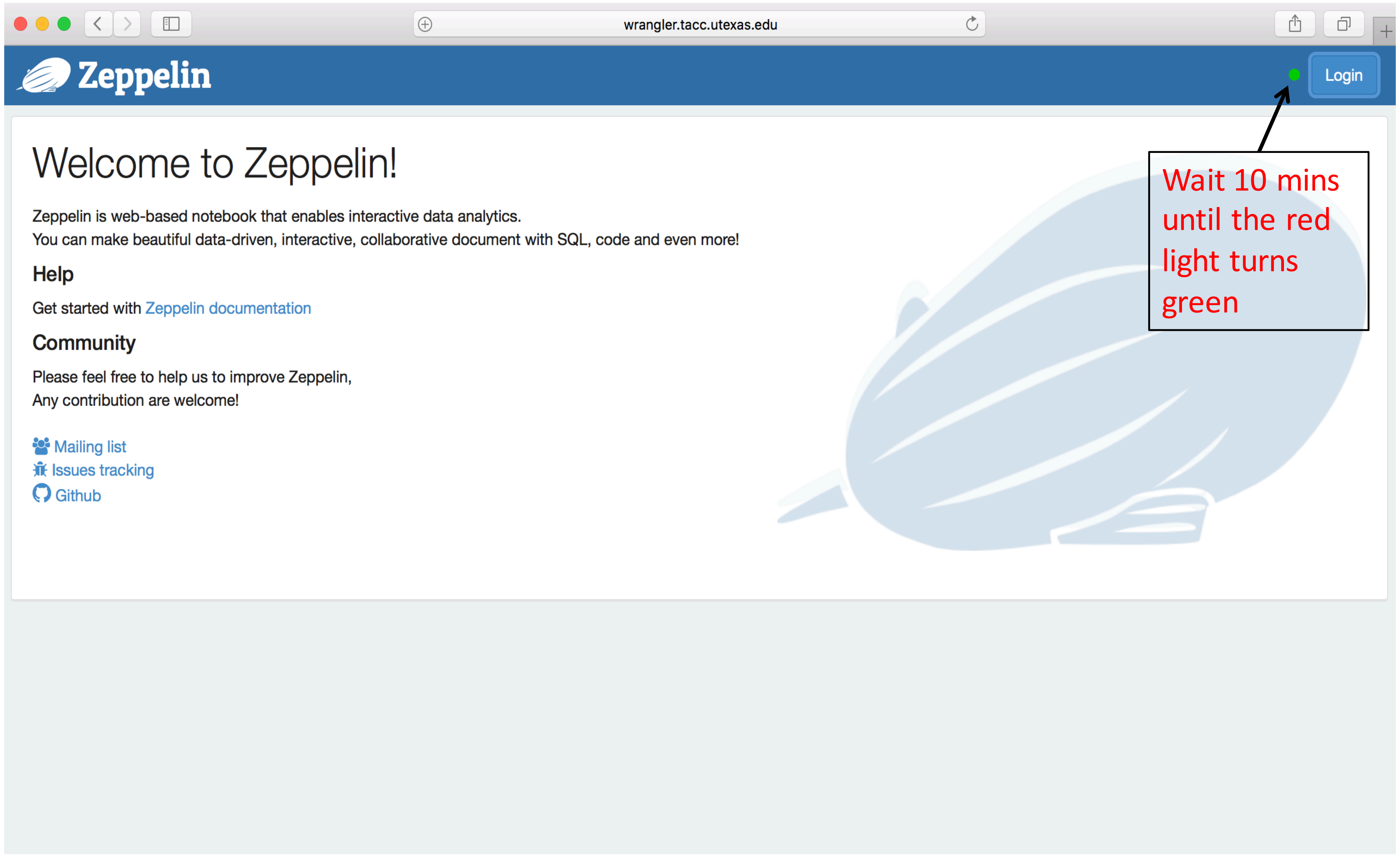
```
login1.wrangler(15)$ tail zeppelin.out |tail -n 3
Your applicatin  is now running!
Application UI is at http://wrangler.tacc.utexas.edu:15211
Zeppelin username and password: user9062
```

Wait 10 minutes for Zeppelin UI to start, copy and paste http://wrangler.tacc.utexas.edu:XXXXX to your web browser.

Use the username and password to login: userXXXX

Wrangler Setup

- Zeppelin



Welcome to Zeppelin!

Zeppelin is web-based notebook that enables interactive data analytics.
You can make beautiful data-driven, interactive, collaborative document with SQL, code and even more!

Help

Get started with Zeppelin documentation

Community

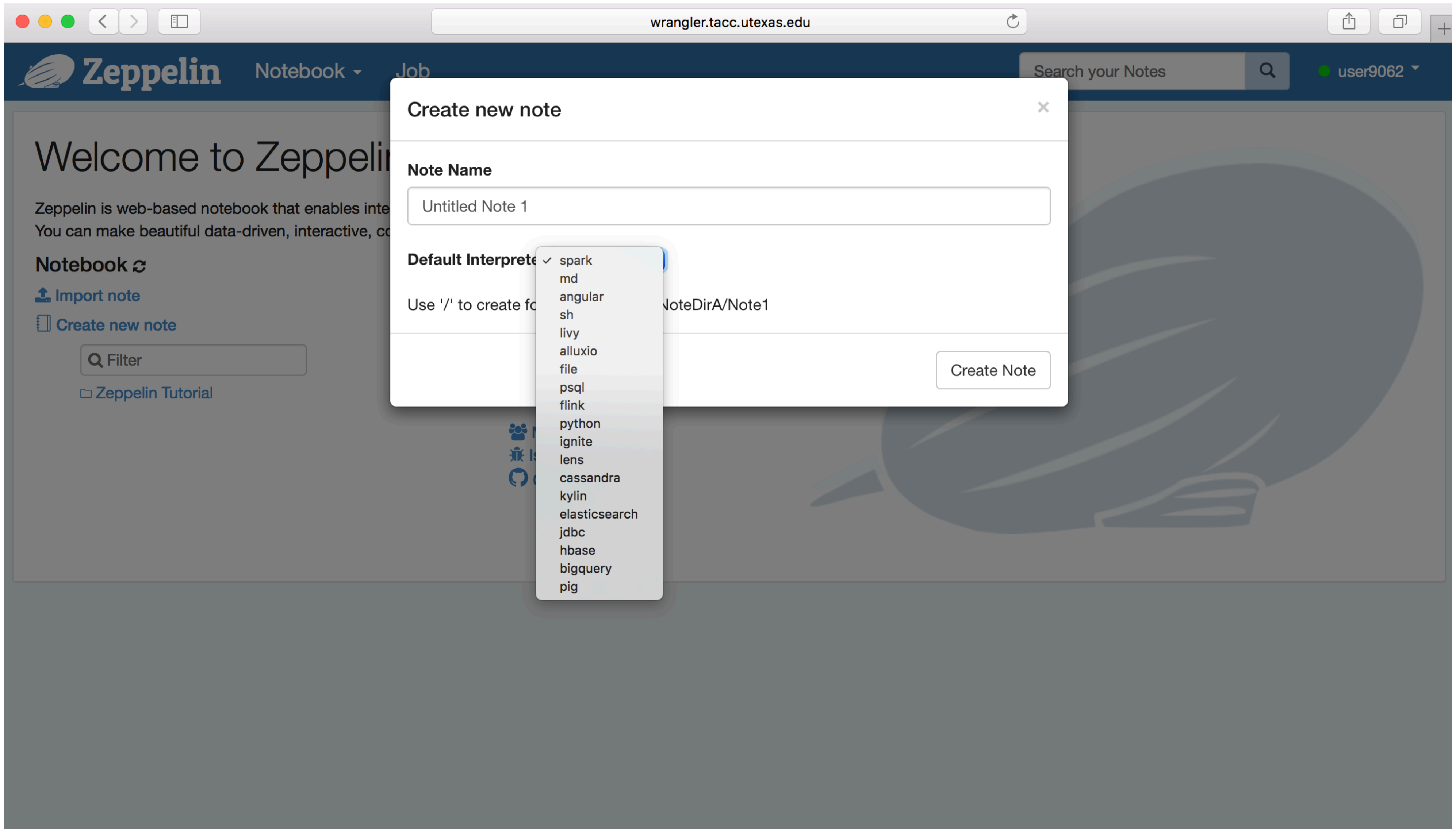
Please feel free to help us to improve Zeppelin,
Any contribution are welcome!

Mailing list
Issues tracking
Github

Wait 10 mins until the red light turns green

Wrangler Setup

- Zeppelin



Introduction to Scala

- Installations
- Hello World!
- Variables
- Functions
- Control Flow

Hello World!

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

Introduction to Scala

- Installations
- Hello World!
- Variables
- Functions
- Control Flow

Scala Variables

- Primitive variables and composite variables
- Immutable variables and mutable variables
- Composite variable iteration

Primitive Variables

- Mutable and immutable variables: ***val*** and ***var***

```
> val i = 5
> i: Int = 5
```

- ***val*** variable is immutable

```
> i = 6
> error: reassignment to val
```

- ***var*** variable is mutable

```
> var j = 5
> j = 6
```

Primitive Variables

- Primitive Types:
  - Double, Float, Long, Int, Short, Byte
  - Char, Boolean, Unit

```
> val a = 5
> val a: Double = 5
```

Composite Variables

- Composite Types — Data Structures
  - List, Map, Seq, Set, Tuple
  - String
- List is immutable

```
> val l = List(1,2,3)
> l(1)
> res0: Int = 2
> l(1) = 5
> error: value update is not a member of List[Int]
```

- What is immutable when we say a list is immutable? see Exercise 1.2
  - The variable *l* ?
  - The length of *l* ?
  - The elements of *l* ?

Composite Variables

- Mutable Counter-structure
  - List   scala.collection.mutable.ListBuffer

- import scala.collection.mutable.ListBuffer
- val l = new ListBuffer[Int]()
- l += 1
- l += 2
- l += 3
- l

Composite Variables

- Tuple

```
> val t = (1,2,3)
> t: (Int, Int, Int) = (1,2,3)
> t._1
> res1: Int = 1
> t._2
> res2: Int = 2

> val (i, j, k) = t
> i: Int = 1
> j: Int = 2
> k: Int = 3
```

Composite Variables

- Is List a type?
  - List[Int], List[Float], List[Double] are type
  - Scala uses type inference for missing type declarations

explain this a bit more

```
> val l = List(1,2,3)
> val l: List[Int] = List(1,2,3)
> val l: List[Double] = List(1,2,3)
> val l = List(1, 2.0, 3)    // what is the type of l ?
> l: List[Double] = List(1.0, 2.0, 3.0)
```

Composite Variables

- Iterate a List(1,2,3) and multiply each element by 2, see Exercise 1.3

Using while:

```
> var r = ListBuffer[Int]()
> val l = List(1,2,3)
> var i = 0
> while (i<l.length){
>   r += l(i)*2
>   i += 1
> }
```

Using for:

```
> val l = List(1,2,3)
> val r = for (x <- l)
    yield(x*2)
```

Using map

```
> val l = List(1,2,3)
> val r = l.map(x => x*2)
```

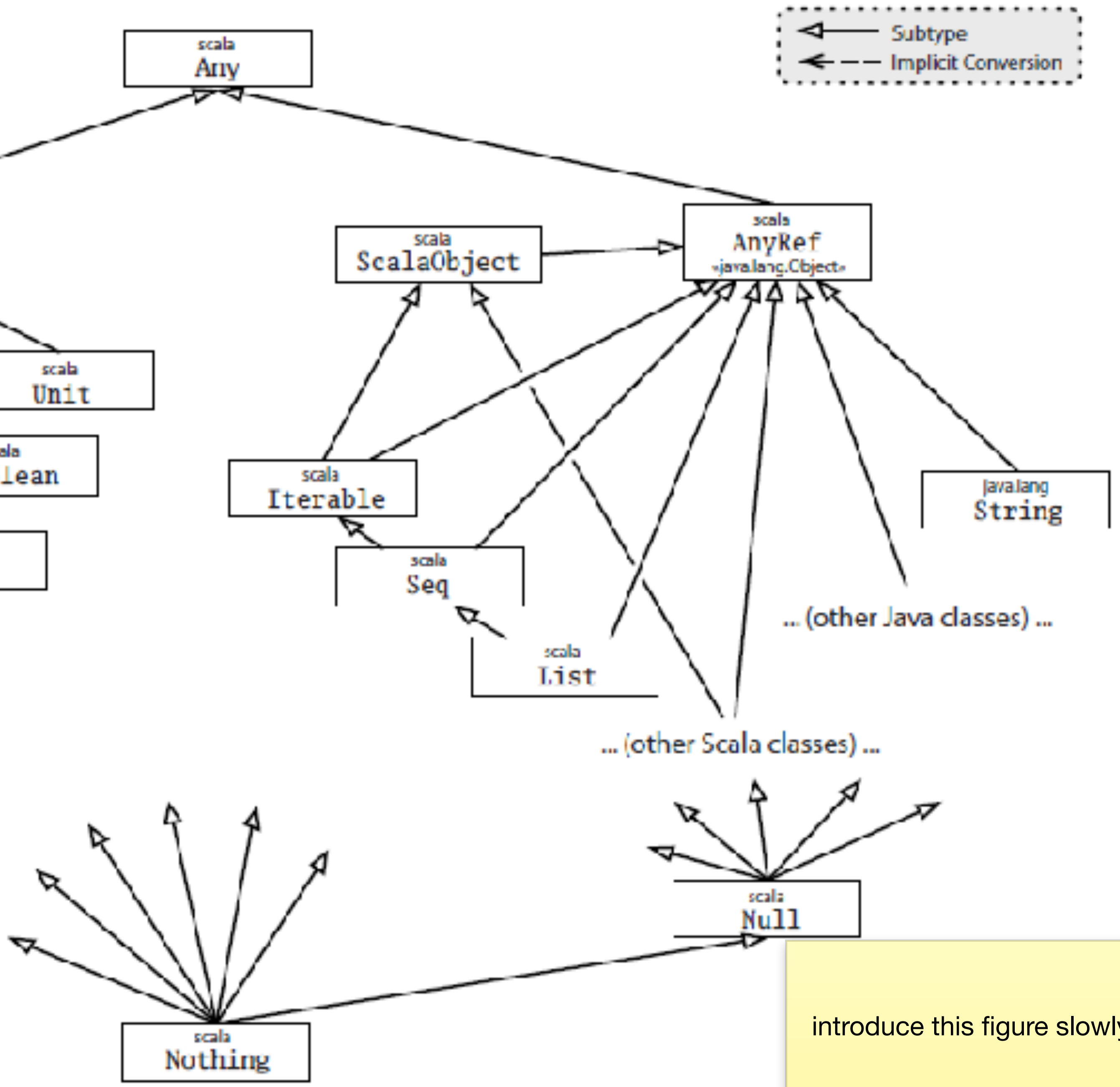
Composite Variables

- List
  - ()
  - head
  - tail
  - last
  - length
  - map
  - reverse
  - sorted
  - …

```
val l = List(3,1,2,4)
  > l(1)
  > l.head
  > l.tail
  > l.last
  > l.length
  > l.map(_*2)
  > l.reverse
  > l.sorted
```

Google Scala API
http://www.scala-lang.org/api/2.11.8/#package

Scala Class Hierarchy



introduce this figure slowly

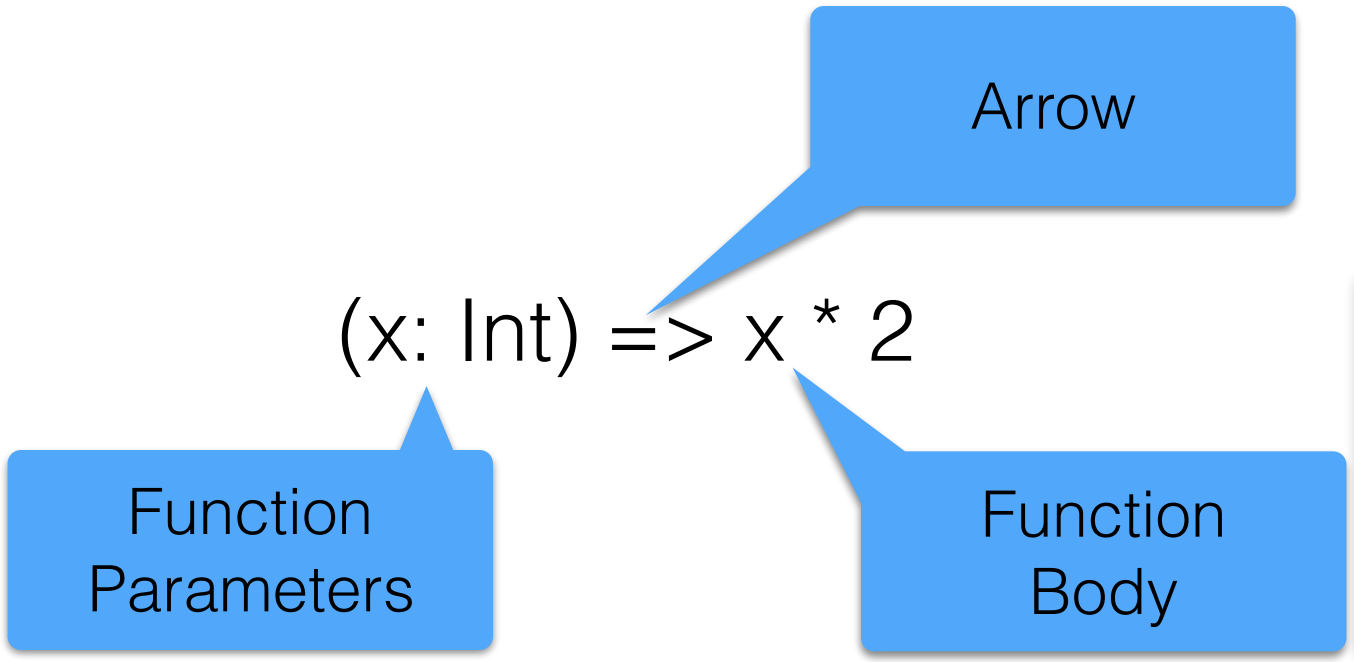
Introduction to Scala

- Installations
- Hello World!
- Variables
- Functions
- Control Flow

Scala Functions

- Anonymous Functions and Named Functions
- Function parameters and return values
- Function body
  - Single-line function
  - Multi-line function

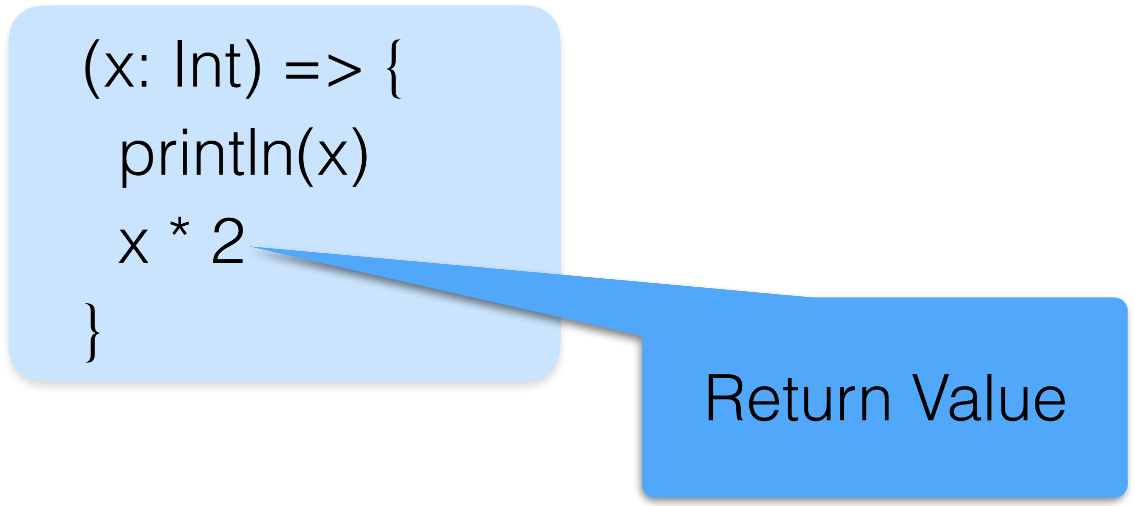
Scala Functions



```
(x: Int) => x * 2
```

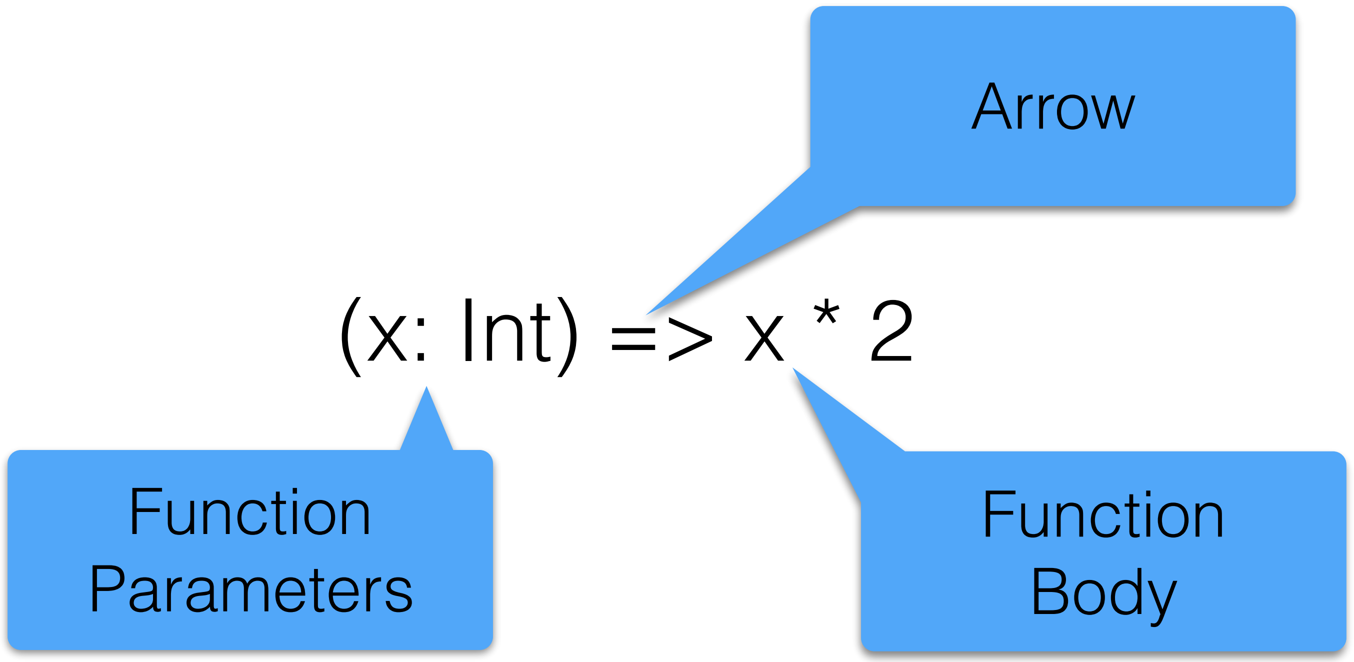
Mentioning return value, and the return keyword

What if we have multiple statements in the body?



```
(x: Int) => {
  println(x)
  x * 2
}
```

Scala Functions



```scala
(x: Int) => x * 2
```

```scala
val l = List(1,2,3)
l.map((x: Int) => x*2)
```

```scala
val l = List(1,2,3)
l.map(x => x*2)
```

```scala
val l = List(1,2,3)
l.map(_*2)
```

Place holder

Scala Functions

Multiple parameters?

```
(x: Int, y: Int) => x + y
```

```
val l = List((1,2),(3,4))
l.map(
  x: (Int, Int) => x._1+x._2
)
```

```
val l = List((1,2),(3,4))
l.map({
   case (x: Int, y:Int) => x+y
})
```

Pattern Matching
Anonymous Function

Scala Functions

Multiple statements?

```
(x: Int, y: Int) => {
    println(x)
    println(y)
    x + y
}
```

Scala Functions

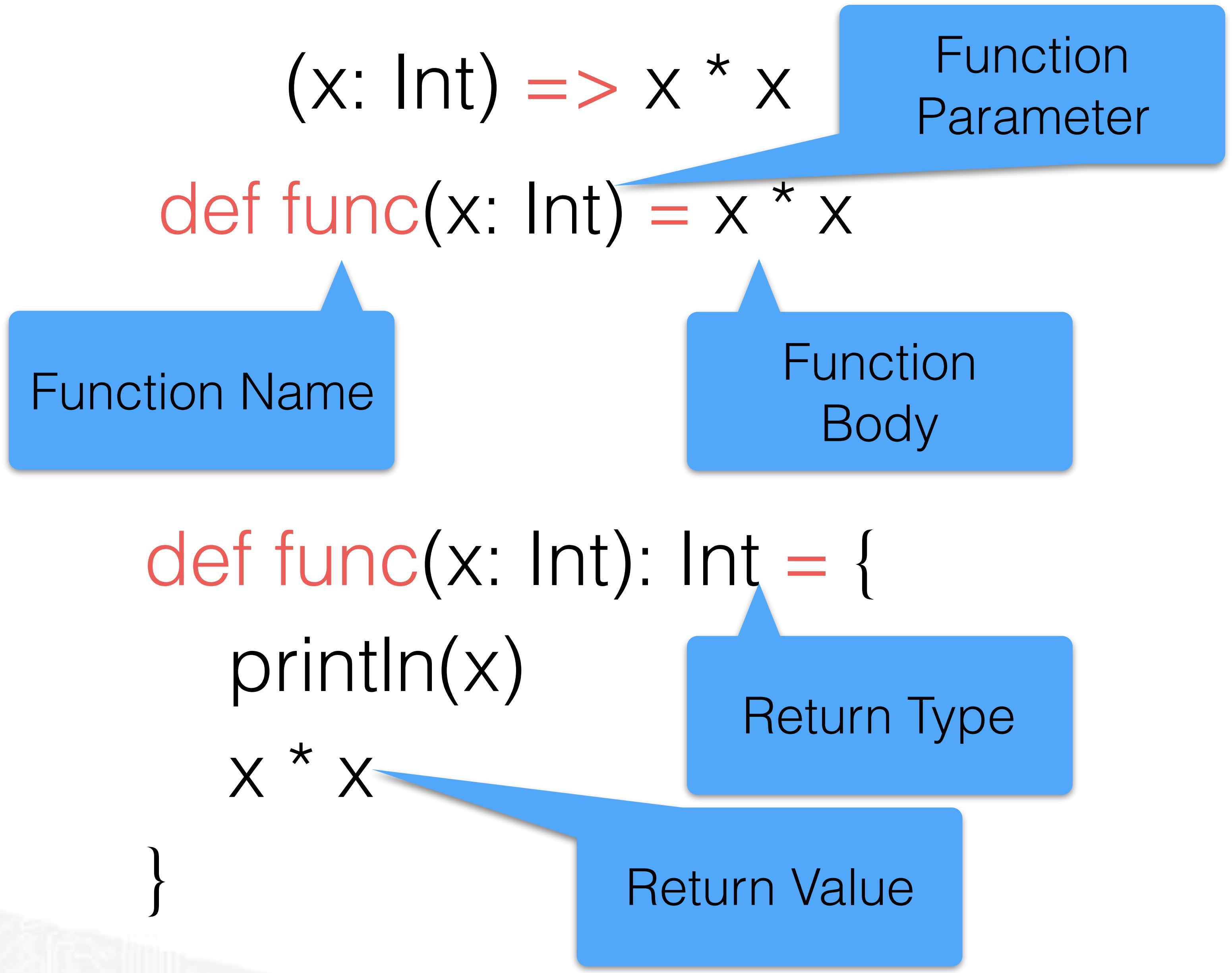
Multiple return values?

```
(x: Int, y: Int) => (x+3, y+5)
```

Return a tuple

Scala Functions

Give it a name



```
(x: Int) => x * x
def func(x: Int) = x * x

def func(x: Int): Int = {
    println(x)
    x * x
}
```

Scala Functions

- Checklist — Exercise 1.4
  - Anonymous function
  - Multiple parameters
  - Multiple statements
  - Multiple return values
  - Naming a function
  - Return types

Introduction to Scala

- Installations
- Hello World!
- Variables
- Functions
- Control Flow

Scala Control Flow

better explanation between foreach and map

- Iterate over List(1,2,3) and print out the element

- While: For: Foreach: Map:

While:

```
> val l = List(1,2,3)
> var i = 0
> while(i < l.length){
>     println(l(i))
>     i += 1
> }
```

For:

```
> val l = List(1,2,3)
> for(i <- l)
>     print(i)
```

Foreach:

```
> val l = List(1,2,3)
> l.foreach(x =>
>     println(x))
```

Map:

```
> val l = List(1,2,3)
> val r = l.map(x =>
>     println(x))
```

Scala Control Flow

- if … else …
- Iterate over List(1,2,3) and print out the odd elements

```
> val l = List(1,2,3)
> l.foreach(x => {
>     if (x%2 == 1)
>         println(x)
> })
```

Scala Control Flow

- Pattern Matching
- match … case …

```
> val l = List(1,2,3)

> l.foreach(x => {

>     if (x%2 == 1)

>         println(x)

> })
```

```
> val l = List(1,2,3)

> l.foreach(x => x%2 match{

>     case 1 => println(x)

>     case _ =>

> })
```

Scala Control Flow

- Pattern Matching — Types — Exercise 1.5
- A function that can handles various paramete

extremely useful when you have
multiple child classes that implements
the same abstract class or traits

also useful in msg processing

```
> def func(a: Any) = a match {
>     case i:Int => println(“a is an int”)
>     case f:Float => println(”a is a float”)
>     case d:Double => println(“a is a double”)
>     case s:String => println(“a is a string”)
>     case l: List[_] => println(“a is a list”)
>     case _ => println(“unknown type”)
> })
```

```
> func(1)
> func(1.0)
> func(1.toFloat)
> func(“abc”)
> func(List(1,2,3))
> func(Array(1,2,3))
```

Lazy Evaluation

- Iterator is just another way to access the data structure
  - val list = List(1,2,3,4,5,6)
  - val i = list.toIterator
  - i.hasNext()
  - i.next()
  - val r = i.map(_*2)
- Lazy Evaluation
- See Exercise 1.6

Use scala.collection.BufferedIterator's head() method to access next value without advancing
val i = l.toIterator.buffered

Scala Control Flow

- Checklist
  - Loop with while, for, foreach, map
  - Condition and Jump with if … else … and pattern matching
  - State-full access method: Iterator
  - Lazy evaluation

Run a Scala Program

- Three ways to run a Scala program
  - Compile then execute
  - Scripting
  - Interactive Shell
- See Exercise 1.1 (http haozhang/ training2017/blob/ma ercise1.md)

start with interactive shell, summarize with this slide in the end

Hello World!

- Run Scala program through compilation and execution
  - Create HelloWorld.scala, then type in the following

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

- Compilation
  - scalac HelloWorld.scala
- Execution
  - scala HelloWorld
  - > Hello, world!

Hello World!

- Run Scala program through scripting
    - Create HelloWorld.sh, then type in the following

```
#!/usr/bin/env scala

object HelloWorld extends App{
    println("Hello, world!")
}
HelloWorld.main(args)
```

- Set execution access code
    - chmod 755 HelloWorld.sh
- Execution
    - ./HelloWorld.sh
    - > Hello, world!

Hello World!

- Run Scala program through interactive shell
  - Type "scala"
  - Type the following code

```
1. println("Hello World!")

> Hello World!
```

Summary

- Wrangler Setup
- Hello World!
- Variables
    - val and var, primitive variables and composite variables
- Functions
    - Anonymous function, multiple parameters, multiple statements, multiple return values, place holder
- Control Flow
    - Loops (while, for, foreach, map), Condition (if … else …), Pattern Matching, Iterator

Exercise

- git clone https://github.com/zhaozhang/training2017/blob/master/exercise/Exercise1.md

- We will be back at 2:50PM

Big Data Analysis - Part II: Programming Spark with Scala

Zhao Zhang
zzhang@tacc.utexas.edu
Texas Advanced Computing Center

April 27, 2017

Exercise

- What is immutable when we say “val l = List(1,2,3)”?
- Pattern Matching?
- Lazy evaluation with Iterator

Goals

- Understanding Spark
- Programming Spark with Resilient Distributed Dataset (RDD)
- Learning professional techniques when you have no idea how to do it

Overview

- Wrangler Setup
- RDD Concept
- RDD Programming Model
- Build Spark Application
- Unit Test

Overview

- Wrangler Setup
- RDD Concept
- RDD Programming Model
- Build Spark Application
- Unit Test

Wrangler Setup

- Shell
- Zeppelin

Overview

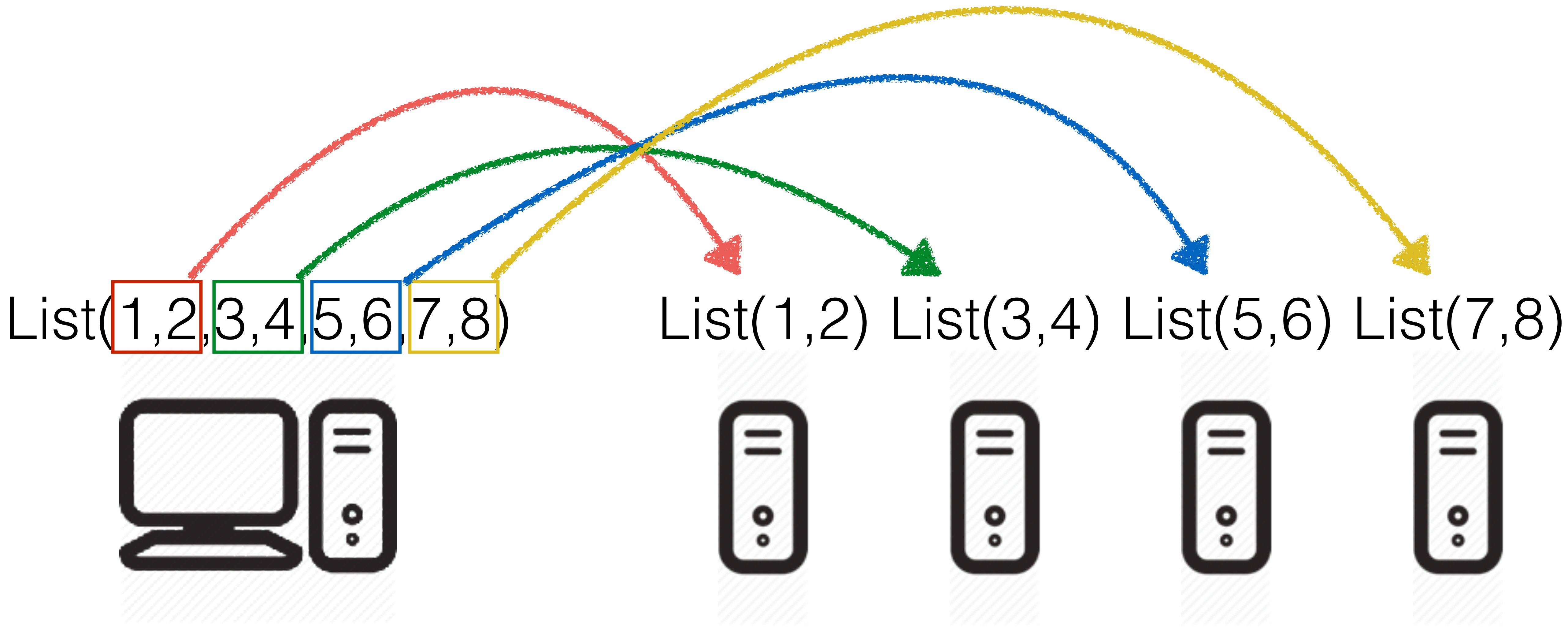
- Wrangler Setup
- RDD Concept
- RDD Programming Model
- Build Spark Application
- Unit Test

Starting with Lists

- ***List[A]*** is a class for **immutable *linked lists*** representing ordered collections of elements of type ***A***.
- val l = List(1,2,3,4,5)
- Operations on List:

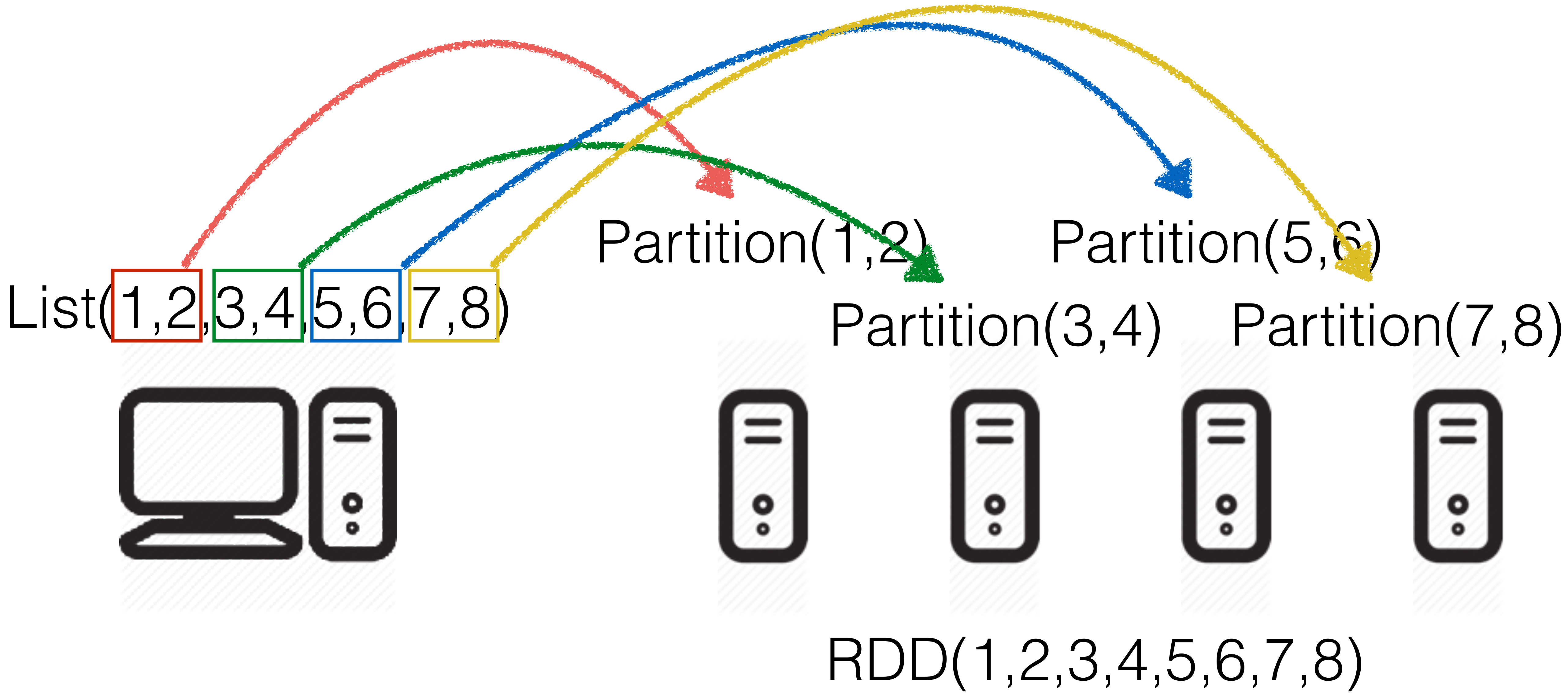
• map	val r = l.map(x => x*2)	List(2,4,6,8,10)
• filter	val r = l.filter(x => x%2 == 0)	List(2,4)
• groupBy	val r = l.groupBy(x => x%2)	Map(1 -> List(1, 3, 5), 0 -> List(2, 4))
• count	val r = l.count(x => x%2 == 0)	2

From Centralized to Distributed



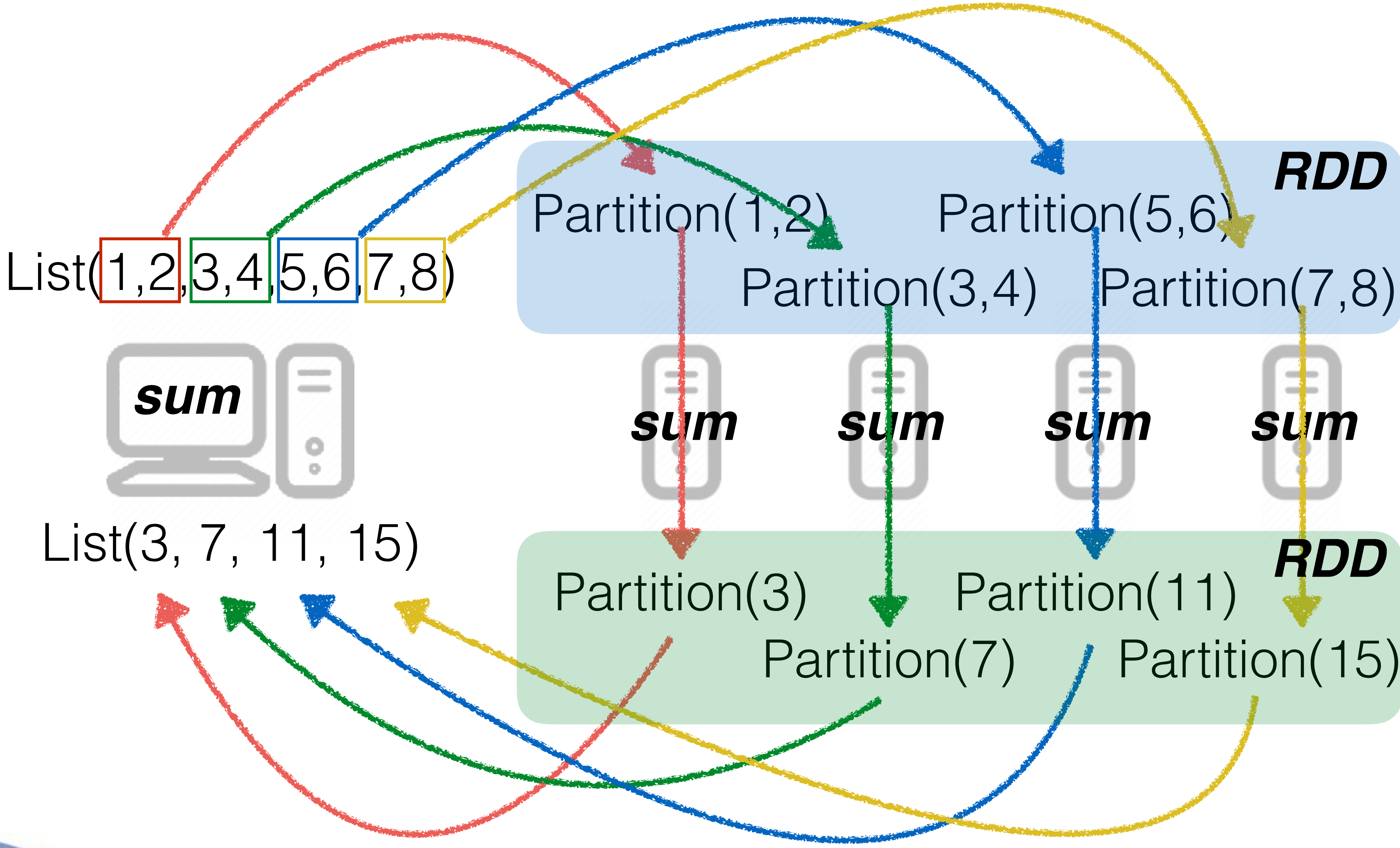
Courtesy Image from https://cdn0.iconfinder.com/data/icons/hardware-outline-icons/60/Hardware_Hardware-01-512.png

From Centralized to Distributed



Courtesy Image from https://cdn0.iconfinder.com/data/icons/hardware-outline-icons/60/Hardware_Hardware-01-512.png

From Centralized to Distributed

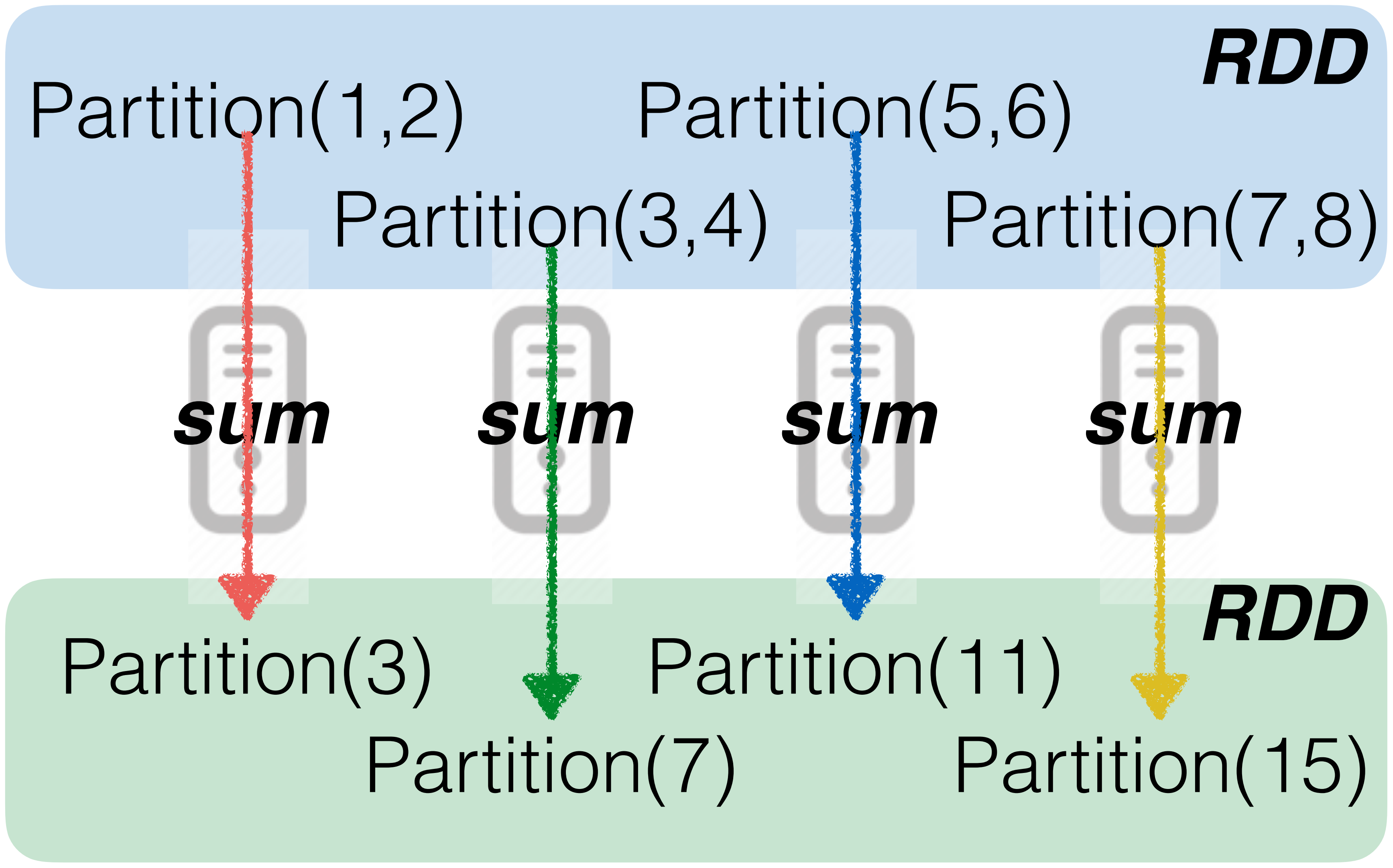


Courtesy Image from https://cdn0.iconfinder.com/data/icons/hardware-outline-icons/60/Hardware_Hardware-01-512.png

Key Concept: RDDs

- A Resilient Distributed Dataset (RDD), the basic abstraction in Spark, represents an immutable, partitioned collection of elements that can be operated on in parallel.
- An RDD contains a set of partitions
- List of dependencies
- Function to compute a partition (as an iterator) given its parents

From Centralized to Distributed

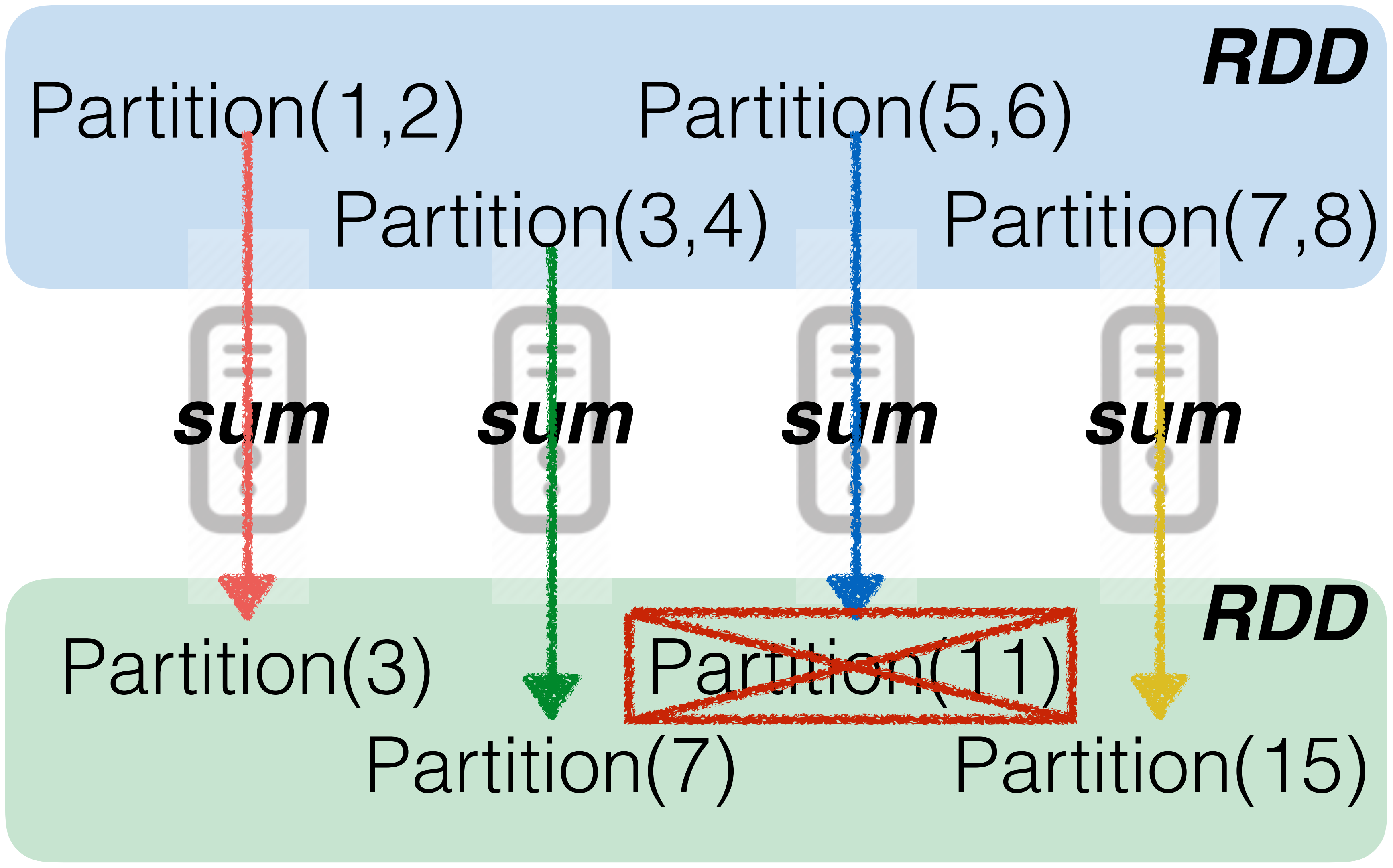


Courtesy Image from https://cdn0.iconfinder.com/data/icons/hardware-outline-icons/60/Hardware_Hardware-01-512.png

What If You Are the Designer?

- Correctness: all computations on a single computer should generate identical results in a distributed environment
- Fault-tolerance: if the data on one computer gets lost, the system should be able to recover it

From Centralized to Distributed



Courtesy Image from https://cdn0.iconfinder.com/data/icons/hardware-outline-icons/60/Hardware_Hardware-01-512.png

Overview

- Wrangler Setup
- RDD Concept
- RDD Programming Model
- Build Spark Application
- Unit Test

RDD Programming Model

- > val rdd = sc.parallelize(List(0,1,…,99))
- Transformations
  - val res = rdd.map(x => x*2)
  - val res = rdd.filter(x => x%2 == 0)
  - val res = rdd.groupBy(x => x%2)
  - val lines = sc.textFile(“path-to-file”)
  - val rdd = sc.binaryFiles(“path-to-file”)
- Actions
  - res.count()
  - res.collect()
  - res.take()
  - res.reduce()
  - res.saveAsTextFiles(“path-to-file”)

RDD Programming Model

- Basic RDD operations: RDD[T]
- Pair RDD operations: RDD[(K,V)]
- Word Count

RDD Transformations

- def map[U](f: (T) => U): RDD[U]
  - Return a new RDD by applying a function to all elements of this RDD
  - val rdd = sc.parallelize(0 until 100)
  - val res = rdd.map(x => x*2)
  - res.collect()

```
scala> res.collect
res0: Array[Int] = Array(0, 2, 4, 6, 8, 10, 12, 14, 16, … 198)
```

RDD Transformations

- def flatMap[U](f: (T) ⇒ TraversableOnce[U]): RDD[U]
  - Return a new RDD by first applying a function to all elements of this RDD, and then flattening the results
  - val rdd = sc.parallelize(List(1,2,3,4))
  - val res = rdd.flatMap(x => List(x, x, x))
  - res.collect()

```
scala> res.collect
res1: Array[Int] = Array(1, 1, 1, 2, 2, 2,3, 3, 3, 4, 4, 4)
```

RDD Transformations

- def filter(p: (A) ⇒ Boolean): List[A]
  - Selects all elements of this traversable collection which satisfy a predicate
  - val rdd = sc.parallelize(0 until 100)
  - val res = rdd.filter(x => x%2 == 0)
  - res.collect()

```
scala> res.collect
res4: Array[Int] = Array(0, 2, 4,…, 96, 98)
```

RDD Transformations

- def mapPartitions[U](f: (Iterator[T]) ⇒ Iterator[U]): RDD[U]
  - Return a new RDD by applying a func
  - What is an iterator?
  - val rdd = sc.parallelize(0 until 8, 2)
  - val res = rdd.mapPartitions(x => List(x.sum).iterator)
  - res.collect()

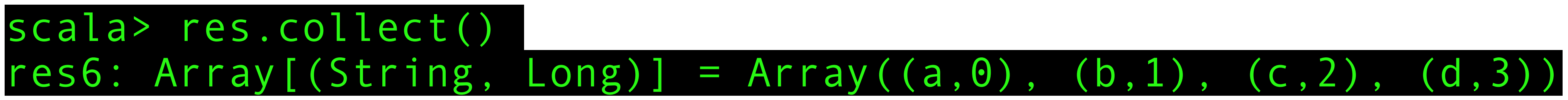
number of partitions
Partition 0: 0,1,2,3
Partition 1: 4,5,6,7

the input is an iterator, the output is also an iterator

```
scala> res.collect
res5: Array[Int] = Array(6, 22)
```

RDD Transformations

- def zipWithIndex(): RDD[(T, Long)]
  - Zips this RDD with its element indices.
  - Use this method when you need index of the element with the value
  - val rdd = sc.parallelize(List("a", "b", "c", "d"))
  - val res = rdd.zipWithIndex()
  - res.collect()



```
scala> res.collect()
res6: Array[(String, Long)] = Array((a,0), (b,1), (c,2), (d,3))
```

RDD Transformations

- groupBy[K](f: (T) ⇒ K): RDD[(K, Iterable[T])]
  - Return an RDD of grouped items. Each group consists of a key and a sequence of elements mapping to that key.
  - val rdd = sc.parallelize(0 until 100)
  - val res = rdd.groupBy(x => x%2 == 0)
  - res.collect()

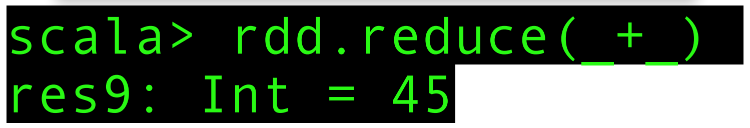
Key, Value
false, CompactBuffer(1, 3, …)

```
scala> res.collect()
res7: Array[(Boolean, Iterable[Int])] =
Array((false,CompactBuffer(1, 3, …, 99)),(true,CompactBuffer(0, 2, 4, …, 98)))
```

RDD Transformations

- def reduce(f: (T, T) ⇒ T): T
  - Reduces the elements of this RDD using the specified commutative and associative binary operator.
  - val rdd = sc.parallelize(0 until 10)
  - val res = rdd.reduce(_ +

find out what these two place holders mean



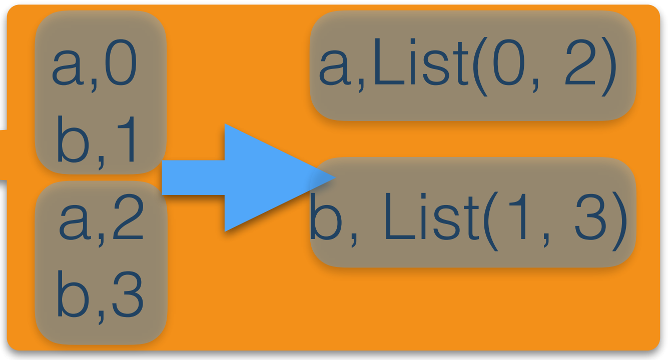
```
scala> rdd.reduce(_+_)
res9: Int = 45
```

PairRDD Transformations

- Pair RDD assumes the elements are tuples [(K, V)]
- def groupByKey(): RDD[(K, Iterable[V])]
  - Group the values for each key in the RDD into a single sequence.
  - val l = List("a", "b", "a", "b").zipWithIndex

```
l: List[(String, Int)] = List((a,0), (b,1), (a,2), (b,3))
```

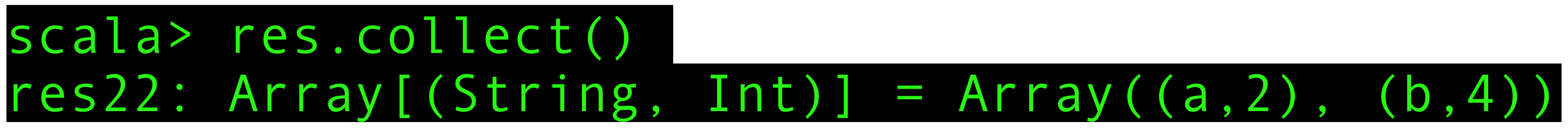
  - val rdd = sc.parallelize(l)
  - val res = rdd.groupByKey()
  - res.collect()



```
scala> res.collect()
res18: Array[(String, Iterable[Int])] =
Array((a,CompactBuffer(0, 2)), (b,CompactBuffer(1, 3)))
```

PairRDD Transformations

- def reduceByKey(func: (V, V) ⇒ V): RDD[(K, V)]
  - Merge the values for each key using an associative reduce function.
  - val l = List(“a”, “b”, “a”, “b”).zipWithIndex
  - val rdd = sc.parallelize(l)
  - val res = rdd.reduceByKey(_ + _)
  - res.collect()



```
scala> res.collect()
res22: Array[(String, Int)] = Array((a,2), (b,4))
```

PairRDD Transformations

- def join[W](other: RDD[(K, W)]): RDD[(K, (V, W))]
  - Return an RDD containing all pairs of elements with matching keys in this and other
  - val person = List("adam", "ben", "chris", "david")
  - val age = List(27, 42, 53, 23)
  - val dept = List("HPC", "Data", "Vis", "Edu")
  - val rdd1 = sc.parallelize(person.zip(age))
  - val rdd2 = sc.parallelize(person.zip(dept))
  - val res = rdd1.join(rdd2)
  - res.collect()

```
res.collect()
res30: Array[(String, (Int, String))] = Array(
(ben,(42,Data)), (david,(23,Edu)),
(chris,(53,Vis)), (adam,(27,HPC)))
```

Word Count

- Now let us consider a word counting problem
- We would like to count the frequency of each word in the /tmp/data/20news-all/alt.atheism directory

Update to Ruizhu’s dataset

Word Count

- Now let us use the RDD transformations and actions to implement a word count program
  - val lines = sc.textFile(“/tmp/data/book.txt”)
  - val words = lines.flatMap(l => l.split(“ ”))
  - words.collect()

```
scala> words.collect
res0: Array[String] = Array(The, Project, Gutenberg, EBook, … )
```

- What shall we do from here?

Word Count

- Now we have words as RDD[String]
  - val kwp = words.map(w => (w, 1))
  - val res = kwp.reduceByKey(_ + _)
  - res.collect()

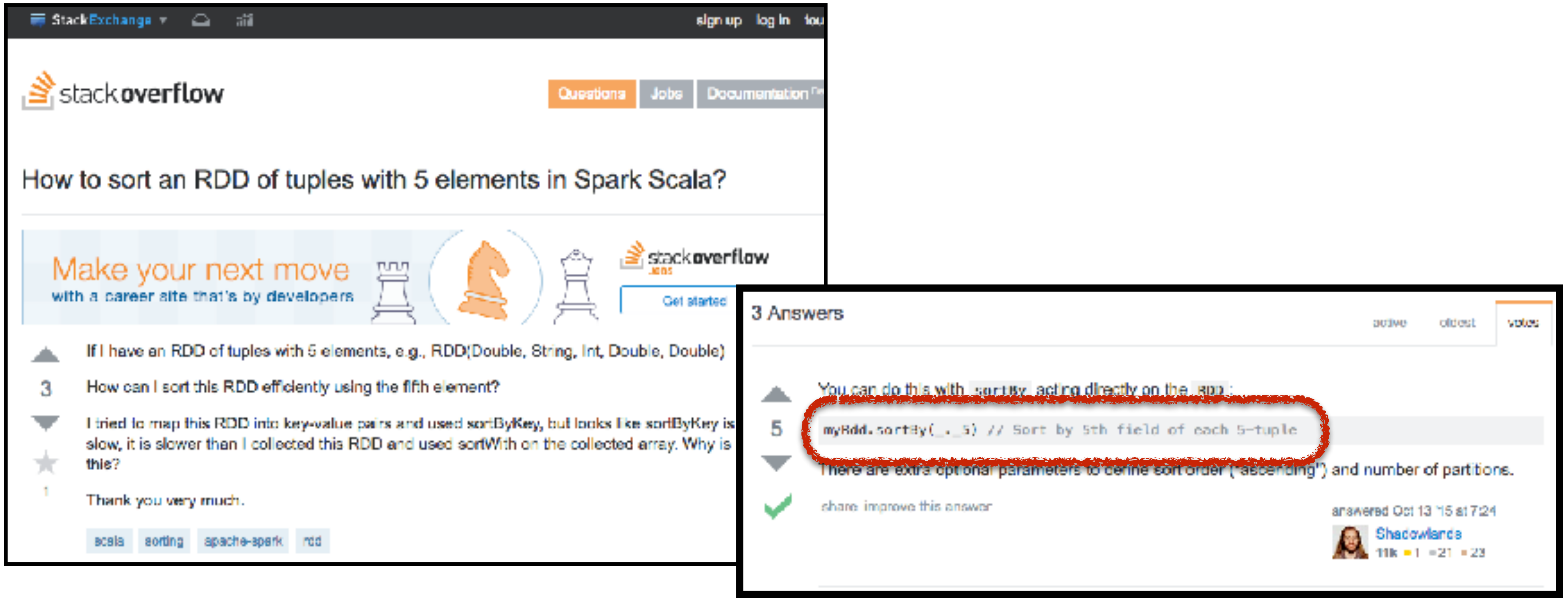
Shall we sort the results?
Use Google!

```
scala> res.collect
res1: Array[(String, Int)] = Array((young,11), (bone,1), (House,1),…)
```

Word Count

- Res is an RDD[(String, Int)], now we want to sort the tuples with the second element
- Google “sort rdd of tuples”

http://stackoverflow.com/questions/33096361/how-to-sort-an-rdd-of-tuples-with-5-elements-in-spark-scala



How to sort an RDD of tuples with 5 elements in Spark Scala?

3

If I have an RDD of tuples with 5 elements, e.g., RDD(Double, String, Int, Double, Double)

How can I sort this RDD efficiently using the fifth element?

I tried to map this RDD into key-value pairs and used sortByKey, but looks like sortByKey is slow, it is slower than I collected this RDD and used sortWith on the collected array. Why is this?

Thank you very much.

scala sorting apache-spark rdd

3 Answers

5

You can do this with `sortBy` acting directly on the `RDD`:

```
myRdd.sortBy(_._5) // Sort by 5th field of each 5-tuple
```

There are extra optional parameters to define sort order ("ascending") and number of partitions.

answered Oct 13 '15 at 7:24
Shadowlands
11k ●1 ●21 ●23

Word Count

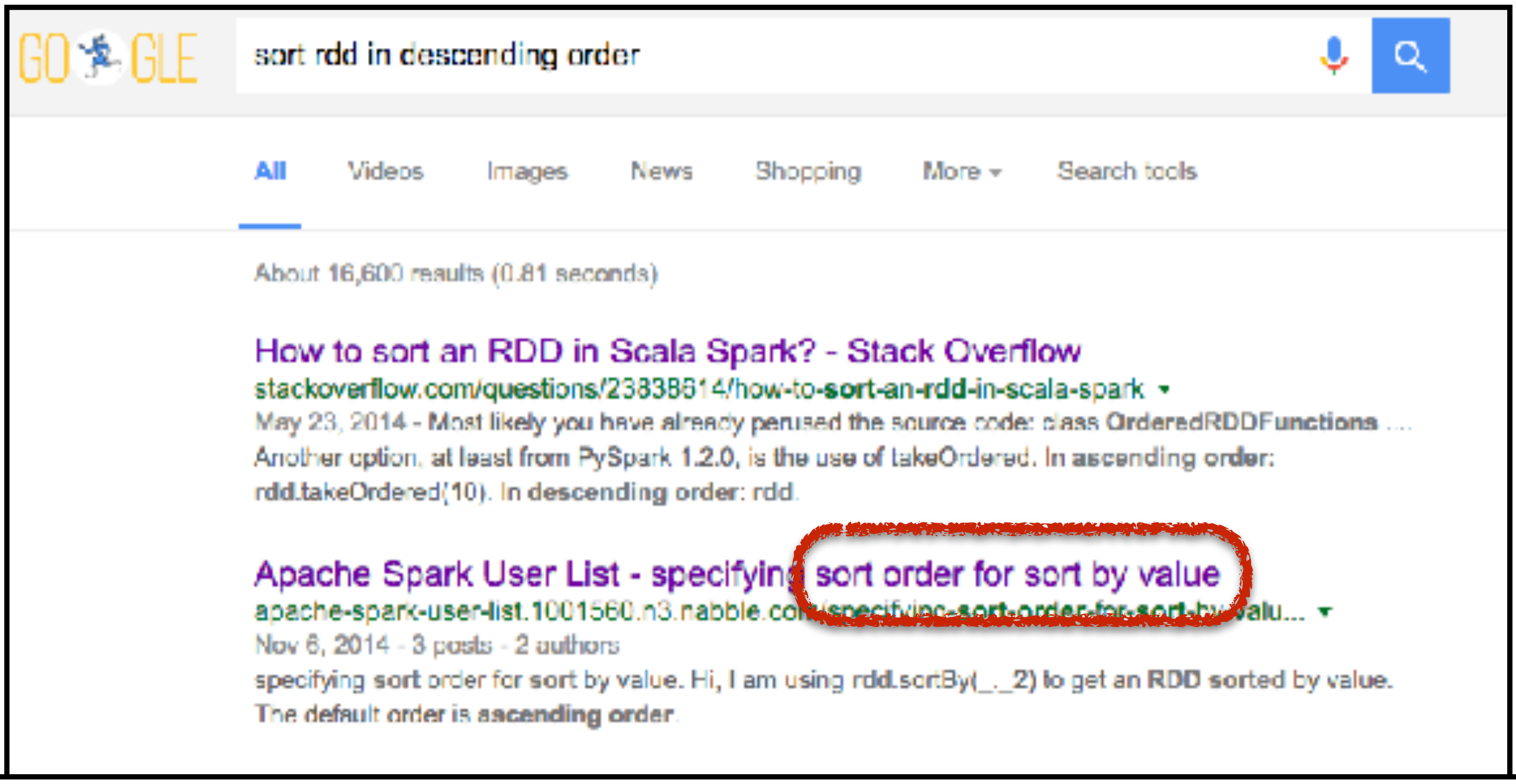
- Try out res.sortBy(_._2)
  - val sorted = res.sortBy(_._2)
  - sorted.collect()

No! I want it in descending order!
Use Google!

```
scala> sorted.collect()
res2: Array[(String, Int)] = Array((House,1), (nobleman,1), (bone,1) …)
```

Word Count

- Google "sort rdd in descending order"

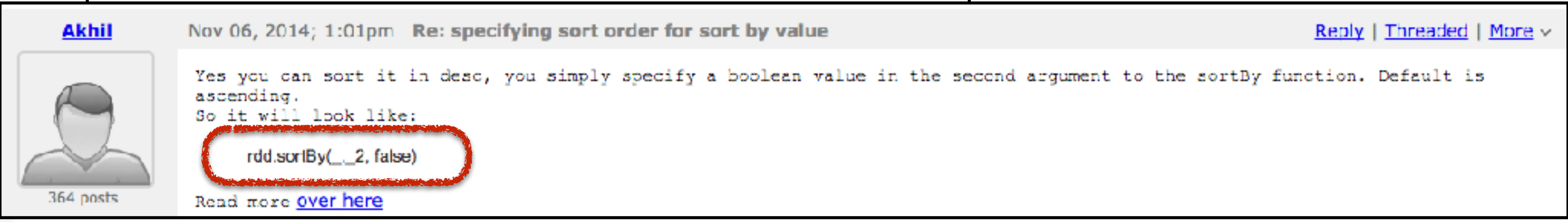


sort rdd in descending order

About 16,600 results (0.81 seconds)

How to sort an RDD in Scala Spark? - Stack Overflow
stackoverflow.com/questions/23838614/how-to-sort-an-rdd-in-scala-spark
May 23, 2014 - Most likely you have already perused the source code: class OrderedRDDFunctions ... Another option, at least from PySpark 1.2.0, is the use of takeOrdered. In ascending order: rdd.takeOrdered(10). In descending order: rdd.

Apache Spark User List - specifying sort order for sort by value
apache-spark-user-list.1001560.n3.nabble.com/specifying-sort-order-for-sort-by-valu...
Nov 6, 2014 - 3 posts - 2 authors
specifying sort order for sort by value. Hi, I am using rdd.sortBy(_._2) to get an RDD sorted by value. The default order is ascending order.



Akhil Nov 06, 2014; 1:01pm Re: specifying sort order for sort by value Reply | Threaded | More

364 posts

Yes you can sort it in desc, you simply specify a boolean value in the second argument to the sortBy function. Default is ascending.
So it will look like:

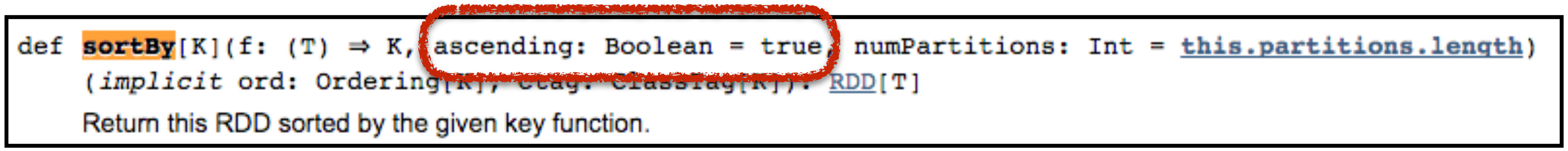
```
rdd.sortBy(_._2, false)
```

Read more over here

Word Count

- Or, as always, look into the Spark API

http://spark.apache.org/docs/1.6.0/api/scala/index.html#org.apache.spark.rdd.RDD



```
def sortBy[K](f: (T) ⇒ K, ascending: Boolean = true, numPartitions: Int = this.partitions.length)
    (implicit ord: Ordering[K], ctag: ClassTag[K]): RDD[T]
```

Return this RDD sorted by the given key function.

Word Count

- Now use res.sortBy(_._2, false)
  - val sorted = res.sortBy(_._2, false)
  - sorted.collect()

```
scala> sorted.collect()
res28: Array[(String, Int)] = Array((the,5060), (of,3626), ("",2313), (and,2230),
(to,1468), … )
```

Overview

- Wrangler Setup
- RDD Concept
- RDD Programming Model
- Build Spark Application
- Unit Test

Build a Self-contained Application using Maven

- Setup Maven On Wrangler
    - module load maven
- “cp -r /work/00791/xwj/DMS/spark-training ~/”
- or “git clone https://github.com/zhaozhang/spark-training.git”
- The directory looks like:

```
├── Hello.scala
├── HelloWorld.sh
├── pom.xml
├── src
    └── main
        └── scala
            └── WordCount.scala
```

Build a Self-contained Application using Maven

- Build the Application
    - mvn package
- The target directory looks like:

```
target/
├── WordCount-1.0-SNAPSHOT.jar
├── classes/
├── classes.timestamp
├── maven-archiver/
└── test-classes
```

- Submit the Application
    - spark-submit --class WordCount target/WordCount-1.0-SNAPSHOT.jar

Overview

- Wrangler Setup
- RDD Concept
- RDD Programming Model
- Build Spark Application
- Unit Test

Unit Test

- We use scalatest module for unit test
- Refer to pom.xml for scalatest dependencies and settings
- The directory structure:

```
src
├──── main
│      └──── scala
│             └──── WordCount.scala
└──── test
       └──── scala
              └──── WordCountSuite.scala
```

Unit Test

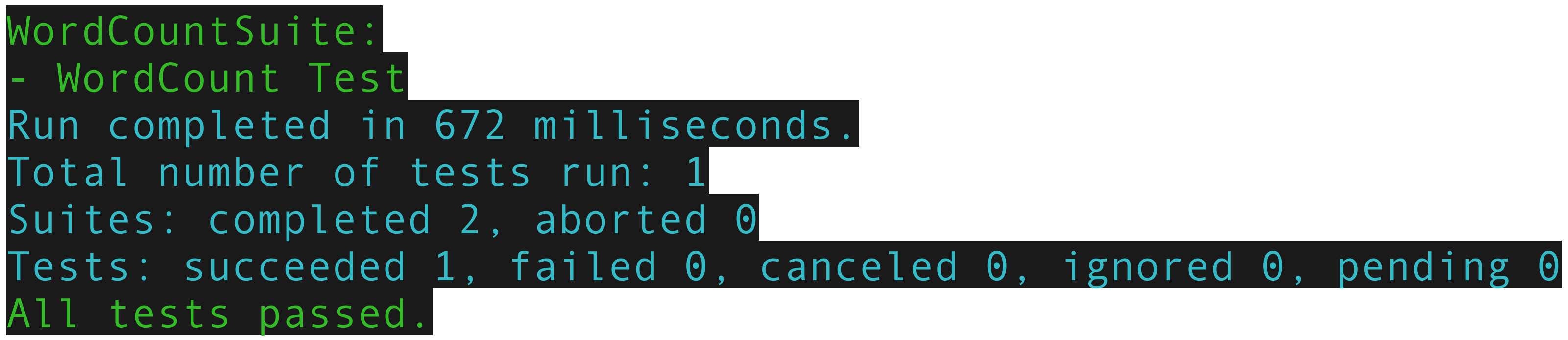
- Inside the test source file

```
cat src/test/scala/WordCountSuite.scala
import org.apache.spark.rdd.RDD
import org.apache.spark.SparkContext
import org.scalatest.FunSuite

class WordCountSuite extends FunSuite{
  test("WordCount Test"){
    val sc = new SparkContext("local", "test")
    val rdd = sc.parallelize(List(1,2,3,4))
    val res = rdd.map(WordCount.func(_))
    assert(res.collect.sameElements(Array(1,4,9,16)))
  }
}
```

- Inside the code source file

```
object WordCount {
  def main(args: Array[String]) {…}

  def func(x: Int) = x*x
}
```

- Then run “mvn test”



```
WordCountSuite:
- WordCount Test
Run completed in 672 milliseconds.
Total number of tests run: 1
Suites: completed 2, aborted 0
Tests: succeeded 1, failed 0, canceled 0, ignored 0, pending 0
All tests passed.
```

Self Checklist

- What is an RDD?
- RDD transformations, actions?
- How to build and submit a Spark standalone application?
- How to do unit test with Spark?
- Where to find useful information about Spark and RDD?