

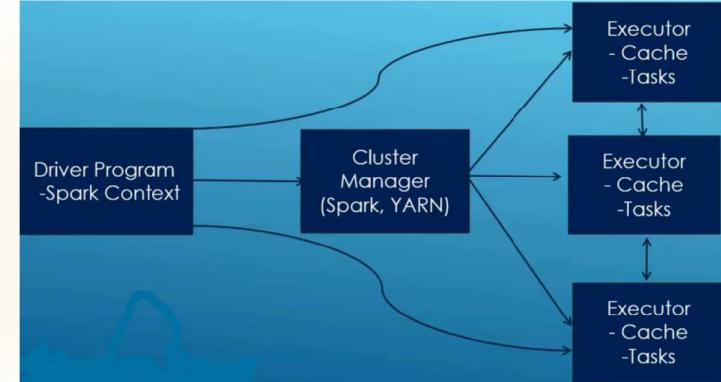
Programmer Avec Apache Spark

Il est rapide

- Il exécute les programmes 100 fois plus rapide que Hadoop MapReduce en mémoire ou 10 fois plus rapide sur disque
- Moteur DAG (Directed Acyclic Graph) optimise les flux de travail

C'est quoi Spark ?

Un moteur général rapide de traitement des données à grande échelle



N'est pas difficile

- Spark est écrit en Scala et s'exécute sur la machine virtuelle Java
- Les programmes Spark peuvent être écrits en    
- Construit autour d'un concept principal, les ensembles de données résilients (Resilient Distributed Data Sets –RDD)

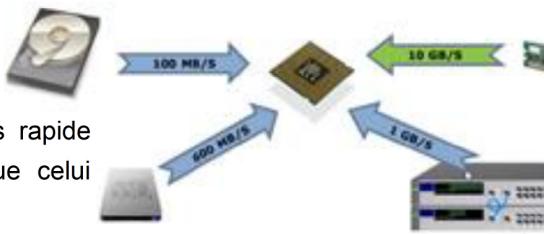
Hadoop Map reduce versus Spark

- Hadoop Map Reduce
 - Résultat écrit sur le disque
 - Opérations map reduce se font sur les disques

- Spark

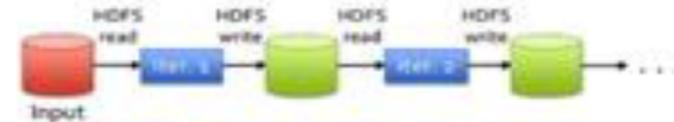
- Résultat écrit en RAM

Taux de transfert beaucoup plus rapide du RAM vers le processeur que celui d'une autre ressource



Hadoop Map reduce versus Spark

MapReduce (Hadoop)



Spark



permet d'exécuter des requêtes de type SQL. Spark SQL permet d'extraire, transformer et charger des données sous différents formats (JSON, Parquet, base de données) et les exposer pour des requêtes ad-hoc.



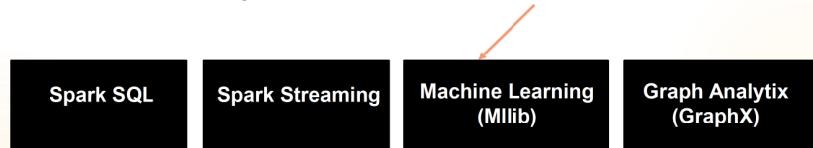
Spark Core

Utilisé pour traitement temps-réel des données en flux. Il s'appuie sur un mode de traitement en "micro batch"



Spark Core

Une librairie de machine learning qui contient tous les algorithmes et utilitaires d'apprentissage classiques, comme la classification, la régression, le clustering, etc.



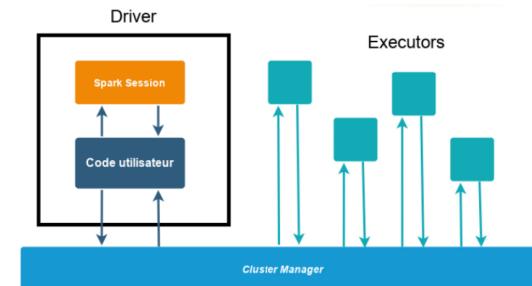
GraphX inclut une collection importante d'algorithme et de builders pour simplifier les tâches d'analyse de graphes.



Architecture système

- Spark est un *framework* qui coordonne l'exécution de tâches des données en les répartissant au sein d'un *cluster* de machines.
- Le programmeur envoie au framework des Spark Applications, pour lesquelles Spark affecte des ressources (RAM, CPU) du cluster en vue de leur exécution.
- la gestion même du cluster de machines peut être déléguée soit au manager de Spark, soit à Yarn ou à Mesos (d'autres gestionnaires pour Hadoop).

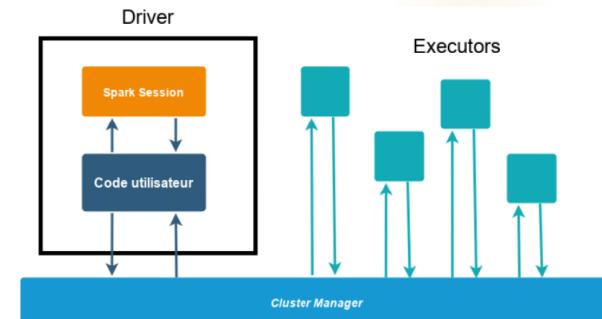
Architecture système



Architecture du système Spark (<http://b3d.bdpedia.fr/spark-batch.html>)

- Une application Spark se compose d'un processus driver et d'executors (executors). Le driver est essentiel pour l'application car il exécute la fonction main() (le programme principal du driver) et est responsable de 3 choses :
 - conserver les informations relatives à l'application ;
 - répondre aux saisies utilisateur ou aux demandes de programmes externes ;
 - analyser, distribuer et ordonner les tâches.

Architecture système



Architecture du système Spark (<http://b3d.bdpedia.fr/spark-batch.html>)

- Un *executor* n'est responsable que de 2 choses : exécuter le code qui lui est assigné par le *driver* et lui rapporter l'état d'avancement de la tâche.
- Le *driver* est accessible par un point d'entrée appelé `SparkSession`, qui se trouve derrière une variable `spark`.

Architecture applicative

L'écosystème des API de Spark comporte essentiellement 3 niveaux :

- les APIs bas-niveau, avec les RDDs (*Resilient Distributed Dataset*);
- les APIs de haut niveau, avec les Datasets, DataFrames et SQL;
- les autres bibliothèques (*Structured Streaming*, *Advanced Analytics*, etc.).

Programmer pyspark sous google collab

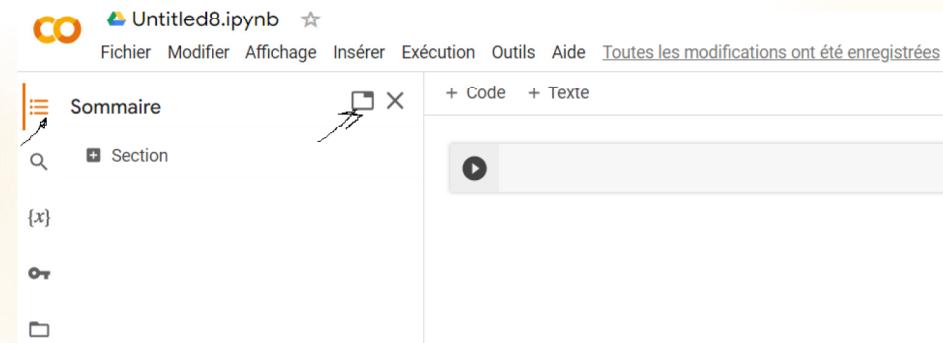
- Ouvrir votre boîte email google, puis ouvrir drive, puis google collab
- Taper les commandes suivantes:

```
!pip install -q findspark
!pip install pyspark
from pyspark.sql import SparkSession
spark = SparkSession.builder.master("local[*]").getOrCreate()

spark=SparkSession.builder.appName('sol').getOrCreate()
sc=spark.sparkContext
```

Programmer pyspark sous google collab

Cliquer sur sommaire puis le schéma du dossier (comme montre la flèche) pour copier dessous vos fichiers de données :



RDD (Resilient Distributed Data Set)

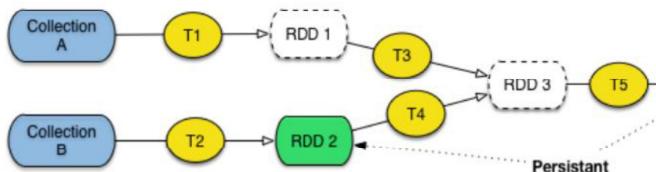
- C'est une collection construite à partir d'une source de données (un flux, un autre RDD). Calculé par une *transformation*, qui sélectionne, enrichit, restructure une collection, ou combine deux collections.
- C'est du code fournit par Spark. Appartient à la plateforme de Spark dans chacun des langages suivants:    
- Spark historie les opérations qui ont permis de constituer un RDD. Ce historique sert à reconstituer le RDD en cas de panne.

RDD (Resilient Distributed Data Set)

- Les RDDs représentent des collections partitionnées et distribuées. Un RDD peut être divisée en partitions logiques multiples qui peuvent être stockées et traitées dans des machines différentes dans un cluster.
- Un RDD constitue un "bloc" non modifiable. Si nécessaire il est entièrement recréé et recalculé avec des opérations coarse-grain.
- Un RDD peut être marquée comme persistante : dans ce cas elle est en mémoire RAM et conservé par Spark.

RDD (Resilient Distributed Data Set)

Les collections forment un graphe construit par application de transformations à partir de collections stockées



RDD persistants et transitoires dans Spark (<http://b3d.bdpedia.fr/spark-batch.html#architecture-applicative>)

- Crée par un programme pilote du programmeur
- Spark est responsable de rendre la RDD résiliente et distribuée
- Spark shell crée un objet dans Spark Context « sc » pour créer la RD

Création d'une RDD

- Spark Context « sc » permet de créer les RDD:
 - ✓ à partir d'une source de données (un fichier, une liste,)
 - ✓ En parallélisant une collection (liste,..)
 - ✓ En appliquant une méthode de transformation

Création d'une RDD

- Création d'une RDD à partir d'un fichier: sc.textFile("/chemin/fichier/fichier.ext")
- ✓ En local, sous linux: rd1=sc.textFile("/user/myfiles/file.ext")
- ✓ En local, sous Windows:rd2=sc.textFile("d:/dossier/file.ext")
- ✓ Sous Hadoop: rd3=sc.textFile("hdfs://user/myfiles/file.ext")
- Création d'une RDD en utilisant parallelize

Création d'une RDD

Créer une RDD nums à partir de la liste [1,2,3,4]

Exemple:

- nums = sc.parallelize ([1,2,3,4])
- Rdd=sc.textFile("d:/bigData/toto.txt")

Un objet RDD est créé à partir d'un fichier toto.txt dans le contexte Spark (sc).

Le fichier peut être dans un disque local ou dans un cluster HDFS:// ou S3n://

Les collections parallélisées sont créées avec le méthode **parallelize** dans contexte de Spark (sc).

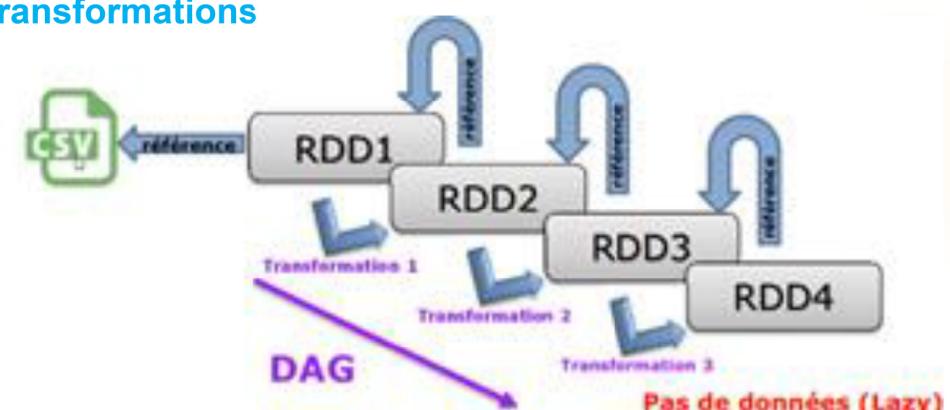
Création d'une RDD

Puissent être créées à partir de:

- JDBC (Java DataBase Connectivity)
- Cassandra
- Hbase
- Elasticsearch
- JSON, csv, sequence files, object files, variété de formats comprimés

Base de programmation d'une RDD

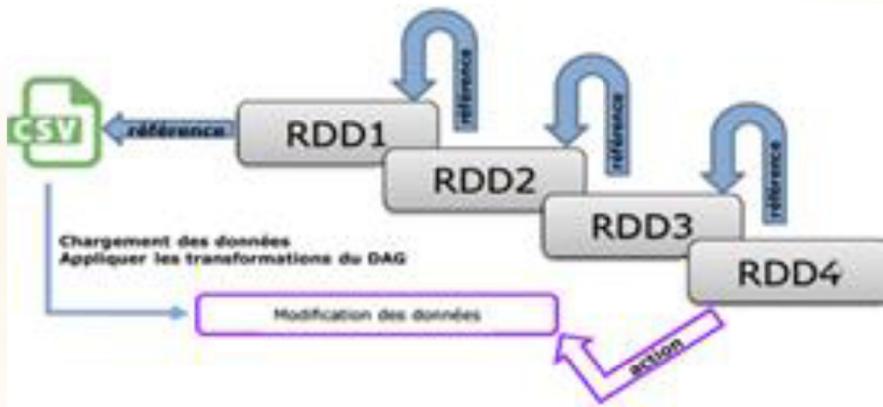
Transformations



Directed Acyclic Graph

Base de programmation d'une RDD

Actions



Transformation d'une RDD

Transformations	Rôle
map(func)	applique une fonction à chacune des données
filter(func)	permet d'éliminer certaines données, exemple rdd.filter(lambda x : permet d'éliminer les nombres pairs
flatMap(func)	similaire à map, mais chaque élément d'entrée peut être mappé à plusieurs éléments de sortie (donc, func devrait retourner une Seq qu'un seul élément)
distinct()	supprime les doublons
groupsByKey()	transforme des clés-valeurs (K,V) en (K,W) où W est un objet itérable exemple, (K,U) et (K,V) seront transformées en (K, [U,V])

Transformation d'une RDD

Transformations Rôle

applique une fonction de réduction aux valeurs de chaque clé. Comme on a pu le voir, la fonction de réduction est avec deux arguments. Il est attendu que la fonction de réduction soit commutative et associative, c'est-à-dire que $func(a,b) = func(b,a)$ et $func(func(a,b),c) = func(a, func(b,c))$. Par exemple, la fonction func peut renvoyer le maximum de deux valeurs.

sortByKey() utilisée pour trier le résultat par clé

join(rdd) permet de réaliser une jointure, ce qui a le même sens qu'en SQL dans les bases de données relationnelles.

Transformation d'une RDD

Transformations Rôle

Crée un nouveau RDD constitué d'un échantillon de n éléments du RDD source. Sample a fraction $fraction$ of the data, without replacement, using a given random number generator seed.

Union() Union de deux ou plusieurs RDD

Sort() Tri du RDD

→ Rien ne se passe pour le programme pilote (driver) jusqu'à une action invoquée.

Transformation d'une RDD

Exemple:

```
rdd=sc.parallelize([1,2,3,4])  
t=rdd.map(lambda x:x*x)
```

C'est quoi l'expression lambda?

Plusieurs méthodes RDD acceptent lambda comme paramètre.

rdd.map(lambda x:x*x)	Est équivalent à:	def auCarre(x) return x*x rdd.map(auCarre(x))
-----------------------	-------------------	---

Transformation d'une RDD

range (start: int, end: Optional[int] = None, step: int = 1, numSlices: Optional[int] = None)

Crée un a RDD d'entier situés entre start et end-1), chaque élément incrémenté par step. numSlices indique le nombre de partitions de la nouvelle RDD.

Si un seul argument est utilisé, l'argument est interprété comme end, and set à 0.

Exemple:

```
sc.range(5).collect() Collect() est utilisé pour pouvoir afficher le résultat de range action)  
[0, 1, 2, 3, 4]
```

Transformation d'une RDD

Exemple:

```
sc.range(2, 4).collect()
```

```
[2, 3]
```

```
sc.range(1, 7, 2).collect()
```

```
[1, 3, 5]
```

Les actions sur les RDD

Action	Rôle
reduce(func)	agrège les éléments de l'ensemble de données en utilisant une fonction func (qui prend deux arguments en renvoie un). La fonction doit être commutative et associative afin qu'elle puisse être calculée correctement en parallèle
take(n)	retourne un tableau avec les n premiers éléments d'un RDD

Les actions sur les RDD

Action	Rôle
Sum()	Effectue une somme des éléments du RDD
Foreach()	Permet d'effectuer des actions spécifiques pour chaque élément du RDD
collect()	retourne toutes les données contenues dans l'RDD sous la forme d'une liste. Transfère les données de l'RDD dans le nœud principal (ou driver) du cluster.
count()	retourne le nombre de données contenues dans l'RDD

Les actions sur les RDD

Si on exécute l'exemple:

```
rdd=sc.parallelize([1,2,3,4])
t=rdd.map(lambda x:x*x)
```

Le résultat de transformation ne sera pas affichée

Si on ajoute t.take(4), on aura le résultat:

```
[1, 4, 9, 16]
```

Des Exemples d'opérations utilisant les RDD

Exemple 1: Le code suivant crée un itérateur de 10 000 éléments, puis il utilise parallelize() pour distribuer ces données en 2 partitions :

```
from pyspark import SparkContext sc = SparkContext()
ma_liste = range(10000)
rdd = sc.parallelize(ma_liste, 2)
nombres_impairs = rdd.filter(lambda x: x % 2 != 0)
nombres_impairs.take(5)
```

Des Exemples d'opérations utilisant les RDD

Exemple 2: Soit un fichier nommé data.txt existante dans l'unité de stockage considérons ce programme en python:

Un fichier texte data.txt est transformée en RDD appelée lines dans le contexte de Spark

```
lines = sc.textFile("d:/data.txt")
lineLengths = lines.map(lambda s: len(s))
totalLength = lineLengths.reduce(lambda a, b: a + b)
```

Une nouvelle RDD lineLengths est créée à partir de lines qui contient la longueur (len) de chaque ligne

map est utilisée pour effectuer une programmation fonctionnelle utilisant une fonction lambda qui invoque une fonction à deux paramètres s et retourne len remplacée par les lignes de lines

Des Exemples d'opérations utilisant les RDD

```
lineLengths = lines.map(lambda s: len(s))
```

- La fonction `map` prend en argument une fonction. Elle crée une nouvelle collection vide (collection vide), applique la fonction à chaque élément de la collection d'origine et insère les valeurs de retour produites dans la nouvelle collection. Finalement, elle renvoie la nouvelle collection.
- Dans l'exemple la nouvelle collection (nouveau RDD) est `lineLengths` à partir de la collection (RDD) `lines` sur laquelle `map` est invoquée

Des Exemples d'opérations utilisant les RDD

`lineLengths` n'est pas exécutée immédiatement à cause de la paresse (lazy).

```
lines = sc.textFile("d:/data.txt")
lineLengths = lines.map(lambda s: len(s))
totalLength = lineLengths.reduce(lambda a, b: a + b)
```

`Reduce` est une action. À la différence de `map`, Spark divise l'exécution en plusieurs parties. Pour l'exécution sur des machines distantes séparées. Chaque machine exécute une partie de `map` et une réduction localisée. Puis, lorsque la partie de `map` est terminée, elle retourne son résultat au pilote.

Des Exemples d'opérations utilisant les RDD

```
lineLengths = lines.map(lambda s: len(s))
```

Ce `map` ne prend pas une fonction nommée en paramètre : il définit une fonction anonyme en définissant à l'aide du mot-clé `lambda` les paramètres de cette fonction. Les paramètres de cette fonction `lambda` sont définis à gauche par un caractère deux-points et le corps de la fonction est défini à sa droite. Le résultat de l'exécution de cette fonction est renvoyé (implicite) par la fonction `map`.

Des Exemples d'opérations utilisant les RDD

```
totalLength = lineLengths.reduce(lambda a, b: a + b)
```

`a` est l'élément courant de l'itération et `b` est l'accumulateur. C'est la valeur qui est renvoyée par l'exécution de la fonction `lambda` sur l'élément précédent. La fonction `reduce()` parcourt les éléments de la liste `lineLengths` et, pour chaque élément, exécute `lambda` sur les valeurs courantes de `b` et de `a` et renvoie le résultat qui devient `a` de l'itération suivante.

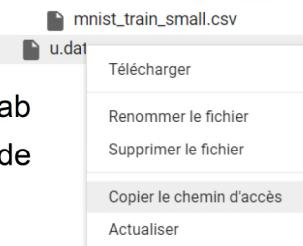
Des Exemples d'opérations utilisant les RDD

```
totalLength = lineLengths.reduce(lambda a, b: a + b)
```

Que vaut **b** lors de la première itération ? Il n'y a pas de résultat d'une itération précédente à lui passer. La fonction `reduce()` utilise la première valeur de a pour b et commence à itérer à partir de la seconde valeur. Autrement dit, la première valeur de **a** est le second élément de la liste.

```
import collections
lines = sc.textFile("/content/u.data")
ratings = lines.map(lambda x: x.split()[2])
result = ratings.countByValue() ← Return the count of each unique value
sortedResults = collections.OrderedDict(sorted(result.items()))
for key, value in sortedResults.items():
    print("%s %i" % (key, value))
```

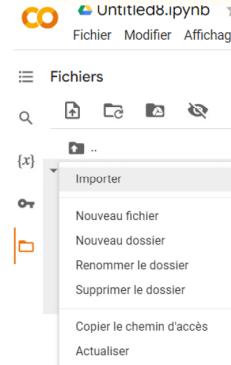
Le chemin du fichier est copié de google collab en cliquant avec le bouton droit sur le nom de fichier comme le montre le figure suivant



Programmer avec Spark

Créer un dossier sous c: ou d: appelé ExercicesSpark qui contiendra les datasets utilisées

Exercice 1. Ecrire un programme qui compte la somme obtenu pour chaque nombre de stars (de 1 à 5) utilisés pour évaluer un film dans le fichier u.data.



Importer le fichier u.data sous le dossier de données sous google collab

Explications

→ Faire les importations des librairies nécessaires

```
import collections
```

- **SparkContext:** pour créer les rdd

Importer la bibliothèque `collections` qui sont des types de conteneur de données comme les listes, les maps, etc.

Charger les données

```
196 242 3 881250949  
186 302 3 891717742  
22 377 1 878887116  
244 51 2 880606923  
166 346 1 886397596  
298 474 4 884182806
```

```
lines = sc.textFile("/content/u.data")
```

- À l'aide `textFile`, le fichier sera lu ligne par ligne. Chaque ligne est un String
- `textFile` permet de retourner un enregistrement par ligne

→ (map) les données voulues dans ce cas la valeur de colonne 2.
(compteur de colonne débute de 0)

```
196 242 3 881250949  
186 302 3 891717742  
22 377 1 878887116  
244 51 2 880606923  
166 346 1 886397596  
298 474 4 884182806
```

```
ratings = lines.map(lambda x: x.split()[2])
```

L'expression fait le split suivant l'espace et garde les données de la colonne 2 dans la variable ratings

```
196 242 3 881250949  
186 302 3 891717742  
22 377 1 878887116  
244 51 2 880606923  
166 346 1 886397596  
298 474 4 884182806
```

```
ratings = lines.map(lambda x: x.split()[2])
```

- L'expression fait le split suivant l'espace et extrait les données de la colonne 2 avec map dans la variable ratings
- map ne transforme pas la rdd source, elle crée une nouvelle: ratings est un vecteur

Effectuer une action: compter par valeur

```
result = ratings.countByValue()
```

3	
3	(3, 2)
1	(1, 2)
2	(2, 1)
1	(4, 1)
4	

→ Trier et afficher le résultat

```
sortedResults = collections.OrderedDict(sorted(result.items()))
for key, value in sortedResults.items():
    print("%s %i" % (key, value))
```

$(3, 2)$	$(1, 2)$
$(1, 2)$	$(2, 1)$
$(2, 1)$	$(3, 2)$
$(4, 1)$	$(4, 1)$

- Utiliser la bibliothèque sur les collections en python OrderedDict (ordered dict) et la méthode sorted qui sera exécutée sur les items du fichier result pour trier les valeurs suivant la clé
 - %s affichage sous format string, %i format integer

Exécuter le programme

Exercice 2. écrire un programme qui cherche le minimum de température pour une location, à partir de la base 1800.csv. Températures seront converties en fahrenheit en utilisant une fonction.