Spark

m2iformation.fr

Sommaire

- Introduction Spark
- Les composants Spark
- Spark Core - RDD
- Spark Core - Transformations
- Spark Core - Actions
- Spark SQL

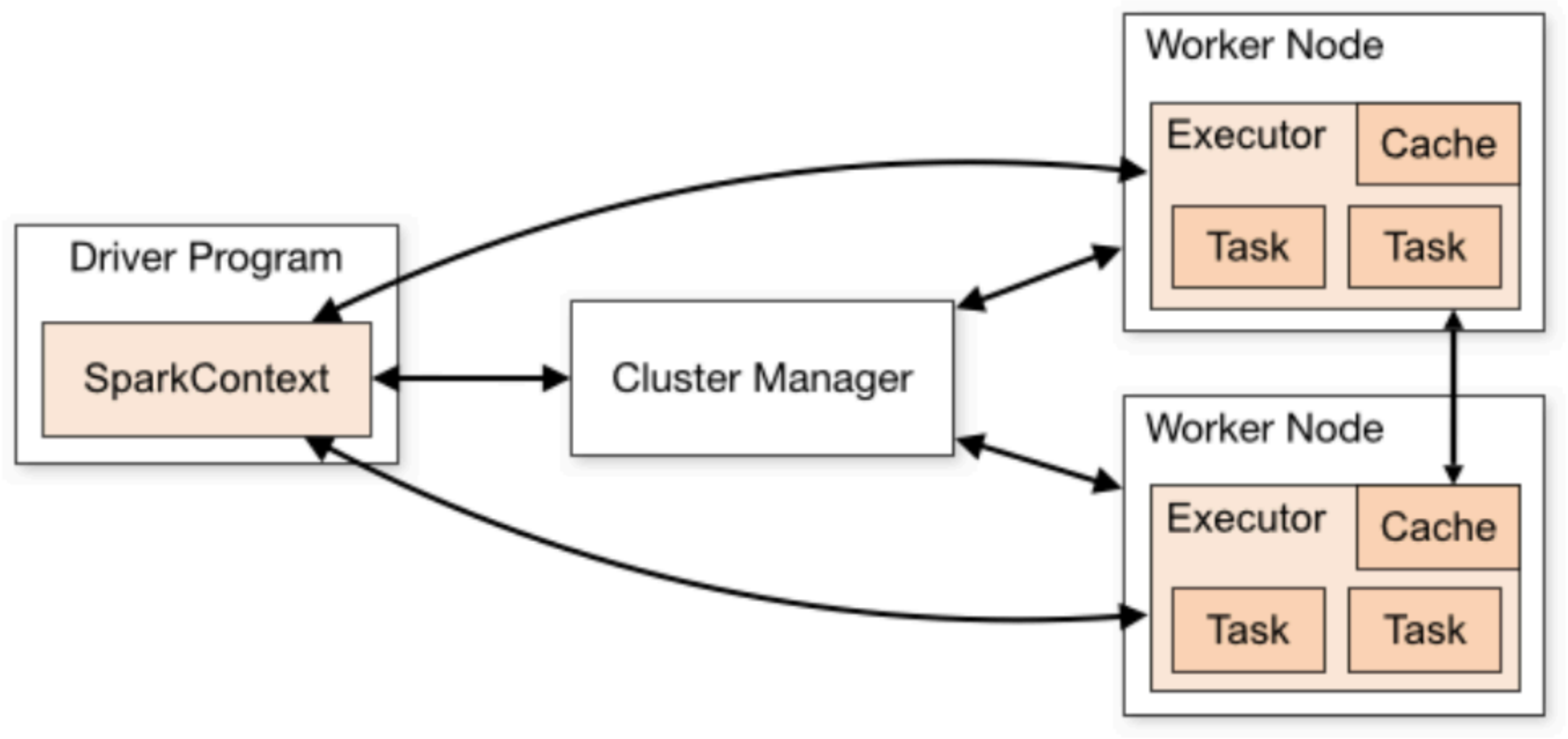
UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Introduction Spark

Introduction Spark

- Spark est un moteur de traitement de données à grande échelle.
- Spark permet d'analyser et transformer une grande quantité de données.
- Spark permet l'exécution et l'analyse sur un cluster.
- Spark est 100 fois plus rapide que Hadoop MapReduce
- Spark utilise DAG Engine pour l'optimisation des workflows.
- Documentation : https://spark.apache.org/docs/latest/

Introduction Spark



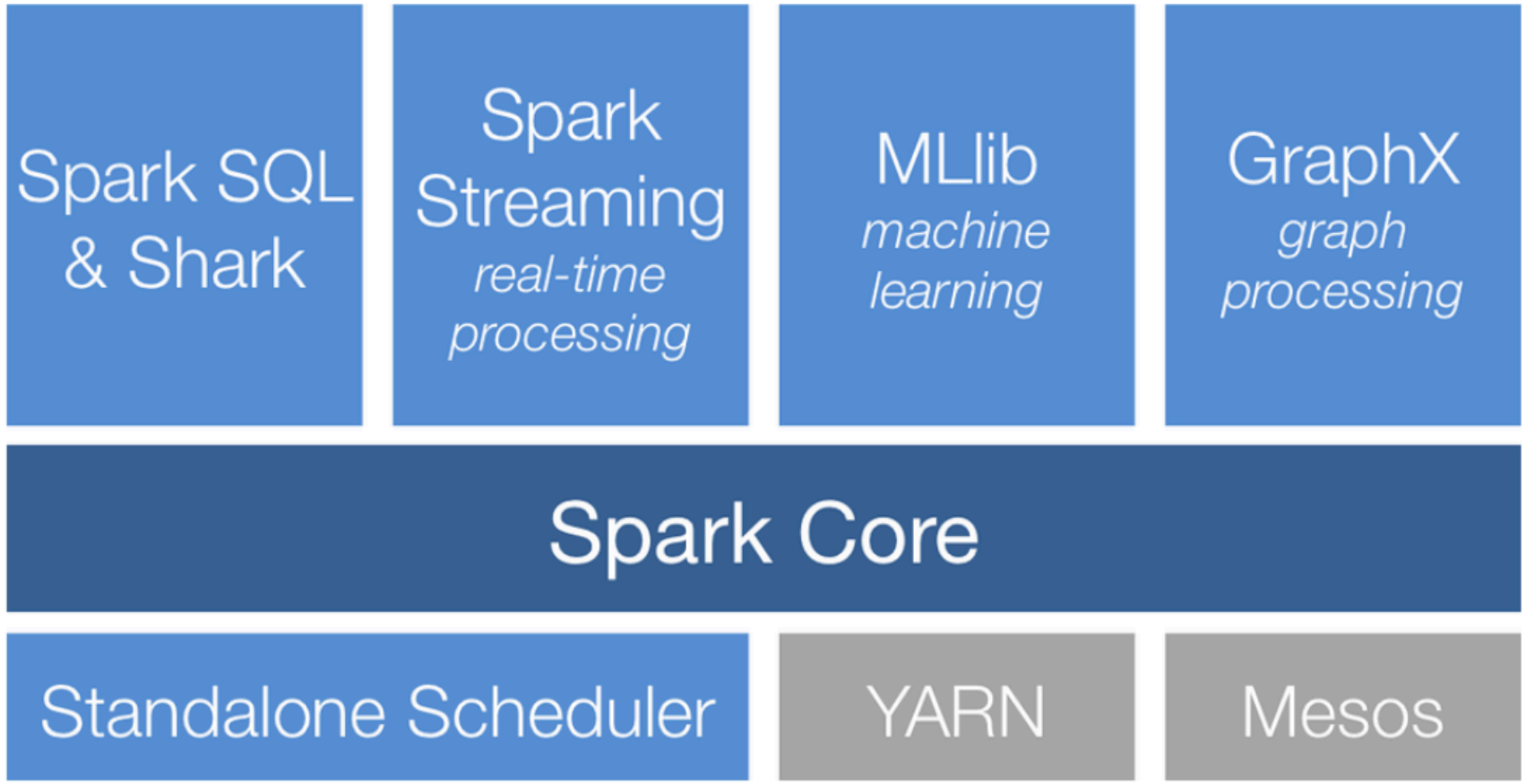
Introduction Spark

- Driver program est un script ou une application qui peut être en Java, scala, python.
- Cluster manager est l'api qui permet de gérer l'orchestration de notre cluster de nœuds.
- Chaque nœud contient un exécuteur.
- Chaque exécuteur possède son propre cache et la liste des tâches à exécuter.
- Les composants de l'architecture spark communiquent et se synchronisent entre eux.

Introduction Spark

- Spark est facile à apprendre.
- Spark supporte une multitude de langages de développement.
- Spark fournit des APIs bas niveau.
- Spark fournit également des Apis Haut niveau.

Les composants Spark



Les composants Spark

- Spark Core est le moteur d'exécution de Spark.
- Spark Streaming est une Api qui permet l'ingestion de données en temps réel.
- Spark SQL permet l'interaction avec Spark à l'aide de requête SQL.
- MLLib est la bibliothèque d'apprentissage automatique de Spark.
- GraphX est une Api qui permet de mettre en place et analyse de connexion entre plusieurs entités.

Nouveautés Spark4

- **API `.plot()` native sur DataFrame** : permet des visualisations directes sans bibliothèques externes (utilise Plotly).
- **Spark Connect activé par défaut** : exécution distante des scripts PySpark avec un client léger.
- **Data Source API Python** : création de connecteurs batch/streaming en pur Python.
- **Support des UDTF en Python** : fonctions retournant des lignes multiples, avec typage flexible.

Comment nous allons utiliser Spark

Comment nous allons utiliser Spark

- Spark 4
- Python avec le package pyspark
- Jupyter Lab Notebooks
- Image Docker

https://quay.io/repository/jupyter/pyspark-notebook
https://jupyter-docker-stacks.readthedocs.io/en/latest/using/specifics.html

Spark Core - RDD (Resilient Distributed DataSet)

- RDD sont des lignes de données muables qui sont :
  - Resilient
  - Distributed
  - DataSet
- Les RDD sont créées par notre Driver Program.
- La création se fait à l'aide d'un context (SparkContext).
- Documentation RDD : RDD

Spark Core - RDD

- La création peut se faire à partir d'un fichier text.
  - File://, s3n://, hdfs://
- La création peut se faire également à partir d'une base de données
  - JDBC
  - Cassandra
  - ElasticSearch
  - JSON, csv…

Spark Core - RDD

```
from pyspark.sql import SparkSession

spark = SparkSession.builder.master("local").getOrCreate()
sc = spark.sparkContext
ma_liste = range(10000)
rdd = sc.parallelize(ma_liste, 2)
nombres_impairs = rdd.filter(lambda x: x % 2 != 0)
nombres_impairs.take(5)
```

Spark Core - Transformations

- Les opérations appliquées aux RDD sont des transformations.
- Chaque transformation donne une nouvelle RDD.
- Spark core offre une liste de transformations possibles :
  - Map
  - Flatmap
  - Filter
  - Distrinct
- Chaque transformation prend comme paramètre une fonction (Expression lambda).

Spark Core - Transformations

- Spark Context ne procède pas directement à l'exécution de la transformation.
- Spark attend l'appel au deuxième type d'opération exécutable sur RDD (action) pour démarrer la transformation.
- Le choix de l'action influence sur la façon d'exécution de la transformation.

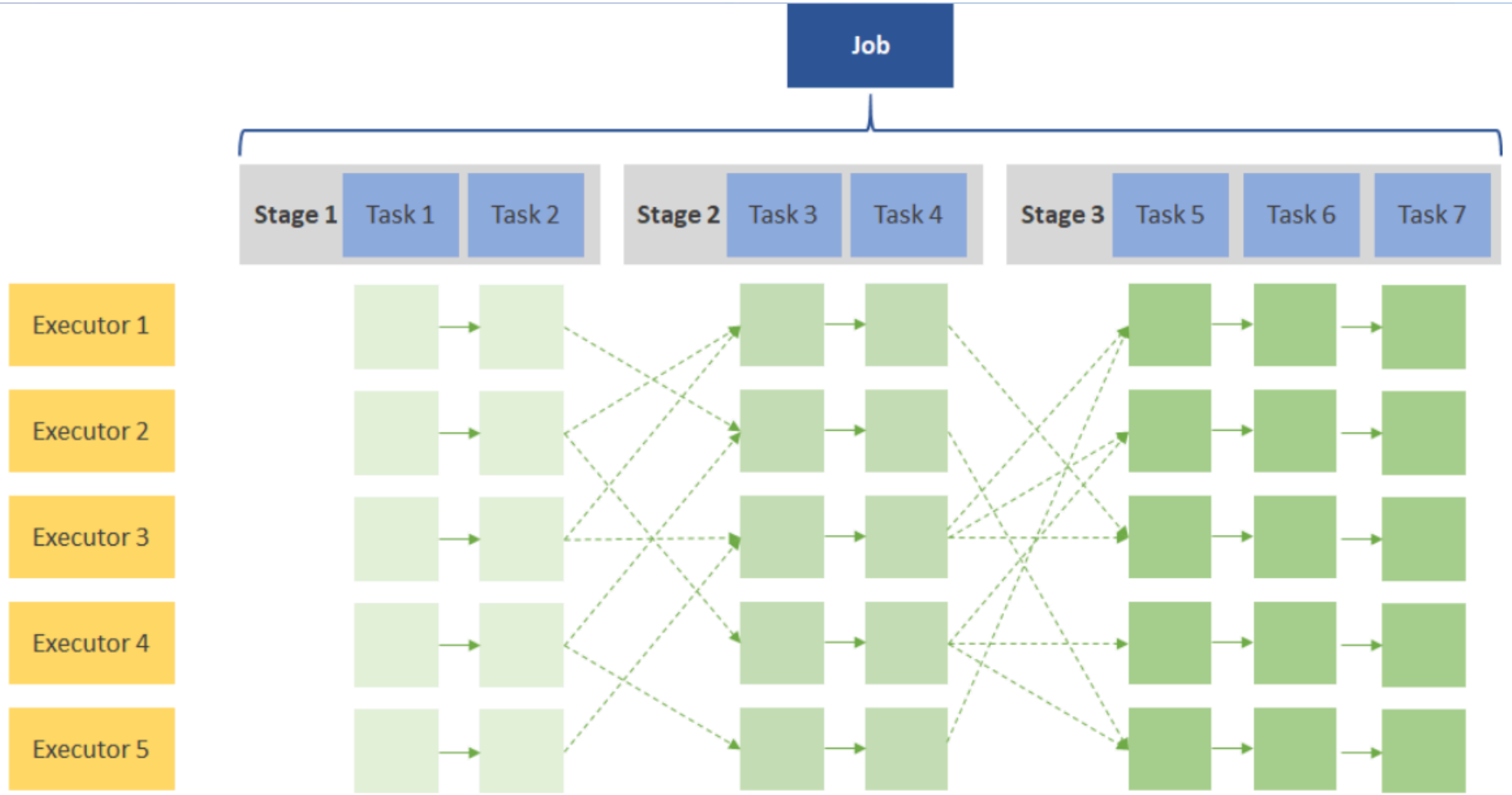
Spark Core - Actions

- L'action est l'opération à appliquer à notre RDD pour récupérer le résultat.
- Spark core offre une liste d'actions possibles.
  - Collect
  - Count
  - CountByValue
  - Reduce
- Une action peut avoir aucun ou plusieurs paramètres.

Spark Core - Rdd - exemple 1

- On utilisera un set de données de plus de 100 000 lignes de notes de films (fichier film.data).
- On souhaite connaitre le nombre de film par note et faire un tri.
- On commence par extraire à partir de chaque ligne la valeur qui nous intéresse.
- On compte le nombre d'éléments par valeur.
- On tri le résultat.

Spark Core - Fonctionnement interne



Spark Core - RDD key value

- RDD key value est une implémentation des données structurées sous format clé valeur.
- RDD key value offre des transformations et des actions en plus.
- Exemple:
  - Pour calculer le nombre moyen d'amis par âge dans un set de données qui contient (personnes avec nombre d'ami).
  - On commence par extraire les données dans une RDD (âge, ami).
  - On applique les transformations mapValues, reducesByKey
  - On applique une action de collect

Spark Core - RDD transformation filter

- Transformation Filter est une opération qui prend comme paramètre une fonction.
- La fonction paramètre est appliquée à chaque élément et renvoie un boolean.
- Le résultat est une nouvelle RDD avec des éléments dont le résultat est positif.
- **Exemple** : A partir d'un set de données météorologique fournit par deux stations météo, on souhaite conserver uniquement les températures minimums.

Spark Core - RDD flatmap

- La fonction map permet de convertir chaque ligne de notre set de données en ligne de RDD.
- Avec map chaque ligne est une row RDD.
- FlatMap permet de convertir chaque ligne de notre set de données en n'importe quel nombre de row RDD.
- **Exemple** :
  - A partir d'un fichier texte, on souhaite compter le nombre de mots.

Fonction de transformation

Transformations	Rôle
map(func)	applique une fonction à chacune des données pour créer un nouveau RDD
filter(func)	permet d'éliminer certaines données. Exemple : rdd.filter(lambda x : x % 2 != 0) permet d'éliminer les nombres pairs
flatMap(func)	similaire à map, mais chaque élément d'entrée peut être mappé à 0 ou plusieurs éléments de sortie (donc, func devrait retourner une liste plutôt qu'un seul élément)
distinct()	supprime les doublons
groupsByKey()	transforme des clés-valeurs (K,V) en (K,W) où W est un object itérable. Par exemple, (K,U) et (K,V) seront transformées en (K,[U,V])
reduceByKey(func)	applique une fonction de réduction aux valeurs de chaque clé. La fonction de réduction est appelée avec deux arguments

Fonction de transformation

Transformations	Rôle
sortByKey	utilisée pour trier le résultat par clé
join(rdd)	join(rdd) permet de réaliser une jointure avec 2 RDD Key Value, ce qui a le même sens que dans les bases de données relationnelles.À noter qu'il existe également des fonctions OuterJoin et fullOuterJoin. Les jointures sont réalisées sur la clé
reduce(func)	agrège les éléments de l'ensemble de données en utilisant une fonction func (qui prend deux arguments et en renvoie un) cela donnera une valeur finale
take(n)	retourne une liste avec les n premiers éléments du RDD
collect()	retourne toutes les données contenues dans le RDD sous la forme d'une liste.
Count()	retourne le nombre de données contenues dans le RDD

Exercice 1

- A partir d'un fichier csv qui contient sur chaque ligne (id client, id article, montant).
- Ecrire un driver programme qui permet de connaître le montant dépensé par chaque client.
- **Aide** : Séparer chaque ligne avec la délimitation « , ».
  - Créer une pair RDD avec id client et montant (transformation map).
  - Réduire par id client (transformation reduceByKey).
  - Récupérer une liste de résultats (action collect)

Spark SQL - RDD structurés

- Un RDD est une ”boî te” (typée) destinée à contenir n’importe quel document, sans aucun préjugé sur sa structure (ou son absence de structure)
- Cela rend le système très généraliste, mais empêche une manipulation fine des constituants des documents, comme par exemple le filtrage en fonction de la valeur d’un champ.
- C’est au programmeur de fournir la fonction effectuant le filtre. Spark propose (depuis la version 1.3, avec des améliorations en 1.6 puis 2.0) des RDD structurées dans lesquels les données sont sous forme tabulaire. Le schéma de ces données est connu.

Spark SQL - RDD structurés

- Ces structures, Datasets et Dataframes, sont assimilables à des tables relationnelles.
- Documentation SparkSQL : https://spark.apache.org/docs/latest/sql-getting-started.html

Spark SQL - Dataset et DataFrames

- La connaissance du schéma - et éventuellement de leur type - permet à Spark de proposer des opérations plus fines, et des optimisations inspirées des techniques d'évaluation de requêtes dans les systèmes relationnels.
- On se ramène à une implantation distribuée du langage SQL.
- En interne, un avantage important de la connaissance du schéma est d'éviter de recourir à la sérialisation des objets Java (opération effectuée dans le cas des RDD pour écrire sur disque et échanger des données en réseau).

Spark SQL - Dataset et DataFrames

- Ces RDDs structurés sont nommées :
  - Dataset quand le type des colonnes est connu
  - Dataframe quand ce n'est pas le cas (ensemble de Row)
- Un Dataframe n'est rien d'autre qu'un Dataset contenant des lignes de type Row dont le schéma précis n'est pas connu

Syntaxe Spark SQL

- En réalité, il est possible avec Spark SQL d'utiliser 2 syntaxe pour arriver à un résultat similaire :
  - La syntaxe SQL « classique », permettant de faire des requêtes SQL à l'aide de la méthode session.sql("SELECT * FROM table"). Cette méthode part d'une table ou d'une vue temporaire de la session et donnera un DataFrame.

Syntaxe Spark SQL

- La syntaxe fonctionnelle, utilisant des méthodes que l'on applique à un DataFrame qui donneront un nouveau DataFrame après leur exécution. Leurs noms sont inspirés des langages SQL classique dans une majorité des cas (sauf par exemple WHERE qui devient filter()) et leur utilisation reste logique par rapport au SQL Certaines sont utilisables directement avec le package pyspark.sql .Il est tout de même possible de récupérer une table ou une vue de la session avec la méthode session.table("nom_table").

Spark SQL - DataSet/DataFrames exemple

- Pour utiliser SparkSQL, on peut créer un point d'entrée de type SqlContext.
- On peut également, depuis la version 2, utiliser le point d'entrée SparkSession.
- **Exemple**:
  - En utilisant notre set de données friends, on souhaite extraire les personnes âgées de 20 à 25 ans.

Spark SQL - DataSet/DataFrames exemple

- **Etapes**:
  - Création du context
  - Extraction des lignes dans une RDD
  - Création d'une View.
  - Exécution d'une requête SQL sur la view.
  - Récupération des résultats

Exercice 2

- On souhaite reprendre notre exemple moyenne d'amis en fonction de l'âge « Section RDD (Key/Val) » avec une utilisation des DataFrames.
- Rappel :
  - Chaque ligne du fichier est composée de :
  - Identifiant de l'utilisateur.
  - Nom de l'utilisateur.
  - Age de l'utilisateur.
  - Nombre d'ami.

Exercice 2 - Aide

- Création d'une sparkSession.
- La lecture du set de données, en utilisant la ligne d'entête.
- La sélection des colonnes.
- L'utilisation des fonctions avg, groupby et show.

Exercice 3

- Reprendre l'exercice "Montant dépensé par client", avec les Dataframes.

Exercice 4

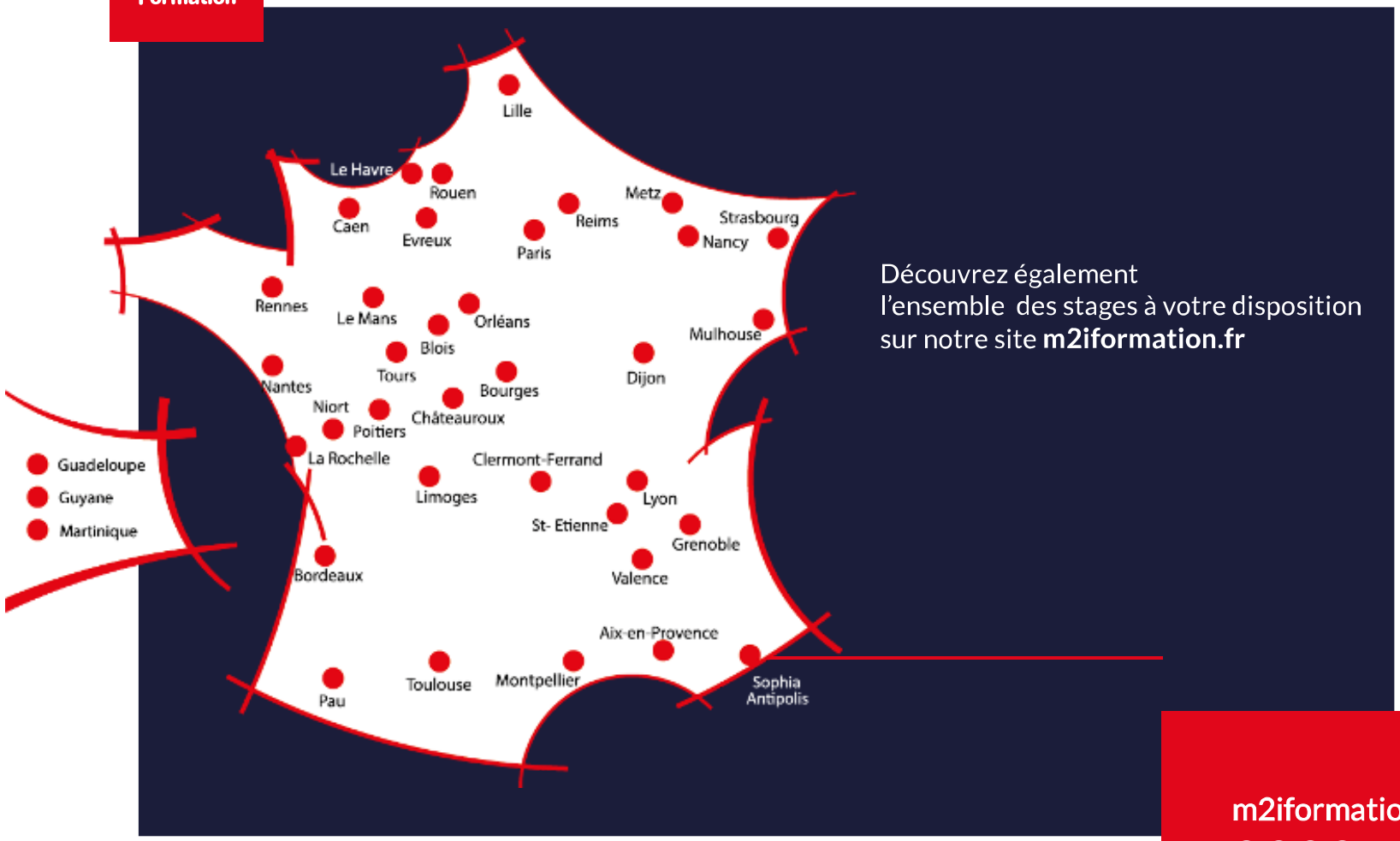
- A partir des données du fichier film.data
  - Créer un dataFrames qui permet de renvoyer un classement des films les mieux noté.
- En utilisant également les données du fichier names.item
  - Trouver le nom des films.
  - Afficher les noms en Majuscule

Tp global

- En utilisant, les set de données marvel-graph et superhero:
  - Donner en fonction des connexions de chaque super héros, le héros le plus populaire.

Merci pour votre attention

Des questions ?



Découvrez également
l'ensemble des stages à votre disposition
sur notre site **m2iformation.fr**

m2iformation.fr