Spark

Traitement des données avec Spark

Objectif

- Apache Spark
- Langage Scala
- Histoire d'Apache Spark
- Architecture Spark
- Les RDDS
- Spark Streaming

Apache Spark- Introduction

- *il s'agit d'un moteur de traitement de données open source et large gamme*
- Permettre le traitement en mode batch et en mode streaming.
- il peut accéder à tous les outils de big data
- Il s’exécute sur un cluster Hadoop
- Apache Spark étend Hadoop MapReduce au niveau supérieur
- Spark possède son propre système de gestion de cluster

Apache Spark- Introduction

- *Spark a une capacité de calcule de cluster en mémoire*
- *Augmente la vitesse du traitement d'une application*
- Il utilise Hadoop à des fins de stockage uniquement
- Il offre une API de haut niveau pour les langages de programmations : Java, Python , Scala et R
- Plus important encore, en **comparant Spark avec Hadoop**, il est 100 fois plus rapide que le mode Hadoop In-Memory et 10 fois plus rapide que le mode Hadoop On-Disk.
- Spark a été développé en scala

Langage Scala - Histoire

- Scala comme « Scalable langage»
- Créé en 2003 par Martin Odersky , professeur à l'Ecole Polytechnique de Lausanne .
- Il a collaboré à la création du compilateur javac1.3 et est à l'origine des Génériques de Java 5.

Langage Scala - Histoire

- Scala comme « Scalable langage»
- Créé en 2003 par Martin Odersky , professeur à l'Ecole Polytechnique de Lausanne .
- Il a collaboré à la création du compilateur javac1.3 et est à l'origine des Génériques de Java 5.

Langage Scala - Histoire

- 2001 : Création du langage
- 2003 : 1.0
- 2006 : 2.0
- 2013 : 2 .10
- 2014 : 2.11
- Octobre 2017 : 2.12.4
- Septembre 2018: 2.12.7

Langage Scala – points clés

- Fusion entre approche fonctionnelle et orientée objets Pas un langage fonctionnel pur comme « Haskell » Support des concepts objets Interopérable avec java.
- implémente les concepts clés de la programmation fonctionnelle dans un contexte orienté performance
- Immutabilité
- Récursivité

Les concepts de Scala – motivations

Langage scala – points clés

- Contexte de forte montée en charge
  - Big data
  - NoSQL
- Contexte agile
  - Utilisation du REPL (Read -Evaluate -Print - Loop)
  - Scripting
- Qualité logicielle
  - Moins de code moins de bugs
  - Compilateur plus exigeant (typage fort statique)

Langage scala – points clés

- SBT : Simple Build Tool
- Equivalent à Maven et Gradle
- Un outil de build pour Scala et java
- Compilation incrémentale
- Support natif des principales Framework de test Scala (Scala test, JUnit)
- Gestion de projet simple ou multi-projets
- Exécution parallèle des tâches (ainsi que les tests unitaires)

Les concepts scala – généralités

- Fichier source « *.scala »
- Fichier compilé « *.class »
- Le ; est optionnel
- Syntaxe proche de celle de java
- Plus concis
- Fortement typé
- Tout est Objet (même les opérateurs)

Les concepts de scala – les variables

- Explicite

```
val immutable : String = "World"
val immutable = "World»
```

- Inférence de type

```
var mutable : String = "Hello«
var mutable = "Hello«
```

```
mutable = "j'ai change d'avis" // OK
immutable = "Moi aussi ! " // KO compiler error!(Reassignment to val)
```

Les concepts de scala – les fonctions

- **Déclaration*:***

```
def functionName ([list of parameters]) : [return type]
```

- **Il n'y a pas de mot clé *return* comme en java**
- **Le résultat de la dernière instruction sera renvoyé**

Apache Spark - Histoire

- Au début, en 2009, Apache Spark a été introduit dans le laboratoire de R&D de l'UC Berkeley,
- désormais connu sous le nom d'AMPLab.
- en 2010, il est devenu open source sous licence BSD.
- en 2013 a été donné à Apache Software Foundation.
- en 2014, il est devenu un projet Apache de haut niveau.

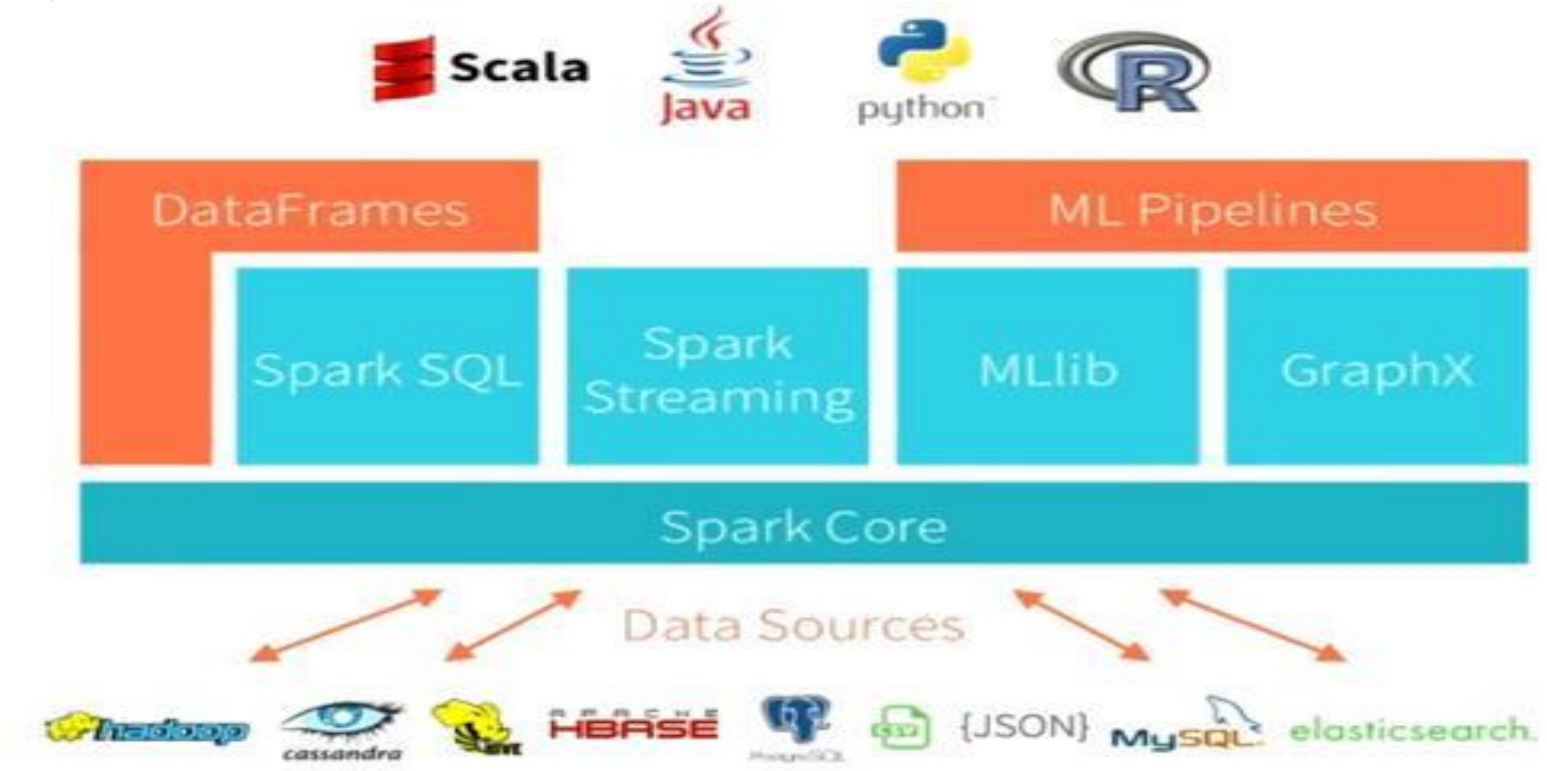
Pourquoi Apache Spark ?

- Comme nous le savons, il n'y avait pas de moteur informatique à usage général dans l'industrie, puisque
- Pour effectuer le traitement par lots, nous **utilisions Hadoop MapReduce**.
- De plus, pour effectuer le traitement des flux, nous utilisions Apache Storm / S4.
- De plus, pour le traitement interactif, nous utilisions Apache Impala / Apache Tez.
- Pour effectuer le traitement des graphes, nous utilisions Neo4j / Apache Giraph.

Pourquoi Apache Spark ?

- il n'y avait pas de moteur puissant dans l'industrie, capable de traiter les données à la fois en temps réel et en mode batch
- il y avait une exigence qu'un moteur puisse répondre en moins d'une seconde et effectuer un traitement en mémoire
- Spark offre un traitement de flux en temps réel, un traitement interactif, un traitement graphique, un traitement en mémoire ainsi qu'un traitement par lots.
- une vitesse très rapide, une facilité
- ces fonctionnalités créent la différence entre Hadoop et Spark.

Écosystème Spark



Écosystème Spark – Spark Core

- Spark Core est un point central de Spark.
- il fournit une plate-forme d'exécution pour toutes les applications
- il fournit une plate-forme généralisée.

Écosystème Spark – Spark SQL

- Spark **SQL** permet aux utilisateurs d'exécuter des requêtes SQL/HQL.
- Nous pouvons traiter des données structurées ainsi que semi-structurées, en utilisant Spark SQL.
- il propose d'exécuter des requêtes non modifiées jusqu'à 100 fois plus rapidement sur les déploiements existants.

Écosystème Spark – Spark Streaming

- Il permet un traitement de flux évolutif, à haut débit et tolérant aux pannes de flux de données en direct
- Les données peuvent être ingérées à partir de nombreuses sources telles que Kafka, Kinesis

Écosystème Spark – Spark Streaming

Écosystème Spark – Spark Streaming

- Il permet un traitement de flux évolutif, à haut débit et tolérant aux pannes de flux de données en direct
- Les données peuvent être ingérées à partir de nombreuses sources telles que Kafka, Kinesis
- Spark Streaming reçoit des flux de données d'entrée en direct et divise les données en lots, qui sont ensuite traités par le moteur Spark pour générer le flux final de résultats par lots.

Écosystème Spark – Spark MLLIB

- La bibliothèque d'apprentissage automatique offre à la fois des efficacités et des algorithmes de haute qualité.
- Puisqu'il est capable de traiter les données en mémoire, cela améliore considérablement les performances de l'algorithme itératif.

Écosystème Spark – Spark Graphx

- Spark GraphX est le moteur de calcul graphique construit sur Apache Spark qui permet de traiter les données graphiques à grande échelle.

Écosystème Spark – SparkR

- pour utiliser Apache Spark à partir de **R**
- Le package R qui donne une interface légère.
- il permet aux data scientists d'analyser de grands ensembles de données.
- il Permet également d'exécuter des tâches de manière interactive sur eux à partir du shell R.
- L'idée principale derrière SparkR était d'explorer différentes techniques pour intégrer la convivialité de R avec l'évolutivité de Spark.

Spark - RDD

- L'abstraction clé de Spark est RDD.
- RDD est un acronyme pour Resilient Distributed Dataset.
- l'unité de données fondamentale dans Spark.
- Il s'agit d'une collection distribuée d'éléments sur les nœuds de cluster.
- Effectue également des opérations parallèles.
- les RDD Spark sont de nature immuable.
- il peut générer un nouveau RDD en transformant le Spark RDD existant.

Spark – Création de RDD

- Il existe trois façons de créer un RDD :
  - **Collections parallélisées**

    En appelant la méthode parallelize dans le programme pilote, nous pouvons créer des collections parallélisées.
  - **Ensembles de données externes**

    On peut créer des RDD Spark, en appelant une méthode textFile. Par conséquent, cette méthode prend l'URL du fichier et la lit comme une collection de lignes.
  - **RDD existants**

    De plus, nous pouvons créer un nouveau RDD dans Spark, en appliquant une opération de transformation sur les RDD existants.

Spark – Opération Spark RDD

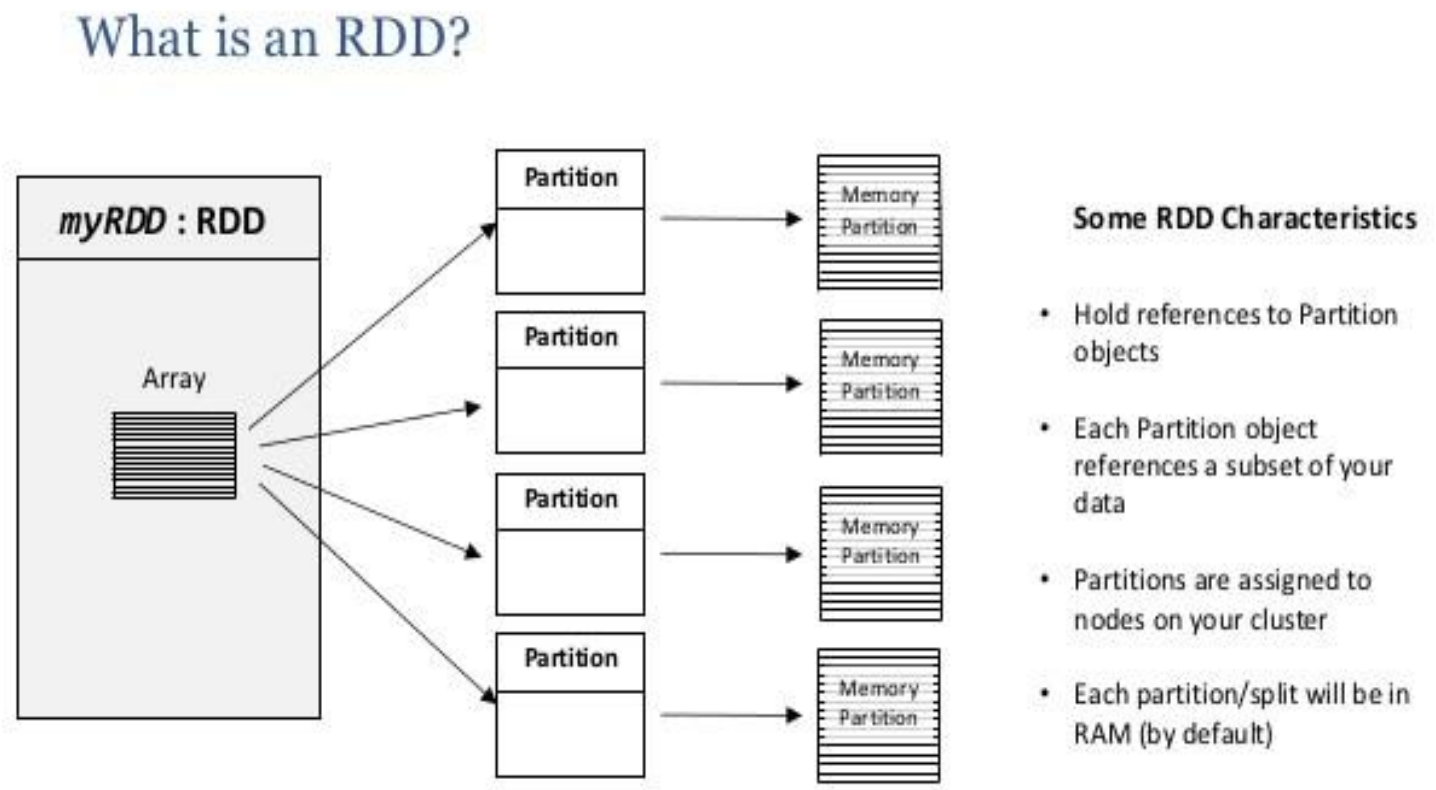
- Il existe deux types d'opérations prises en charge par les RDD Spark :
  - Opérations de transformation :

    Il crée un nouveau Spark RDD à partir de celui existant. De plus, il transmet l'ensemble de données à la fonction et renvoie un nouvel ensemble de données.

  - Opérations d'action

    Action renvoie le résultat final au programme pilote ou l'écrit dans le magasin de données externe.

Spark RDD– les caractéristiques



Spark RDD– les caractéristiques

- **Calcul en mémoire**
  - Fondamentalement, lors du stockage des données dans RDD, les données sont stockées en mémoire aussi longtemps que vous le souhaitez. Il améliore les performances d'un ordre de grandeur en gardant les données en mémoire.

Spark RDD– les caractéristiques

- **Lazy Evaluation**
  - Spark Lazy Evaluation signifie que les données à l'intérieur des RDD ne sont pas évaluées en déplacement. seulement après qu'une action a déclenché tous les changements ou que le calcul est effectué. Par conséquent, cela limite la quantité de travail qu'il doit faire.

Spark RDD– les caractéristiques

- Immutabilité

  L'immuabilité signifie qu'une fois que nous créons un RDD, nous ne pouvons pas le manipuler. De plus, nous pouvons créer un nouveau RDD en effectuant n'importe quelle transformation. De plus, nous atteignons la cohérence grâce à l'immuabilité.

Spark RDD– les caractéristiques

- Immutabilité

  L'immutabilité signifie qu'une fois que nous créons un RDD, nous ne pouvons pas le manipuler. De plus, nous pouvons créer un nouveau RDD en effectuant n'importe quelle transformation. De plus, nous atteignons la cohérence grâce à l'immutabilité.

Spark RDD– les caractéristiques

- La persistance

  En mémoire, nous pouvons stocker le RDD fréquemment utilisé. De plus, nous pouvons les récupérer directement de la mémoire sans passer par le disque. Il en résulte une rapidité d'exécution.
  nous pouvons effectuer plusieurs opérations sur les mêmes données. Cela n'est possible qu'en stockant explicitement les données en mémoire en appelant la fonction persist() ou cache().

Spark RDD– les caractéristiques

- La partitionnement :

  Le RDD partitionne les enregistrements de manière logique. Distribue également les données sur différents nœuds du cluster.

  les divisions logiques sont uniquement destinées au traitement et en interne, il n'y a pas de division. Par conséquent, il fournit le parallélisme.

Spark RDD– les caractéristiques

- Le traitement parallèle

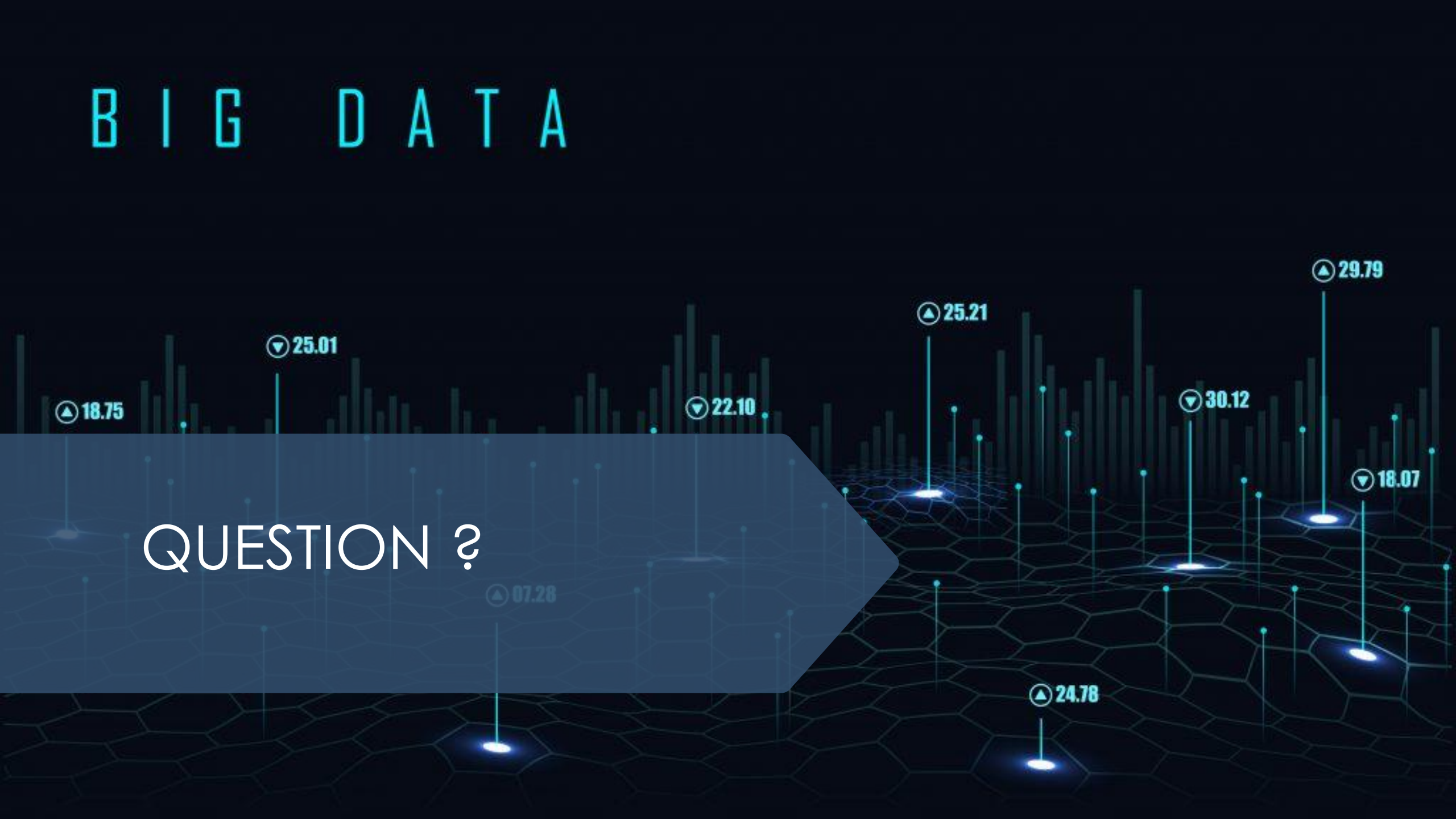
  RDD traite les données en parallèle sur le cluster.


LAB.

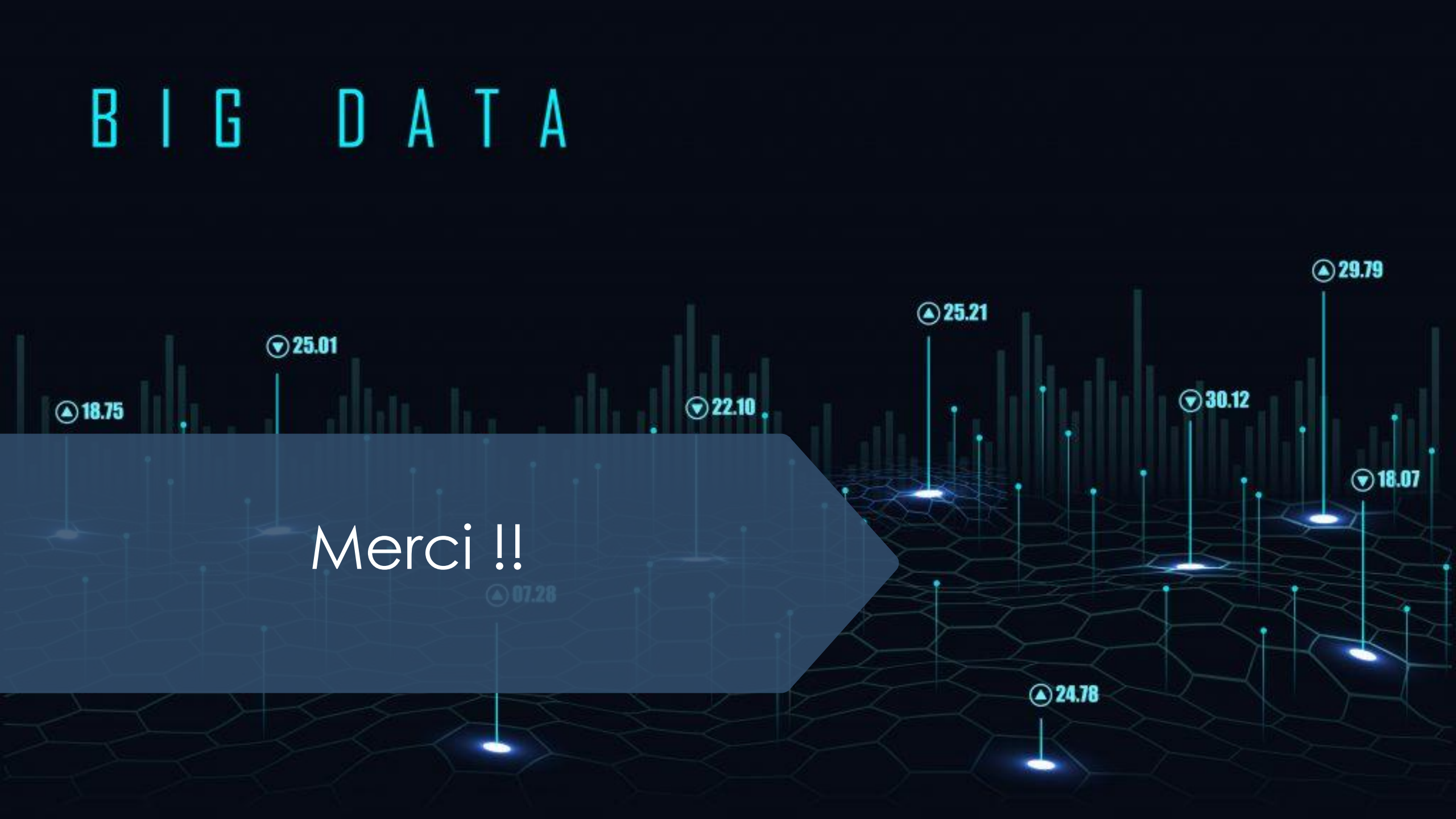
Lancement du spark shell

```
[root@sandbox-hdp ~]# spark-shell
SPARK_MAJOR_VERSION is set to 2, using Spark2
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://sandbox-hdp.hortonworks.com:4040
Spark context available as 'sc' (master = local[*], app id = local-1658877586923).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.3.0.2.6.5.0-292
      /_/

Using Scala version 2.11.8 (OpenJDK 64-Bit Server VM, Java 1.8.0_171)
Type in expressions to have them evaluated.
Type :help for more information.
```



QUESTION ?



Merci !!