Scala Simple Tutorial

CS-562

Introduction to data analysis using Apache Spark

Instructor: Kondylakis Haridimos

T.A.: Sideri Sophia

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.

Java vs Scala WordCount

```java
public class WordCountJava {
    public static void main(String[] args) {
        StringTokenizer st
                = new StringTokenizer(args[0]);
        Map<String, Integer> map =
                new HashMap<String, Integer>();
        while (st.hasMoreTokens()) {
            String word = st.nextToken();
            Integer count = map.get(word);
            if (count == null)
                map.put(word, 1);
            else
                map.put(word, count + 1);
        }
        System.out.println(map);
    }
}
```

```scala
object WordCountScala extends App {
  println(
    args(0)
    .split(" ")
    .groupBy(x => x)
    .map(t => t._1 -> t._2.length))
}
```

```
> runMain WordCountJava "a b a c a b"
[info] Running WordCountJava a b a c a b
{a=3, b=2, c=1}
```

```
> runMain WordCountScala "a b a c a b"
[info] Running WordCountScala a b a c a b
Map(b -> 2, a -> 3, c -> 1)
```

var vs val

- Var -> **Var** - iable

- Val -> **V**ariable + Fin**al**

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.

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

Zip & ZipWithIndex

Zip

Returns a list formed from this list and another iterable collection by combining corresponding elements in pairs.
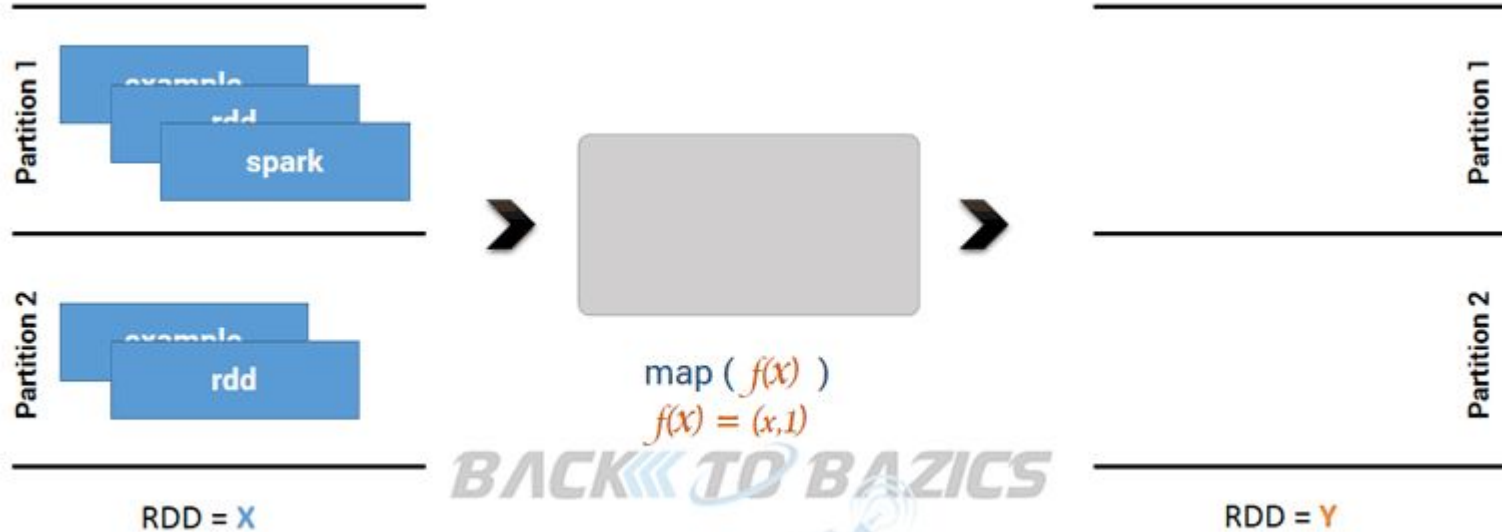
zipWithIndex

Zips this list with its indices.
Returns: A new list containing pairs consisting of all elements of this list paired with their index. Indices start at 0.
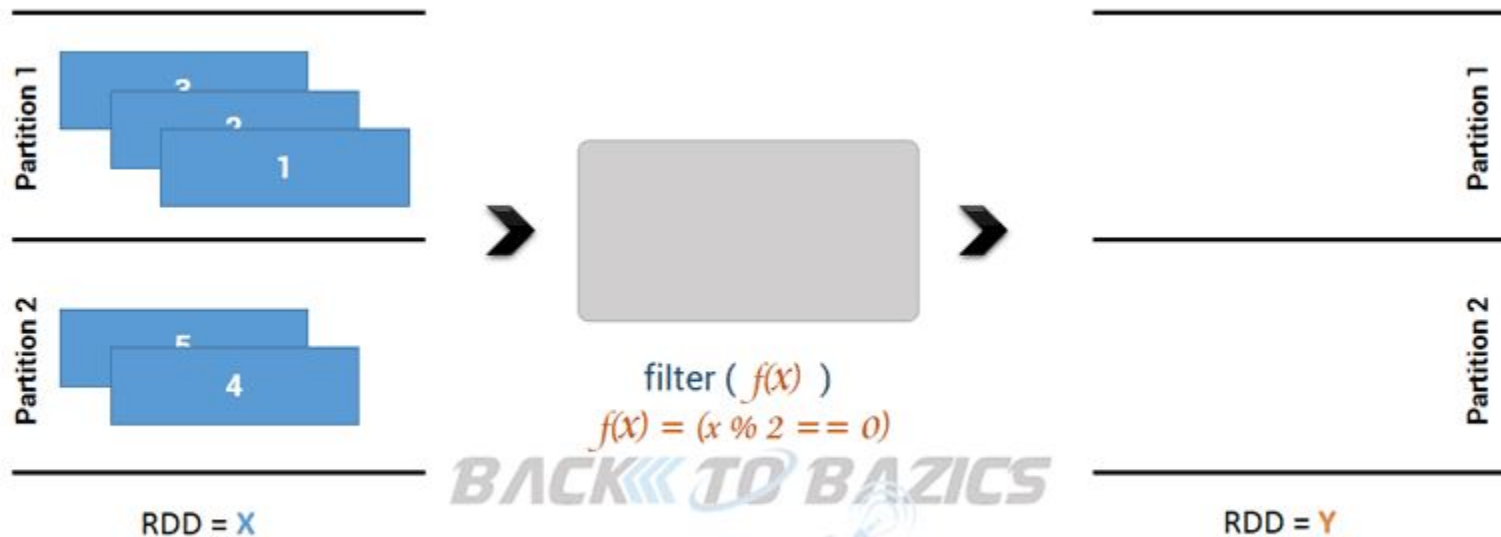
Map

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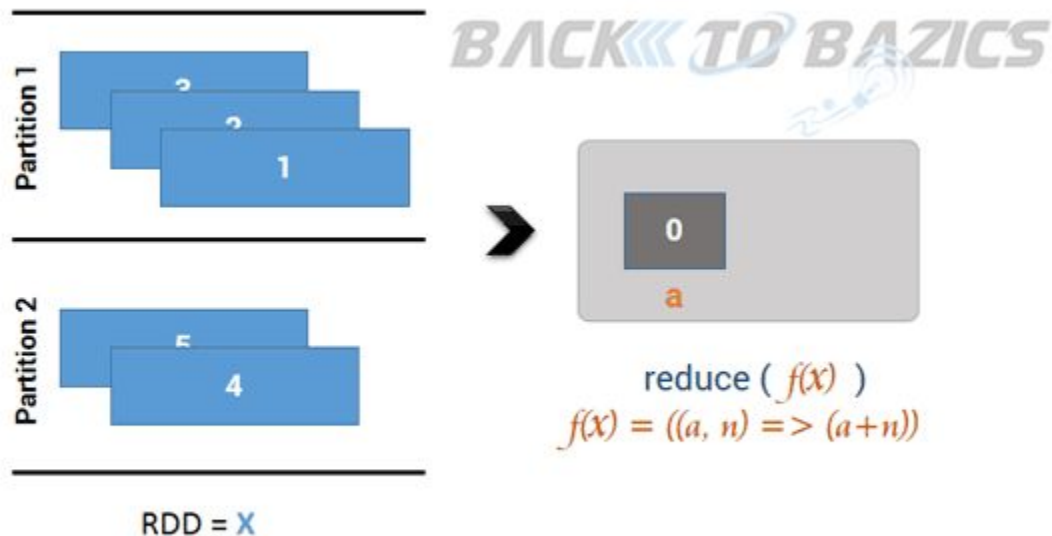
Filter

Spark RDD filter function returns a new RDD containing only the elements that satisfy a predicate.
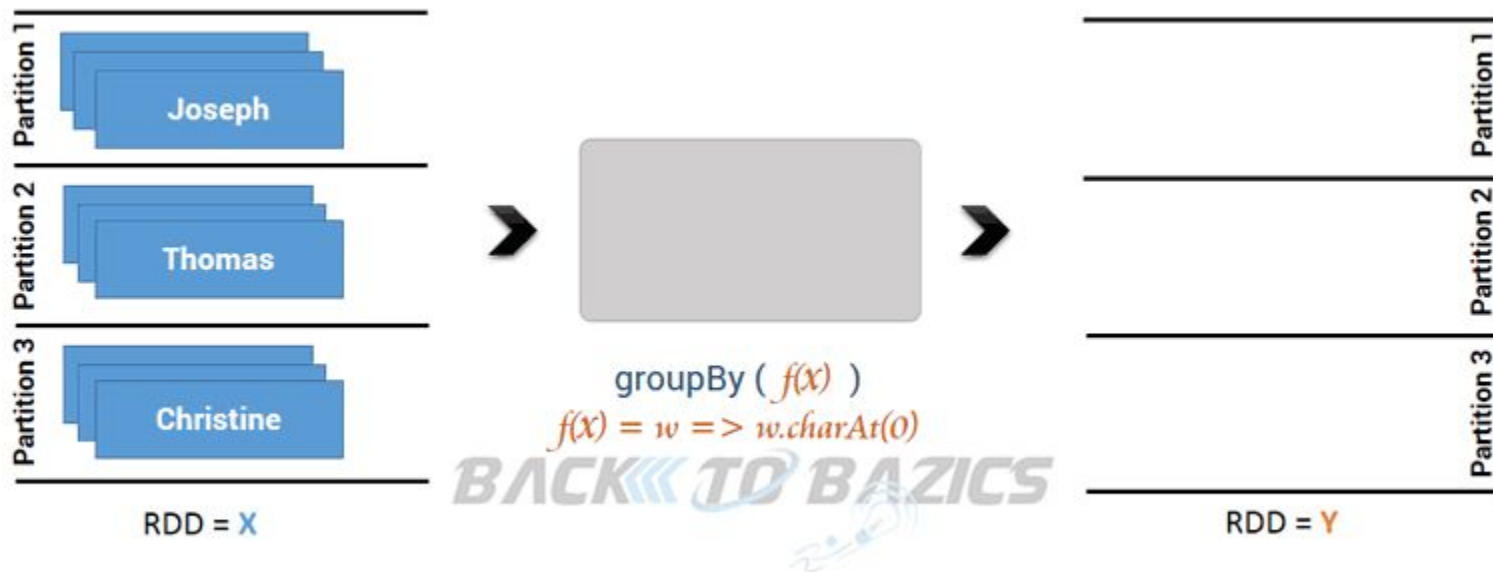
Reduce

Spark RDD reduce function reduces the elements of this RDD using the specified commutative and associative binary operator.

GroupBy

Spark RDD groupBy function returns an RDD of grouped items.

$$groupBy\ (\ f(x)\ )$$
$$f(x) = w \implies w.charAt(0)$$

RDD = X

RDD = Y

Scala functions (closures)

```scala
(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
```

databricks™

Quick Tour of Scala Part 2

(electric boogaloo)

Processing collections with functional programming

```scala
val lst = List(1, 2, 3)
list.foreach(x => println(x)) // 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.

databricks™

Functional methods on collections

There are a lot of methods on Scala collections, just **google Scala Seq** or http://www.scala-lang.org/api/2.10.4/index.html#scala.collection.Seq

Method on Seq[T]	Explanation
map(f: T => U): Seq[U]	Each element is result of f
flatMap(f: T => Seq[U]): Seq[U]	One to many map
filter(f: T => Boolean): Seq[T]	Keep elements passing f
exists(f: T => Boolean): Boolean	True if one element passes f
forall(f: T => Boolean): Boolean	True if all elements pass
reduce(f: (T, T) => T): T	Merge elements using f
groupBy(f: T => K): Map[K, List[T]]	Group elements by f
sortBy(f: T => K): Seq[T]	Sort elements
…..	

Lets Make our First Crash-Test

Our Data:

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*

```scala
// load error messages from a log into memory
// then interactively search for various patterns
// https://gist.github.com/ceteri/8ae5b9509a08c08a1132

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

// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()

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

// action 2
messages.filter(_.contains("php")).count()
```
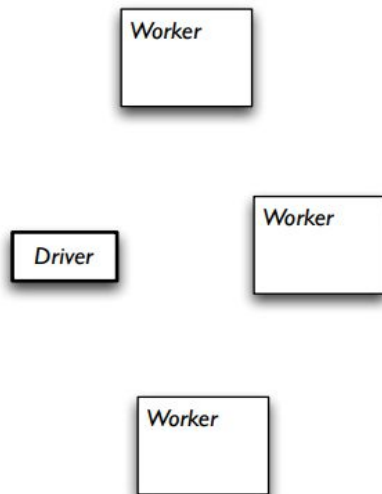
Spark Deconstructed: *Log Mining Example*

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

Worker

Worker

Driver

Worker

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("\t")).map(r => r(1))
messages.cache()

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

//
messages.filter(_.contains("php")).count()
```

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("\t")).map(r => r(1))
messages.cache()

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

// action 2
messages.filter(_.contains("php")).count()
```
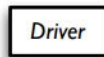
discussing the other part

Worker

Worker

Driver

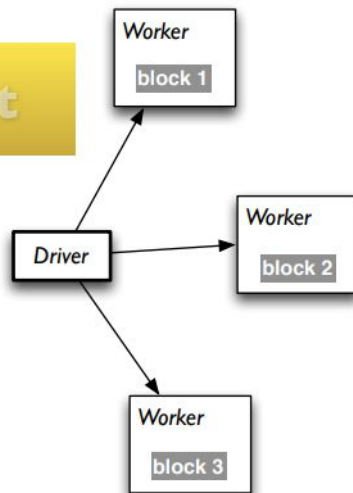Worker

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("\t")).map(r => r(1))
messages.cache()

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

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

discussing the other part

Worker
block 1

Driver

Worker
block 2

Worker
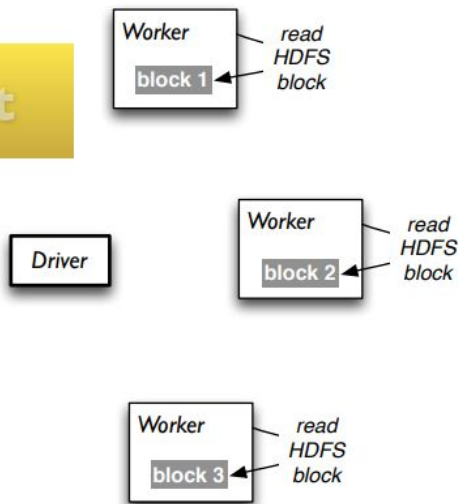block 3

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("\t")).map(r => r(1))
messages.cache()

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

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

discussing the other part
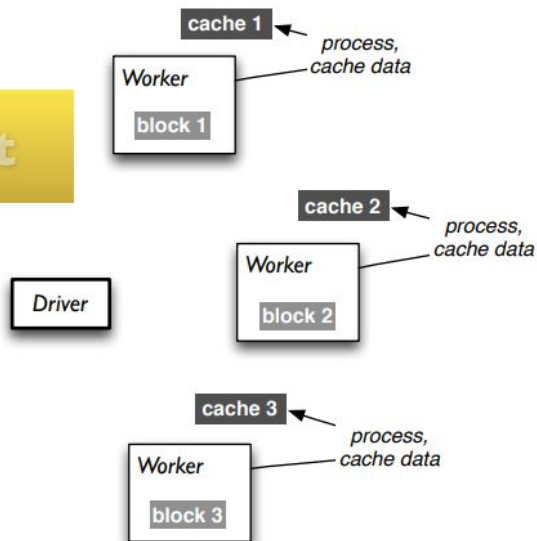
Spark Deconstructed: *Log Mining Example*

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

// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()

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

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

discussing the other part

Worker
block 1
read HDFS block

Driver

Worker
block 2
read HDFS block

Worker
block 3
read HDFS block
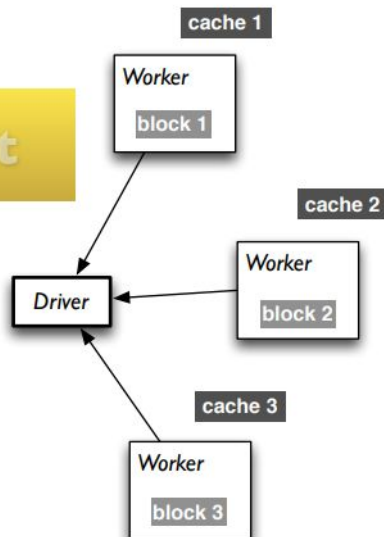
Spark Deconstructed: *Log Mining Example*

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

// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()

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

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

discussing the other part

cache 1

process, cache data

Worker

block 1

cache 2

process, cache data

Worker

block 2
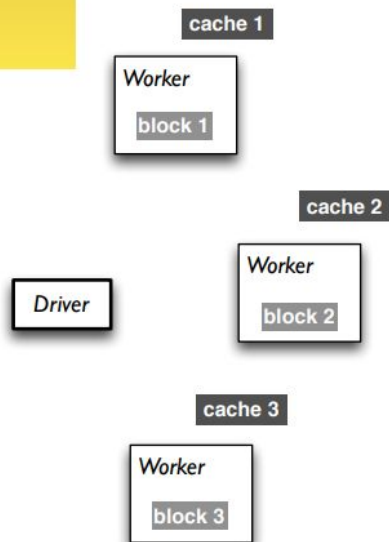
Driver

cache 3

process, cache data

Worker

block 3

Spark Deconstructed: *Log Mining Example*

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

// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()

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

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

discussing the other part

Spark Deconstructed: *Log Mining Example*

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

// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()

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

// action 2
messages.filter(_.contains("php")).count()
```
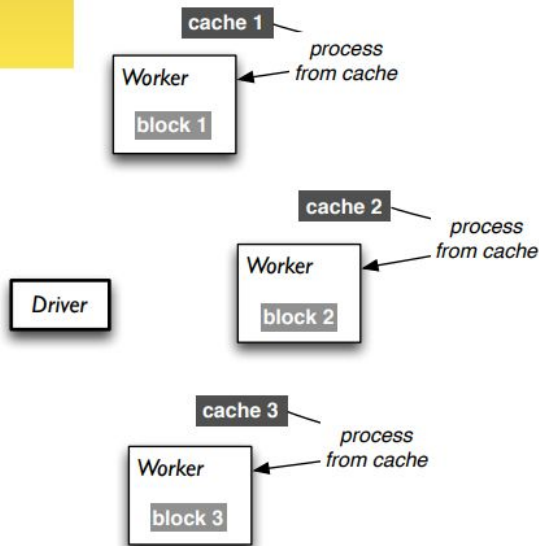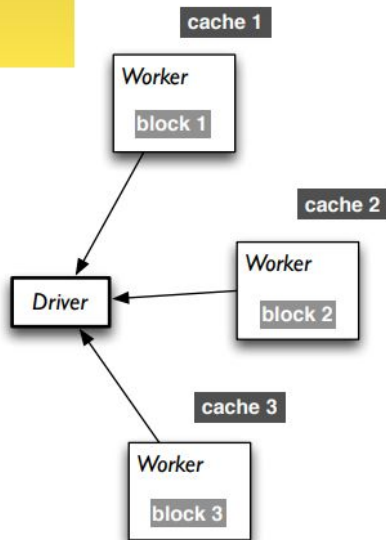
discussing the other part

cache 1

Worker

block 1

cache 2

Worker

block 2

Driver

cache 3

Worker

block 3

Spark Deconstructed: *Log Mining Example*

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

// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()

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

discussing the other part

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

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("\t")).map(r => r(1))
messages.cache()

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

discussing the other part

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

Spark Deconstructed:

Looking at the RDD transformations and actions from another perspective…

```
// load error messages from a log into memory
// then interactively search for various patterns
// https://gist.github.com/ceteri/8ae5b9509a08c08a1132

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

// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()

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

// action 2
messages.filter(_.contains("php")).count()
```
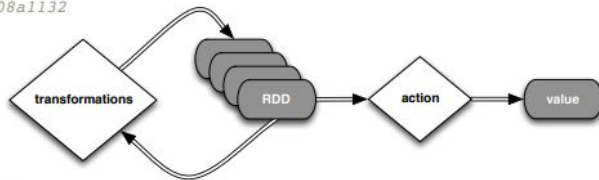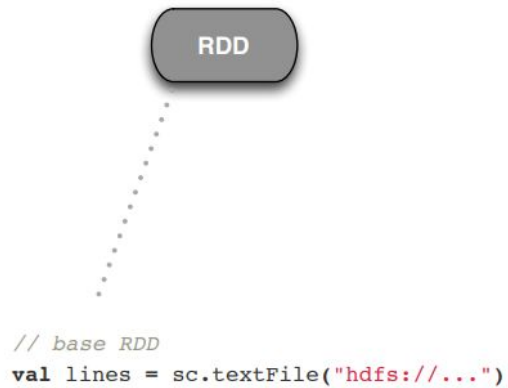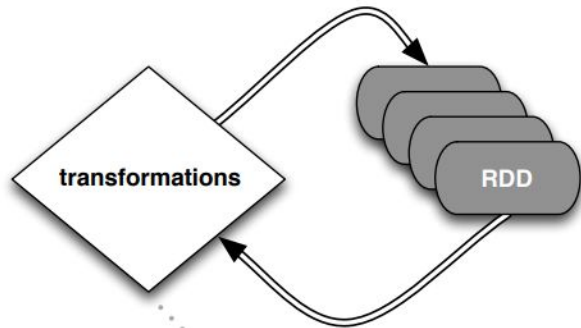
Spark Deconstructed:

```
RDD
```

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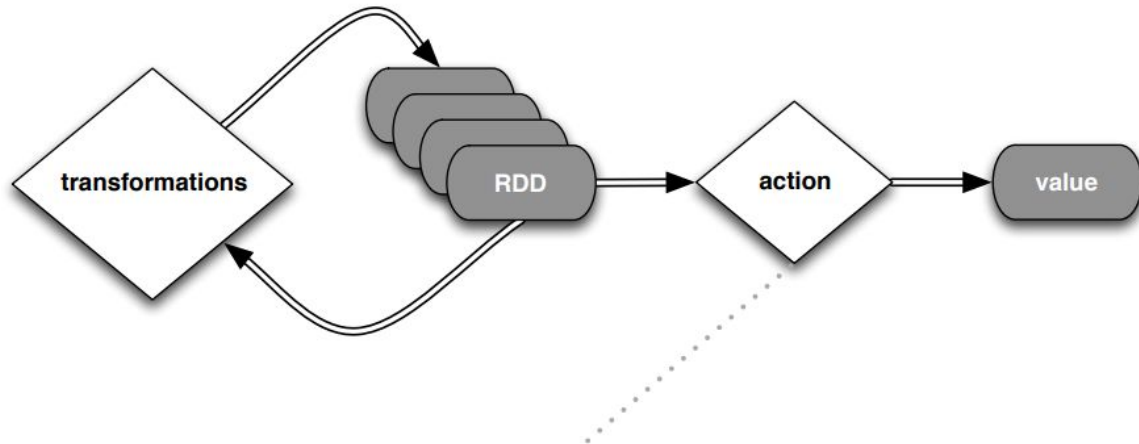
Spark Deconstructed:


```scala
// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()
```

Spark Deconstructed:


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

Special Thanks

- CS543 Presentations

- Coursera Introduction to Apache Spark,University of California,Databricks

- https://backtobazics.com/big-data/spark