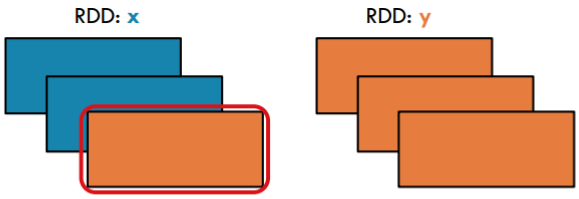
University of
South Australia

Advanced Transformations and actions

1

Map is your best friend

MAP



map(*f*, *preservesPartitioning=False*)

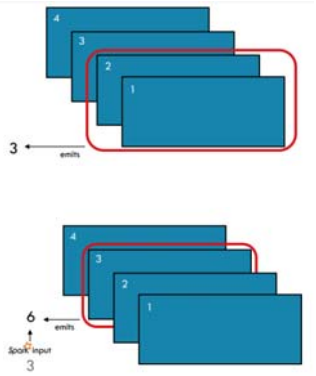
Return a new RDD by applying a function to each element of this RDD

```
x = sc.parallelize(["b", "a", "c"])
y = x.map(lambda z: (z, 1))
print(x.collect())
print(y.collect())
```

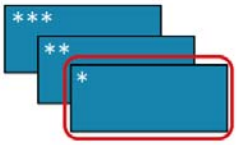
```
x: ['b', 'a', 'c']
y: [('b', 1), ('a', 1), ('c', 1)]
```


2

Reduce is a bit trickier…



REDUCE



reduce(*f*)

Aggregate all the elements of the RDD by applying a user function pairwise to elements and partial results, and returns a result to the driver

```
x = sc.parallelize([1,2,3,4])
y = x.reduce(lambda a,b: a+b)

print(x.collect())
print(y)
```

```
val x = sc.parallelize(Array(1,2,3,4))
```

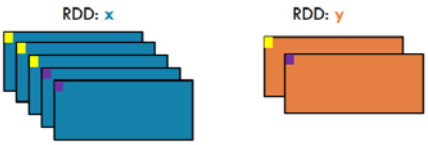
```
x: [1, 2, 3, 4]
y: 10
```

University of South Australia

3

groupByKey is tricky in a different way

GROUPBYKEY



groupByKey(*numPartitions=None*)

Group the values for each key in the original RDD. Create a new pair where the original key corresponds to this collected group of values.

```
x = sc.parallelize([('B',5),('B',4),('A',3),('A',2),('A',1)])
y = x.groupByKey()
print(x.collect())
print(list((j[0], list(j[1])) for j in y.collect()))
```

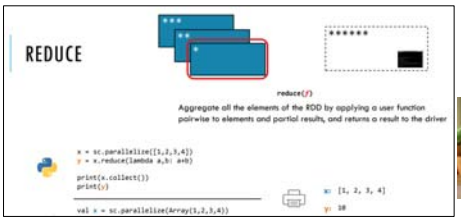
```
x: [('B', 5),('B', 4),('A', 3),('A', 2),('A', 1)]
y: [('A', [2, 3, 1]),('B',[5, 4])]
```

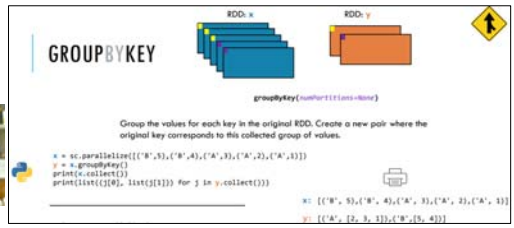
Output key is standard, value is an iterable object

z = y.map(lambda x: (x[0],list(x[1])))

University of South Australia

4





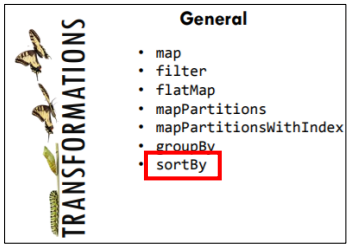
reduceByKey

x = sc.parallelize([('B',5),('B',4),('A',3),('A',2),('A',1)])
y = x.reduceByKey(lambda v1,v2: v1+v2)
print(y.collect())

y = [(A,6),(B,9)]

Some useful transformations are not included in the databricks resource. For example:

sortBy



x = sc.parallelize([('B',5),(1,4),('A',3),(5,2),(4,1)])

y = x.sortBy(lambda x: x[0])
print(y.collect())
y = [(1,4),(4,1),(5,2),('A',3),('B',5)]

z = x.sortBy(lambda x: x[1])
print(z.collect())
Z = [(4,1),(5,2),('A',3),(1,4),('B',5)]

A final note on PySpark…

University of South Australia

7