SPARK TRANSFORMATION IN DEPTH

By www.HadoopExam.com

Note: These instructions should be used with the HadoopExam Apache Spark: Professional Trainings. Where it is executed and you can do hands on with trainer.

Transformation In RDD

map()
filter()
flatMap()

Element-wise Transformation:-

- Transformation works on each element
- Two most common transformation
  - map()
  - filter()

=> map() :

List [1,2,3,4].map(?)

① What do you want to do with each element of list

② I want to double each element

③ Replace ? with function, which can double an element

List [1,2,3,4].map(x => x*2)

↓

List [2,4,6,8]

Filter :

List[1,2,3,4,5,6].filter(?)

② What do you want to do with each element of list

② We want to filter/select values other than 3

③ Pass the function which filter values other than 3

$$x \Rightarrow x != 3$$

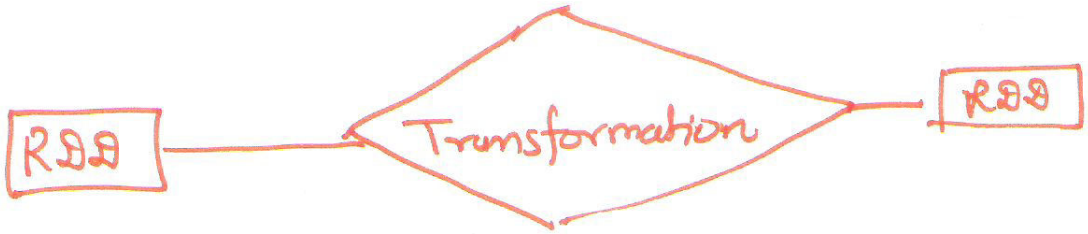
List[1,2,3,4,5,6].filter(x ⇒ x != 3)

Input RDD

⇓

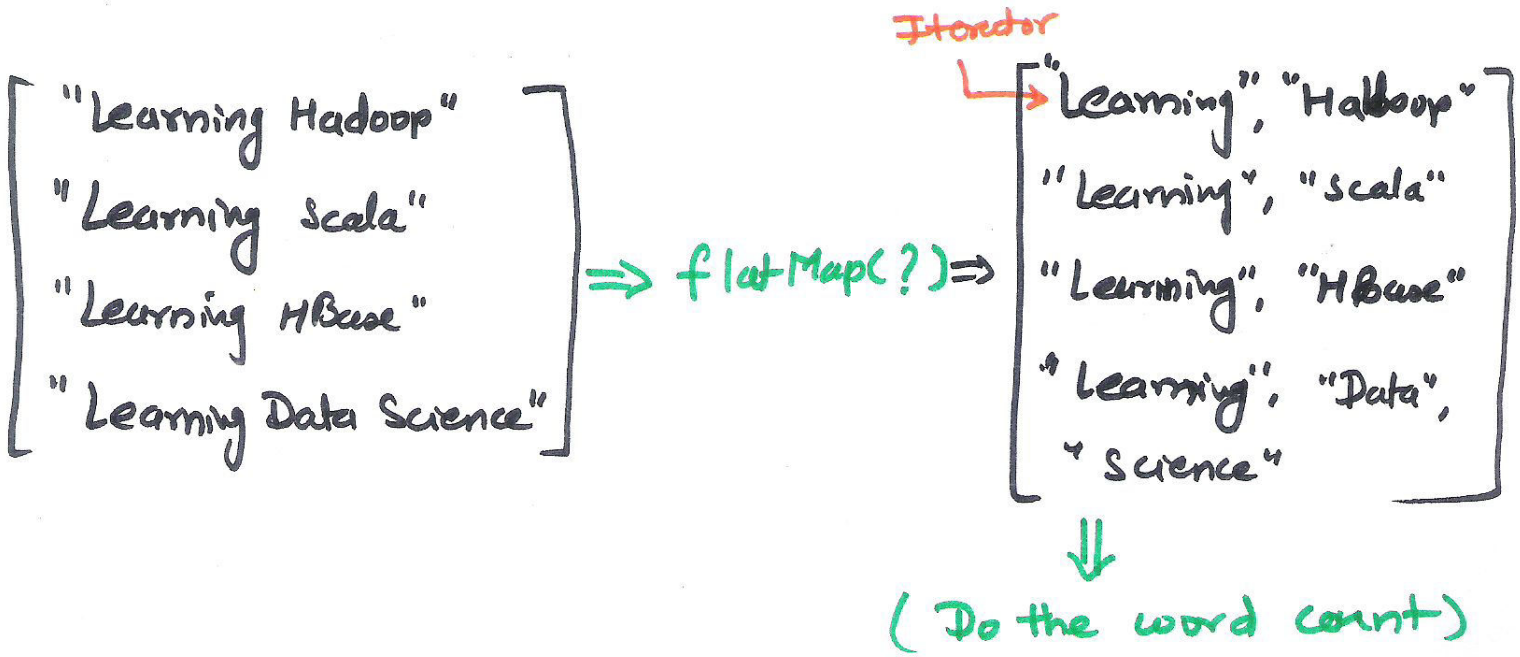
List[1,2,4,5,6] ← Output RDD

⇒ map() and filter() both are the function which independently applied on each element of input RDD



⇒ Return type does not have to be the same as its input type.

FlatMap: produce multiple output for each element



flatMap(?) ← Provide the function which will help to create multiple elements (Return iterator) e.g. Split() function

⇒ The function (?) we provide to flatMap() is called individually for each element to inputRDD

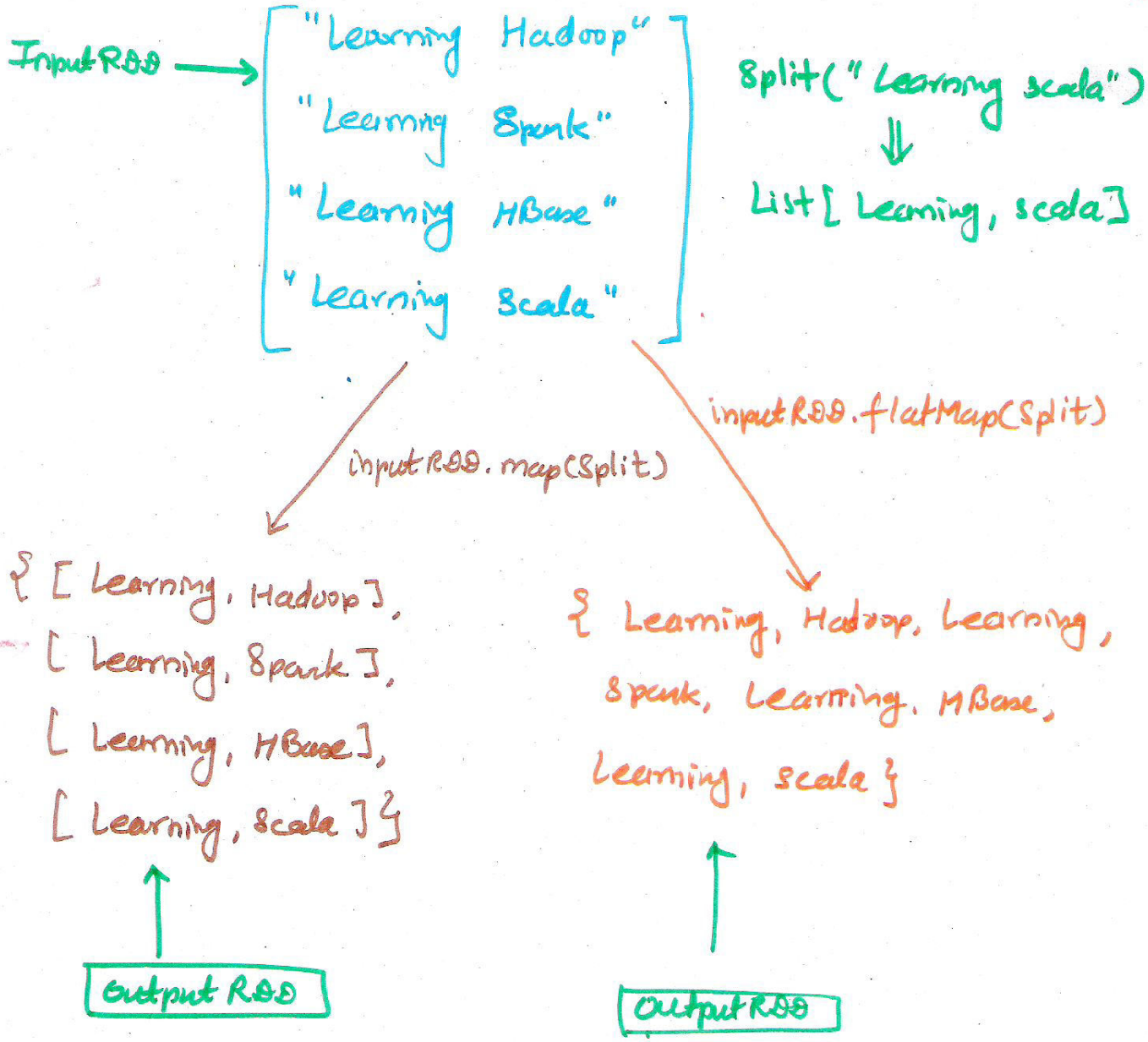
⇒ This "?" function will not return individual element, it return an iterator with return values.

flatMap [List [Iterator1, Iterator2, ... Iterator n]

⇒ Now this flatMap will not return list of iterators

⇒ It will return all the elements combined from all iterator.

→ Given ~~Sim~~ Sample split all the words from line


InputRDD
"Learning Hadoop"
"Learning Spark"
"Learning HBase"
"Learning Scala"
split("Learning scala")
List[Learning, scala]
inputRDD.map(Split)
inputRDD.flatMap(Split)
{[Learning, Hadoop],
[Learning, Spark],
[Learning, HBase],
[Learning, Scala]}
{Learning, Hadoop, Learning,
Spark, Learning, HBase,
Learning, Scala}
Output RDD
Output RDD

Hands on Exercise Lab

- Map function squaring the values

```
val input = sc.parallelize(List(1, 2, 3, 4))
val result = input.map(x => x * x)
println(result.collect().mkString(","))
```

- Splitting lines in multiple words.

```
val lines = sc.parallelize(List("hello HadoopExam.com", "hi"))
val words = lines.flatMap(line => line.split(" "))
words.first() // returns "hello"
```

- Return an RDD consisting of only elements that pass the condition passed to filter().

```
val result = input.filter(x => x != 1)
println(result.collect().mkString(","))
```

- Another example on map()

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

val result = l.map( x => x*2 )
println(result.collect().mkString(","))

def f(x: Int) = if (x > 2) Some(x) else None

val result = l.map(x => f(x))
println(result.collect().mkString(","))


def g(v:Int) = List(v-1, v, v+1)
val result = l.map(x => g(x))
println(result.collect().mkString(","))


val result = l.flatMap(x => g(x))
println(result.collect().mkString(","))
```

- Set operation on RDD

```
val rdd = sc.parallelize(List(1,2,3))
val other = sc.parallelize(List(3,4,5))

val result = rdd.union(other)
```

```
println(result.collect().mkString(","))

val result = rdd.intersection(other)
println(result.collect().mkString(","))

val result = rdd.subtract(other)
println(result.collect().mkString(","))


val result = rdd.cartesian(other)
println(result.collect().mkString(","))
```

ORACLE JAVA CERTIFICATION

- Java 1z0-808
- Java 1z0-809

ORACLE DATABASE CLOUD CERTIFICATION

- 1z0-060 (Oracle 12c)
- 1z0-061 (Oracle 12c)