Introduction Spark

Mr DIATTARA Ibrahima

Consultant Senior Big Data

Objectif du cours

- Comprendre l'architecture du framework de calcul distribué **Spark**
- Utiliser **Databricks**
- Coder en **PySpark**
- Utiliser **Spark SQL**
- Orchestrer un job Spark avec **Airflow**

Lien: https://community.cloud.databricks.com

Sommaire

Qu'est-ce que Spark

Spark est un Framework pour les calculs distribués (répartis ou partagés), on repartie le traitement sur plusieurs microprocesseur de différentes machines

Spark est en développé en scala

Qu'est-ce que Spark

Construit pour effectuer des analyses sophistiquées et conçu pour la rapidité et

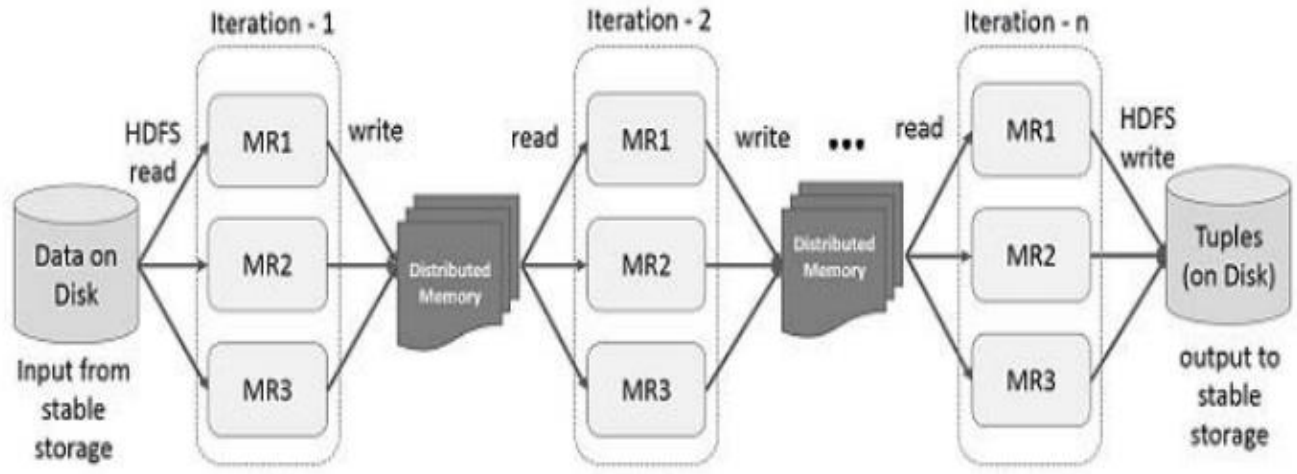
la facilité d’utilisation

Mapreduce Hadoop vs Spark

Traitement des données en mémoire :

•**MapReduce** : MapReduce traite les données de manière séquentielle et stocke les résultats intermédiaires sur le disque après chaque étape du traitement. Cela peut entraîner des E/S disque coûteuses, ce qui peut ralentir les performances.

•**Spark** : Spark effectue le traitement en mémoire autant que possible, minimisant ainsi les E/S disque. Il utilise une structure de données résiliente distribuée (RDD) pour stocker les données en mémoire, ce qui accélère considérablement les opérations.



Mapreduce Hadoop vs Spark

En résumé, Spark offre des performances améliorées grâce au traitement en mémoire, une API plus conviviale et une plus grande polyvalence par rapport à MapReduce, qui est mieux adapté aux tâches de traitement par lots traditionnelles.

Cependant, le choix entre les deux dépendra des besoins spécifiques de votre application et de votre écosystème de données existant

Mapreduce Hadoop vs Spark

En résumé, Spark offre des performances améliorées grâce au traitement en mémoire, une API plus conviviale et une plus grande polyvalence par rapport à MapReduce, qui est mieux adapté aux tâches de traitement par lots traditionnelles.

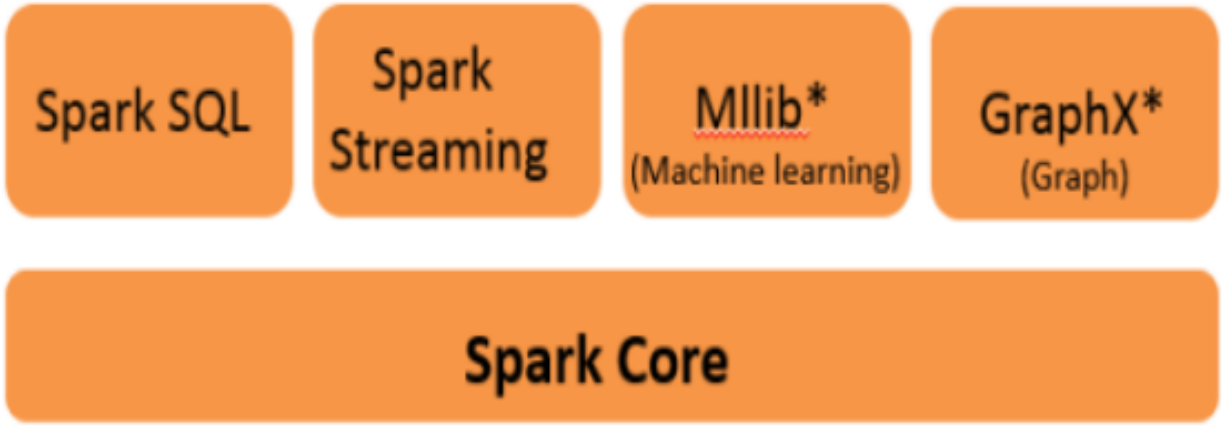
Cependant, le choix entre les deux dépendra des besoins spécifiques de votre application et de votre écosystème de données existant

Fonctionnalités

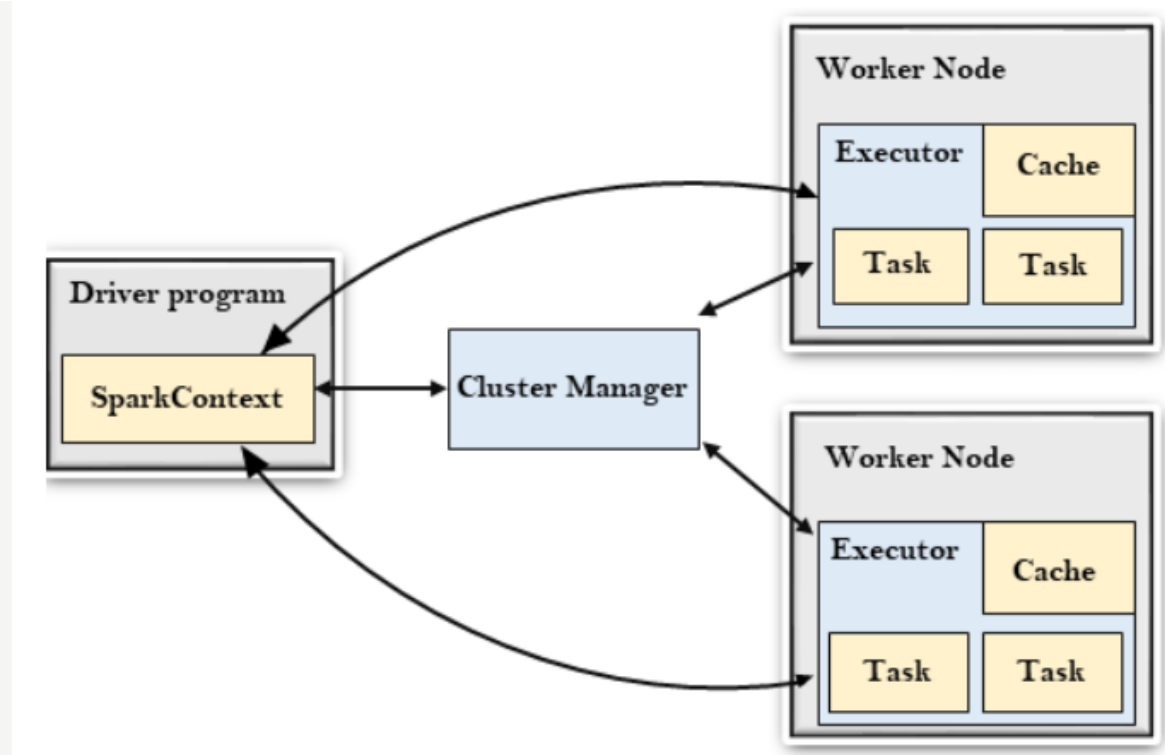
Le Framework Spark possède plusieurs fonctionnalités.

• Le Spark Core: c'est le système central de Spark. C'est une brique dédiée au traitement distribué des données (comme Hadoop MapReduce).

• Les modules avancés: Ils sont développés au-dessus de Spark Core et permettent de faire des traitements complexes (Streaming, machine learning, SQL…)



Mini Architecture



Mini Architecture

SparkContext :

Rôle : Le SparkContext est un objet qui représente la connexion à un cluster Spark. Chaque application Spark a un SparkContext.
Fonctions :
Il initialise les configurations et les ressources nécessaires à l'application Spark.
Il permet de créer des RDD (Resilient Distributed Datasets) et de les manipuler.
Il coordonne l'ensemble du traitement dans l'application Spark.

Driver :
Rôle : C’est le point d'entrée principal d'une application Spark. Il contient la fonction main() de l'application Spark et coordonne l'exécution globale du programme.
Fonctions :
Il convertit le code source Spark en un plan d'exécution logique.
Il communique avec le Cluster Manager pour obtenir des ressources (comme des nœuds d'exécution) pour les tâches Spark.
Il planifie et distribue les tâches aux nœuds d'exécution Spark (les "executors")

Mini Architecture

Cluster Manager (gestionnaire de cluster) :

Rôle : Le gestionnaire de cluster est responsable de la gestion des ressources du cluster, notamment la fourniture de nœuds d'exécution (executors) à l'application Spark.

Fonctions :

Il alloue des ressources (CPU, mémoire) aux tâches Spark.

Il surveille l'état des nœuds d'exécution et des tâches.

Il peut être intégré à des gestionnaires de cluster tels que YARN, Mesos, ou il peut être autonome (Standalone).

Worker Node (nœud d'exécution) :

Rôle : Les nœuds d'exécution sont les machines physiques ou virtuelles où les tâches Spark sont exécutées. Chaque cluster Spark peut avoir plusieurs nœuds d'exécution.

Fonctions :

Ils exécutent les tâches Spark, telles que les mappers et les reducers.

Ils stockent les données en mémoire pour un accès rapide par les tâches.

Ils communiquent avec le driver et le cluster manager pour signaler leur disponibilité.

Task (tâche) :

Rôle : Une tâche est une unité d'exécution de travail dans Spark. Les tâches sont exécutées sur les executors.

Fonctions :

Elles effectuent des opérations spécifiques sur les données, telles que des transformations et des actions, comme un map ou un reduce.

Elles sont distribuées sur les nœuds d'exécution pour le parallélisme.

Opérartions

Lazy Evaluation : L'évaluation paresseuse est une stratégie d'évaluation qui maintient l'évaluation d'une expression jusqu'à ce que sa valeur soit nécessaire. Cela évite l'évaluation répétée ce qui permet une optimisation des étapes du traitement.

Les transformations : Ce sont des fonctions qui retournent une nouvelle SD(structure de données) . Rien n'est évalué lorsque l'on fait appel à une fonction de transformation, cette fonction prend juste une nouvelle et retourne un nouveau SD. Les fonctions de transformation sont par exemple : map, filter, flatMap, groupByKey, reduceByKey, aggregateByKey, Exp: df.filter(_.age > 21),

Nous avons deux type de transformation Narrow opération qui nécessitent un shuffle après un stage et Wide operation qui n' ont pas besoin

Les actions : les actions évaluent et retournent une nouvelle valeur. Au moment où une fonction d'action est appelée sur une SD, toutes les requêtes de traitement des données sont calculées et le résultat est retourné. Les actions sont par exemple : reduce, collect, count, first, take, countByKey et foreach

Les Types de données

- **RDD** (Resilient Distributed Dataset) -depuis la version 1.0.

Un RDD est une collection distribuée immutable de données calculée à partir d'une source et conservée en mémoire vive (tant que la capacité le permet). Les données sont organisées en objets par ligne .

- **DataFrames** - Spark introduit des DataFrames dans la version 1.3

Un DataFrame est une collection distribuée immutable de données. Contrairement à un RDD, les données sont organisées en colonnes nommées, comme une table dans une base de données relationnelle.

- **DataSet -** Spark a introduit le jeu de données dans 1.6. DataSet =Dataframe+RDD
- **Discretized Stream (DStream)**: est une séquence continue de RDD (du même type) , une fonction qui est utilisée pour
- générer un RDD après chaque intervalle de temps

!!!!Dans ce cours on se limitera que sur les DataFrame

https://github.com/idiattara/SDA_2025/blob/main/RDD_DF.py