5f_transformations_set_operations

February 21, 2022

0.0.1 Understanding set operations

[1]:
```
# Import SparkContext and SparkConf
from pyspark import SparkContext, SparkConf

# Initialize spark
conf = SparkConf().setAppName("union")
sc = SparkContext(conf=conf)
```

```
22/02/21 14:31:16 WARN Utils: Your hostname, ThinkCentre resolves to a loopback
address: 127.0.1.1; using 10.180.5.223 instead (on interface eno1)
22/02/21 14:31:16 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another
address
22/02/21 14:31:17 WARN NativeCodeLoader: Unable to load native-hadoop library
for your platform… using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
setLogLevel(newLevel).
22/02/21 14:31:18 WARN Utils: Service 'SparkUI' could not bind on port 4040.
Attempting port 4041.
22/02/21 14:31:18 WARN Utils: Service 'SparkUI' could not bind on port 4041.
Attempting port 4042.
22/02/21 14:31:18 WARN Utils: Service 'SparkUI' could not bind on port 4042.
Attempting port 4043.
22/02/21 14:31:18 WARN Utils: Service 'SparkUI' could not bind on port 4043.
Attempting port 4044.
22/02/21 14:31:18 WARN Utils: Service 'SparkUI' could not bind on port 4044.
Attempting port 4045.
22/02/21 14:31:18 WARN Utils: Service 'SparkUI' could not bind on port 4045.
Attempting port 4046.
```

[2]:
```
# Create two RDDs from two lists
L1 = [1,2,3,4]
L2 = [2,4,6,8]

# Create two RDDs
num1RDD = sc.parallelize(L1,4)
num2RDD = sc.parallelize(L2,4)
```

[3]:
```
# 1. Union
# Signature:
#       num1RDD.union(other)
# Return the union of this RDD and another one.
num1RDD.union(num2RDD).collect() # Duplicates are not removed
```

[3]:
```
[1, 2, 3, 4, 2, 4, 6, 8]
```

[4]:
```
# 2. Join - Joins values of the matching key

# Signature: xRDD.join(yRDD, numPartitions=None)

# Return an RDD containing all pairs of elements with matching keys in
# C{self} and C{other}.

# Each pair of elements will be returned as a (k, (v1, v2)) tuple,
# where (k, v1) is in C{self} and (k, v2) is in C{other}.

xRDD = sc.parallelize([("a", 1), ("b", 4)])
yRDD = sc.parallelize([("a", 2), ("a", 3), ("b", 5)])
xRDD.join(yRDD).collect()
```

[4]:
```
[('a', (1, 2)), ('a', (1, 3)), ('b', (4, 5))]
```

[5]:
```
# 3. distinct
# Signature: num1RDD.distinct(numPartitions=None)
#
# Return a new RDD containing the distinct elements in this RDD.

L1 = [1,1,2,2,3]
num1RDD = sc.parallelize(L1,2)
num1RDD.distinct(4).collect()
```

[5]:
```
[1, 2, 3]
```

[6]:
```
# 4. coalesce - Reduce the number of partitions by combining them
# num1RDD.coalesce(numPartitions, shuffle=False)
# Return a new RDD that is reduced into `numPartitions` partitions.

L1 = [1,2,3,4,5,6,7,8]
num1RDD = sc.parallelize(L1,4)
num1RDD.glom().collect()
```

[6]: `[[1, 2], [3, 4], [5, 6], [7, 8]]`

[7]:
```
# coalesce 4 partitions into two partitions
num1RDD.coalesce(2).glom().collect()
```

[7]: `[[1, 2, 3, 4], [5, 6, 7, 8]]`

[8]:
```
# Get the number of partitions in a RDD
L1 = [1,2,3,4,5,6,7,8]
num1RDD = sc.parallelize(L1,8)
num1RDD.getNumPartitions() # Not a transformation
```

[8]: `8`