5a_Transformations_map_flatmap

February 21, 2022

0.1 Demonstrating transformations of RDDs

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

[2]:
```python
# Initialize spark
conf = SparkConf().setAppName("LearnTransormations")
sc = SparkContext(conf=conf)
```

```
22/02/21 11:56:06 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 11:56:06 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another
address
22/02/21 11:56:07 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 11:56:08 WARN Utils: Service 'SparkUI' could not bind on port 4040.
Attempting port 4041.
```

0.1.1 1. map

[3]:
```python
# Example1 (A) : Subtract 1 from every number of a list
# Using lambda expression

# Create an RDD
data = [1, 2, 3, 4, 5, 6, 7]
RDD1 = sc.parallelize(data, 4)

# Transform xrangeRDD through map transformation using subtract lambda function
subRDD = RDD1.map(lambda x: x-1)

# Collect the results to driver
subRDD.collect()
```

```
[3]: [0, 1, 2, 3, 4, 5, 6]
```

```
[5]: # Example1 (B) : Subtract 1 from every number of a list

     # Using Functions - We can pass a function to a transformation

     # Create an RDD
     data = [1, 2, 3, 4, 5, 6, 7]
     RDD1 = sc.parallelize(data, 4)
```

```
[4]: # Create a function called sub, which subtracts 1 from each element
     def sub(x):
         return x-1
```

```
[6]: # Transform xrangeRDD through map transformation using sub function
     subRDD = RDD1.map(sub)
```

```
[7]: # Collect the results to driver
     subRDD.collect()
```

```
[7]: [0, 1, 2, 3, 4, 5, 6]
```

```
[8]: # Check: What is the number of elements in a transformed RDD after map?

     # Example1 (C) : Remove all the odd numbers

     # Using lambda expression

     # Create an RDD
     data = [1, 2, 3, 4, 5, 6, 7]
     RDD1 = sc.parallelize(data, 4)
```

```
[9]: # Function to return only even numbers
     def even(x):
         if x%2 == 0:
             return x
```

```
[10]: # Transform xrangeRDD through map transformation using sub function
      evenRDD = RDD1.map(even)

      # Collect the results to driver
      evenRDD.collect()
```

```
[10]: [None, 2, None, 4, None, 6, None]
```

0.1.2 2. flatmap

[11]:
```
# Bazic map example in python
x = sc.parallelize(["Apache Spark", "Flat Map"], 2)

# map operation will return List of Lists in following case (check the result)
y = x.map(lambda x: x.split(' '))
y.collect()
```

[11]:
```
[['Apache', 'Spark'], ['Flat', 'Map']]
```

[12]:
```
# flatMap operation will return List of words in following case (check the␣
 ↪result)
y = x.flatMap(lambda x: x.split(' '))
y.collect()
```

[12]:
```
['Apache', 'Spark', 'Flat', 'Map']
```