# Key-Value Operations

DSC 232R

# Operations on (key,val) RDDs

RDDs are similar to Python lists in that each element can be of a different type and size.

(key.value) RDDs are useful to store and retrieve records. Examples:

- Key = SNN, Value = personal information
- Key = (Longitude, Latitude), Value = address
- Key = word, Value = number of occurrences

# 1.1 Three types of Spark operations

- Transformations
- Actions
- Shuffles

# **Transformations:** RDD → RDD.

- **Examples:** map, filter, sample, more
- **No** communication needed.

# **Actions:** RDD → Python-object in head node.

- **Examples:** reduce, collect, count, take, more
- **Some** communication needed.

# **Shuffles:** RDD → RDD, **shuffle** needed

- **Examples:** sort, distinct, repartition, sortByKey, reduceByKey, join more
- **A LOT** communication might be needed.

# 1.4 Transformations on (key,value) rdds

# 1.4.1 reduceByKey(func)

Apply the reduce function on the values with the same key.

```
In [4]: rdd = sc.parallelize([(1,2), (2,4), (2,6)])
        print("Original RDD :", rdd.collect())
        print("After transformation : ", rdd.reduceByKey(lambda a,b: a+b).collect())

Original RDD : [(1, 2), (2, 4), (2, 6)]
After transformation :  [(1, 2), (2, 10)]
```

# 1.4.2 sortByKey():

Sort RDD by keys in ascending order.

```
In [5]: rdd = sc.parallelize([(2,2), (1,4), (3,6)])
        print("Original RDD :", rdd.collect())
        print("After transformation : ", rdd.sortByKey().collect())
```

```
Original RDD : [(2, 2), (1, 4), (3, 6)]
After transformation :  [(1, 4), (2, 2), (3, 6)]
```

# 1.4.3 mapValues(func):

Apply func to each value of RDD without changing the key.

```
In [6]: rdd = sc.parallelize([(1,2), (2,4), (2,6)])
        print("Original RDD :", rdd.collect())
        print("After transformation : ", rdd.mapValues(lambda x: x*2).collect())

Original RDD : [(1, 2), (2, 4), (2, 6)]
After transformation :  [(1, 4), (2, 8), (2, 12)]
```

# 1.5 Transformations on two (key-value) RDDs

# 1.5.1 1. subtractByKey:

Remove from RDD1 all elements whose key is present in RDD2.

```
In [11]: print('rdd1=',rdd1.collect())
         print('rdd2=',rdd2.collect())
         print("Result:", rdd1.subtractByKey(rdd2).collect())
```

```
rdd1= [(1, 2), (2, 1), (2, 2)]
rdd2= [(2, 5), (3, 1)]
Result: [(1, 2)]
```

# 1.5.2 2. join:

- A fundamental operation in relational databases.
- Assumes two tables have a **key** column in common.
- Merges rows with the same key.

Suppose we have two `(key,value)` datasets.

| dataset 1 | | ..... | dataset 2 | |
|---|---|---|---|---|
| key=name | (gender,occupation,age) | | key=name | hair color |
| John | (male,cook,21) | | Jill | blond |
| Jill | (female,programmer,19) | | Grace | brown |
| John | (male, kid, 2) | | John | black |
| Kate | (female, wrestler, 54) | | | |

# Accessible Table of Page 14

| Dataset 1 | | Dataset 2 | |
|---|---|---|---|
| key=name | (gender, occupation, age) | key=name | hair color |
| John | (male, cook, 21) | Jil | blond |
| Jill | (female, programmer, 19) | Grace | brown |
| John | (male, kid, 2) | John | black |
| Kate | (female, wrestler, 54) | | |

When `Join` is called on datasets of type `(Key, V)` and `(Key, W)`, it returns a dataset of `(Key, (V, W))` pairs with all pairs of elements for each key. Joining the 2 datasets above yields:

| key = name | (gender,occupation,age),haircolor |
|---|---|
| John | ((male,cook,21),black) |
| John | ((male, kid, 2),black) |
| Jill | ((female,programmer,19),blond) |

# Accessible Table of Page 16

| key = name | (gender, occupation, age), hair color |
|---|---|
| John | ((male, cook, 21), black) |
| John | ((male, kid, 2), black) |
| Jill | ((female, programmer, 19), blond) |

```
In [12]: print('rdd1=',rdd1.collect())
         print('rdd2=',rdd2.collect())
         print("Result:", rdd1.join(rdd2).collect())
```

```
rdd1= [(1, 2), (2, 1), (2, 2)]
rdd2= [(2, 5), (3, 1)]
Result: [(2, (1, 5)), (2, (2, 5))]
```

# 1.7 Actions on (key,val) RDDs

# 1.7.1 1. countByKey():

Count the number of elements for each key. Returns a dictionary for easy access to keys.

```
In [16]: print("RDD: ", rdd.collect())
         result = rdd.countByKey()
         print("Result:", result)

RDD:  [(1, 2), (2, 4), (2, 6)]
Result: defaultdict(<class 'int'>, {1: 1, 2: 2})
```

# 1.7.2 2. collectAsMap():

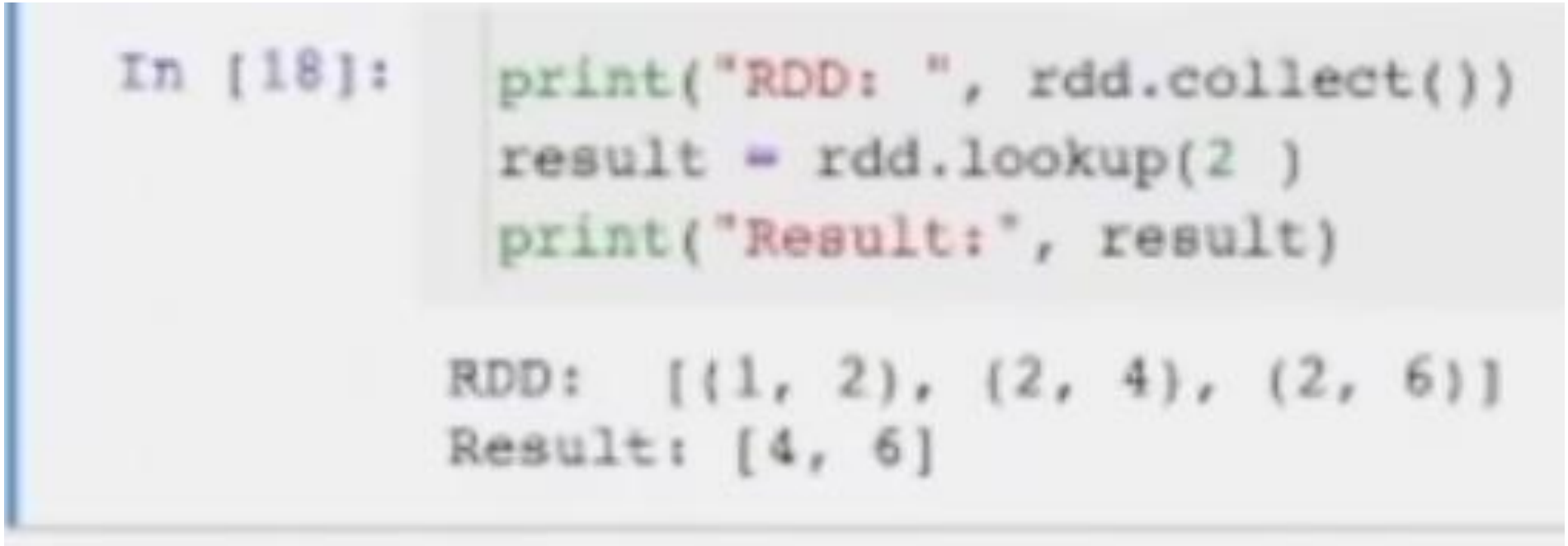
Collect the result as a dictionary to provide easy lookup.

```
In [17]: print("RDD: ", rdd.collect())
         result = rdd.collectAsMap()
         print("Result:", result)

RDD:  [(1, 2), (2, 4), (2, 6)]
Result: {1: 2, 2: 6}
```

# 1.7.3 3. lookup(key):

Return all values associated with the provided key.



# 2 Summary

- We saw some more of the operations on Pair RDDs.
- For more, see the Spark RDD programming guide.
- An example of an actual Spark program: Word-Count.