**Filière : Master Science et ingénierie de données**

# ETAT DE L'ART

## Sujet de TER :

**Architecture pour le traitement de données horodatées**


**Encadré par :**

BOUCELMA Omar

**Réalisé par :**

RHOUNAIM Haitam

SI JASSI Anas


Année universitaire 2021/2022

# 1. Concept de base du Big data

## 1.1. Introduction

Chaque jour, nous générons des trillions d'octets de données (Big Data). Ces données proviennent de partout : de capteurs utilisés pour collecter les informations climatiques, de messages sur les sites de médias sociaux, d'images numériques et de vidéos publiées en ligne, d'enregistrements transactionnels d'achats en ligne et de signaux GPS de téléphones mobiles, pour ne citer que quelques sources. Les Big Data se caractérisent par leur volumétrie (données massives); ils sont connus aussi par leur variété en termes de formats et de nouvelles structures, ainsi, qu'une exigence en termes de rapidité dans le traitement. Alors les problématiques du Big Data font partie de notre quotidien, et il faudrait des solutions plus avancées pour gérer cette masse de données dans un petit temps. Le calcul distribué concerne le traitement de grandes quantités de données. Ce traitement ne peut être réalisé avec les paradigmes classiques de traitement de données, il nécessite l'utilisation de plateformes distribuées. Dans la littérature, il existe plusieurs solutions, pour l'implémentation de ces paradigmes. Parmi ces solutions on trouve l'exemple Google qui a développé un modèle de programmation très fiable pour le traitement de Big Data : c'est le modèle MapReduce. Ce modèle est implémenté sur plusieurs plateformes comme la plateforme Hadoop. Malgré tous ces avantages, Hadoop souffre de problèmes de la latence qui est la cause principale de développement d'une nouvelle alternative pour améliorer les performances du traitement, c'est la plateforme Spark qui est plus puissante, plus souple et rapide que Hadoop MapReduce.

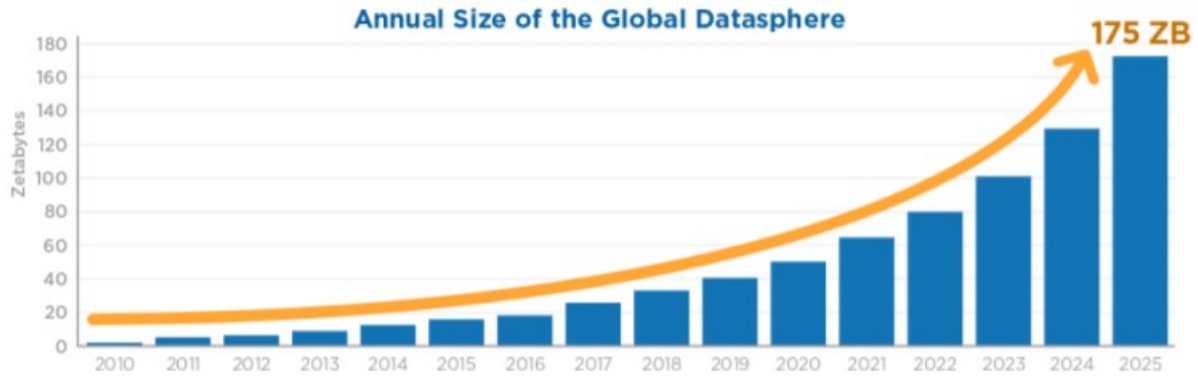



Figure 1 :augmentation annuelle des données par Zeta bytes

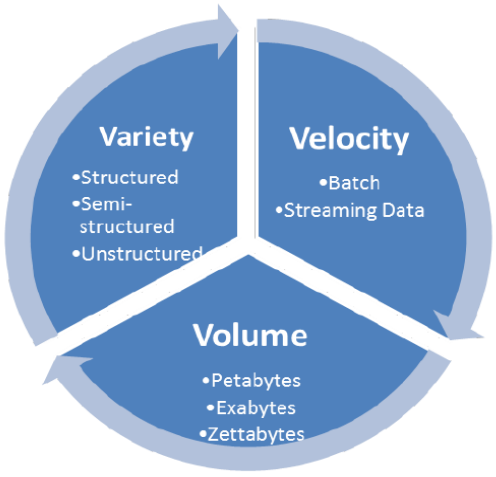
## 1.2. Définition

Le Big Data désigne un très grand volume de données souvent hétérogènes qui ont plusieurs formes et formats (texte, données de capteurs, son, vidéo, données sur le parcours, fichiers journaux, etc.), et comprenant des formats hétérogènes : données structurées, non structurées et semi-structurées. Le Big Data a une nature complexe qui nécessite des technologies puissantes et des algorithmes avancés pour son traitement et stockage. Ainsi, il ne peut être traité en utilisant des outils tels que les SGBD traditionnels. La plupart des scientifiques et experts des données définissent le Big Data avec le concept des 3V comme suit » :

Vélocité : Les données sont générées rapidement et doivent être traitées rapidement pour extraire des informations utiles et des informations pertinentes. Par exemple, Wallmart (une chaîne internationale de détaillants à prix réduits) génère plus de 2,5 petabyte(PB) de données toutes les heures à partir des transactions de ses clients. YouTube est un autre bon exemple qui illustre la vitesse rapide du Big Data.

Variété : Les données volumineuses sont générées à partir de diverses sources distribuées dans plusieurs formats (vidéos, documents, commentaires, journaux, par exemple). Les grands ensembles de données comprennent des données structurées et non structurées, publiques ou privées, locales ou distantes, partagées ou confidentielles, complètes ou incomplètes, etc.

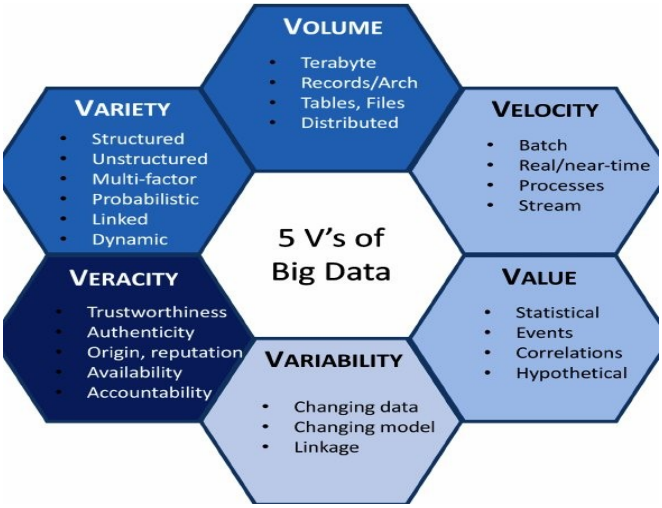
Volume : il représente la quantité de données générées, stockées et exploitées. Le volume des données stockées aujourd'hui est en pleine explosion il est presque de 800.000 Péta-octets, Twitter générer plus de 7 téraoctets chaque jour de données, Facebook générer plus de 10 téraoctets et le volume de données dans 2020 peut atteindre 40 zêta-octets




Par la suite, les trois dimensions initiales sont élargies par deux autres dimensions des données Big Data (on parle aussi des « 5 V du Big Data») :

Véracité : La véracité (ou validité) des données correspond à la fiabilité et l'exactitude des données, et la confiance que ces Big Data inspirent aux décideurs. Si les utilisateurs de ces données doutent de leur qualité ou de leur pertinence, il devient difficile d'y investir davantage.

Valeur : Ce dernier V joue un rôle primordial dans les Big Data, la démarche Big Data n'a de sens que pour atteindre des objectifs stratégiques de création de va- leur pour les clients et pour les entreprises dans tous les domaines.




## 1.3. Les plateformes de traitement des Big Data

### 1.3.1. La plateforme hadoop pour le calcul distribué de Big Data

Hadoop est la plateforme logicielle open source de la fondation Apache permettant de répondre aux besoins du big data, à savoir la volumétrie, la véracité et vélocité des données, ainsi qu'au paradigme de traitement MapReduce pour avoir une meilleure vitesse d'exécution d'algorithme. Elle permet de stocker et d'effectuer des traitements sophistiqués sur des données de différents types et différentes provenances à grande échelle et à moindre coût.

Grâce aux différentes contributions, Hadoop est aujourd'hui un écosystème complexe. En effet si le cœur (Kernel) de Hadoop est basé sur le système de fichier HDFS (Hadoop Distributed File System) et le paradigme MapReduce, pour les traitements que nous allons voir en détail par la suite, on a également une multitude d'outils autour de ce noyau. On peut citer entre autres :

- HBase, basée sur BigTable le système de stockage de Google, une base de données NoSQL de type colonne ;

- ZooKeeper, un système de gestion de service distribué ;
- des outils d'interrogation et de traitement des données, notamment Hive et Pig ;
- des outils de planification et coordination des traitements MapReduce Oozie ;
- des outils de transfert de données entre hadoop et les SGBDR classiques, Sqoop

## ➢ Le system de stockage de fichier Hdfs

**HDFS (Hadoop Distributed File System)** est le système de stockage de fichiers d'Hadoop, basé sur le système de fichiers développé par Google, GDF (Google File System) et dont la théorie a été publiée dans des papiers de recherche de Google.

❖ HDFS a pour objectifs d'être :

- Tolérant aux pannes d'une manière native (Fault tolerant) : c'est-à-dire qu'il peut résister à l'erreur, en partant du principe que la défaillance est la règle générale plutôt que l'exception, en premier lieu la panne matérielle et notamment le disque dur ;
- Scalable : il peut supporter une montée en charge horizontale, avec très peu de contraintes et de pertes de performance ;
- Modèle d'accès immuable : écriture unique (il n'est plus modifiable) et multiples lectures pour les fichiers, ce qui favorise la cohérence des données.
- Déplacer les calculs vers les données : parce que le déplacement des fichiers volumineux peut conduire à un goulot d'étranglement réseau et impacter les performances globales du système.
- Simple à mettre en place en assurant la portabilité sur différentes plateformes.

❖ Il propose de nombreuses fonctionnalités :

- gestion des fichiers par blocs : chaque fichier sera découpé en blocs de 64 Mo ;
- réplication et distribution : tous les blocs seront à la fois répliqués et distribués pour assurer un stockage fiable des fichiers de très grand volume. Répliqués sur différents endroits sur l'HDFS, par défaut trois endroits, le premier est le **serveur pour positionner le fichier, le second est un serveur voisin sur le même** rack et le troisième sur autre rack ou un autre Datacenter, pour permettre d'avoir la meilleure distribution possible ;
- gestion des droits : les permissions en lecture et écriture et l'accès au répertoire à la fois pour le propriétaire et pour le groupe. L'authentification peut être soit absente, soit gérée par login et mot de passe ou bien par Kerberos ;
- accès aux données en continu (Streaming) : il a un accès continu aux jeux de données, avec un haut débit d'accès au détriment de la faible latence, parce qu'il est adapté au traitement par lots ;

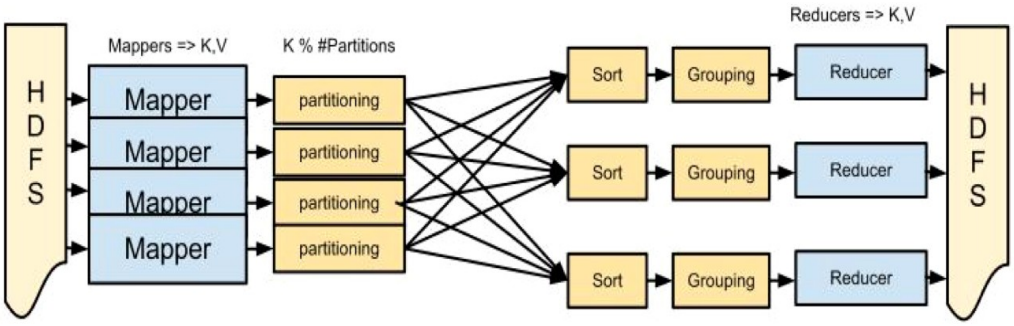
- stockage des grands jeux de données, les fichiers stockés dans HDFS sont typiquement de plusieurs gigaoctets ou téraoctets, avec une bande passante de haut débit pour les différents nœuds du cluster.

➢ **MapReduce**

MapReduce est un paradigme de programmation qui permet de distribuer des traitements parallèles sur des volumétries de données dépassant typiquement 1To, dans un cluster composé de centaines, voire de milliers de nœuds - machines - peu coûteux, avec une architecture de type maître/esclave, grâce à la séparation des données et des traitements.

Le traitement parallèle des données avec MapReduce est simple et s'effectue avec deux opérations qu'on appelle map et reduce. Les données d'entrée doivent être structurées sous forme d'une liste de clé/valeur (une Map) qui peut être très volumineuse (téraoctets voire pétaoctets). Ce format unique facilite les entrées\ sorties.

Cette simplicité est composée d'une couche logicielle ou d'un Framework. Le Framework open source le plus connu est Apache Hadoop, (que nous allons détailler dans la section suivante) qu'on peut découper en quatre phases, dont deux à la charge du développeur (map/reduce) et le reste à la charge du Framework :

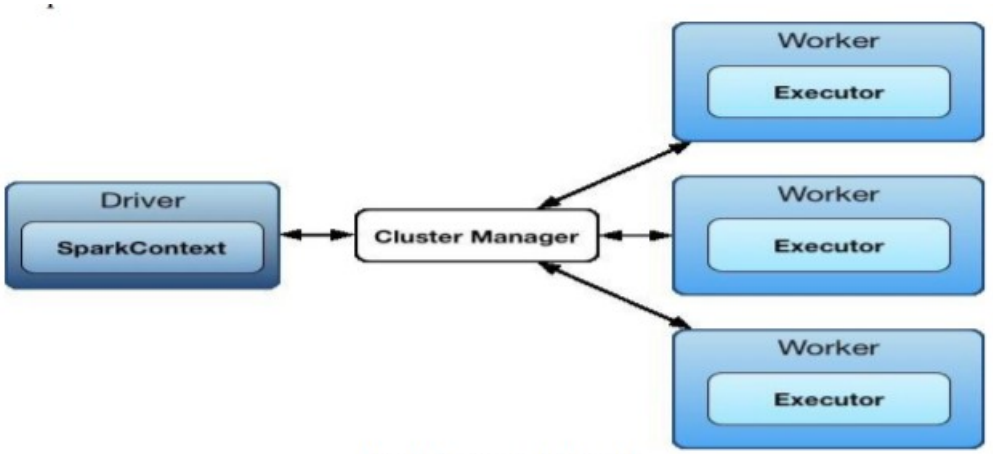



The MapReduce Pipeline

A mapper receives (Key, Value) & outputs (Key, Value)
A reducer receives (Key, Iterable[Value]) and outputs (Key, Value)
Partitioning / Sorting / Grouping provides the Iterable[Value] & Scaling

## 1.3.2. La plateforme Spark pour le calcul distribué de Big Data

Apache Spark est un Framework open source de traitement, il est construit autour de la vitesse, la facilité d'utilisation et capacité de gérer des grands ensembles de don- nées qui sont de nature diverse (données de texte, données de graphes, etc.), Spark étend le modèle MapReduce pour soutenir efficacement plusieurs types de calculs, y compris le traitement itératif, les requêtes interactives et le traitement de flux.



Architecture de Spark

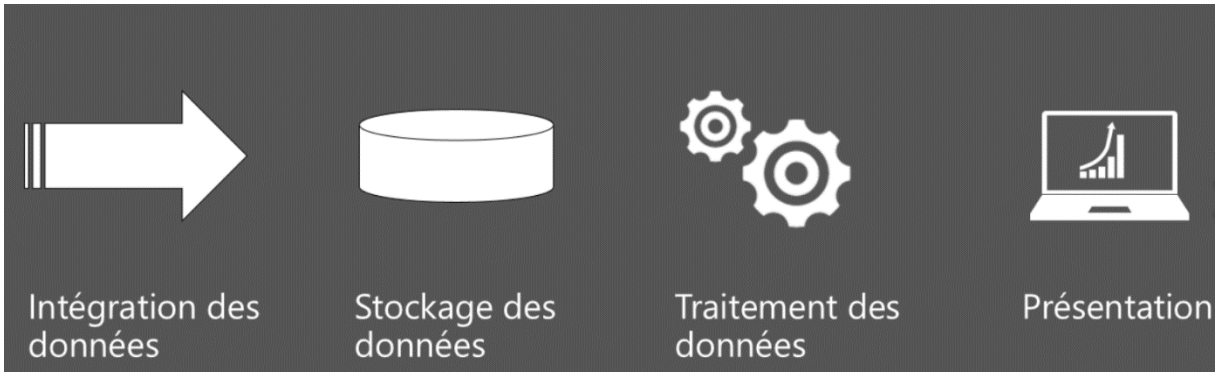
Apache Spark sert à faire des traitements, avec un modèle (transform/action) plus flexible que le paradigme Map/Reduce, en utilisant un modèle plus général des données nommé RDD (Resilient Distributed Dataset), sachant qu'il est « Storage agnostic », c'est-à-dire qu'il fait abstraction du mode de stockage, contrairement à Map/Reduce qui utilise HDFS.

Les traitements sont réalisés en mémoire et non dans HDFS, ce qui rend leur exécution plus rapide.

## 1.4. Batch processing / Stream processing

### 1.4.1. Batch processing

- Traitement de blocs de données déjà stockes sur une période donnée.
- Ces données contiennent des millions d'enregistrement pour chaque jour pouvant être stockes sous forme de fichiers textes (CSV) ou d'enregistrement stockes dans HDFS, SGBD SQL, NoSQL, etc.
- Exemple de Framework :
  - MapReduce
  - spark



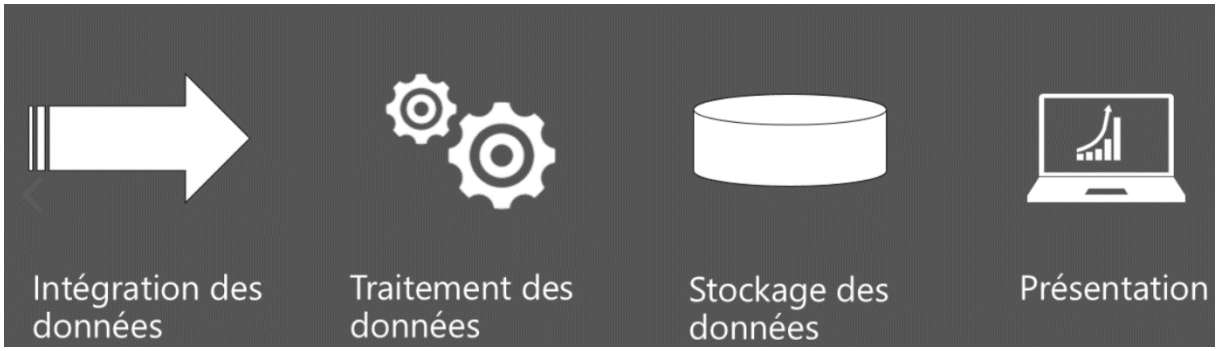
### 1.4.2. Stream Processing (Traitement de flux) :

- Contrairement au traitement par lots où les données sont liées avec un début et une fin dans un traitement qui se termine après le traitement de données finies,

- Le Stream Processing est destiné au traitement de flux de données sans fin arrivant en temps réel de façon continue pendant des jours, des mois, des années et à jamais.

- Le traitement de flux nous permet de traiter les données en temps réel

- Le traitement de flux permet d'introduire des données dans des outils d'analyse dès qu'elles sont générées et d'obtenir des résultats d'analyse instantanés.




Deux approches pour mettre en place un Framework Streaming:

- Native Streaming (Real Time Processing)

  - Chaque enregistrement entrant est traité dès son arrivée, sans attendre les autres.

  - Exemples: Storm, Flink, Kafka Streams, Samza.

- Micro Batch Processing (Micro Batching)

  - Les enregistrements entrants toutes les quelques secondes sont mis en lots, puis traités en un seul mini-lot avec un délai de quelques secondes.

  - Exemples: Spark Streaming, Storm-Trident.

## 1.5. Conclusion :

Le Big Data permet d'analyser et d'évaluer tout type de production humaine et les feedbacks clients. Le Big Data peut être utilisé pour améliorer la prise de décision, pour l'ajuster au mieux à la demande du marché.