RDA Assignment 1

April 26, 2021

[6]:
```
import os

import findspark
import pyspark
import time
from operator import add
```

[4]:
```
from pyspark import SparkConf
from pyspark import SparkContext

conf = SparkConf()
conf.setMaster("local")
conf.setAppName("spark-basic")
sc = SparkContext(conf = conf)#Start the environment/context.
```

1 Another Exercise!¶

Let's code a shopping list! We will have a list of elements like this:

x = sc.parallelize([["Apple",3,0.2],["Pear",5,0.35],["Milk",2,1.1],["Apple",3,0.2]])

Where the first element of each list is the product, the second the number of unit we bought and the third the unit price.

We want to have the list of how much we have spent in each product (ordered), and the total amount of money we have spent.

(Optional) If we buy more than 10 products of the same type, we have a 10% discount of the final price

[7]:
```
x = sc.parallelize([["Apple",3,0.2],["Pear",5,0.35],["Milk",2,1.1],["Apple",3,0.
↪2]])
rdd = x.map(lambda x:(x[0], x[1]*x[2])).reduceByKey(add)
rdd.sortBy(lambda x: x[1], False).collect()
```

[7]:
```
[('Milk', 2.2), ('Pear', 1.75), ('Apple', 1.2000000000000002)]
```

2 Last one...

Replicate the last exercise, but the structure of the data is different. We have one object with products and prices. On the other hand, we have one list of the following form:

x = sc.parallelize([[“Maria”,“Apple”,1],[“Maria”,“Pear”,2],[“Pau”,“Milk”,4],[“Laura”,“Apple”,3]])

We want to know how much each of the have spent in total.

[9]:
```
y = sc.parallelize([["Apple",3,0.2],["Pear",5,0.35],["Milk",2,1.1],["Apple",3,0.
↪2]])
y = y.map(lambda x: (x[0],x[2])).distinct()
```

[10]:
```
y.collect()
```

[10]:
```
[('Apple', 0.2), ('Pear', 0.35), ('Milk', 1.1)]
```

[11]:
```
x = sc.
↪parallelize([["Maria","Apple",1],["Maria","Pear",2],["Pau","Milk",4],["Laura","Apple",3]])
z = x.map(lambda x: (x[1],x[0],x[2])).join(y)#.collect()
```

[12]:
```
z.collect()
```

[12]:
```
[('Apple', ('Maria', 0.2)),
 ('Apple', ('Laura', 0.2)),
 ('Milk', ('Pau', 1.1)),
 ('Pear', ('Maria', 0.35))]
```

[13]:
```
z.map(lambda x:x[1]).collect()
```

[13]:
```
[('Maria', 0.2), ('Laura', 0.2), ('Pau', 1.1), ('Maria', 0.35)]
```

[14]:
```
z.reduceByKey(lambda x,y:x).collect()
#z.groupByKey().mapValues(list).collect()
```

[14]:
```
[('Apple', ('Maria', 0.2)), ('Milk', ('Pau', 1.1)), ('Pear', ('Maria', 0.35))]
```

[ ]:

[ ]:

3 Exercise

- A proposal of a MapReduce process and its implementation in Pyspark. Doesn’t have to be really complex. Think about the example we worked with last week (counting words), and create something similar.

Read a Barcelona match plays file and determine if Messi or Griezman appeared the most on the game.

[15]:
```
rdd=sc.textFile('en-vivo.txt')
```

[16]:
```
def lenguajes_map(x):
    if "Messi" in x:
        return ("Count",(1,0))
    elif "Griezman" in x:
        return ("Count", (0,1))
    else:
        return ("Count",(0,0))
```

[17]:
```
rdd.map(lenguajes_map).take(5)
```

[17]:
```
[('Count', (0, 0)),
 ('Count', (0, 0)),
 ('Count', (0, 0)),
 ('Count', (0, 0)),
 ('Count', (0, 0))]
```

[18]:
```
rdd.map(lenguajes_map).reduceByKey(lambda x,y:(x[0]+y[0], x[1]+y[1])).collect()
```

[18]:
```
[('Count', (7, 11))]
```

[ ]:
```

```