Data 831

Partie 1 :

Assurez-vous de l'installation de Spark et des outils nécessaires :

- Il faut avoir Java 8, pour ceux qui ont déjà une version Java 11 sur leur machine, il est possible d'utiliser jenv pour switcher entre les différentes versions de Java. En effet, il « suffit » de changer le JAVA_HOME pour que cela fonctionne
- Il faut avoir Python en version 3 (3.6, 3.7) pour avoir accès à pyspark (à installer avec pip)
- Il faut installer Spark, prendre la version 2.4.5 avec un pre-built for Haddop, pas grave si vous n'avez pas Hadoop sur votre machine, cela peut fonctionner aussi en local
- Il n'est pas obligatoire d'avoir Scala d'installer sur sa machine
- Si vous voulez utiliser un Jupyter Notebook pour votre code, il faut avoir le package findspark à installer avec pip

Afin de vérifier le fonctionnement, allez sur une invite de commandes et entrez la commande spark-shell, si tout se passe bien vous allez voir :

```
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLeve
l(newLevel).
20/03/25 12:45:15 WARN Utils: Service 'SparkUI' could not bind on port 4040. Att
empting port 4041.
20/03/25 12:45:15 WARN Utils: Service 'SparkUI' could not bind on port 4041. Att
empting port 4042.
Spark context Web UI available at http://mbpdemaippe1507.home:4042
Spark context available as 'sc' (master = local[*], app id = local-1585136715865
).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.4.5
      /_/

Using Scala version 2.11.12 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_241)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

Et dernière étape, vérifiez que cela fonctionne pour pyspark, vous devez obtenir à peu près la même fenêtre.

Partie 2 :

Nous allons faire un ensemble de petits exercices en Spark pour comprendre le principe.

Spark c'est un peu comme Hadoop, vous allez effectivement faire du Map Reduce, la grosse différence avec Hadoop est que nous changeons la façon de décomposer le programme. Si vous vous souvenez comment nous avons fait pour le TP 1. Il s'agissait d'écrire un programme pour le Map et un programme pour le Reduce que nous avons relié par Streaming.

En Spark, tout se trouve ensemble et en fait, c'est le Driver (l'équivalent du MasterNode d'Hadoop) qui va se charger de répartir entre les différents Workers. Donc si vous écrivez sur l'invite de commandes :

```
pyspark --master "local[4]"
```

Vous créez un master et quatre worker. Le driver se chargera de placer les données sur chacun des Worker. Et pour vous, c'est transparent.

Maintenant, nous allons un ensemble de petits exemples :

```
from pyspark import SparkContext, SparkConf

if __name__ == "__main__":

  # create Spark context with Spark configuration
  conf = SparkConf().setAppName("Word Count -
Python").set("spark.hadoop.yarn.resourcemanager.address",
"mbpdemaippe1507.home:4040")
  sc = SparkContext(conf=conf)

  # read in text file and split each document into words
  words = sc.textFile("/Users/marc-
philippehuget/Downloads/spark.txt").flatMap(lambda line: line.split(" "))

  # count the occurrence of each word
  wordCounts = words.map(lambda word: (word, 1)).reduceByKey(lambda a,b:a +b)

  wordCounts.saveAsTextFile("/Users/marc-philippehuget/Downloads/pyoutput2")
```

Voilà l'exemple le plus classique que nous trouvons un peu partout, le Word Count. Pensez bien à changer l'adresse de lancement qui est chez moi « mbpdmaippe1507.home :4040

Ensuite, il faut indiquer que nous lisons un fichier, nous utilisons pour cela le contexte de Spark (sc) et l'ensemble du fichier spark.txt est ensuite converti en RDD. Resilient Distributed Dataset ou RDD est la structure qui conserve les données. La différence avec habituellement est que la donnée ne sera chargée qu'au moment de son utilisation donc votre fichier peut bien peser 1To, il ne sera pas en mémoire à l'instant donné.
Une fois que vous avez votre RDD, vous pouvez effectuer des Map et des Reduce. Ici, nous utilisons beaucoup les lambda functions mais vous pouvez aussi écrire des fonctions normales. Ici la lambda indique que pour chaque élément du fichier (appelé line) sera ensuite décomposé en mots et seront ajoutés à words.
Le flatMap est uniquement pour décomposer la ligne, et car un élément peut donner plusieurs éléments de RDD. Si cela n'avait été qu'un seul élément, on aurait mis un map.

Ensuite, pour chacun de ces mots, nous ajoutons un compteur qui vaut forcément 1. La dernière étape est le Reduce. Il suffit ensuite de tout additionner pour donner le résultat. Ce qui se retrouve dans wordCounts. La dernière ligne enregistre dans le répertoire pyoutput2. Vous pouvez aller voir le résultat pour vous en convaincre.

Comment exécuter ce programme ? Allez sur une invite de commandes et vous écrivez :

```
spark-submit <nom du fichier.py>
```

Les autres exemples que nous trouverons sont sur le même principe.

Voici quelques exemples à tester :

https://www.tutorialkart.com/apache-spark/spark-print-contents-of-rdd/

Vous apprendrez ce qu'est .collect(). Nous avons vu des transformations (dans le langage Spark) maintenant .collect() est une action. Elle va transformer le RDD en quelque chose qui ne sera pas un RDD.

Lire un fichier CSV avec Spark n'est pas plus compliqué, vous continuez à utiliser sc.textFile et cela devient un RDD. Essayez. Pour ensuite voir le contenu, vous pouvez faire <nom du RDD>.show(<nb de lignes>).

Je vous encourage à regarder le dépôt Github : https://github.com/jadianes/spark-py-notebooks pour quelques exemples supplémentaires. Surtout l'exemple nb10-sql-dataframes.

Histoire de se familiariser avec Spark sans pour autant à chaque fois faire un spark-submit, nous allons dans la partie 3, travailler sur des DataFrame pandas qui sont équivalents à ceux de Spark.

Partie 3 :

Prenez le fichier epl1.csv qui se trouve sur l'EAD et sur lequel nous allons travailler. L'objectif sera d'abord de lire le fichier CSV et en faire un DataFrame. Puis à partir de ce DataFrame, apprendre à faire les actions de base que sont map, filter, reduce. Il n'est pas obligatoire d'utiliser une lambda function.

Par exemple, sans map, filter et reduce, utilisez le DataFrame pour filtrer pour n'avoir les données que pour un seul club, ou seulement certaines colonnes. Il paraît même que vous pouvez faire des graphiques avec matplotlib…

Sortez des résultats comme le nombre moyen de spectateurs par match, le nombre moyen de buts par match, le nombre de buts pour chaque équipe. Et si nous voulons aller plus loin dans la difficulté, essayez de récolter le nombre de buts par joueur. A chaque fois, il s'agit de pratiquer de façon normale puis en map et reduce, et finalement en Spark qui restera équivalent.