SUPPORT DE COURS

Big Data et Architectures Associées

Niveau : 2ème année Licence Fondamentale en Big Data

Elaboré par *: Dr. Mohamed HMIDEN*

Année Universitaire : 2020-2021

Big Data et Architectures Associées

2ème année Big Data

Mohamed HMIDEN
Mohamed.Hmiden@gmail.com

1

Objectifs

- Les objectifs de ce cours sont :
  - Introduire les concepts liés aux données massives et leur émergence
  - étudier une architecture distribuée dédiée à stocker à gérer les données massives et les outils nécessaires à la manipulation de données massives
  - Étudier le stockage de données massives dans des bases de données NoSQL et quels sont les points de différence par rapport aux bases de données relationnelles.

Références

Livres

Titre : Les bases de données NoSQL et le BigData Comprendre et mettre en oeuvre

Auteur : Rudi Bruchez

Editeur : Eyrolles

Edition : 2015

Titre : *Hadoop. The definitive Guide. 3rd edition.* . 2013.

Auteur : Tom White

Editeur : O'Reilly

Année : 2013

Cours

- Big data

  Ensignants : Stéphane Vialle & Patrick Mercier

  URL : Cours de Big Data - 2A Supelec.

- Cours : Initiation aux Big Data :

  Enseignant : Cédric du Mouza et Nicolas Travers

  Cours : Big data et ses technologies

  Enseignant : Philippe Laflamme

  URL : BigData_Technologies_PL.pdf (etsmtl.ca)

Plan

- Chapitre 1 : Introduction aux Big Data
- Chapitre 2 : Hadoop et MapReduce
- Chapitre 3 : Base de données NoSql

Chapitre 1: Introduction au Big data

ISAMM
2ème BIG DATA 20-21

Plan

- Définitions des concepts liés aux données
- Emrgence de big data
- Big data
  - Définition
  - les 5V de Big Data
- Domaines d'application
- Technologies de big data

Gestion de données

■ Définitions

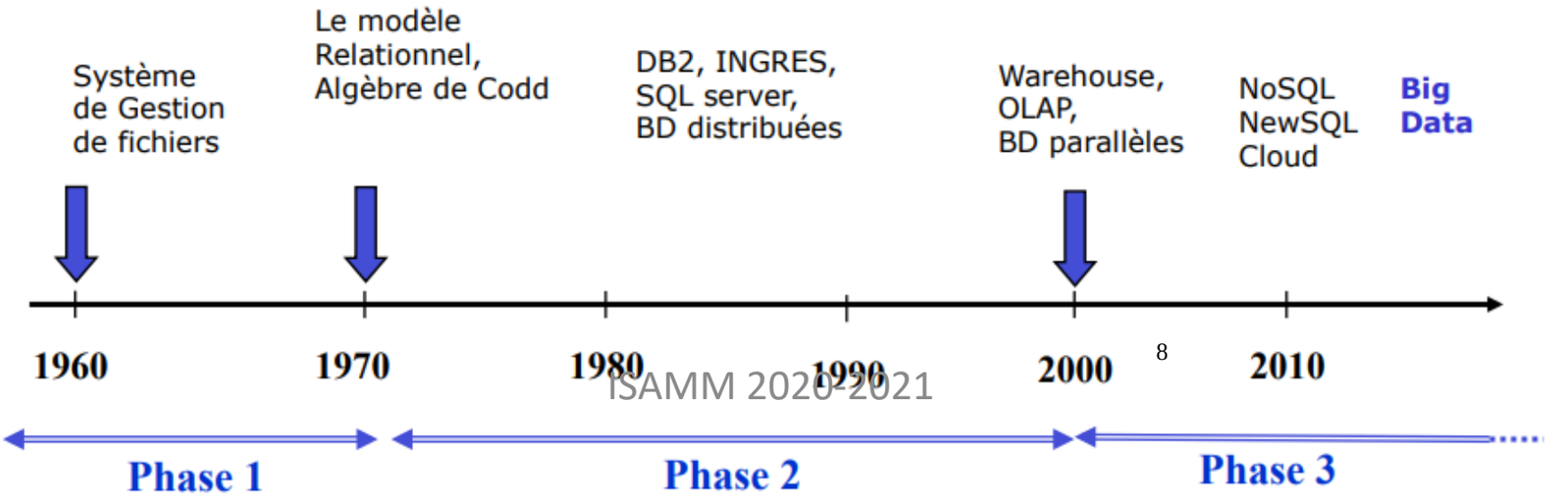
- ❖ **Donnée** : C'est l'enregistrement d'une observation, d'un objet ou d'un fait destiné à être interprété, traité par l'homme.
  - ➢ Généralement, la donnée est objective.

- ❖ **Information :** c'est le signifiant attaché à la donnée ou à un ensemble de données. L'information est généralement subjective, définie selon un contexte.
- ❖ **Connaissance :** C'est une nouvelle information, apprise par l'association d'informations de base, de règles, de raisonnement, d'expérience ou d'expertise

❑ Historique de la gestion de données

La gestion des données a passé par les phases suivantes :

- ❖ Phase 1 : période préhistorique 96-69
- ❖ Phase 2 : période phare 70- 2000
- ❖ Phase 3 : période nouvelle 2000-maintenant



❑ Phase 1 :

- Avant 1960, il n'y a pas des bases de données, il n'y a que des applications manipulant des fichiers gérés par de système de gestion de fichiers.
- À partir du milieu des années 1960, les BD hiérarchique et celles réseau sont apparues.
- Ces systèmes sont caractérisés par :
  - Plusieurs applications partagent des données
  - Séparation des données et des traitements sur les données
  - Données gérées par un serveur central
  - Minimisation de la redondance et de l'inconsistance
  - Amélioration du contrôle des données
  - Accès par langage standardisé (COBOL)

❑ Phase Phare

- C'est la phase du modèle du modèle relationnel des bases de données. Ce modèle est inventé par Edgar **Franck Codd** en 1970. Il est fondé sur les principes suivants :
  - Algèbre relationnelle, logique de prédicat de 1er ordre
  - Indépendance des données
  - Langages : SQL, QUEL, QBE
  - Dépendances fonctionnelles, formes normales

- Les bases de données transactionnelles OLTP sont conçues
  OLTP : On-Line Transanctional Processing.

Données et Gestion de données

- Elle Permettent Traiter des données de manière transactionnelle et fiable.
- Ces BD se basent sur le modèle relationnel permettant la gestion des transactions.
  - ✓ Une transaction est définie comme étant une unité logique du travail sur la base de données.

- Ce modèle respecte les propriétés ACID suivantes
  - ✓ **Atomicité** : une transaction forme une unité indivisible,
  - ✓ **Cohérence** : Une transaction transforme la base de données d'un état cohérent à un autre état cohérent
  - ✓ **Isolation** : les transactions s'exécutent de manière indépendante les unes des autres.
  - ✓ **Durabilité** : les effets d'une transaction complètement achevée sont inscrits de manière durable dans la base de données et ne peuvent subir de perte à la suite d'une défaillance subséquente.

❖ Données gérées par des systèmes de gestion de bases de données (SGBD). Un SGBD permet de stocker, de manipuler et à partager des Données.

Exemples de SGBD

- Oracle
- SQL SERVER
- DB2
- INFORMIX

❑ Phase 3: Bases de données décisionnelles

Les bases de données décisionnelles se basent sur :

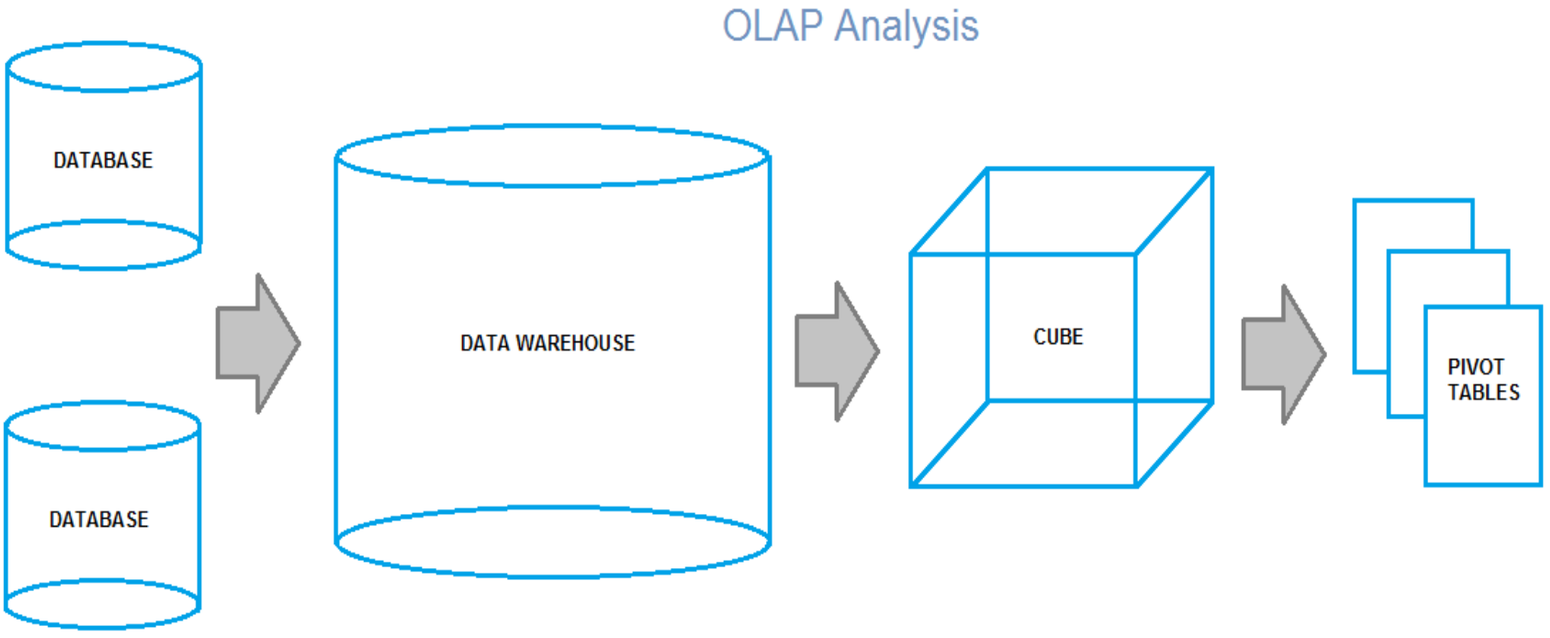
❖ *les entrepôts de données, Data Warehouse*, sont des bases de données dédiées au stockage de l'ensemble des données Utilisées.

- Utilisées dans le cadre de la prise de décision et de l'analyse décisionnelle.
- Elles sont alimentées en données depuis les bases transactionnelle de production avec des ETLs.
- ETL : Extract Transform Load : permet d'extraire, de transformer et de charger les données dans le data warehouse.

❖ Les analystes décideurs et décideurs accèdent aux données collectées pour :

- ❖ étudier des cas précis de réflexion.
- ❖ Construire des modèles d'étude et de prospective pour limiter la part d'incertitude lors du processus de prise de décision.

- *bases de données multidimensionnelles :* permet de stocker des données , OLAP d'analyses, sous forme des cubes.
  - L'OLAP (OnLine Analysis Processing) est une technologie permettant d'effectuer des analyses de données multidimensionnelles

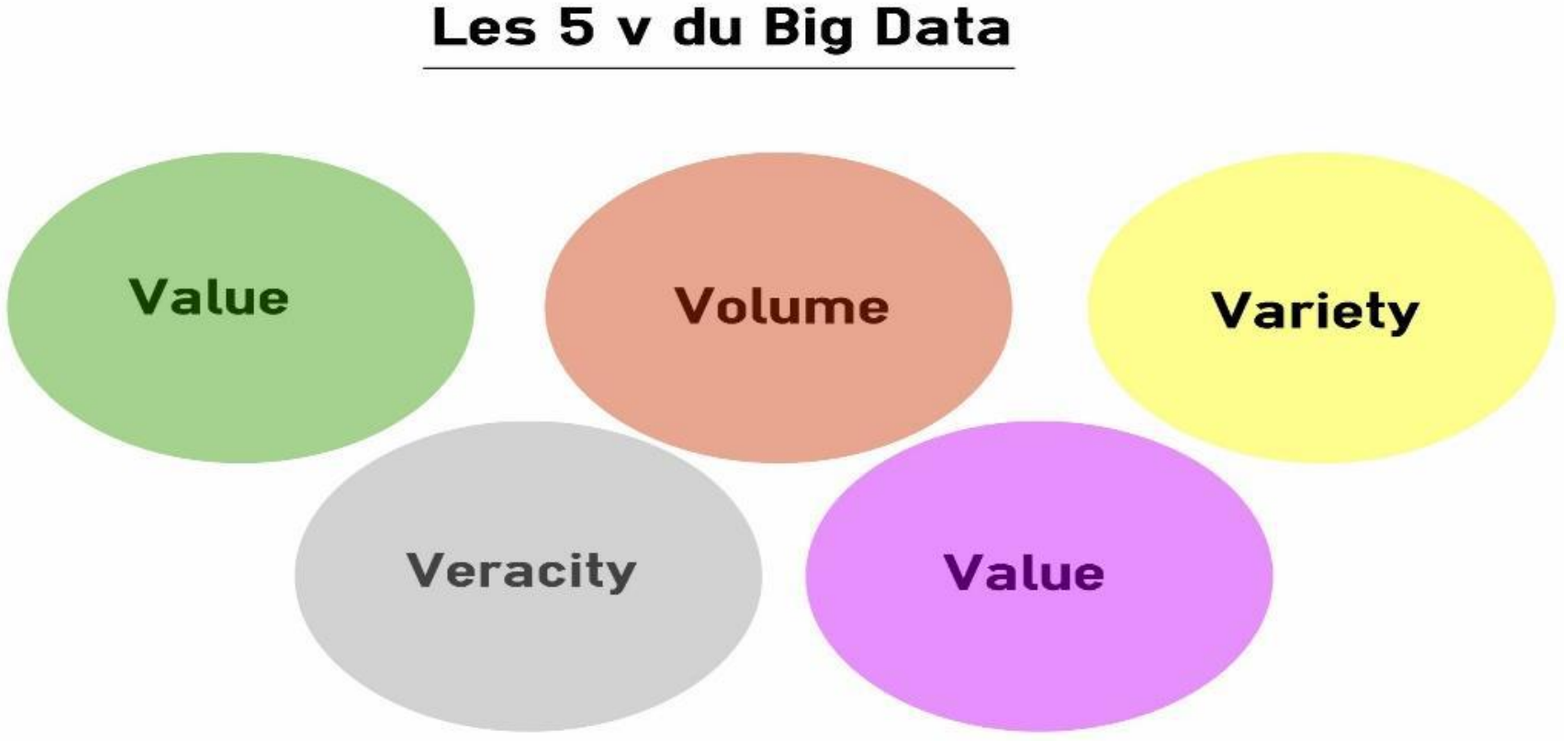


II. BIG DATA

- Le **big data**, « grosses données » ou **méga-données .**
  « *des ensembles de données devenus si* ***volumineux*** *qu'ils* ***dépassent*** *l'intuition et les capacités humaines* ***d'analyse*** *et même celles des outils informatiques classiques de gestion de base de données ou de l'information »*

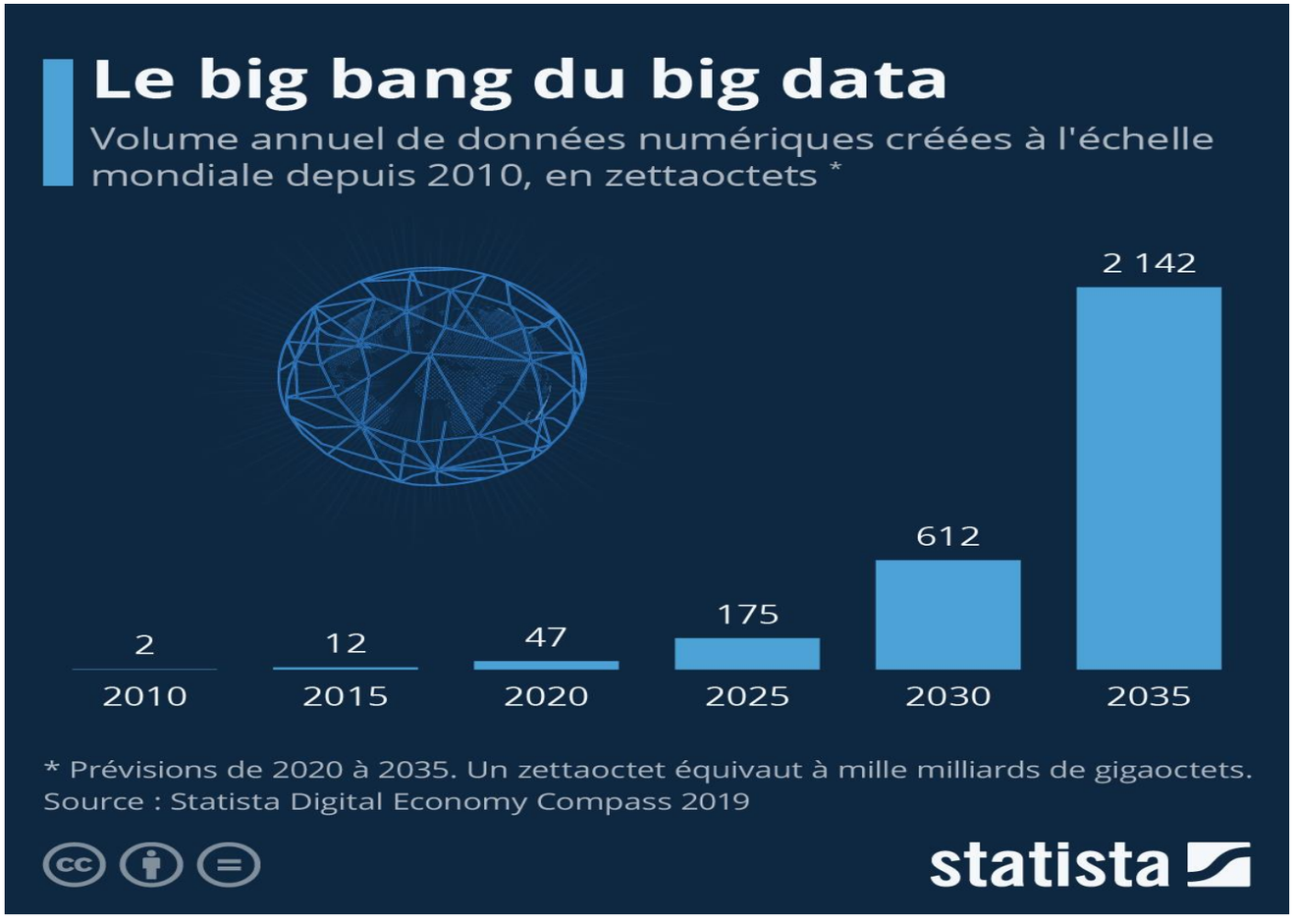
- **Caractéristiques de Big data**

Le big data est caractérisé par les 5 V suivants :



Big Data | Caractéristiques

- Volume : la masse d'informations produite et qui doit être analysée et traitée.
  - Il est en accroissement exponentiel
  - 90 % des données ont été engendrées durant les années générées par des applications web (internet)
  - Le graphique suivant présente l'évolution du volume de données



12

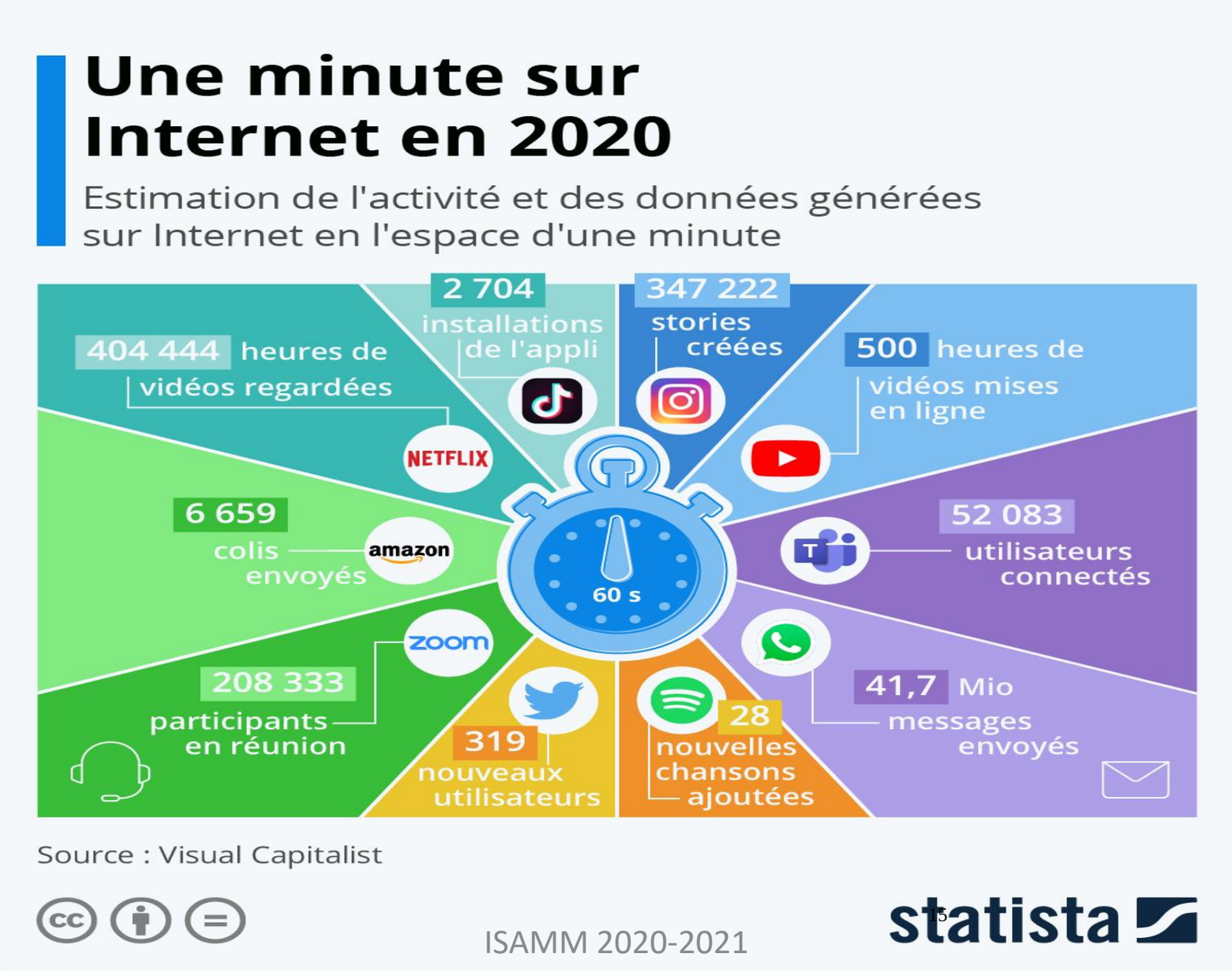
Big Data | Caractéristiques

- **Vélocité : c'est la vitesse** du déploiement des nouvelles données.

  Par exemple : si on diffuse des messages sur les réseaux sociaux, ils peuvent devenir « viraux » et se répandre en un rien de temps.

  → Analyser les données au décours de leur acquisition avant même de les stocker.

- Une minute sur internet génère des données énormes



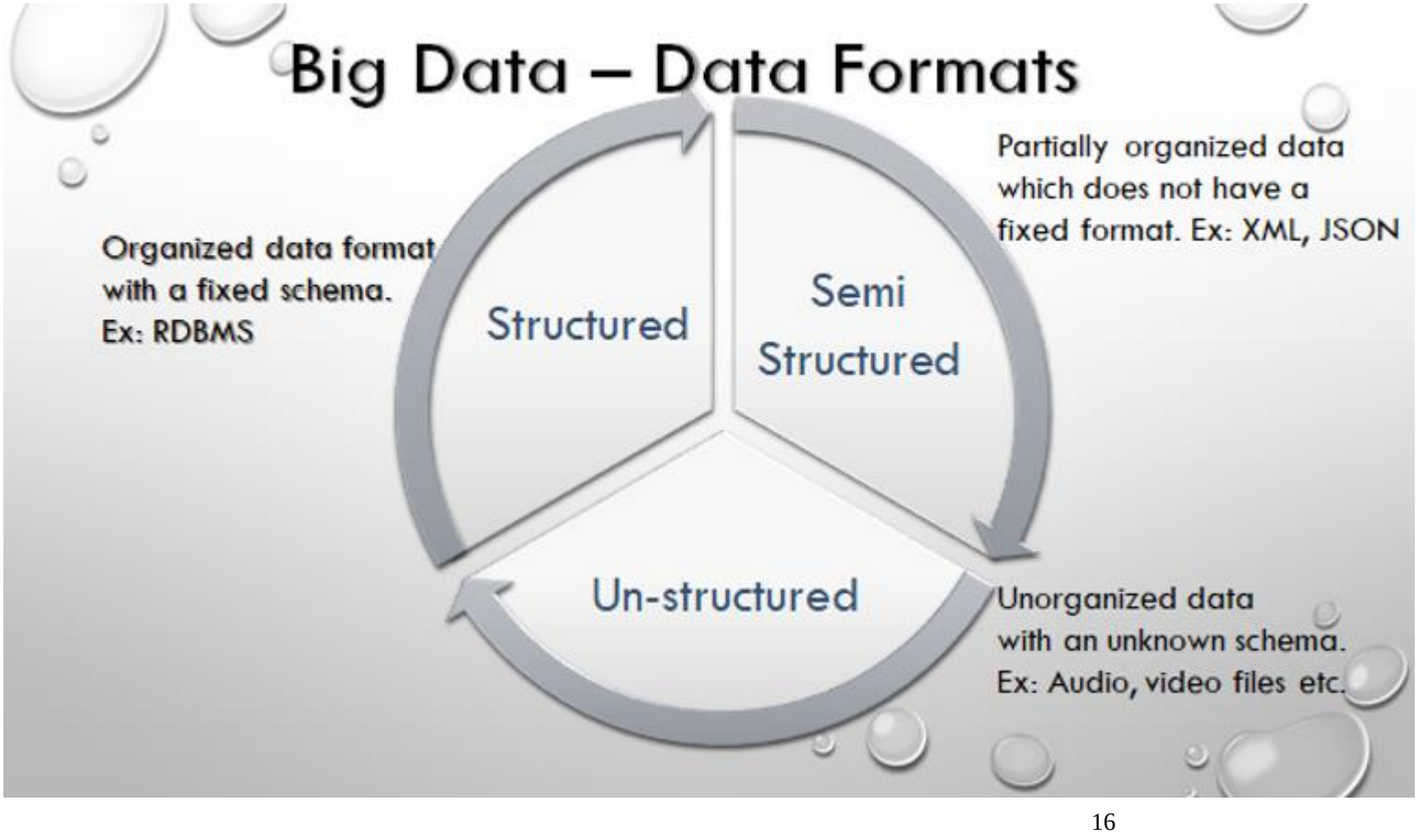
Big Data | variété

❑ Variété de Données big data :

❖ Données structurées : 20% des données sont structurées puis stockées dans des tables de bases de données relationnelles similaire à celles utilisées en gestion comptabilisée.

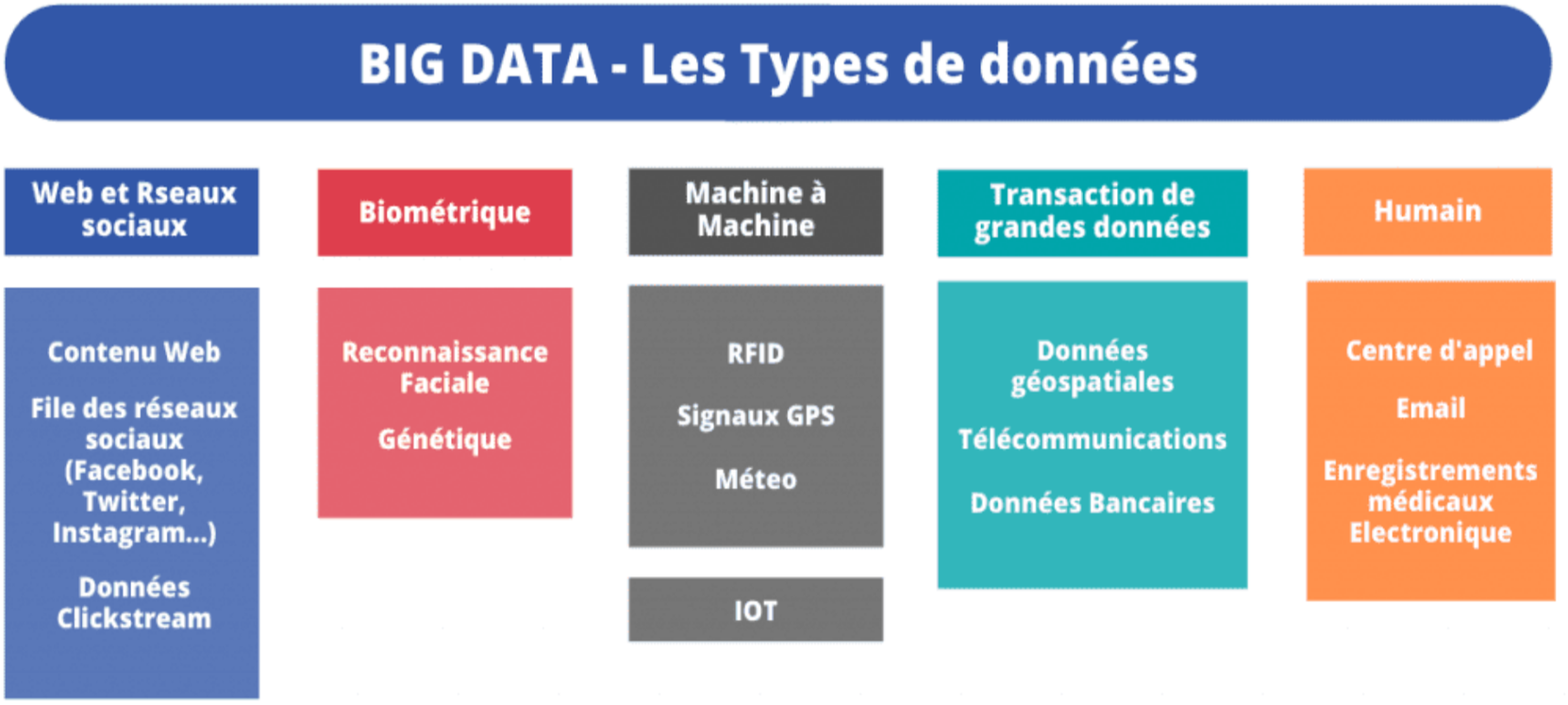
❖ Données non structurées: 80% des données sur internet sont non structurées ou non-structurées.

- images,
- des vidéos,
- des textes,
- des voix



Big data | Sources de données

- Les types de données dépendent de leurs sources qui sont :
  - Web et réseaux socieux
  - Biométrique et génétique
  - Machine
  - Transaction de données
  - Humain.

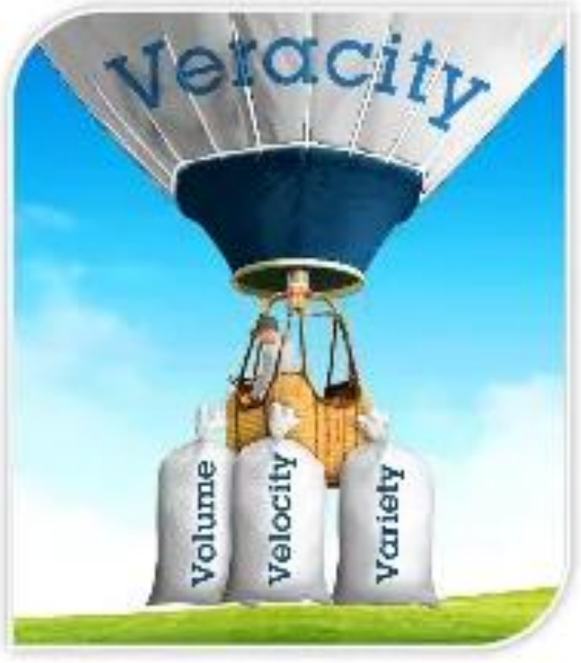


Big data | Véracité

- La véracité
  - Elle concerne la fiabilité et la crédibilité des informations collectées.
    - Big Data permet de collecter un nombre indéfini et plusieurs formes de données, il est difficile de justifier l'authenticité des contenus.

      - Donnes de réseaux sociaux : facebook, twitter sont écrites avec les abréviations, le langage familier, les hashtag, les coquilles etc.
      - Ces données ne sont pas fiables

  - La véracité de données dépendent des caractéristiques volume , variété et vélocité:



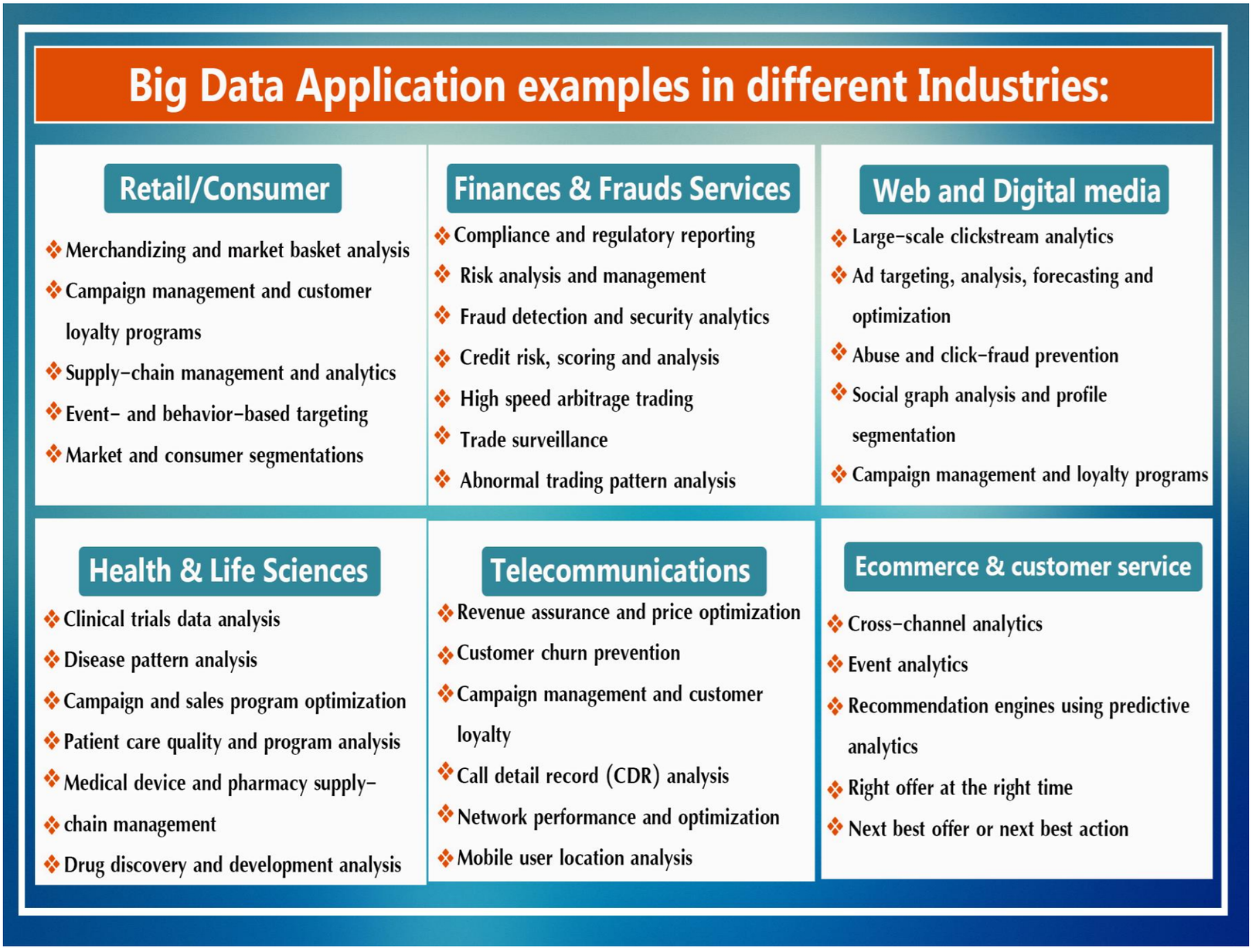
Big data | Valeur

- Valeur : elle correspond au profit qu'on puisse tirer de l'usage du Big Data.

  - Ce sont généralement les entreprises qui commencent à obtenir des avantages incroyables de leurs Big Data.
    Selon les experts gestionnaires et les économistes , les entreprises qui ne s'intéressent pas sérieusement au Big Data risquent d'être pénalisées et écartées.

Domaines d'application

- Domaines d'application de big data



❑ Big data et decision d'entreprise

❖ Aide à la décision pour le pilotage d'activités :

- ➢ il aide les managers-décideurs en charge d'une unité la BI. Il les fournis les outils nécessaires à leur travail :

  - ✓ Des tableaux de bords
  - ✓ Des analyses multi-dimensionnelles

❖ Analyse de données :

- ➢ Élaborations des états statistiques et de prévisions
- ➢ ils étudient des hypothèses de réflexion et bâtissent des modèles pour pousser plus avant la connaissance :clients, produits, processus

❖ Le Big Data permet de bâtir des modèles bien plus complets

❖ Il améliore la connaissance des thèmes habituellement prospectés et ouvre de nouveaux champs d'étude.

Technologies de big data

❑ Technologies du big data

❖ **Map Reduce :**

- ➢ **C'est** une méthode de traitement massivement parallèle développée par Google.
- ➢ **Elle stocke les données** avec un système de gestion de fichiers spécifiques (Google File System)
- ➢ Elle est tolérante aux pannes

❖ **Hadoop**

- ➢ un framework développé par Apache Software Fundation
- ➢ permet de mieux généraliser l'usage du stockage et traitement massivement parallèle de Map Reduce et de Google File System.
- ➢ Hadoop possède ses limites.
- ➢ c'est une solution de big data très largement utilisée pour effectuer des analyses sur de très grands nombres de données.

20

Bases No SQL

- ➢ Les bases NoSQL autorisent la redondance pour mieux servir les besoins en matière de flexibilité.
- ➢ Elles sont tolérantes aux pannes et d'évolutivité.
- ➢ Permet de stocker des données sans schéma.

Chapitre 2
Hadoop et MapReduce

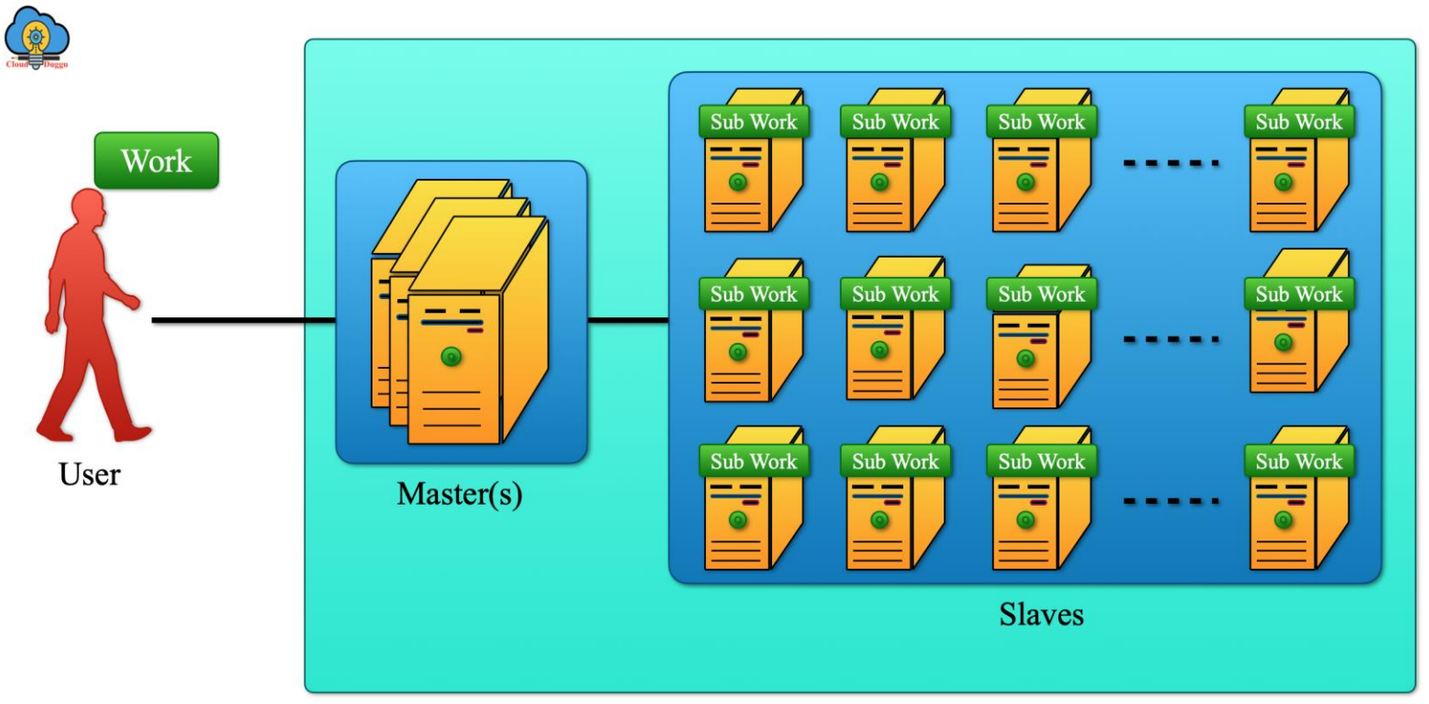
Hadoop

- **Hadoop**
  - Hadoop est un framework libre et open source écrit en Java.
  - Il est destiné à faciliter la création d'applications distribuées et échelonnables.
  - Il permet aux applications de travailler avec des milliers de nœuds et des pétaoctets de données.
  - Hadoop s'installe sur un cluster

- Hadoop repose sur les modules suivants :
  - Hadoop distributed file system (HDFS) : utilisé pour le stockage de données
  - Yet Another Resource Negociator (YARN) : c'est un gestionnaire des réssources. Il permet de planifier les tâches et surveiller les nœuds du cluster.
  - Module MapReduce : aide les programmes à effectuer des calculs parallèles.
  - hadoop common utilise des bibliothèque standards Java.

  - Hadopp s'installe sur un cluser composé d'un mastère et plusieurs machine slaves.

Hadoop



Hadoop Cluster

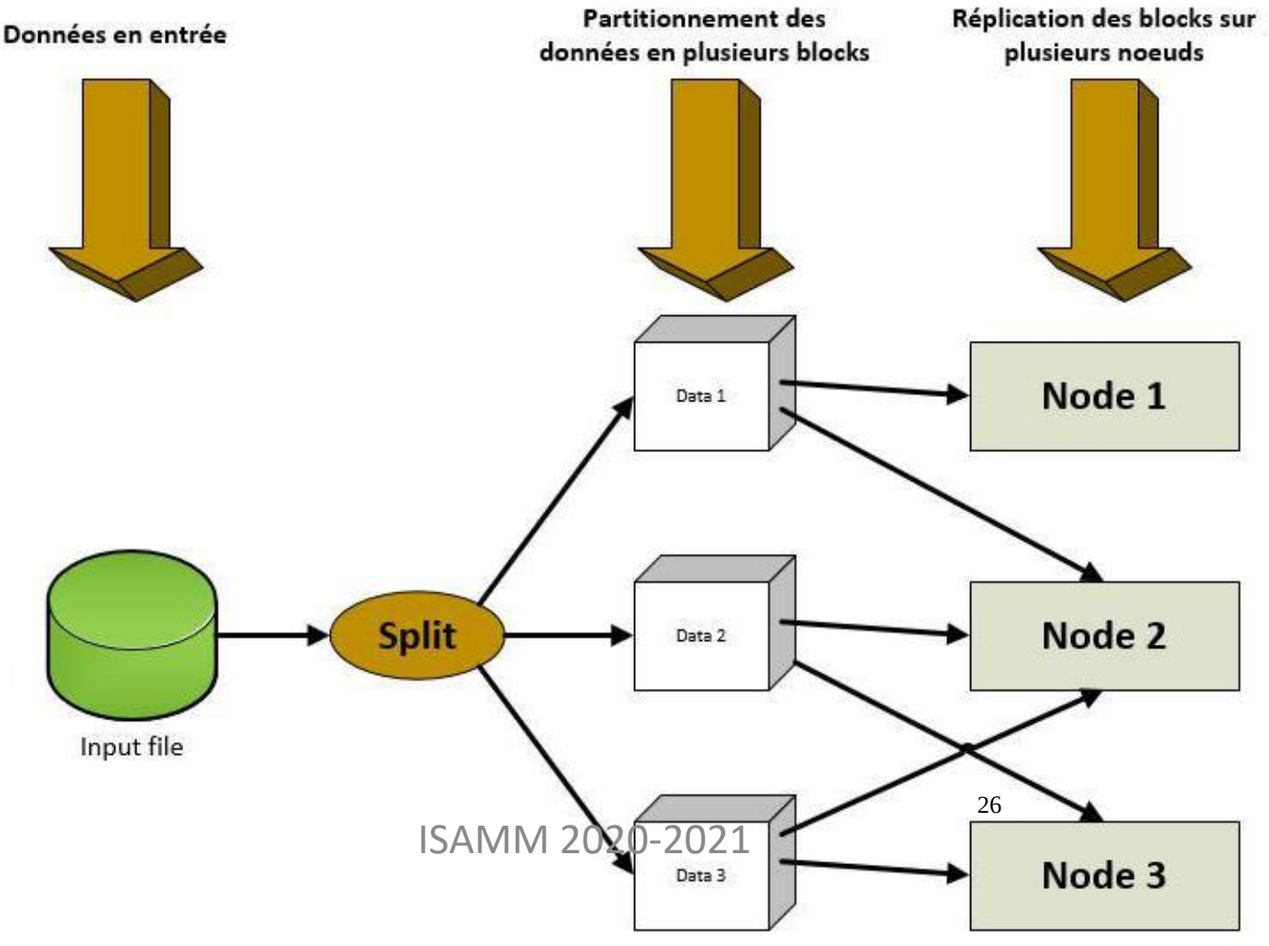
Deux types de machines dans un cluster hadoop :

- **Master : Il est dit Name Node. C'**est la pièce centrale dans du système hadoop.
  - il maintient une arborescence de tous les fichiers du système et gère l'espace de nommage.
  - Il centralise la localisation de données répartis sur le système.
  - Il s'occupe de reconstituer un fichier à partir des données réparties dans les différents slaves.
  - Il n'y a qu'un « master » dans un cluster hadoop .

- **Slaves : il est dit data node :** Il permet le stockage de données.
  - Il communique périodiquement au « Name Node » une liste des blocs qu'il gère.
  - Un système hadoop contient plusieurs noeuds de données ainsi que des réplications d'entre eux.

❑ Hadoop distributed file system HDFS :

- ❑ Permet de stocker toutes les données
- ❑ Structures, semi-structures et non structures.
- ❑ Le HDFS partitionne les données en entrée en plusieurs blocs de données (bloc physique)
- ❑ Les blocs de données sont distribuées sur les différentes machines slaves.
- ❑ Un bloc de données est dupliqué sur plusieurs machine pour garantir sa disponibilité même en cas de panne. Par défaut chaque bloc est dupliqué sur trois machines slaves.
- ❑ Les différents noeuds de données peuvent communiquer entre eux pour rééquilibrer les données.



Hadoop | HDFS

Hadoop repose sur les cinq composants suivants :

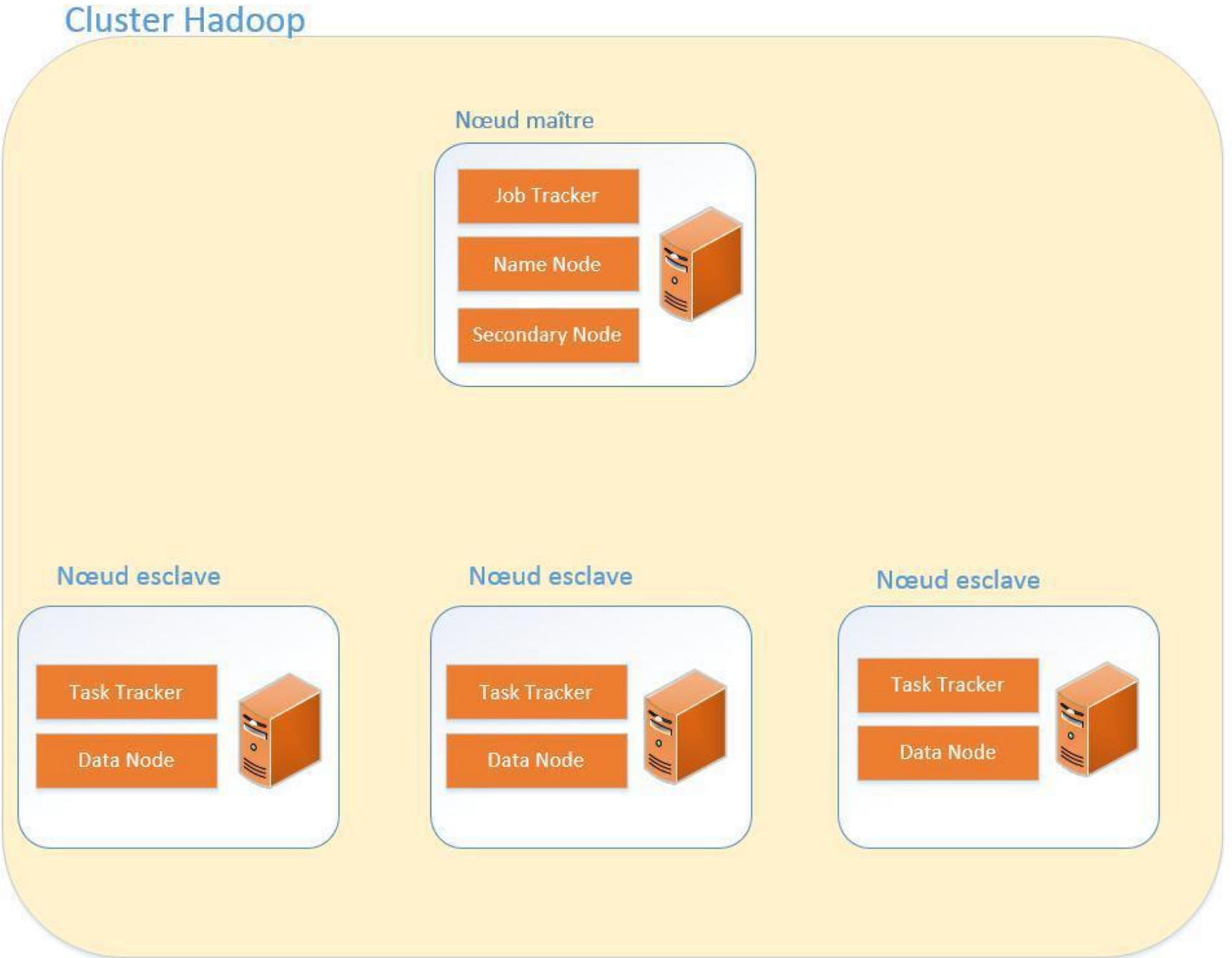
- **Name Node. C'**est la pièce centrale dans le HDFS,
  - il maintient une arborescence de tous les fichiers du système et gère l'espace de nommage.
  - Il centralise la localisation des blocs de données répartis sur le système. Sans le « Name Node » , les données peuvent être considérées comme perdues car il s'occupe de reconstituer un fichier à partir des différents blocs répartis dans les différents « Data Node ».
  - Il n'y a qu'un seul « Name Node » par cluster hadoop.

- **Job Tracker :**
  - **I**l s'occupe de la coordination des tâches sur les différents clusters.
  - Il attribue les fonctions MapReduce aux différents « Task Trackers ».
  - Le « Job Tracker » est un « Daemon » cohabitant avec le « Name Node » et ne possède donc qu'une instance par cluster.

- **Data node :** Il permet le stockage des blocs de données.
  - Il communique périodiquement au « Name Node » une liste des blocs qu'il gère.
  - Un HDFS contient plusieurs noeuds de données ainsi que des réplications d'entre eux. Ce sont les noeuds esclaves.

- **Task tracker : Il permet de :**
  - Exécuter les ordres de MapReduce.
  - Lire les blocs de données en accédant aux différents « Data Nodes ».
  - notifier de façon périodique au « Job Tracker » le niveau de progression des tâches qu'il exécute, ou alors d'éventuelles erreurs pour que celui-ci puisse reprogrammer et assigner une nouvelle tâche.

- **Le noeud secondaire (Secondary node) :** c'est une copie de la machine mastère (name node).
  - Périodiquement, le nœud secondaire va donc faire une copie des données du « Name Node » afin de pouvoir prendre la relève en cas de panne de ce dernier.


Cluster Hadoop
Nœud maître
Job Tracker
Name Node
Secondary Node
Nœud esclave
Task Tracker
Data Node
Nœud esclave
Task Tracker
Data Node
Nœud esclave
Task Tracker
Data Node

MapReduce

- Le traitement de données de façon distribuée soulève certaines questions,
  - Comment distribuer le travail entre les serveurs ?
  - Comment synchroniser les différents résultats ?
  - Comment gérer une panne d'une unité de traitement ?

- Map-Reduce répond à ces problèmes.

  - C'est un modèle de programmation s'inspirant des langages fonctionnels et plus précisément du langage Lisp.
  - Il permet de traiter une grande quantité de données de manière parallèle, en les distribuant sur les noeuds du cluster.
  - Ce modèle a été mis en place par Google en 2004 et il est utilisé par les sociétés utilisant des Data Centers telles que Facebook ou Amazon.

MapReduce

- **Principe de MapReduce**: il consiste à découper une tâche manipulant un gros volume de données en plusieurs tâches traitant chacune un sous-ensemble de ces données.

- MapReduce est constitué de deux étapes :

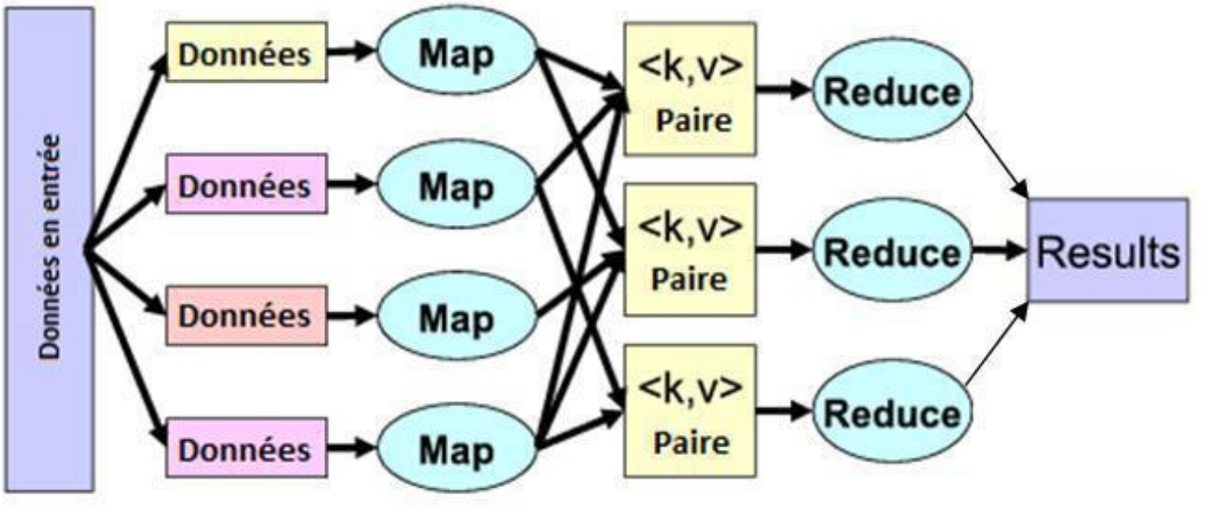
  - **MAP** : consiste à dispatcher une tâche, ainsi que les données quelle traite en plusieurs sous-tâches traitant chacune d'elles un sous-ensemble de données.
  - **Reduce** : il s'agit de récupérer les résultats des différents serveurs et de les consolider.

- Ce découpage permet aux développeurs de se focaliser sur le traitement. Le fait que MapReduce soit développé en java permet de abstraire de l'architecture matérielle pour qu'un framework MapReduce puisse tourner sur un ensemble de machines hétérogènes.

MapReduce

Le modèle MapReduce consiste en deux étapes, représentées toutes deux par des fonctions.

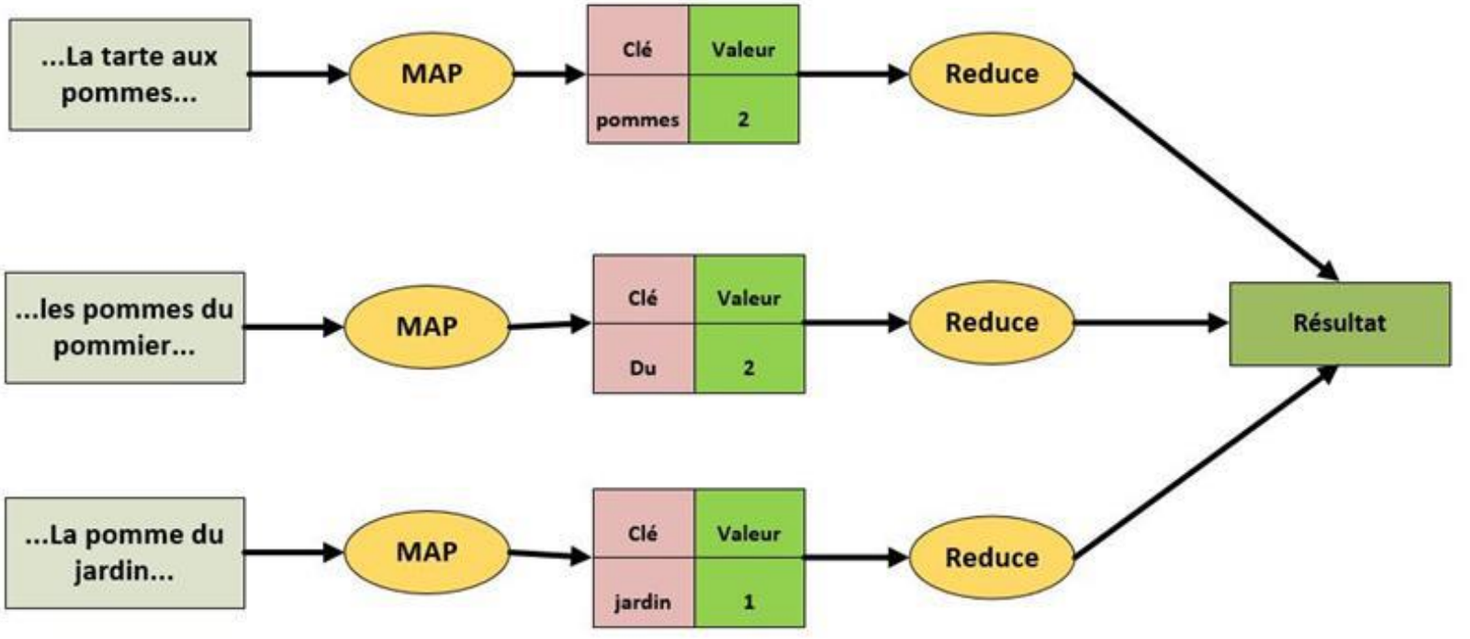
- La fonction map on prend en entrée un ensemble de « clé-valeurs »,
  - le noeud fait une analyse du problème et il va le séparer en sous-tâches, pour pouvoir les redistribuer sur les autres noeuds du cluster.
  - Dans le cas nécessaire, les noeuds recevant les sous-tâches refont le même processus de manière récursive.
  - Les sous-tâches des différents noeuds sont traitées chacune d'entre elles dans leur fonction map respective et vont retourner un résultat intermédiaire.
- La fonction « Reduce », consiste à faire remonter tous les résultats à leur noeud parents respectif.
  - Les résultats se remonte donc du noeud le plus bas jusqu'à la racine.
  - Avec la fonction Reduce, chaque noeud va calculer un résultat partiel en associant toutes les valeurs correspondant à la même clé en un unique couple (clé – valeur).
  - Une fois obtenu ce couple (clé-valeur), le résultat est remonté au noeud parent, qui va refaire le même processus de manière récursive jusqu'au noeud racine. Quand un noeud termine le traitement d'une tâche,
  - un nouveau bloc de données est attribué à une tâche Map.



❑Exemple d'utilisation.

Le problème est de compter le nombre d'occurrence de chaque mot.

- Données en entrée : fichiers texte.
- La fonction Map aura pour but de décomposer le texte en couple de mots clé-valeur.
- La fonction Reduce va prendre les couples avec la même clé et compter la totalité des occurrences pour ne faire qu'une seul paire clé-valeur par mot.



❑La fonction Map

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;

public class WC_Mapper extends MapReduceBase implements
Mapper<LongWritable,Text,Text,IntWritable>{
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

 public void map(LongWritable key, Text value, OutputCollector<Text,IntWritable>
output,   Reporter reporter) throws IOException{
        String line = value.toString();
        StringTokenizer  tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens())
       {
            word.set(tokenizer.nextToken());
            output.collect(word, one);
        }

    }

}
```

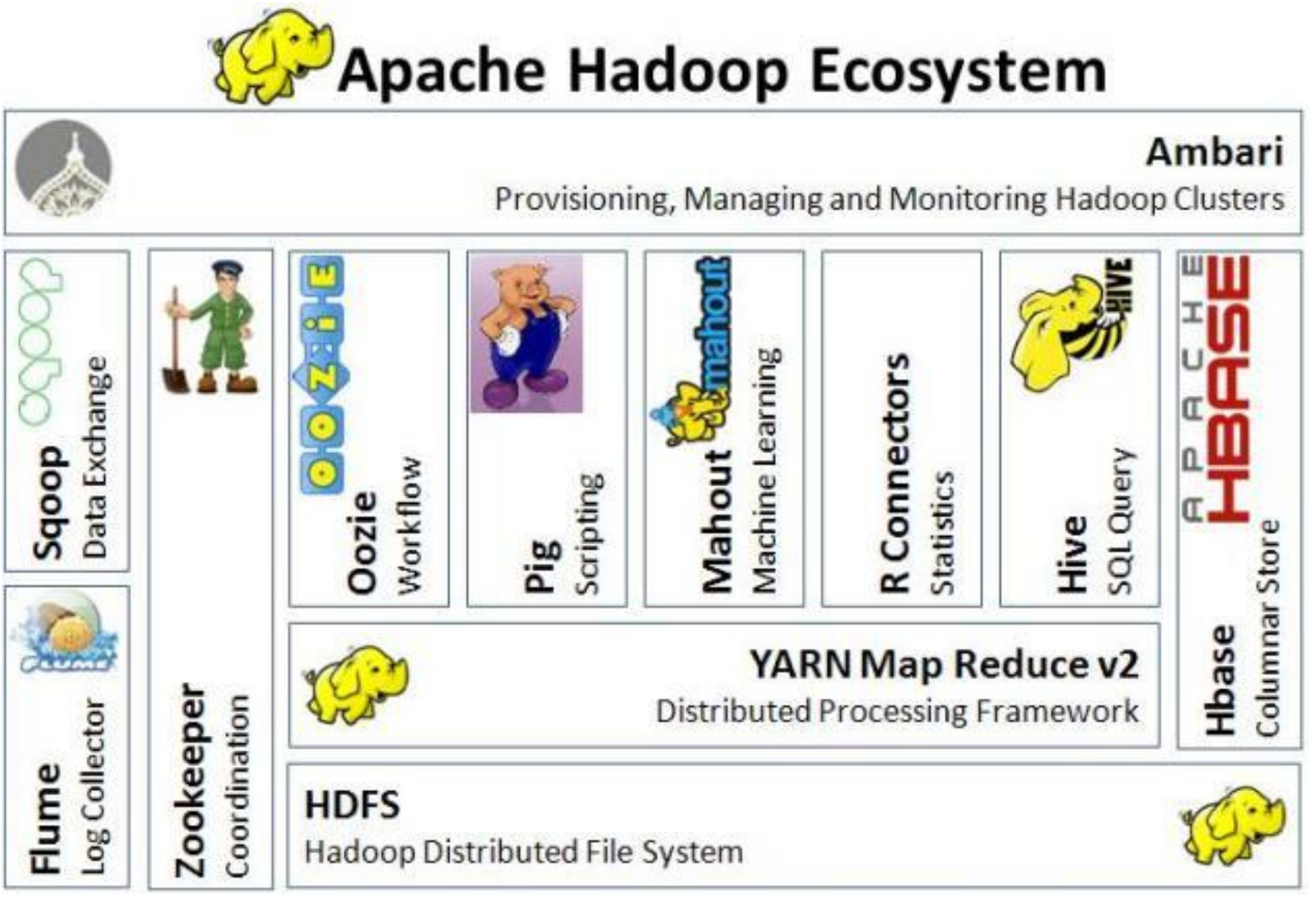
❑La fonction Reduce

```
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;

public class WC_Reducer  extends MapReduceBase implements
         Reducer<Text,IntWritable,Text,IntWritable> {

public void reduce(Text key, Iterator<IntWritable>
     values,OutputCollector<Text,IntWritable> output,
 Reporter reporter) throws IOException
 {
    int sum=0;
    while (values.hasNext())
       {
        sum+=values.next().get();
       }
    output.collect(key,new IntWritable(sum));
   }

}
```

❑ Le programme principal

```
import java.io.IOException;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.TextInputFormat;
import org.apache.hadoop.mapred.TextOutputFormat;
public class WC_Runner {
    public static void main(String[] args) throws IOException{
        JobConf conf = new JobConf(WC_Runner.class);
        conf.setJobName("WordCount");
        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(IntWritable.class);
        conf.setMapperClass(WC_Mapper.class);
        conf.setCombinerClass(WC_Reducer.class);
        conf.setReducerClass(WC_Reducer.class);
        conf.setInputFormat(TextInputFormat.class);
        conf.setOutputFormat(TextOutputFormat.class);
        FileInputFormat.setInputPaths(conf,new Path(args[0]));
        FileOutputFormat.setOutputPath(conf,new Path(args[1]));
        JobClient.runJob(conf);
    }
}
```

Les applications connexes de Hadoop



Les applications connexes de Hadoop

❑ Gestionnaire des données en Hadoop

❑ HDFS

- ❖ C'est le Système de stockage de fichiers distribué de Hadoop.
- ❖ Il permet de stocker au format natif n'importe quel objets (image, vidéo, fichiers texte)
- ❖ Il duplique les données sur plusieurs serveur pour garantir la disponibilité des données à n'importe quel moment.

❑ HBASE :

- ❖ Base NoSQL orientée colonnes. Le stockage est dans le HDFS.
- ❖ Hbase est une BD surcouche qui permet de se bénéficier des avantages des bases orientées colonnes :
  - ➢ Colonnes dynamiques : les colonnes nulles ne sont considérées.
  - ➢ Historisation de la valeur et non-plus de la ligne (suppression des doublons, quand une valeur change seule la valeur concernée est sauvegardée et non toute la ligne.

❑ Hive :

- ❖ C'est base de données relationnelle dans Hadoop.
- ❖ **Elle** permet de stocker la structure des tables et des bases de données créées
- ❖ Les données sont stockées dans le hdfs
- ❖ Le métadonnées Hive apporte une surcouche qui permet de connaître les structures de données stockéesdans hdfs.
- ❖ Utilise le langage SQL

Les applications connexes de Hadoop

❑Récupération des données

- ❑ **Apache Flume** :
  - ❖ C'est un système distribué permettant :
    - de récupérer des logs de plusieurs sources,
    - d' agréger les logs et
    - d'enregistrer les logs dans HDFS.
  - ❖ Il est fiable, flexible, assez intuitif et porté par toutes les distributions Hadoop.
- ❑ **Apache Storm** : Système distribué qui permet la récupération et le traitement en temps réel. Il ajoute des capacités de traitements de données à Hadoop.
- ❑ **Spark Streaming** : Librairie de spark permettant la récupération et le traitement de donnée en temps réel
- ❑ **Kafka** : Système distribué de messagerie pour l'agrégation de log, le traitement en temps réel et le monitoring.
- ❑ **Flink** : Il fournit un framework de traitements distribué en mémoire. Il est similaire à spark

❑**Apache Sqoop** : il permet de faire le lien entre un système de gestion de base de données relationnel (SGBDR) et HDFS. Moteurs d'interrogation et de manipulation des données

Les applications connexes de Hadoop

❑ Gestionnaires de ressources

- ❑ **Yarn (Yet Another Resource Negotiator)** :C'est gestionnaire des ressources (CPU, Ram
- ❑ **Mesos** : Projet Apache plus récent que Yarn,
  - ❖ C'est un système de gestion de ressource de clusters.
  - ❖ Il gère et partage de manière dynamique les ressources entre différents clusters entre les diverses applications.

Chapitre 3
Base de données NOSQL

Emergence de Base de données Nosql

- Base de données relationnelle
  - SGBDR : gère les bases de données relationnelles.
    - système transactionnel : réalise des transactions atomiques, cohérentes, isolées et durables
    - Propriétés ACID
    - Les contraintes d'intégrité sont respectées .
    - Base de données conçues selon le modèle entité/association
    - Gère uniquement les données structurées
    - Elle n'est dédiées à stocker les données non structurées
      - Vidéo, audio, texte, images
    - Base de données avec des schémas relationnels
    - SGBDR comporte :
      - Un LDD langage de définition de données
      - Un LMD : langage de manipulation de données
      - Un LCD : langage de contrôle de données
    - SQL est un LDD, LMD et LCD
    - GBDR exige des logiciels et des ordinateurs coûteux et des compétences en optimisation peu répandues pour gérer de très gros volumes de données
    - Problème de SGBD au niveau de distribution de données. Il gère les données d'une manière centralisée.

Caractéristiques des systèmes distribués

- Caractéristiques des systèmes distribués
  - Un système distribué ne peut garantir la satisfaction de trois contraintes suivantes :
    - **Cohérence (Consistency)** : toutes les machines du système ont les mêmes données à tout moment.
    - **Disponibilité (Availability)** : le système répond aux requêtes même en présence des pannes.

    → Si une machine tombe en panne, le système continue à répondre requête.
    - Tolérance au partitionnement (**Partition tolerance**) : aucune panne ne doit empêcher le système de répondre correctement.
  - Tout système ne garantit qu'au plus deux contraintes parmi ces trois.

→ **transactions ACID impossibles dans un environnement distribué.**

- **Exemple** : si un utilisateur modifie la valeur d'un champ sur un des nœuds du système. Un deuxième utilisateur doit attendre un certain temps pour pouvoir y accéder. Donc, le système n'est pas disponible durant le temps de modification de données.

  - Trois niveau de cohérence sont identifiés :
    - Cohérence forte : tous les serveurs sont cohérents après une MAJ
    - Cohérence faible : aucune garantie de cohérence après une MAJ (s'il y a une défaillance système)
    - Cohérence éventuelle : s'il y a aucune MAJ encours, le système est censé être cohérent.

- Les systèmes distribués favorisent la propriété BASE au détriment de la cohérence et de l'exactitude de réponse.

- **propriété BASE**
  - **Basically Available** : le système est toujours accessible. Il peut inaccessible pour des courtes périodes.
  - **Soft state** : l'état de Base de données ne sont pas garantis. Les mises à jour ne sont pas en temps réel.
  - **Eventual consistency** : la cohérence des données à un instant donné n'est pas primordiale.

Bases de données NoSQL | Types

- Base de Données NoSQL
  - NoSQl : Not Only SQL
  - Il y a quatre types de Base de données NoSql
    - Bases de données clés-Valeurs : Clé-valeur (Key-value (KV)) Collection de couples (clé, valeur) (key-value pairs) → tableau associatif
    - Bases de données orientés document : Modèle clé-valeur permettant de disposer d'une valeur complexe (un document), chaque document étant composé de champs et de valeurs associées.
    - Bases de données orientées colonne : modèle clé-valeur permettant de disposer de plusieurs valeurs il permet de représenter les associations un à plusieurs)
    - Bases de données orientées graphe : Gestion de relations multiples entre les objets

Bases de données NoSQL | Types

- Base de données Clé-valeur
  - Clé est l'identfiant unique
  - La valeur peut être scalaire ou structurée

- Les bases de données clé-valeur sont hautement divisibles et permettent une mise à l'échelle horizontale à des échelles que d'autres types de bases de données ne peuvent pas atteindre

- 

- **Manipulation des données**
  - Création : créer un nouveau couple (clé, valeur). La valeur peut être n'importe quel objet.
  - La lecture : lire un objet en connaissant sa clé
  - La modification : mettre à jour l'objet associé à une clé
  - La suppression : supprimer un objet connaissant sa clé
- **Avantages**
  - Simplicité, scalabilité, disponibilité
  - Très bonnes performances puisque les lectures et écritures sont réduites à un accès disque simple

- Faiblesse :
  - Pas de requêtes sur le contenu des objets stockés
  - Pas de requêtes impliquant des relations entre les objets

Base de données orientées documents

- BD orientée documents = collection de documents
- Structure arborescente : liste de champs (dont la valeur peut être une liste de champs)
- Deux documents peuvent avoir des champs différents

- Caractéristiques :
  - Modèle simples et puissant
  - Evolutive du schéma.
  - Expressivité de langage de requête Json
- Exemples
  - MongoDB
  - CouchDB

Bases de données NoSQL | Types

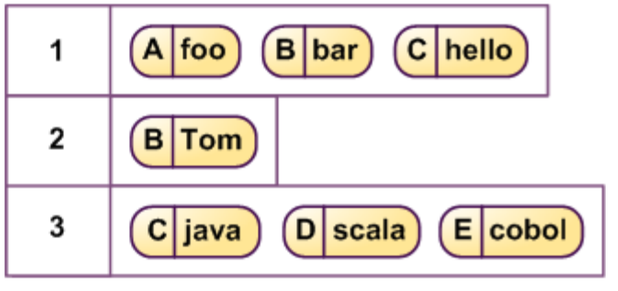
- Base de données :
  - Ensemble de collections
  - Espace de stockage
- Collection ←→ Table en modèle relationnel
  - Ensemble de documents qui partage un objectif commun
  - Pas de schéma prédéfini
- Document
  - Un enregistrement dans une collection
  - Syntaxe et stockage en langage BSON (Binary JSON)
  - Une valeur peut être un objet complexe

Base de données orientées colonne

- Une **base de données orientée colonnes** est une BD qui stocke les données par colonne et non par ligne.
- L'orientation colonne permet d'ajouter des colonnes plus facilement aux tables.

	A	B	C	D	E
1	foo	bar	hello		
2		Tom			
3			java	scala	cobol

Organisation d'une table dans une BDD relationnelle



Organisation d'une table dans une BDD orientée colonnes

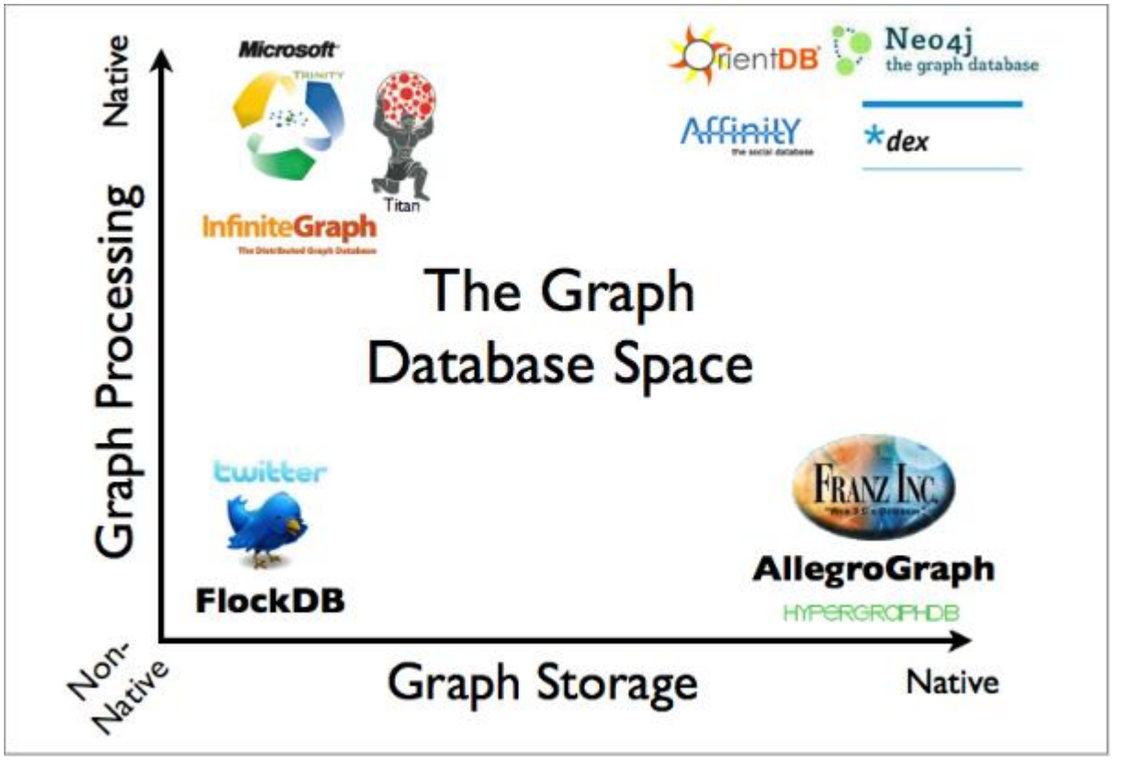
- Elles similaires aux bases de données classiques (SGBDR).
  - ✓ il y a le concept des tables avec des lignes et des colonnes

- Les deux principales grosses différences :
  - **Les colonnes sont dynamiques.** Au sein d'une même table deux individus peuvent ne pas avoir le même nombre de colonnes.
    - les valeurs nulles ne sont pas stockées
      - → libérer de la place de stockage
      - → améliorer les performances de traitement
    - Créer une seule table contenant toutes les données
      - Pas besoin de plusieurs tables comme c'est le cas dans les modèles relationnels
      - L'absence de 'jointure' entre les tables améliore les performances.
- L'historisation des données se fait à la valeur et non pas à la ligne comme dans les SGBDR.
- → empêche le stockage d'informations en double.
- Exemples des bases de données Orientées colonnes

- BD orientée graphe correspond à un système de stockage capable de fournir une adjacence entre éléments voisins.

  → chaque voisin d'une entité est accessible grâce à un pointeur physique.

- C'est une base de données orientée objet adaptée à l'exploitation des structures de données de type graphe ou arbre.
- Exemples de BD orientées graphes



- Similarités des BDs NoSql
- Schéma implicite :
  schéema de données non prédéfini coté serveur,
  Application cliente structures les données
- Absence de relation : Pas de relation entre données, pas de références entre éléments, Sauf exception : orientée graphe (structurel),
  Hive avec jointure (fonctionnel)
- Le langage de manipulation : souvent un langage déclaratif est utilisé