IMT Mines Alès
École Mines-Télécom

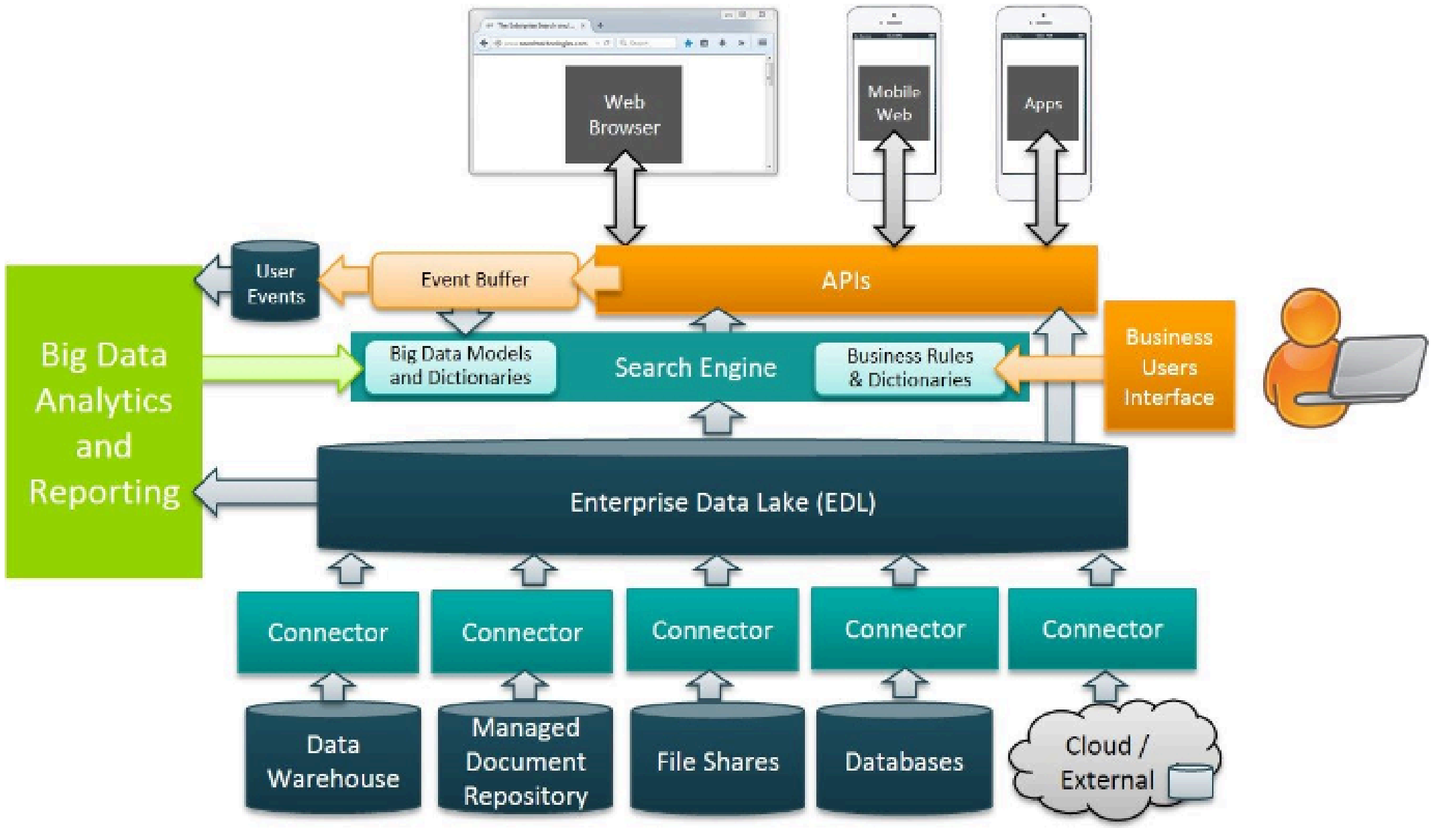
05 FÉVRIER 2026

IMT MINES ALÈS

3e année Développement Logiciel

Big Data et modélisation

Architectures Big Data, Hadoop, Spark et Data Lakes



FORMATEUR : DIALLO ALIMOU

Objectifs pédagogiques

- Concevoir une architecture Big Data
- Comprendre Spark, Hadoop, Data Lake
- Manipuler Spark en conditions quasi réelles
- Comprendre les problèmes de scalabilité
- Implémenter un pipeline de traitement distribué

Hadoop : fondations du Big Data

Pourquoi Hadoop ?

- Les systèmes classiques ne sont pas adaptés :
  - volumes trop importants
  - coûts élevés
  - faible tolérance aux pannes
- Besoin de :
  - stockage distribué
  - traitement parallèle
  - passage à l’échelle horizontal

Hadoop est né pour **stocker** et **traiter** des données massives sur des **clusters** de machines standards.

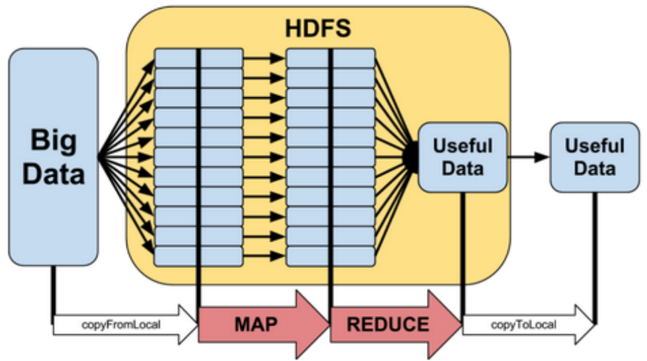
Hadoop : vue d’ensemble

Hadoop est donc une plateforme permettant d'établir un dialogue entre plusieurs machines d'un **cluster**. Ses objectifs sont de résoudre les deux principales problèmes de la manipulation de grandes quantités de données :

- stocker ces données (limitation de la taille des disques dur)
- rechercher rapidement dans ces données (limitation des puissances de calcul)

Pour résoudre ces deux problèmes, **Hadoop** se structure en deux principales couches :

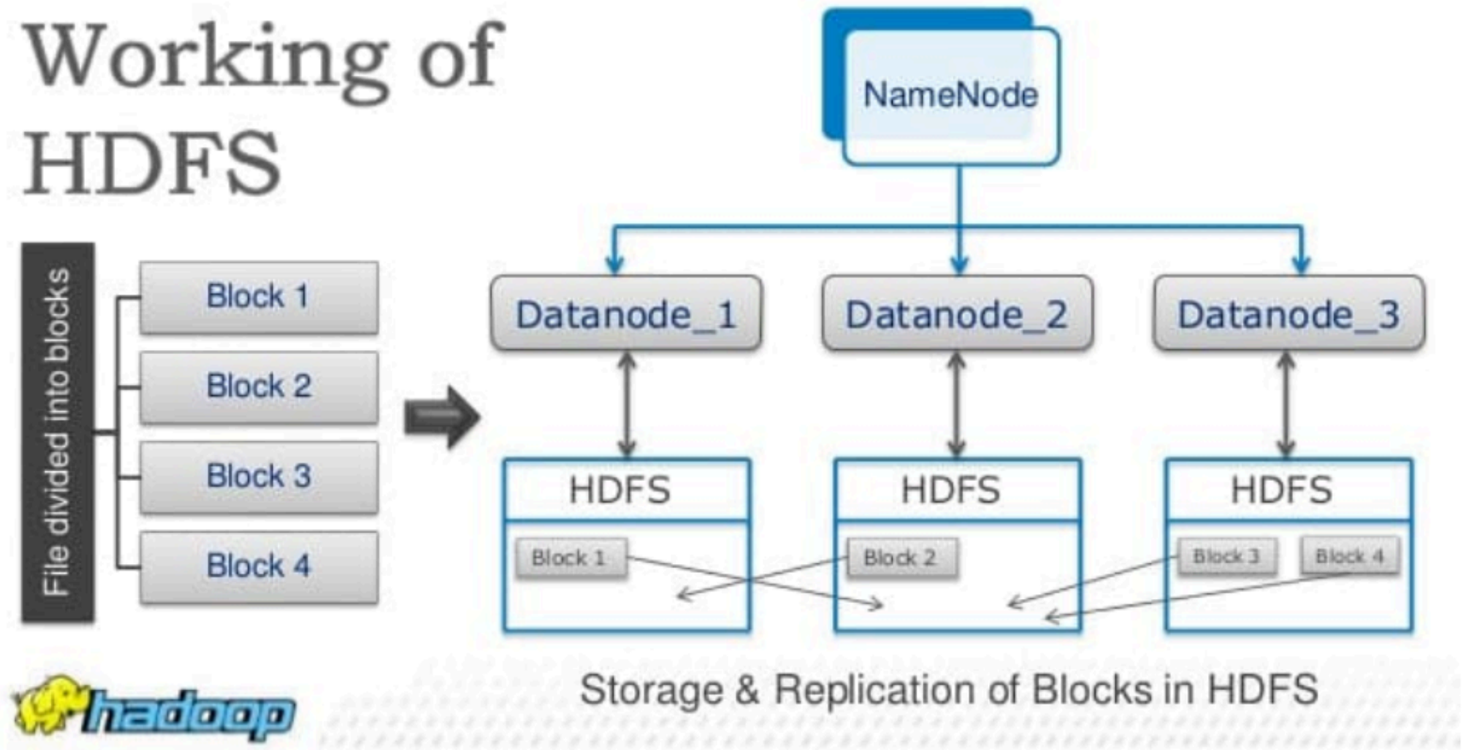
- **HDFS** : Hadoop Filesystem, un système de fichiers virtuel aggrégant le stockage de plusieurs machines d'un cluster
- **Hadoop MapReduce** : un framework logiciel en Java permettant de développer des programmes exécutables de manière distribués grâce à l'utilisation de l'algorithme **MapReduce** développé par Google



HDFS : rôle

HDFS (Hadoop Distributed File System)

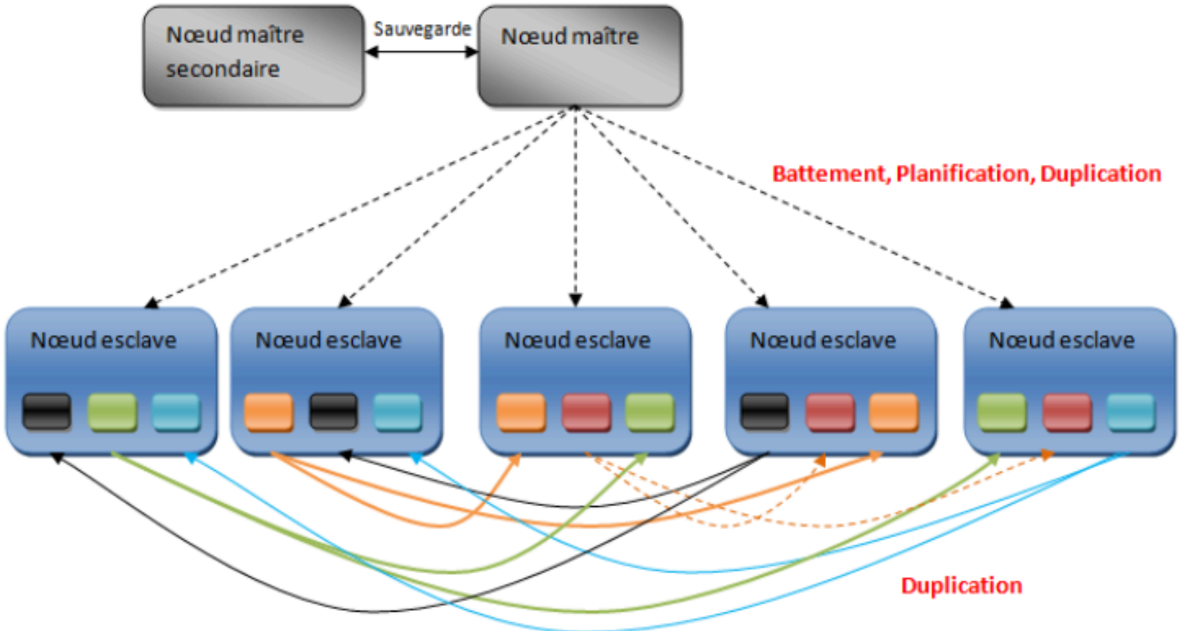
HDFS est un système de fichiers distribué conçu pour stocker de très gros volumes de données de manière fiable et scalable.



HDFS : principes de fonctionnement

Principes clés

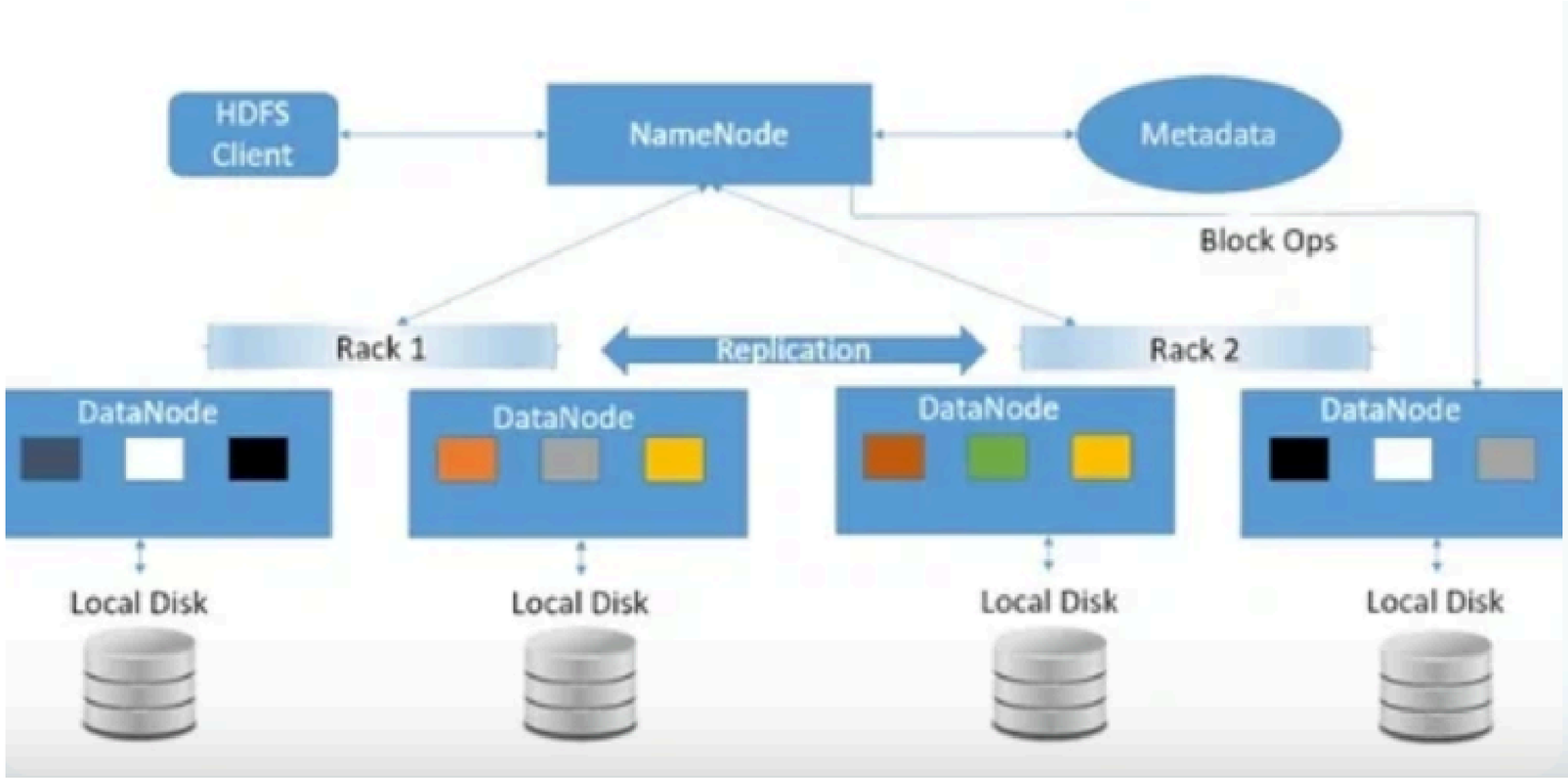
- **Découpage en blocs :**
  - les fichiers sont découpés en gros blocs (ex : 128 Mo)
- **Réplication** :
  - chaque bloc est copié sur plusieurs machines
- **Tolérance aux pannes :**
  - si un nœud tombe, les données restent accessibles ailleurs



HDFS : architecture

Architecture HDFS

- **NameNode** :
  - gère les métadonnées (où sont les fichiers, blocs, etc.)
- **DataNodes** :
  - stockent réellement les blocs de données

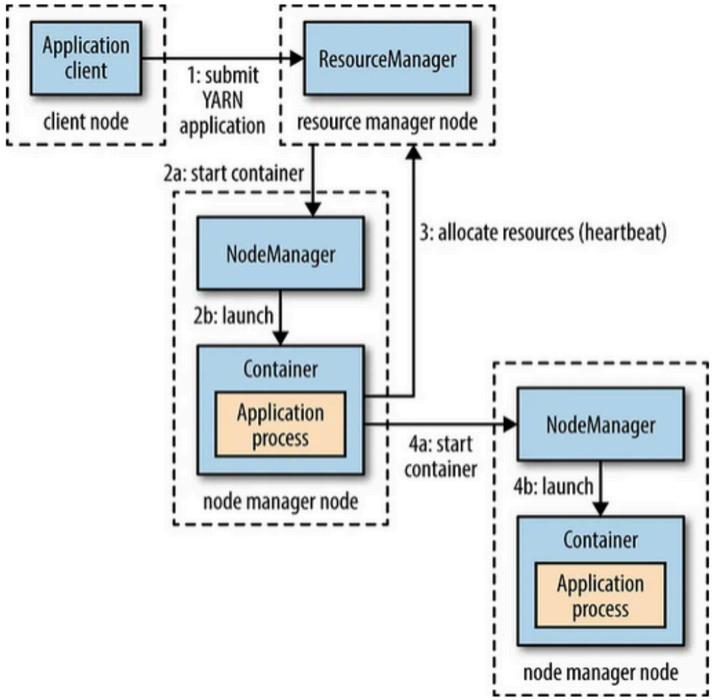


YARN : gestion des ressources

YARN (Yet Another Resource Negotiator) : est le gestionnaire de ressources du cluster Hadoop.

Rôles

- Alloue CPU et mémoire aux applications
- Planifie l’exécution des jobs
- Permet à plusieurs applications de partager le cluster



MapReduce : modèle de calcul

Principe MapReduce

- **Map** : transformation des données en paires clé/valeur
- **Shuffle** : regroupement par clé
- **Reduce** : agrégation finale

Modèle simple pour traiter de très grandes quantités de données en parallèle.

Exemple : WordCount

Map : (mot, 1)

Shuffle : regrouper par mot

Reduce : somme des occurrences

Lorsque l'application MapReduce est lancée, elle crée un composant « Master » responsable de la distribution des données et de la coordination de l'exécution de différentes unités de travail ou « Workers ».

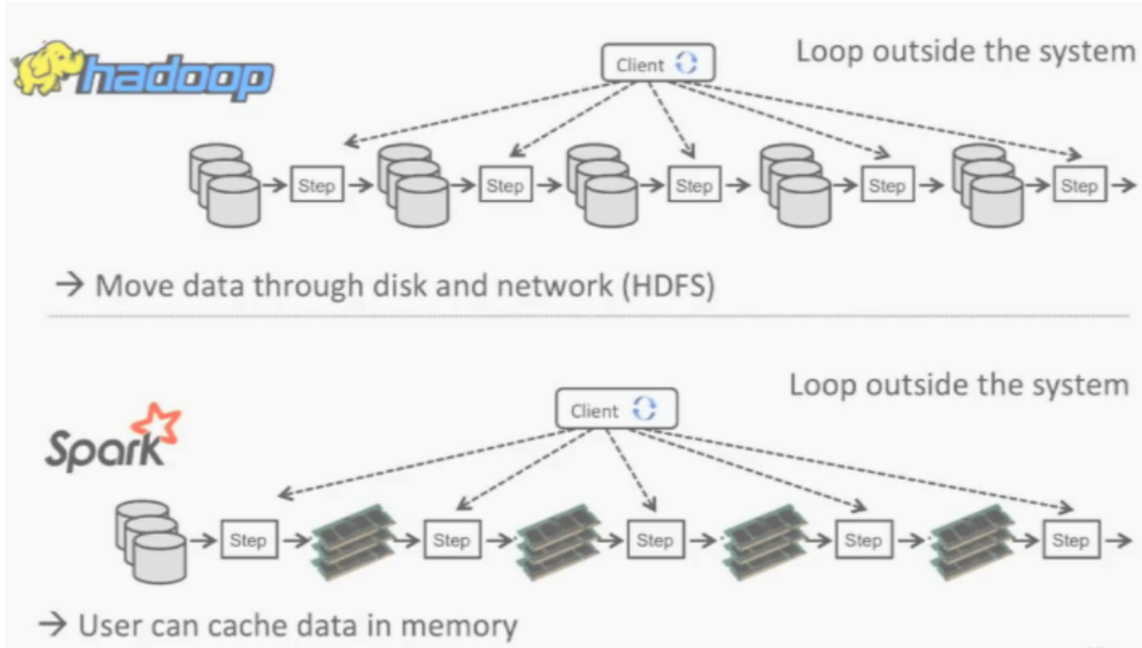
• Le Master attribue aux Workers les tâches **Map et Reduce**

Limites de MapReduce

Problèmes

- Trop lent (beaucoup d’accès disque)
- Pas adapté :
  - au streaming
  - aux traitements interactifs
  - aux algorithmes itératifs (ML)

Ces limites ont conduit à l’émergence de **Spark**.



Apache Spark : moteur moderne du Big Data

Apache Spark est un moteur de calcul distribué pour le traitement de grandes quantités de données, en batch ou en streaming.

Spark a été conçu pour :

- être beaucoup plus rapide
- travailler en mémoire
- supporter plusieurs types de traitements

Spark est aujourd'hui le moteur de calcul Big Data le plus utilisé.

Types de traitements supportés

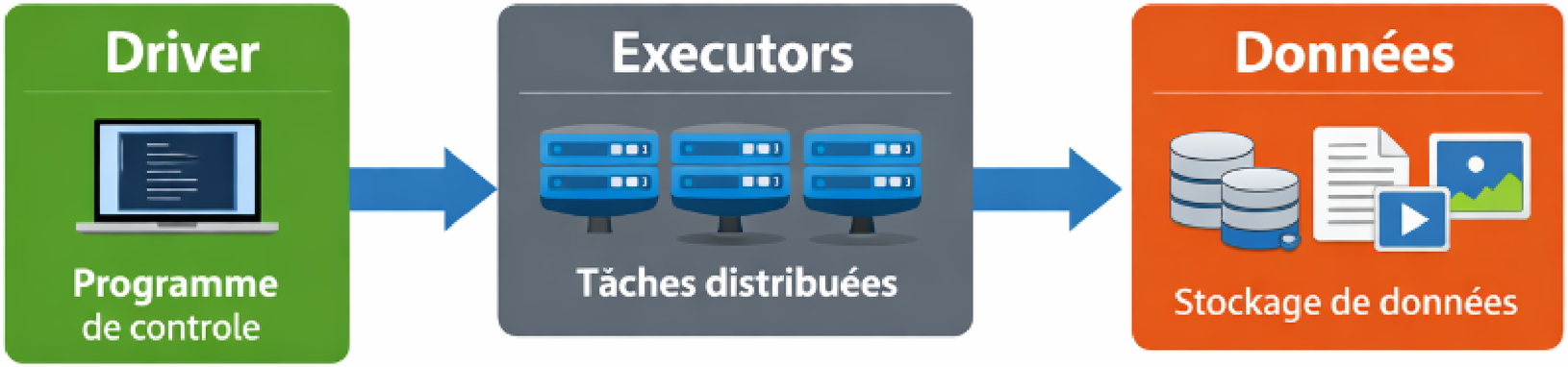
Spark permet :

- Traitement batch
- Traitement streaming
- Requêtes SQL
- Machine Learning
- Traitement de graphes

Architecture Spark

Architecture

- Driver :
  - programme principal
  - orchestre l’exécution
- **Executors** :
  - exécutent les tâches sur le cluster
- **Cluster Manager :**
  - YARN, Kubernetes, Standalone



Abstractions de données

- RDD : structure bas niveau (distribuée, immuable)
- DataFrame : table distribuée optimisée (la plus utilisée)
- Dataset : typé (surtout en Scala/Java)

En pratique, on utilise surtout les DataFrames.

```
val rdd = session.sparkContext.parallelize(0 to 9)
print(rdd.a) // 4
```

RDD : Deux types d'opérations

Transformation	signification
Map()	Crée un nouveau RDD avec les conditions spécifiées dans la fonction anonyme
Filter()	Crée un nouveau RDD contenant les données répondant aux conditions de filtre
Flatmap()	Crée un nouveau RDD au même titre que map(), à la seule différence que le nouveau RDD est applati.
Sample()	Crée un nouveau RDD constitué d'un échantillon de données du RDD source.
Groupbykey()	Effectue une opération GroupBy()

Action	signification
Count()	Compte le nombre d'éléments présents dans le RDD
Collect()	Transfère les données du RDD dans le nœud principal (ou driver) du cluster Spark
Reduce()	Effectue une agrégation entre les éléments du RDD. Il combine les données en utilisant une fonction associative qui produit un résultat au programme pilote.
Save()	Persiste les données du RDD sur le disque dur

Principe clé : Lazy Evaluation

Évaluation paresseuse

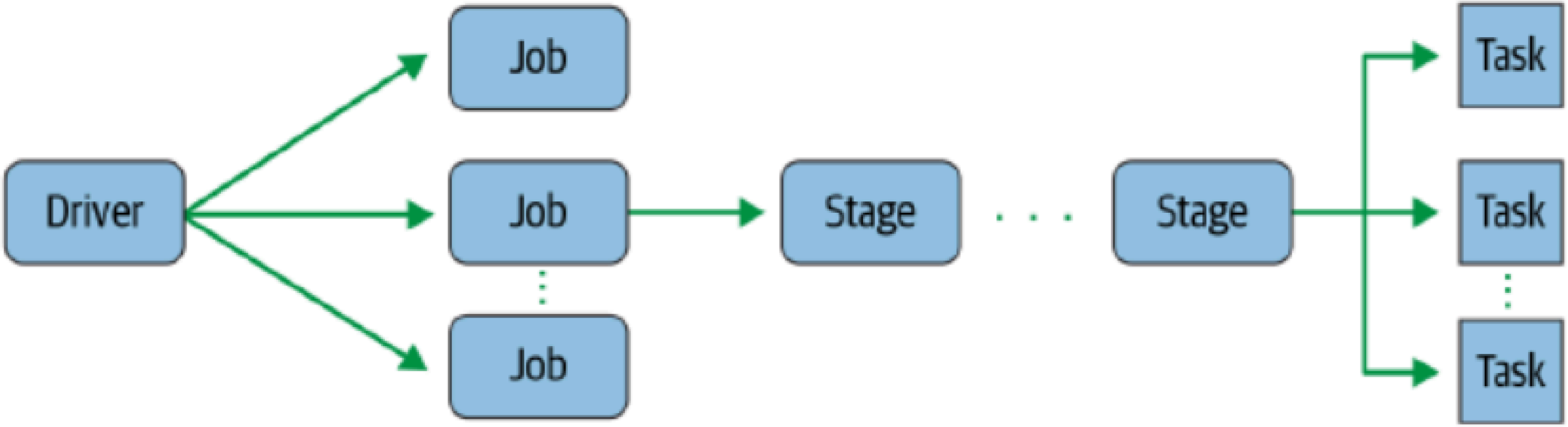
- **Spark** :
  - ne calcule rien tant qu’on n’a pas demandé un résultat
- **Les transformations** :
  - construisent un plan d’exécution (DAG)

L'évaluation paresseuse est un concept clé dans Apache Spark, où les transformations sur les données ne sont pas immédiatement exécutées, mais plutôt leur exécution est retardée jusqu'à ce qu'une action soit déclenchée.

Job, Stage, Task

Organisation interne

- **Job** : déclenché par une action
- **Stage** : groupe d’opérations sans shuffle
- **Task** : unité de calcul exécutée sur une partition



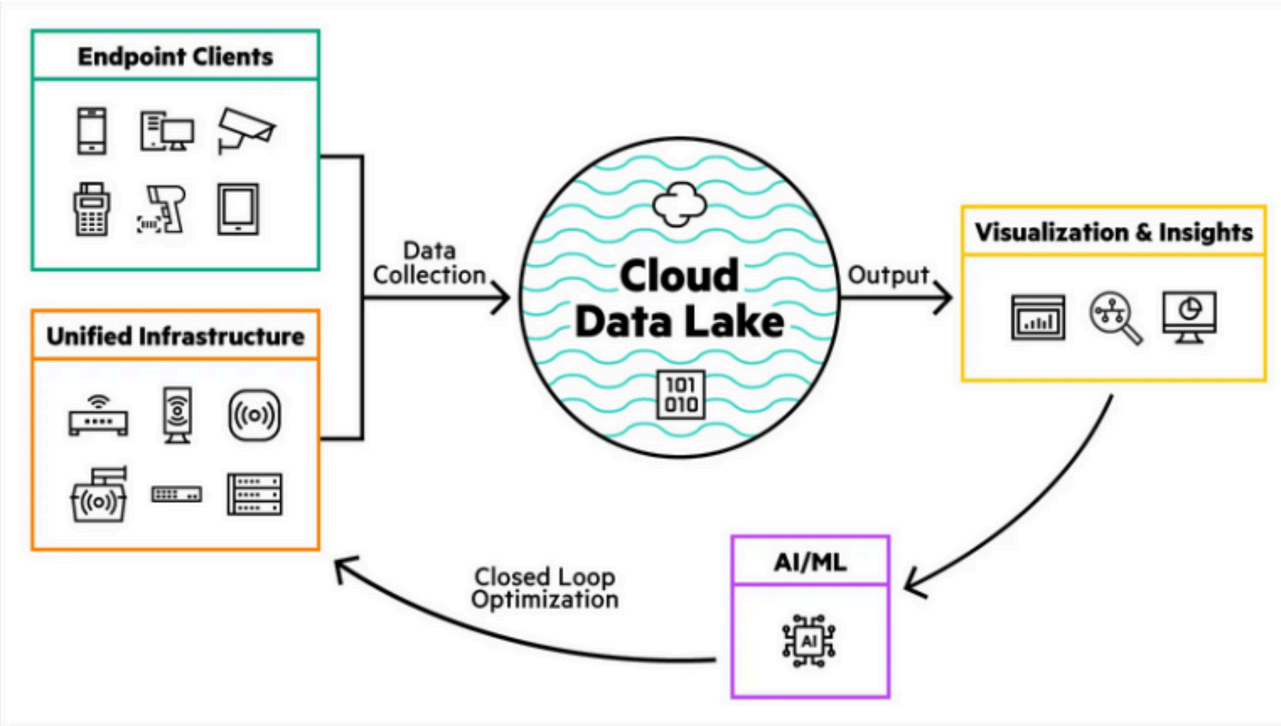
Spark dans une architecture Big Data

Spark :

- lit depuis le Data Lake
- transforme
- écrit les résultats

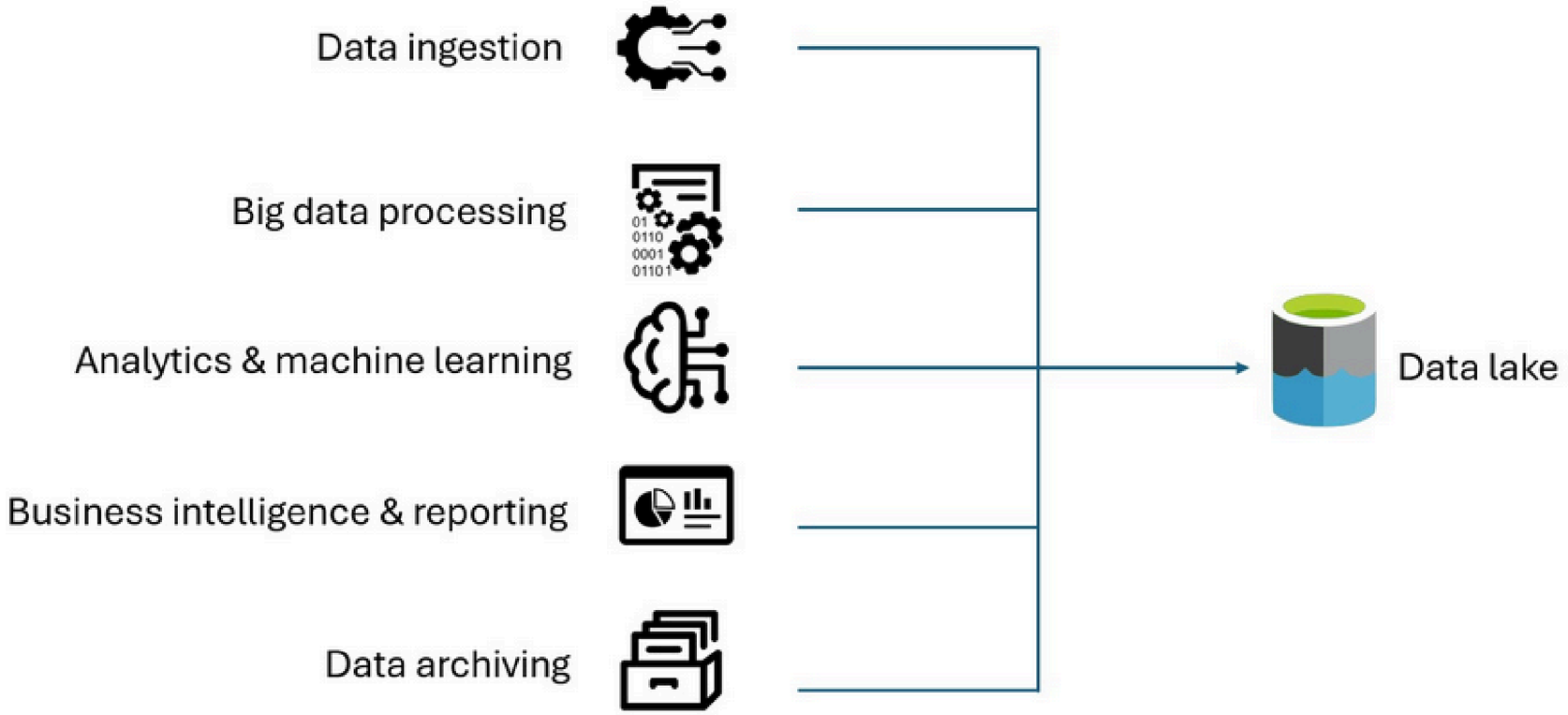
utiliser des lacs de données d'entreprise pour l'analyse moderne et l'intelligence d'affaires

« big data » et « **data lake** » n’ont de sens que pour la vision d’une organisation lorsqu’ils résolvent des problèmes commerciaux en permettant la démocratisation, la réutilisation, l’exploration et l’analyse des données. aux technologies de recherche, nous utilisons **des architectures de mégadonnées** pour améliorer la recherche et l’analyse, et nous aidons les organisations à faire des choses étonnantes en conséquence.



Qu’est-ce qu’un data lake ?

un lac de données est un grand dépôt de stockage qui contient une grande quantité de **données brutes** dans son format natif jusqu'à ce qu'il soit nécessaire. un "lac de données d'entreprise" (edl) est simplement un lac de données pour le stockage et le partage d'informations à l'échelle de l'entreprise.



Les cas d’usage courants de Data Lake sont les suivants :

- **Ingestion et déplacement des données** : collectez et consolidez des données à partir de services cloud, d’appareils IoT, de systèmes locaux et de sources de diffusion en continu dans un référentiel unique.
- **Traitement de big data** : gérez des données à grand volume et à grande vitesse à grande échelle à l’aide de frameworks de traitement distribué.
- **Analytique et Machine Learning** : prendre en charge l’analyse exploratoire, l’analytique avancée et l’apprentissage du modèle IA et le réglage précis sur les jeux de données volumineux et diversifiés.
- **Intelligence d'affaires et rapportage** : Permettez les tableaux de bord et les rapports en intégrant des sous-ensembles organisés de données de lac dans des entrepôts ou des outils décisionnels.
- **Archivage et conformité des données** : stockez les jeux de données historiques ou bruts pour la rétention à long terme, l’auditabilité et les besoins réglementaires.

Architecture d’une solution Data Lake moderne

Une solution de lac de données moderne comprend deux éléments principaux :

- Stockage : conçu pour la durabilité, la tolérance de panne, l’extensibilité infinie et l’ingestion à débit élevé de différents types de données.
- Traitement : alimenté par des moteurs tels qu’Apache Spark dans Azure Databricks, Microsoft Fabric, permettant des transformations, des analyses et du Machine Learning à grande échelle.

Avantages d’un Data Lake

Un lac de données offre au client les avantages suivants :

- Des bases de référence dynamiques pour les performances réseau de son site, sans avoir à définir manuellement les attentes de niveau de service (SLE).
- Des comparatifs qui indiquent où des sites similaires ont rencontré des problèmes sur la base de leurs propres données.
- Conseils d’optimisation fondés sur les données de performance du comportement d’un site client similaire.
- Réentraînement continu de l’IA/ML à mesure qu’émergent des technologies, des infrastructures et des terminaux de nouvelle génération.

Data Lake vs Data Warehouse

Un Data Lake stocke des données brutes, de tous types, sans définir à l'avance leur usage.

Un Data Warehouse stocke des données déjà structurées et préparées pour des usages analytiques précis.

Critère	Data Lake	Data Warehouse
Type de données	Brutes, structurées, semi-structurées, non structurées	Structurées uniquement
Schéma	À la lecture (schema-on-read)	À l'écriture (schema-on-write)
Usage	Exploration, Data Science, ML, Big Data	Reporting, BI, analyse métier
Flexibilité	Très élevée	Faible à moyenne
Coût	Généralement plus faible (stockage objet)	Plus élevé
Scalabilité	Très élevée	Limitée ou coûteuse
Données	Non filtrées, complètes	Filtrées, nettoyées, préparées
Cas d'usage	Analyse avancée, exploration, IA	Tableaux de bord, KPI, reporting

TP2: Mise en œuvre du pipeline Big Data et traitement des données