first_pyspark

FINISHED

Prise en main de PySpark

Ceci est un support de cours ;-).

Les Resilients distributed datasets (RDD). Kesako?

Un RDD est une collection partitionnée d’enregistrements en lecture seule qui ne peut être créée que par des opérations déterministes. En gros, analogiquement, c'est une liste partitionnée dont les morceaux vivent dans la RAM des différents **workers** spark d'un cluster.

Took 0 sec. Last updated by anonymous at August 20 2019, 3:16:15 PM.

FINISHED

```
%spark2.pyspark

# la methode parallelize converti la liste en un RDD
words = sc.parallelize (
   ["scala",
   "java",
   "hadoop",
   "spark",
   "akka",
   "spark vs hadoop",
   "pyspark",
   "pyspark and spark"]
)
```

Took 0 sec. Last updated by anonymous at August 20 2019, 4:07:47 PM.

SPARK JOB (http://sandbox-hdp.hortonworks.com:4040/jobs/job?id=7) FINISHED

```
%spark2.pyspark

# notez que l'execution de la methode count initie un spark job (normal le RDD e
counts = words.count()
print "Number of elements in RDD -> %i" % (counts)
```

```
Number of elements in RDD -> 8
```

Took 0 sec. Last updated by anonymous at August 20 2019, 4:07:51 PM.

ERROR

```
%spark2.pyspark

# notez aussi qu'un RDD N'EST PAS UNE LISTE au sens de python. La preuve
print words[2]
```

```
Fail to execute line 1: print words[2]
Traceback (most recent call last):
  File "/tmp/zeppelin_pyspark-6097419831649483967.py", line 380, in <module>
    exec(code, _zcUserQueryNameSpace)
  File "<stdin>", line 1, in <module>
TypeError: 'RDD' object does not support indexing
```

Took 0 sec. Last updated by anonymous at August 20 2019, 3:01:43 PM. (outdated)

first_pyspark

SPARK JOB (http://sandbox-hdp.hortonworks.com:4040/jobs/job?id=2) FINISHED

```
%spark2.pyspark

# Pour récupérer le contenu d'un RDD (i.e. le désérialiser) on se sert de la met
print "Elements in RDD -> {}".format(words.collect())
```

```
Elements in RDD -> ['scala', 'java', 'hadoop', 'spark', 'akka', 'spark vs hadoop'
, 'pyspark', 'pyspark and spark']
```

Took 1 sec. Last updated by anonymous at August 20 2019, 3:06:58 PM. (outdated)

FINISHED

Effectuer des traitements parallèles à l'aide de Map / Reduce / Filter

Ce que nous allons montrer ici n'est pas très différent de la **programmation fonctionnelle** locale telle qu'on peut la faire en Python ou R. La seule différence tient ici à la parallélisation

Un bon article sur la PF en python (http://sametmax.com/map-filter-et-reduce/)

Took 0 sec. Last updated by anonymous at August 20 2019, 3:24:32 PM.

FINISHED

```
%spark2.pyspark

# Regardons déjà comment cela se présente en python local. J'ai une liste de nom
noms = ["Marie", "Jean", "Pierre" , "Bethsabee"]

# On peut adopter une approche itérative (i.e. boucler sur la structure)
noms_longs = []
for n in noms:
    if len(n)>5:
        noms_longs.append(n)
print noms_longs
```

```
['Pierre', 'Bethsabee']
```

Took 0 sec. Last updated by anonymous at August 20 2019, 3:23:43 PM.

first_pyspark

```
%spark2.pyspark                                                     FINISHED

# ou alors filter en utilisant une fonction
noms_longs = list(filter(lambda x: len(x)>5, noms))
print noms_longs
```

```
['Pierre', 'Bethsabee']
```

Took 0 sec. Last updated by anonymous at August 20 2019, 3:26:24 PM. (outdated)

```
%spark2.pyspark  SPARK JOB (http://sandbox-hdp.hortonworks.com:4040/jobs/job?id=3)  FINISHED
# revenons maintenant à notre RDD et récupérons les noms de plus de 5 caractères

long_words = words.filter(lambda x: len(x)>5)
print "les mots de plus de 5 caractères: {}".format(long_words.collect())
```

```
les mots de plus de 5 caractères: ['hadoop', 'spark vs hadoop', 'pyspa
rk and spark']
```

Took 0 sec. Last updated by anonymous at August 20 2019, 3:29:56 PM. (outdated)

```
%spark2.pyspark  SPARK JOB (http://sandbox-hdp.hortonworks.com:4040/jobs/job?id=4)  FINISHED

# remplaçons tous les espaces par un _ à l'aide de MAP

words_without_spaces = words.map(lambda x: x.replace(" ", "_"))
print "les mots sans espaces: {}".format(words_without_spaces.collect())
```

```
les mots sans espaces: ['scala', 'java', 'hadoop', 'spark', 'akka', 'spark_vs_had
oop', 'pyspark', 'pyspark_and_spark']
```

Took 0 sec. Last updated by anonymous at August 20 2019, 3:43:25 PM. (outdated)

```
%spark2.pyspark  SPARK JOB (http://sandbox-hdp.hortonworks.com:4040/jobs/job?id=16)  FINISHED
from operator import add

# enfin concatenons l'ensemble des mots en une seule str à l'aide de REDUCE

concatenated = words.reduce(add)
print "les mots concatene: {}".format(concatenated)
```

```
les mots concatene: scalajavahadoopsparkakkaspark vs hadooppysparkpyspark and spa
rk
```

Took 0 sec. Last updated by anonymous at August 20 2019, 4:25:38 PM.

```
%spark2.pyspark  SPARK JOB (http://sandbox-hdp.hortonworks.com:4040/jobs/job?id=19)  FINISHED

#TODO: à la suite de la fusion des boites x et y, constituez une équipe composée

# NB 1) on peut définir un RDD comme la somme de deux RDD ou plus à l'aide de "+
#    2) rien n'empèche d'appliquer plusieurs map / filter successifs

boite_x = sc.parallelize([("Claire", 37, "dev"),("Pierre", 24, "po"),("Marie",22
boite_y = sc.parallelize([("Charles", 33, "po"),("Ali", 23, "dev"),("Karl", 40,

# ICI
```

first_pyspark

```
[('Pierre', 24, 'po'), ('Charles', 33, 'po'), ('Claire', 37, 'dev'), ('Karl', 40,
'dev')]
```

Took 0 sec. Last updated by anonymous at August 20 2019, 4:33:20 PM. (outdated)

```
%spark2.pyspark
```

READY