6_actions

February 21, 2022

1 Actions on RDDs

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

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

```
22/02/21 14:30:44 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 14:30:44 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another
address
22/02/21 14:30:44 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 14:30:46 WARN Utils: Service 'SparkUI' could not bind on port 4040.
Attempting port 4041.
22/02/21 14:30:46 WARN Utils: Service 'SparkUI' could not bind on port 4041.
Attempting port 4042.
22/02/21 14:30:46 WARN Utils: Service 'SparkUI' could not bind on port 4042.
Attempting port 4043.
22/02/21 14:30:46 WARN Utils: Service 'SparkUI' could not bind on port 4043.
Attempting port 4044.
22/02/21 14:30:46 WARN Utils: Service 'SparkUI' could not bind on port 4044.
Attempting port 4045.
22/02/21 14:30:46 WARN Utils: Service 'SparkUI' could not bind on port 4045.
Attempting port 4046.
22/02/21 14:30:46 WARN Utils: Service 'SparkUI' could not bind on port 4046.
Attempting port 4047.
```

Action 1: reducce

Syntax: num1RDD.reduce(f) Reduces the elements of this RDD using the specified commutative and associative binary operator. Currently reduces partitions locally.

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

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

# Find the sum of all elements of L1
from operator import add

# Using inbuilt function add
print(num1RDD.reduce(add))
```

```
[Stage 0:>                                                          (0 + 4) / 4]
```

```
10
```

[3]:
```
# Usinf lambda experession
num1RDD.reduce(lambda x,y: x+y)
```

[3]:
```
10
```

2. Count

[4]:
```
# Number of elements in a RDD
num1RDD.count()
```

[4]:
```
4
```

3. max, min, sum, stdev, collect, take

[5]:
```
num1RDD.max()
```

[5]:
```
4
```

[6]:
```
num1RDD.min()
```

[6]:
```
1
```

[7]:
```
num1RDD.sum()
```

[7]:
```
10
```

[8]:
```
num1RDD.stdev()
```

[8]:
```
1.118033988749895
```

[9]:
```
num1RDD.collect()
```

[9]:
```
[1, 2, 3, 4]
```

[10]:
```
num1RDD.take(2)
```

[10]:
```
[1, 2]
```

countByValue, first, takeSample

[11]:
```
# countByValue
repetitiveRDD = sc.parallelize([1, 2, 3, 3, 1, 2, 1, 2, 3, 4, 5, 4, 6])
repetitiveRDD.countByValue()
```

[11]:
```
defaultdict(int, {1: 3, 2: 3, 3: 3, 4: 2, 5: 1, 6: 1})
```

[12]:
```
# first

L1 = [1,2,3,4]

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

[12]:
```
1
```

[13]:
```
# takeSample

# Signature: num1RDD.takeSample(withReplacement, num, seed=None)

# Return a fixed-size sampled subset of this RDD.

#.. note:: This method should only be used if the resulting array is expected
#    to be small, as all the data is loaded into the driver's memory.


L1 = [1,2,3,4]

# Create two RDDs
num1RDD = sc.parallelize(L1,4)
num1RDD.takeSample(False, 2, 100)
```

[13]:
```
[1, 3]
```