Hadoop

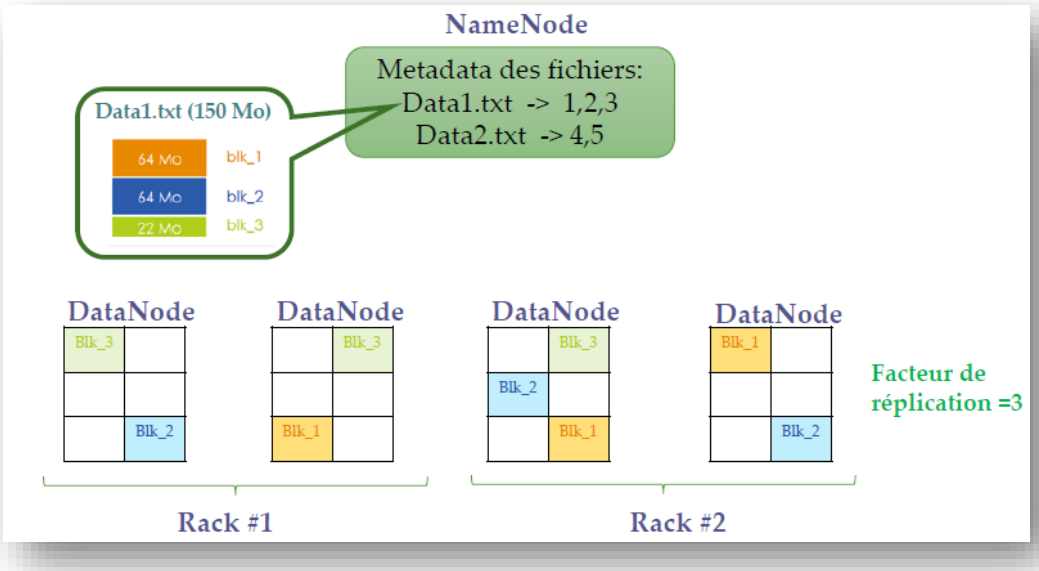
- Diviser les données en plusieurs parties pour les stocker sur plusieurs machines
- Sert dans le traitement de grands volumes de données
- Principes :
  - Division des données
  - Sauvegarder dans des Clusters (collection de machines)
  - Traiter les données dans les Clusters
- Avantages :
  - Forte tolérance aux pannes
  - Sécurité des données
  - Complexité réduite
  - Coût réduit

⟹ HDFS : Stockage des données

⟹ MapReduce : Traitement des données

HDFS

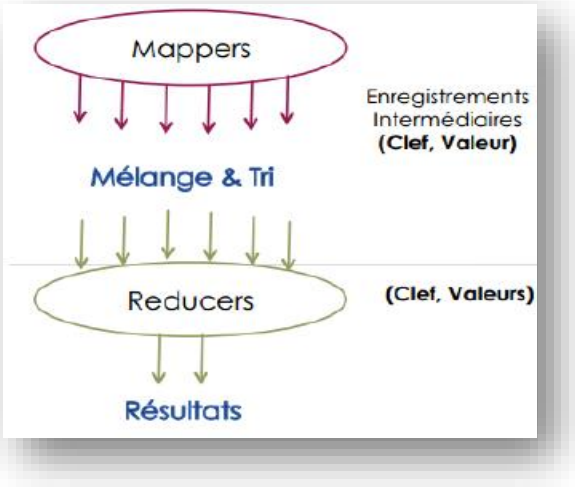
- Avantages :
  - Traitement rapide
  - Garantir la lecture malgré la défaillance d'une machine



- **Problèmes du DataNode :**
  - **Si un nœud a un problème, les données sont perdues**
- **Solutions :**
  - **Hadoop réplique chaque bloc 3 fois**
  - **Place une copie du bloc dans 3 nœuds au hasard**
  - **Si un nœud est endommagé, le NN réplique ses blocs encore**
- **Commandes :**
  - **Hadoop fs -commande (-help)**
  - **Cat : Afficher le contenu d'un doc**
  - **Cp : Copier un fichier de HDFS vers HDFS**
  - **Ls : Lister**
  - **create table A(x "type", ...) row format delimited field terminated by ',' stored as textfile;**
  - **load data local inpath 'CHEMIN' overwrite into table A; (=> Add file in an existing table in the database)**

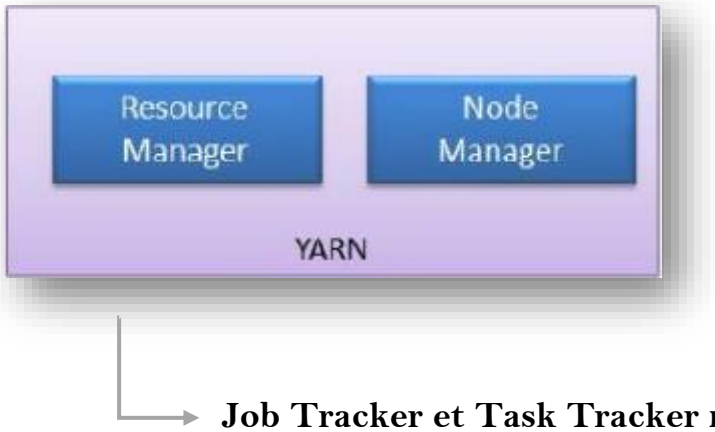
MapReduce

- **Permet de traiter des données volumineuses de manière parallèle et distribuée**
- **Il est basé sur 2 étapes :**
  - **Mapping : Analyser les données brutes du HDFS afin de les sortir**
  - **Réduction : Récupérer les données sorties et les analyser pour extraire les données les plus importantes**



- **Fonctionne avec 2 processeurs :**
  - **Job Tracker :**
    - **Planifie les taches**
    - **Affecte les taches au Task Tracker**

        - Gère le Map Reduce
        - Récupère les erreurs et redémarre les taches lentes ou qui ont échoué
    - **Task Tracker :**
        - Notifie le Job Tracker du niveau de progression d'une tache et la notifie lors d'une erreur
        - S'exécute sur chacun des nœuds
        - Traite un bloc sur la même machine que lui
- Modèle de gestion de mémoire basé sur les slots :
    - Configurés au démarrage
    - Une tache est exécutée sur un slot
- **YARN :**
    - Traitement de grandes quantités de données (PetaBytes …) dans HDFS en utilisant des applications



Job Tracker et Task Tracker n'existent plus

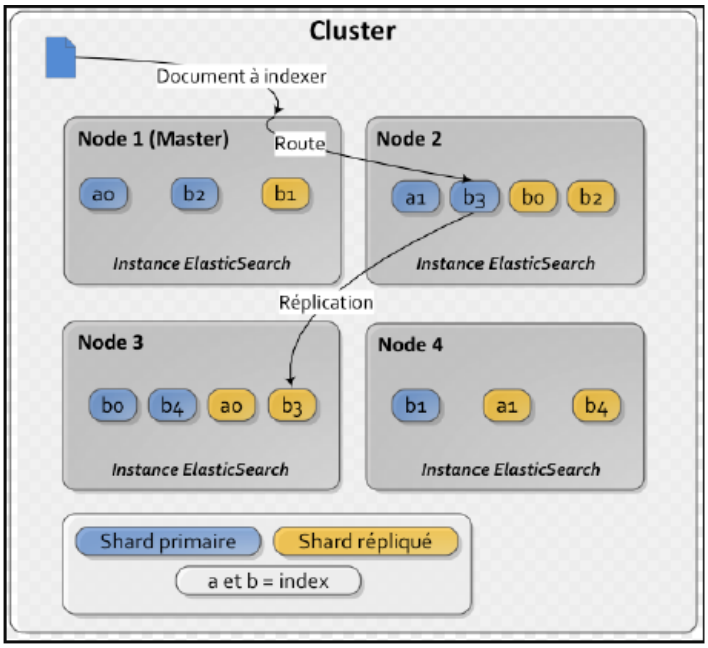
- Commandes:
    - $ Hadoop jar app.jar data.txt output

Exécution d'un job

Elastic Search

- C'est un outil de recherche et d'analyse
- Un stockage de document temps réel distribué où tous les champs sont indexés et consultables
- Architecture :
    - **Nœud :** une instance
    - **Cluster :** Composé de plusieurs nœuds, dont un nœud maître
    - **Index :** Espace logique de stockage de documents
    - **Shard :** Instance Lucéne

- **Prim. Shard : 5 Shards primaires impossible de les changer après création**
- **Sec. Shard : Partition répliquées (infinité)**



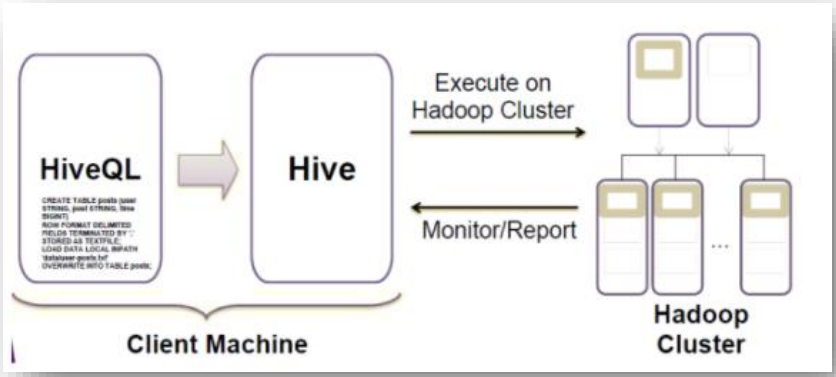
- **Commandes :**

Add	**PUT /boutique/_doc/1** **{ "titre" : "19 Nocturnes",** **"artiste" : "Arthur Rubinstein",** **"compositeur" : "Fryderyk Chopin",** **"genre" : "romantique "}**
View	**GET /boutique/_doc/1**
View all	**GET /boutique/_search**
Seach (=x)	**GET /boutique/_search ?q=x**
Delete doc	**GET /boutique/_doc/1**
Delete index	**GET /boutique/**

Hive

- **Il consiste à réduire la taille des programmes Java**

- **Il traduit les requêtes HiveQL en un ensemble de jobs MapReduce qui seront exécutés dans un cluster Hadoop**



- **Hive utilise le langage SQL (insert, drop, select …)**
- **Commandes :**

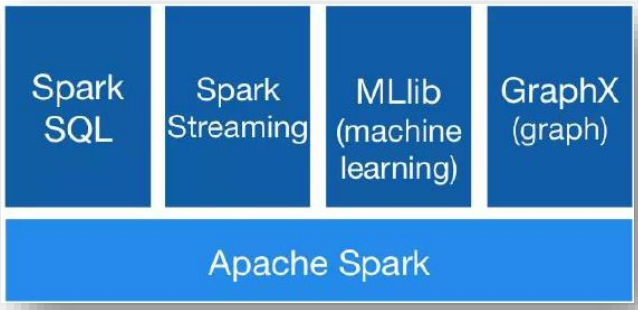
mysql-u root–p
MySQL> show databases;
MySQL> show tables;
MySQL> select TBL_NAME from TBLS;
Hive
Create database test;
Use test;
…

Spark

- **Il essaye de stocker le plus possible en mémoire avant de basculer sur disque**
- **Il est capable de travailler avec une partie des données en mémoire et une autre sur disque**
- **Spark Streaming : Traitement des données à temps réel [reçu de diff. Sources et envoyé à un système d'ingestion de données (Kafka, …)**

- Inconvénients :
  - Pas de récupération auto en cas d'erreur
  - Combinaison Streaming, batch, interactif impossible
- Avantages :
  - Découpe les données en micro Batch

- Spark SQL : Extraction et transformation des données sous plusieurs formats (JSON, BD, Parquet)
- Spark MLIB : Librairie de machine learning
- SparkGraphX : Traitement et parallélisation des graphes



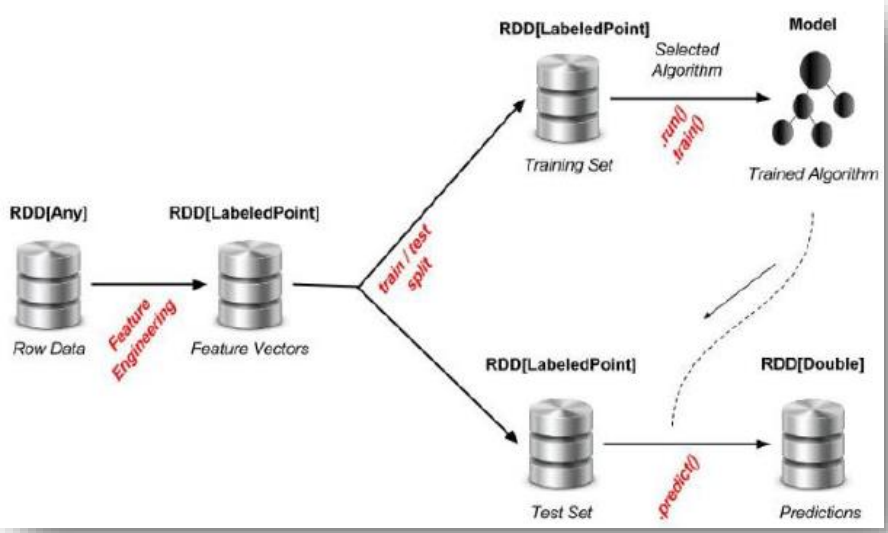
- Driver :
  - Exécute la fonction Main et crée le SparkContext
  - Contient plusieurs composants qui sont responsables de la traduction du code Spark en Job (ensembles de taches Tasks)
  - Planifie l'exécution des jobs et négocie les ressources avec le cluster manager
- Cluster Manager :
  - Service externe responsable de l'allocation des ressources aux jobs et peut être de type Yarn ou autre
- Une partition (qui constitue le RDD) est une division logique de données qui est immuable
- Pandas : sert dans la création des graphes
- Commandes :

RDD.collect()	Retourne le contenu de RDD
RDD.count()	Retourne le nombre d'éléments
RDD.first()	Retourne le premier élément
RDD.take(n)	Retourne les n premiers éléments
RDD.reduce(F)	Joindre les éléments de RDD avec une fonction F
RDD.persist() RDD.cache()	Sauvegarde RDD en mémoire
RDD.saveAsTextFile(path)	Sauvegarder le RDD sous forme txt
sc.parallelize(array)	Ajouter un tab.
sc.textfile('path')	Prendre un fichier
RDD.map(F)	Retourne une valeur mise en dans le RDD
RDD.flatMap(F)	Items du RDD source = 0 ou autres
RDD.filter(F)	Filtre de recherche
df.sql(requête).show()	Afficher une requête SQL
Df.printSchema()	Description d'un schéma
Df.select(x)	Affichage d'un champ x
Df.limit(n)	Retourne un data frame avec les n premiers n-uplets
Df.join(x, condition, type)	Join de df avec x

Spark MLab

- **Supervisé :**
  - On dispose d'un Data Set compose de features associées à des labels (target)
    - **Algo de classification :** le label est une classe (mail : spam ou non)
    - **Algo de régression :** le label est prédit (la taille en fonction du poids et âge)

- **Non supervisé :**
  - **On ne dispose pas de label pour nos données**
  - **On doit alors trouver des similarités entre les objets observés, pour les regrouper au sein de clusters**
- **Les algos implémentés nécessitent en entrée :**
  - **RDD Vector : Vecteurs de doubles**
  - **RDD Labeled Point : Vecteur + Label**
  - **RDD Rating : Tuple**

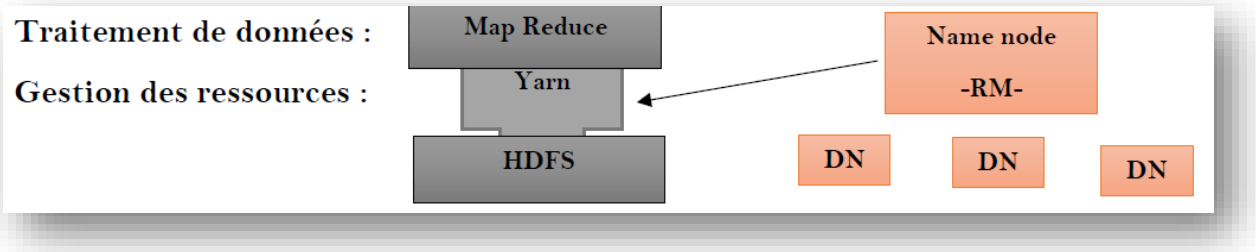


- **Commandes :**

df = sqlContext.read.load('bank.csv', format='com.databricks.spark.csv', header='true',inferSchema='true')	**Ouvrir le fichier bank.csv**
df.printSchema()	**Afficher les colonnes et leurs types**
Df.drop('x')	**Supprimer la colonne x**
Df. groupBy(« x »)	**Trier par x**

Others

- **Architecture Hadoop 2 :**



- **Dans Kibana, l'Index Pattern sert à accélérer la recherche**
-

```
sqoop import-all-tables --connect jdbc:mysql://localhost:3306/regions --username=root
–warehouse-dir=/user/hive/warehouse/regions.db --m=1
```

Le nombre de process qui seront utilisés

Import des données et struct.

```
-as-avrodatafile \
```