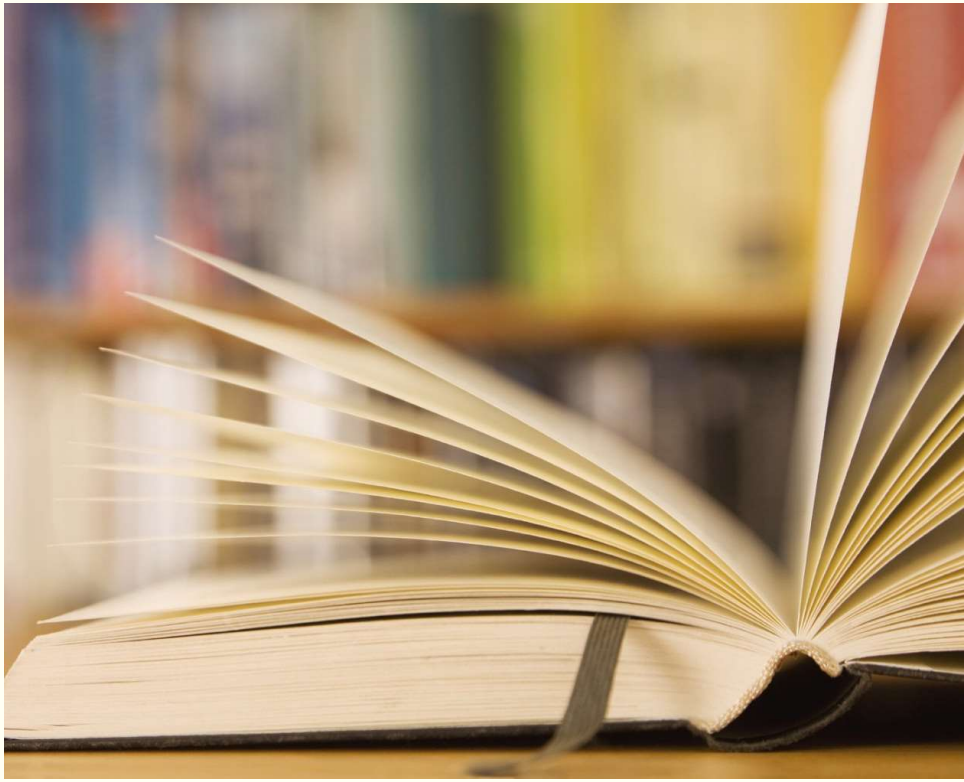
FONDAMENTAUX ÉCOSYSTÈME HADOOP

Durée : 1 jour

Mbengue Mohamadou



Disposition de titre et de contenu avec liste

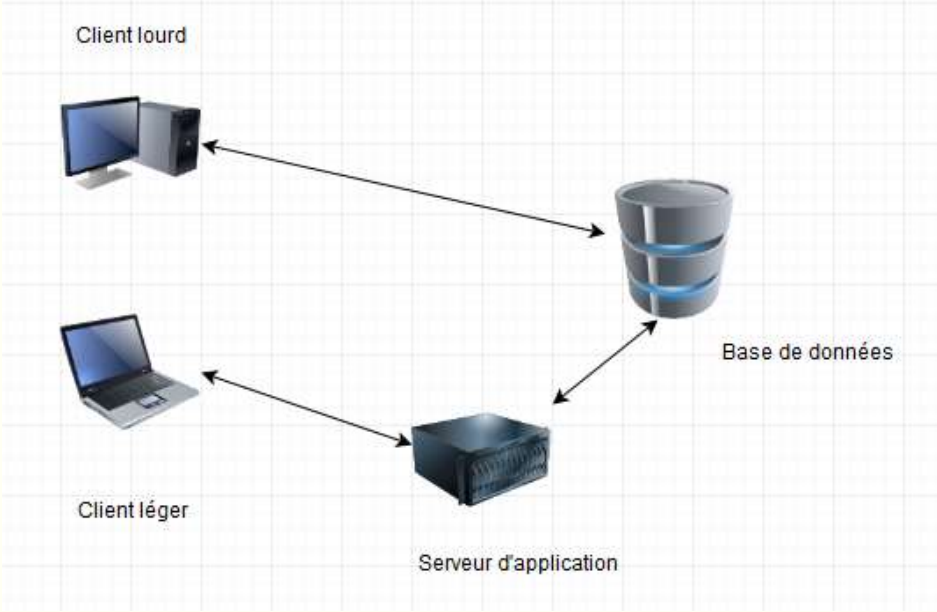
- Module 1 : Introduction au Big Data
- Module 2 : Hadoop: présentation
- Module 3 : L'architecture Hadoop
- Module 4 : L’Ecosystème
- Module 5 : Exemples / Démos interactifs

MODULE 1

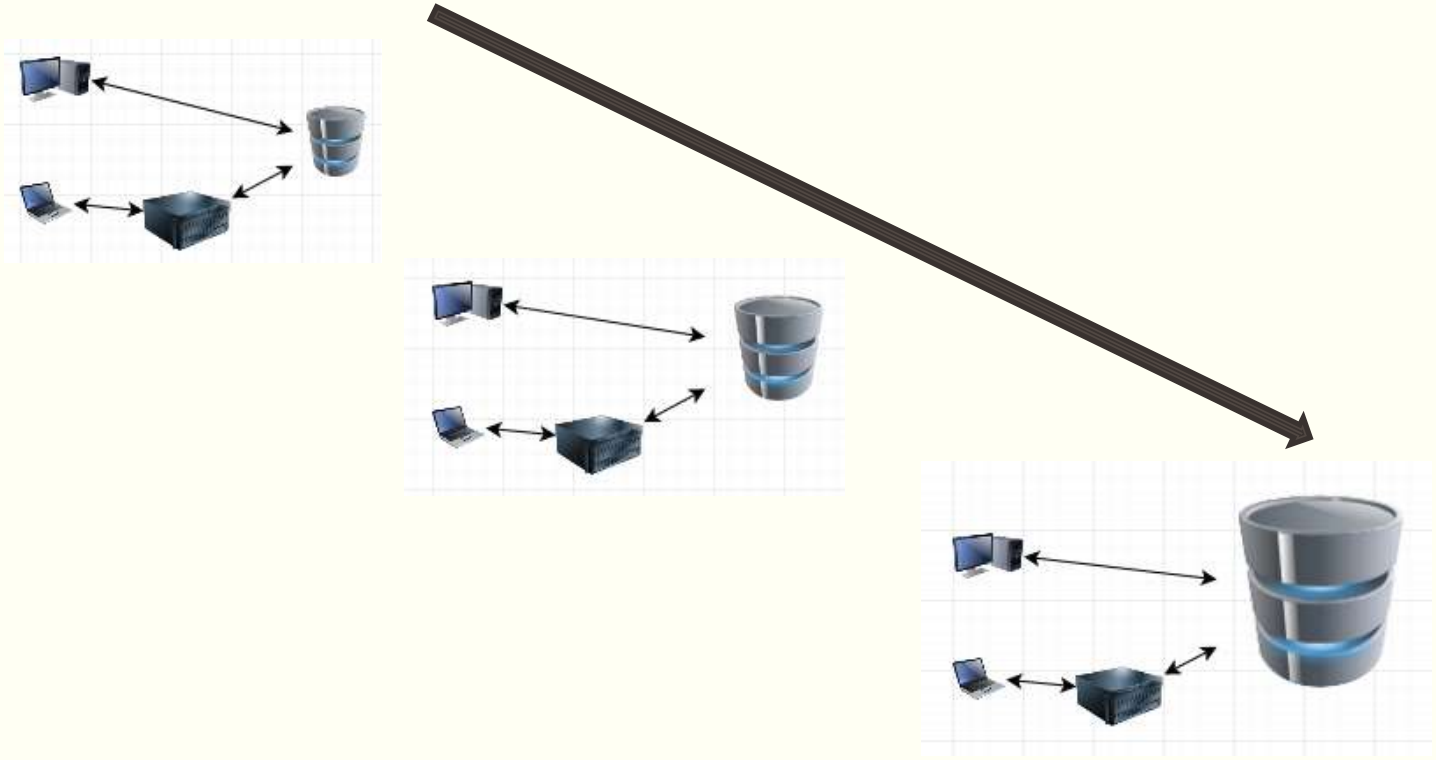
Introduction au Big Data

Le monde avant …

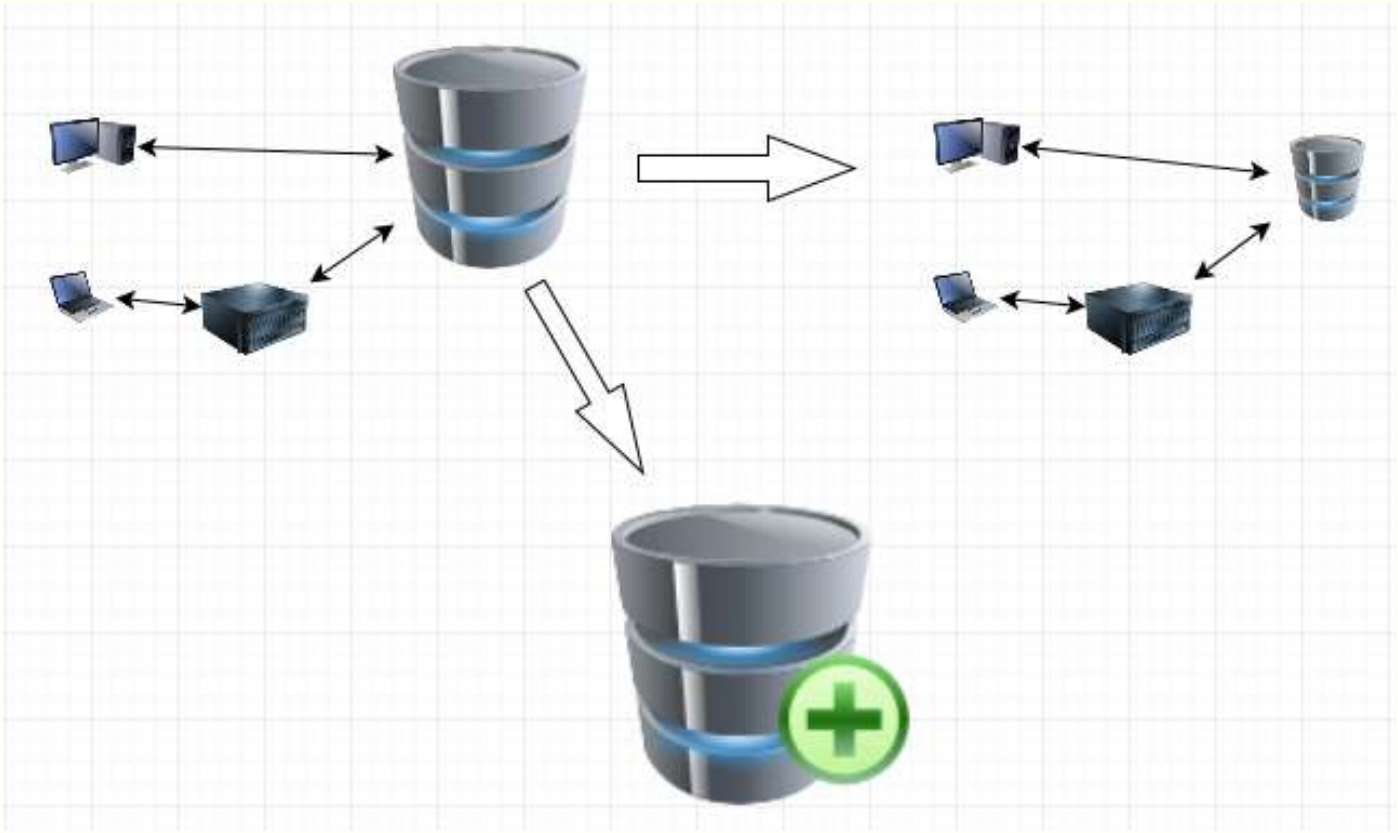
- Tout commence dans le monde applicatif



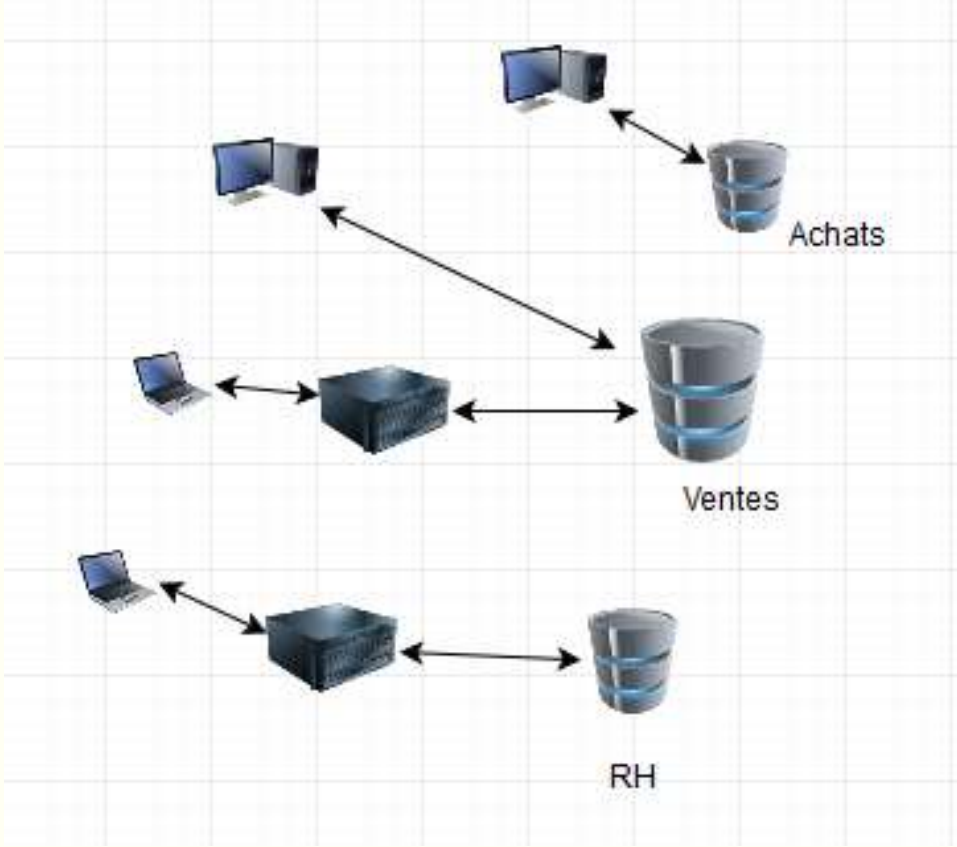
1ER Besoin Métier



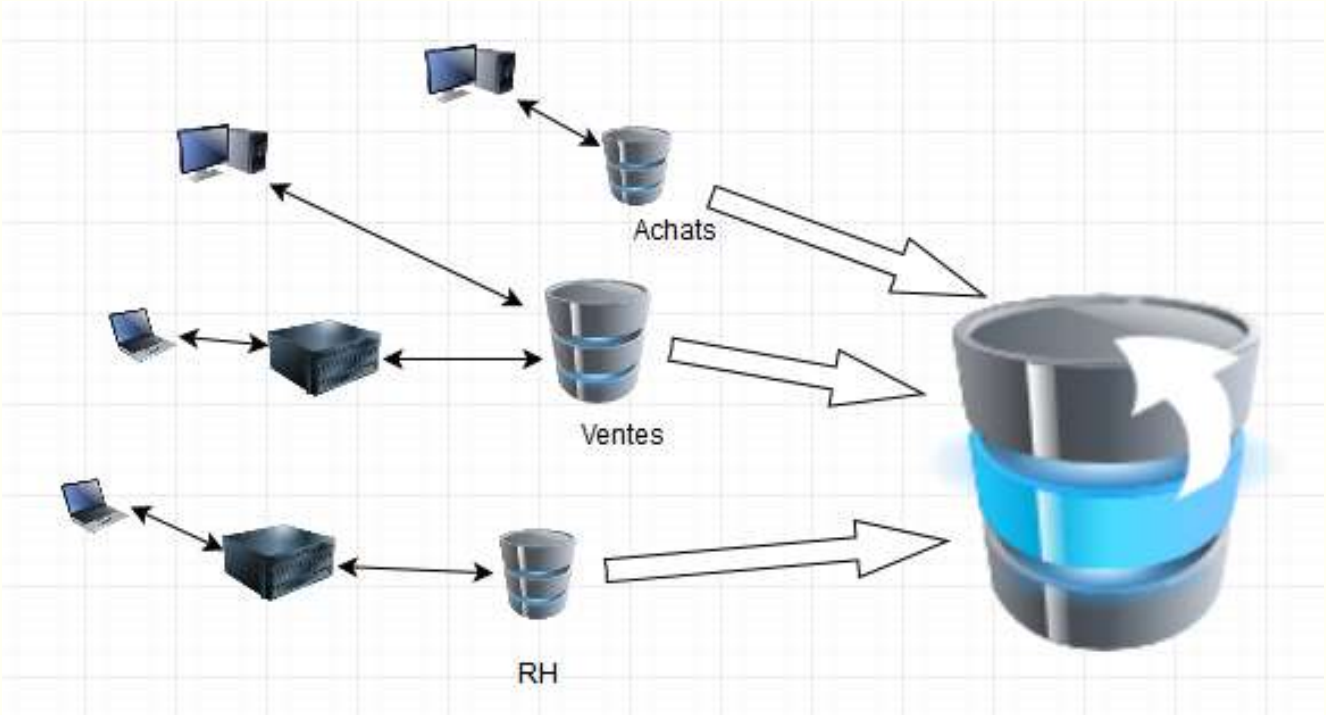
Besoin : Historisation



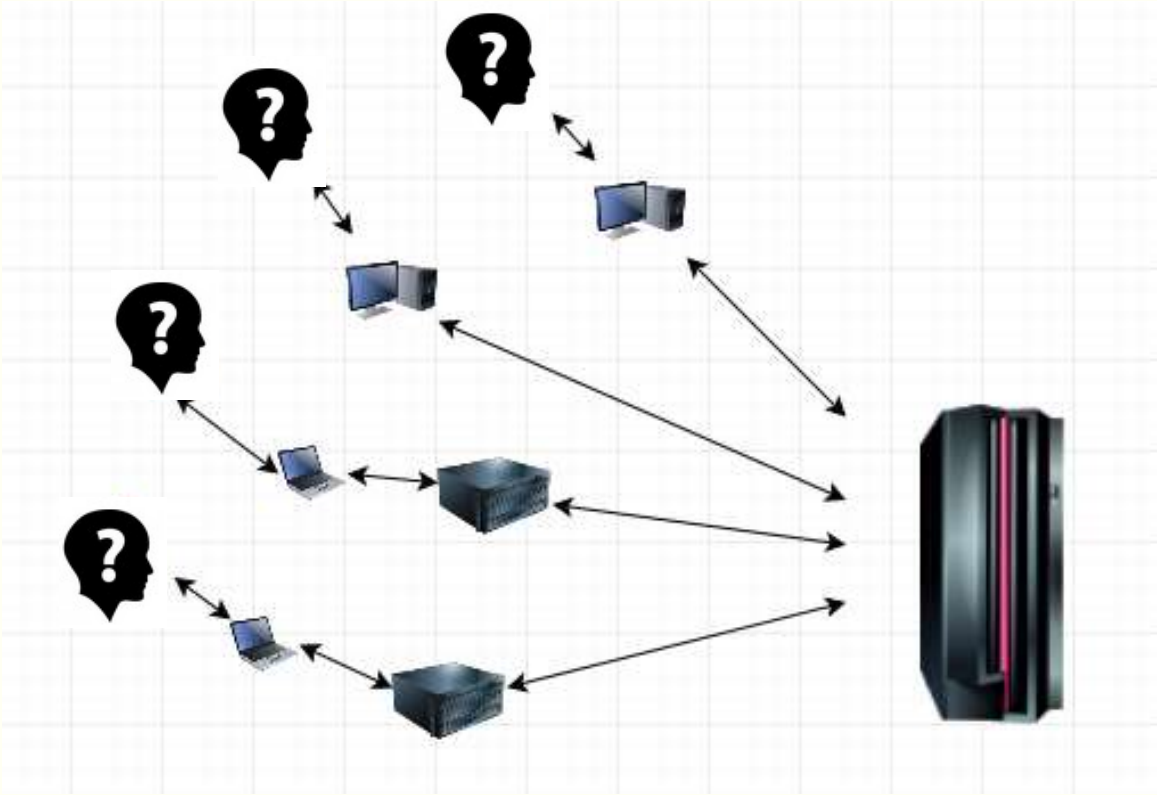
2ème Besoin Métier



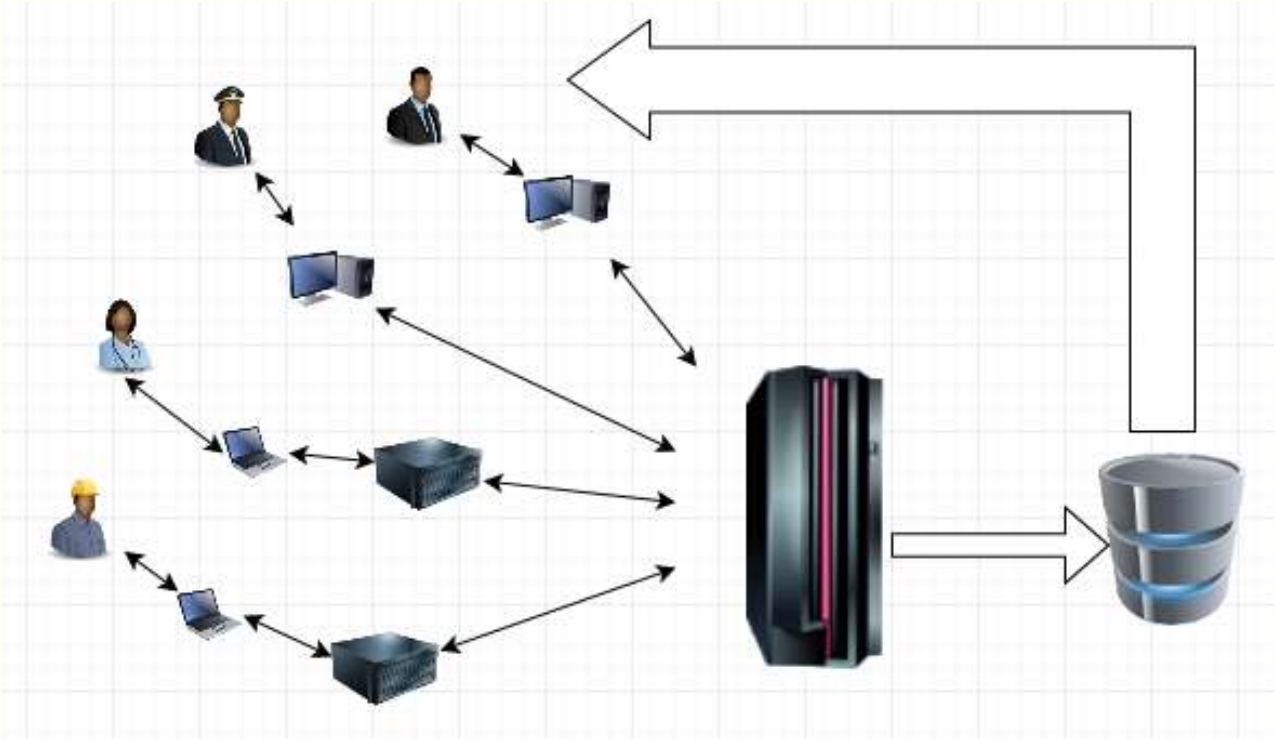
Besoin : Centralisation



3éme Besoin Métier



Besoin : Analyser



5 Etapes pour 3 besoins

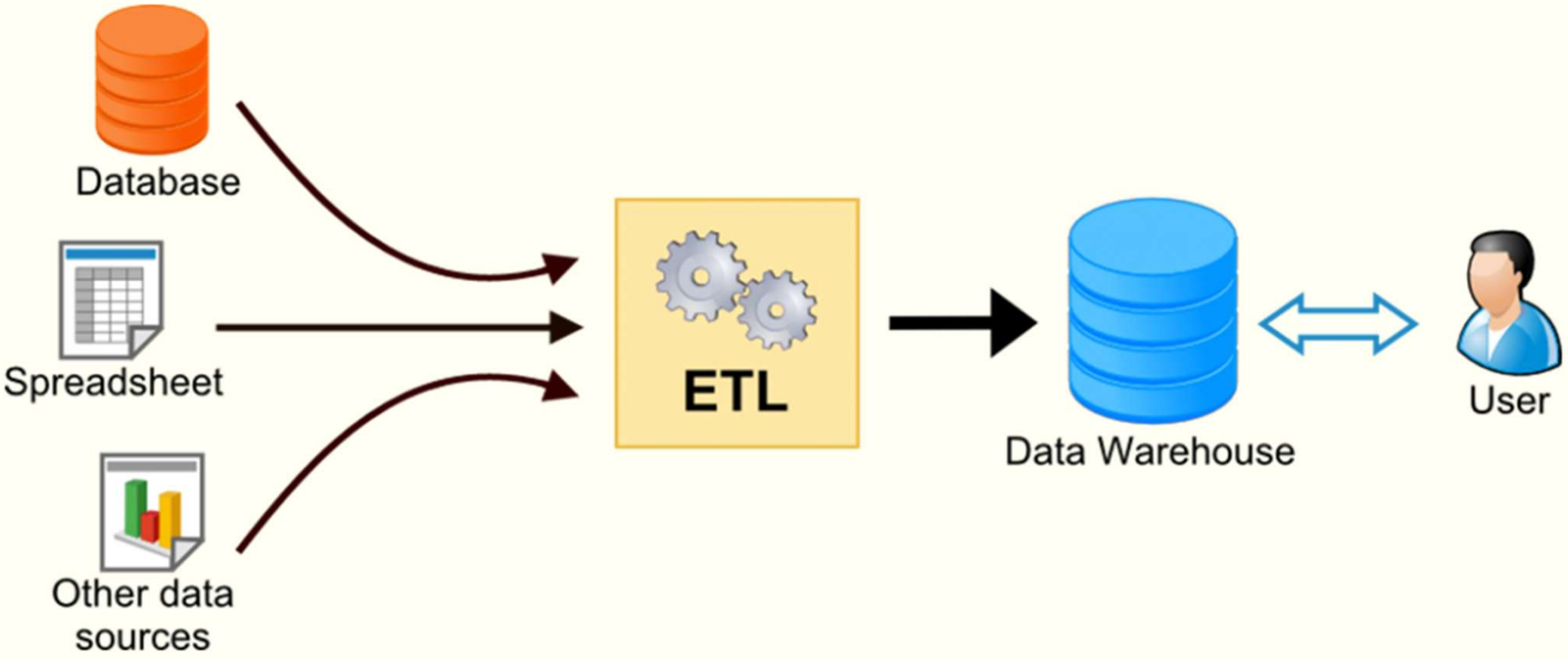
Préparation

- Extraction
- Nettoyage
- Stockage

Présentation

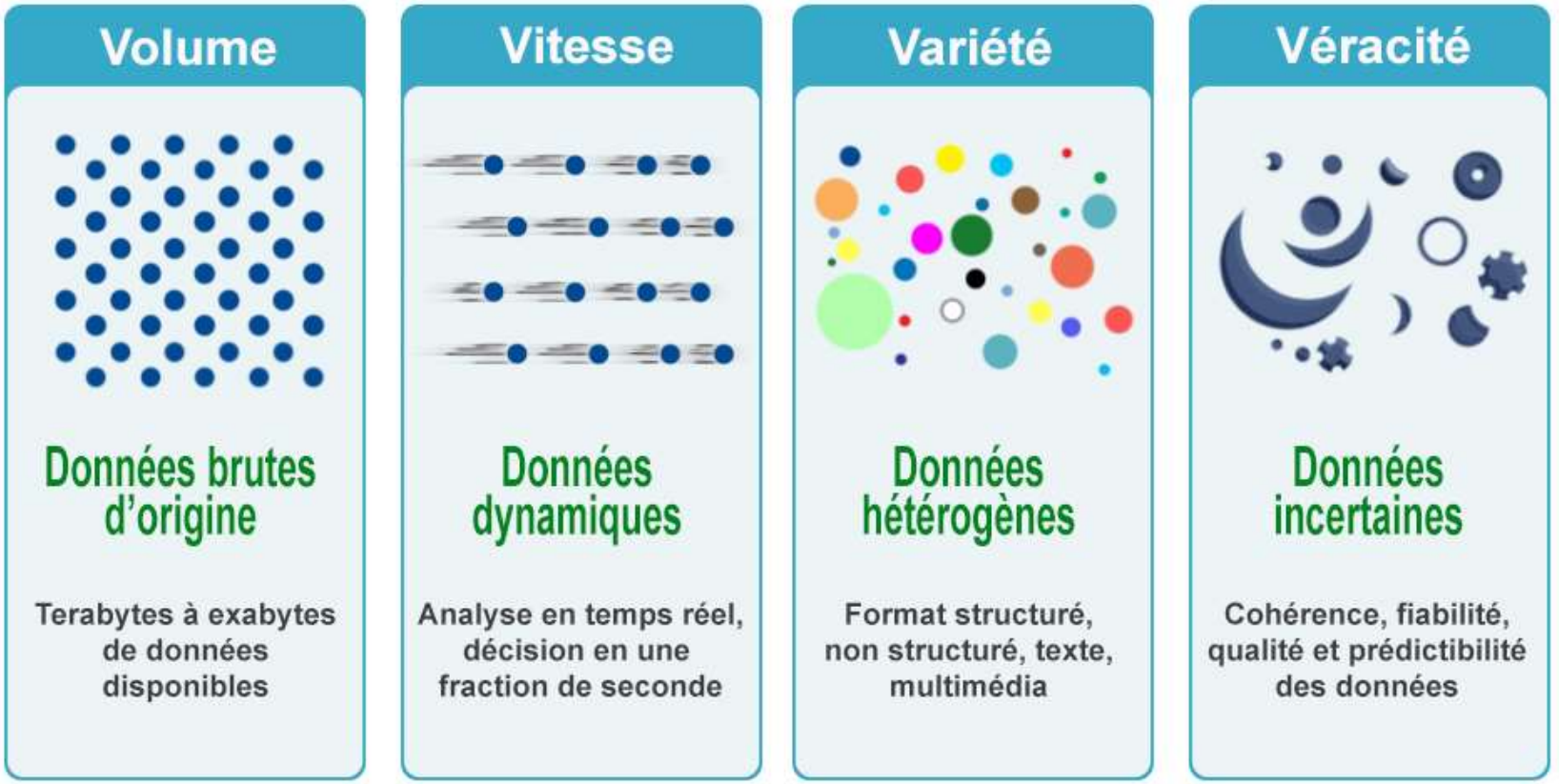
- Analyse
- Reporting

Traitement classique



Qu’est ce que le Big Data ?

Le Big Data couvre quatre dimensions : volume, vélocité, variété et véracité.



Les "4 V" ou principaux attributs du Big Data. Documents IBM et T.Lombry.

Volume 1/4

Volume : les entreprises sont submergées de volumes de données croissants de tous types, qui se comptent en téraoctets, voire en pétaoctets.

Exemple:

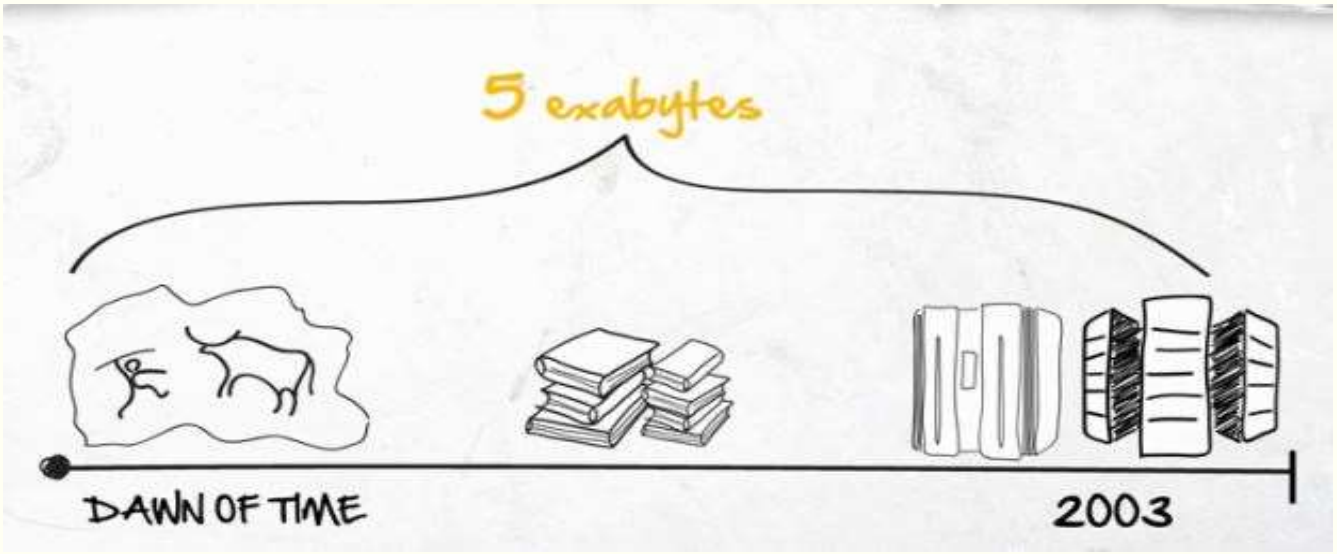
- Google
- Youtube
- FaceBook
- Twiter
- ...

Volume 2/4

Quelques repères sur les volumétries

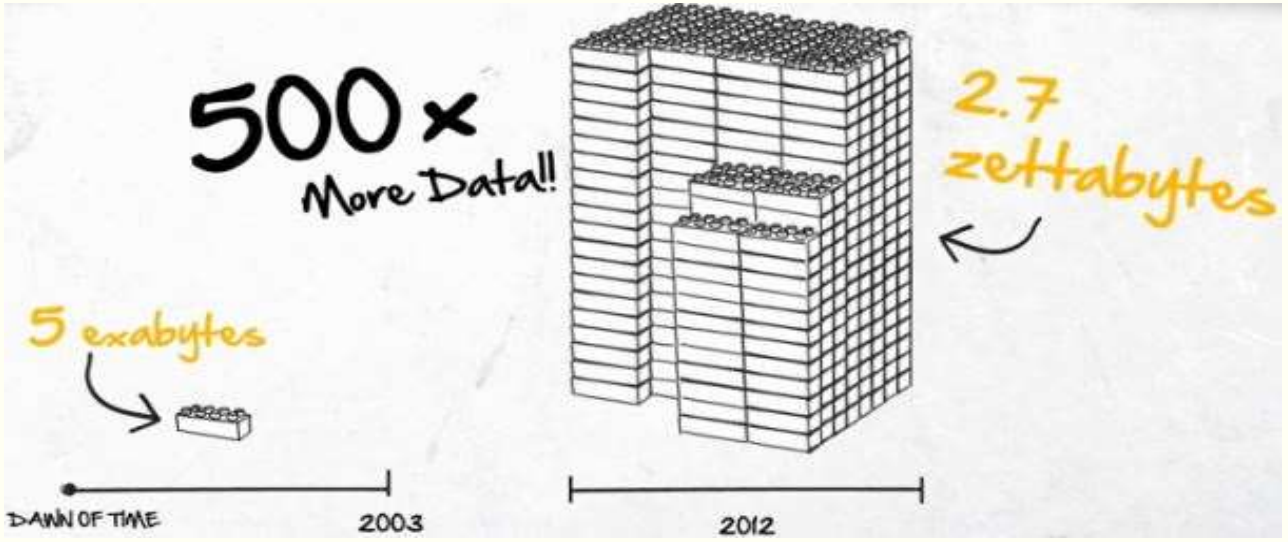
Volume 3/4

Evolution du volume des Données



A l'origine ….

Volume 4/4



Rien qu'en 2012 ...

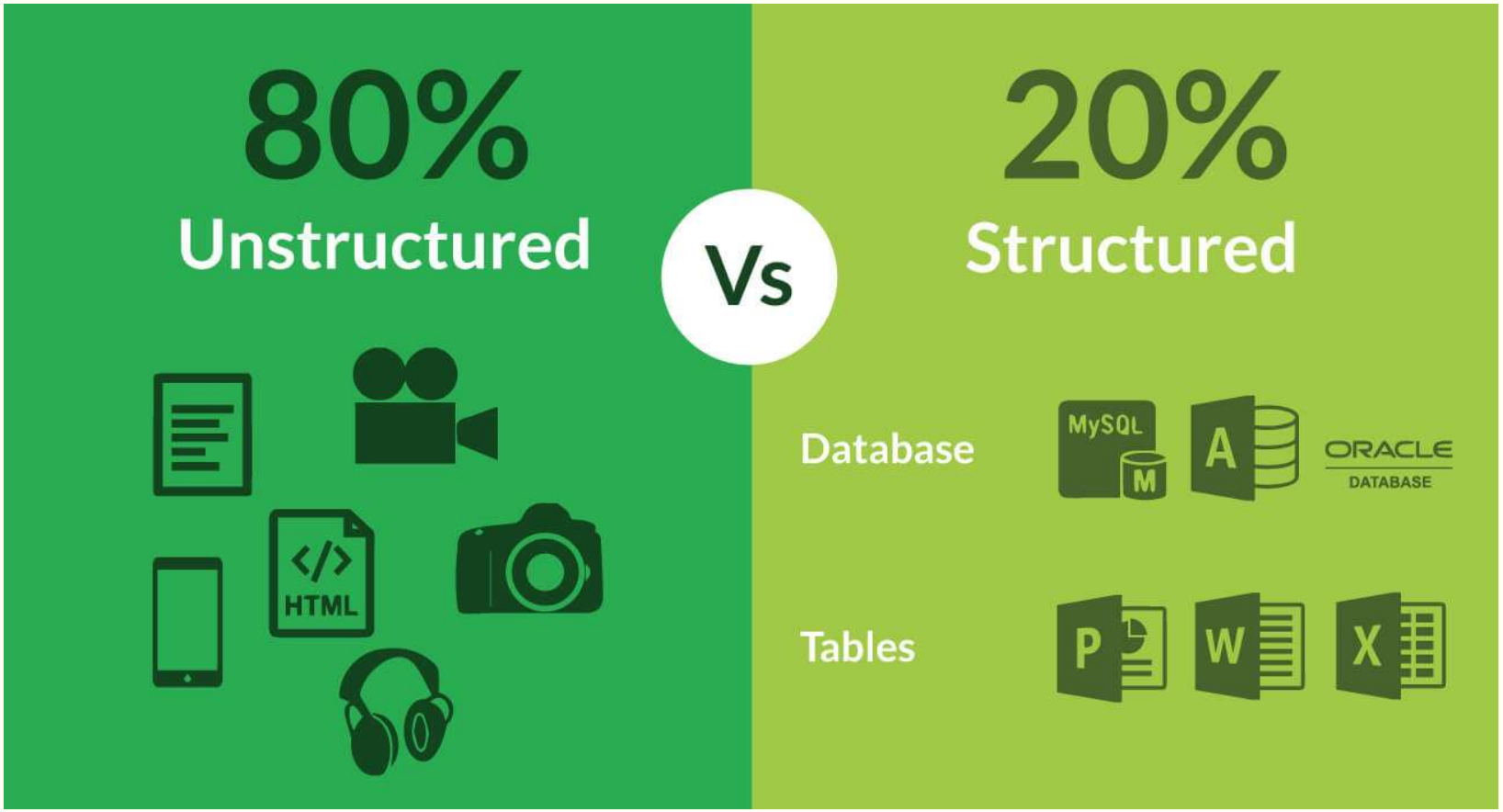
Variété 1

Variété : le Big Data se présente sous la forme de données structurées ou non structurées (texte, données de capteurs, son, vidéo, données sur le parcours, fichiers journaux, etc.).

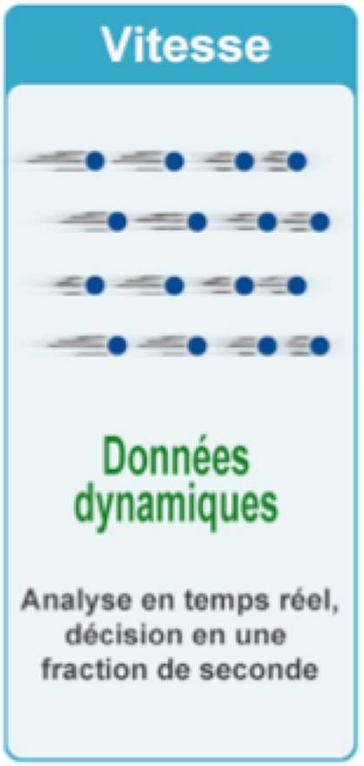
Exemple:

- De nouvelles connaissances sont issues de l'analyse collective de ces données.
- Utiliser les centaines de flux vidéo des caméras de surveillance pour contrôler les points d'intérêt.
- Tirer parti de la croissance de 80 % du volume de données image, vidéo et documentaires pour améliorer la satisfaction client.

Variété 2


80%
Unstructured
Vs
20%
Structured
HTML
Database
MySQL
ORACLE
DATABASE
Tables

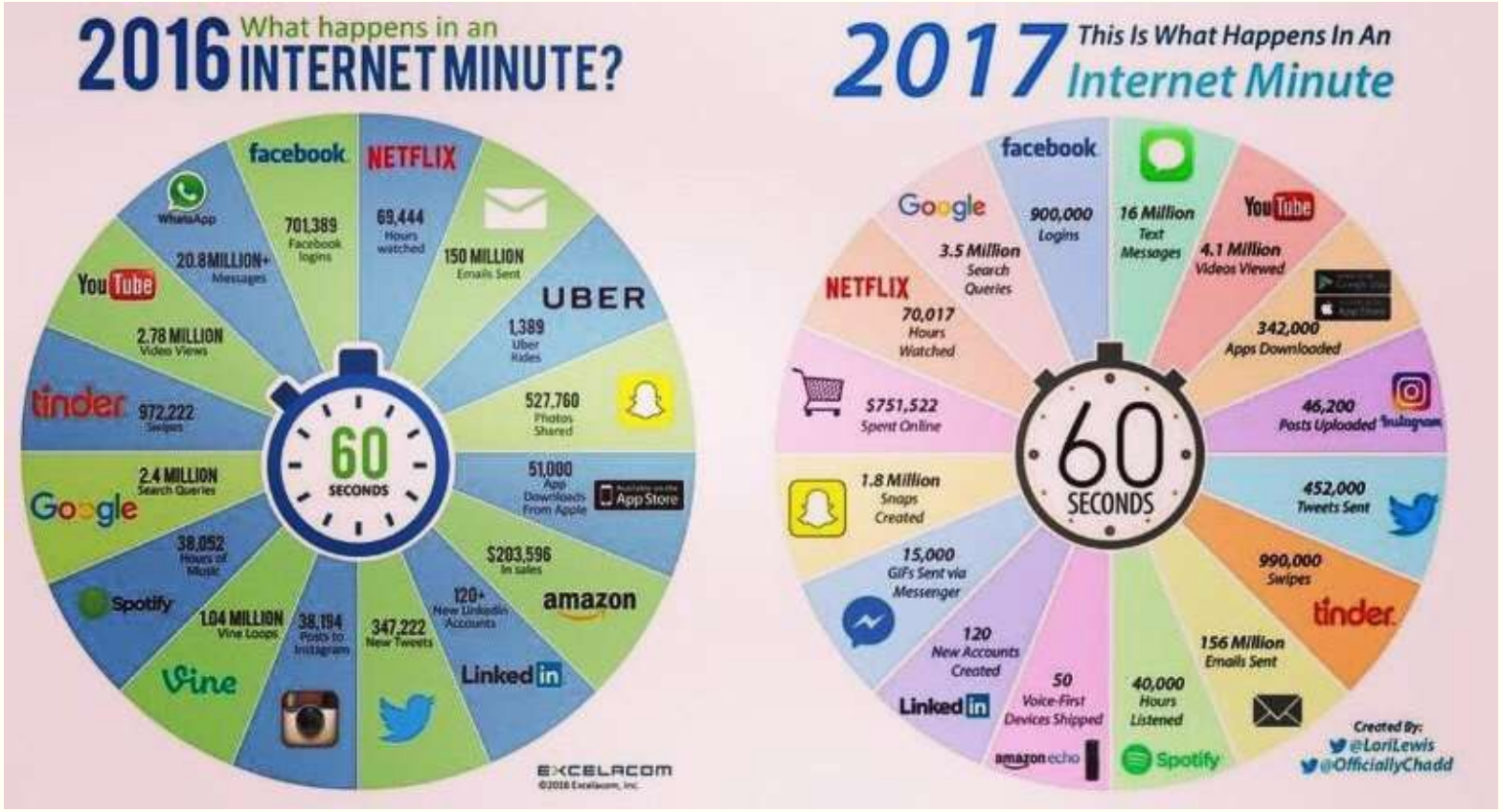
Vélocité 1/2



Vélocité : parfois, 2 minutes c'est trop.

- Pour les processus chrono-sensibles tels que la détection de fraudes, le Big Data doit être utilisé au fil de l'eau, à mesure que les données sont collectées par votre entreprise afin d'en tirer le maximum de valeur.
- Scruter 5 millions d'événements commerciaux par jour afin d'identifier les fraudes potentielles.
- Analyser en temps réel 500 millions d’enregistrements détaillés d’appels quotidiens.

Vélocité 2/2


2016 What happens in an INTERNET MINUTE?
facebook
701,389 Facebook logins
NETFLIX
69,444 Hours watched
150 MILLION Emails Sent
UBER
1,389 Uber Rides
527,760 Photos Shared
51,000 App Downloads From Apple
App Store
$203,596 In sales
amazon
120+ New LinkedIn Accounts
Linked in
347,222 New Tweets
38,194 Posts to Instagram
1.04 MILLION Vine Loops
Vine
38,052 Hours of Music
Spotify
2.4 MILLION Search Queries
Google
972,222 Swipes
tinder
2.78 MILLION Video Views
YouTube
20.8 MILLION+ Messages
WhatsApp
60 SECONDS
EXCELACOM
2017 This Is What Happens In An Internet Minute
facebook
900,000 Logins
16 Million Text Messages
YouTube
4.1 Million Videos Viewed
342,000 Apps Downloaded
46,200 Posts Uploaded
452,000 Tweets Sent
990,000 Swipes
tinder
156 Million Emails Sent
40,000 Hours Listened
Spotify
50 Voice-First Devices Shipped
amazon echo
120 New Accounts Created
Linked in
15,000 GIFs Sent via Messenger
1.8 Million Snaps Created
$751,522 Spent Online
NETFLIX 70,017 Hours Watched
3.5 Million Search Queries
Google
60 SECONDS
Created By:
@LoriLewis
@OfficiallyChadd

Véracité

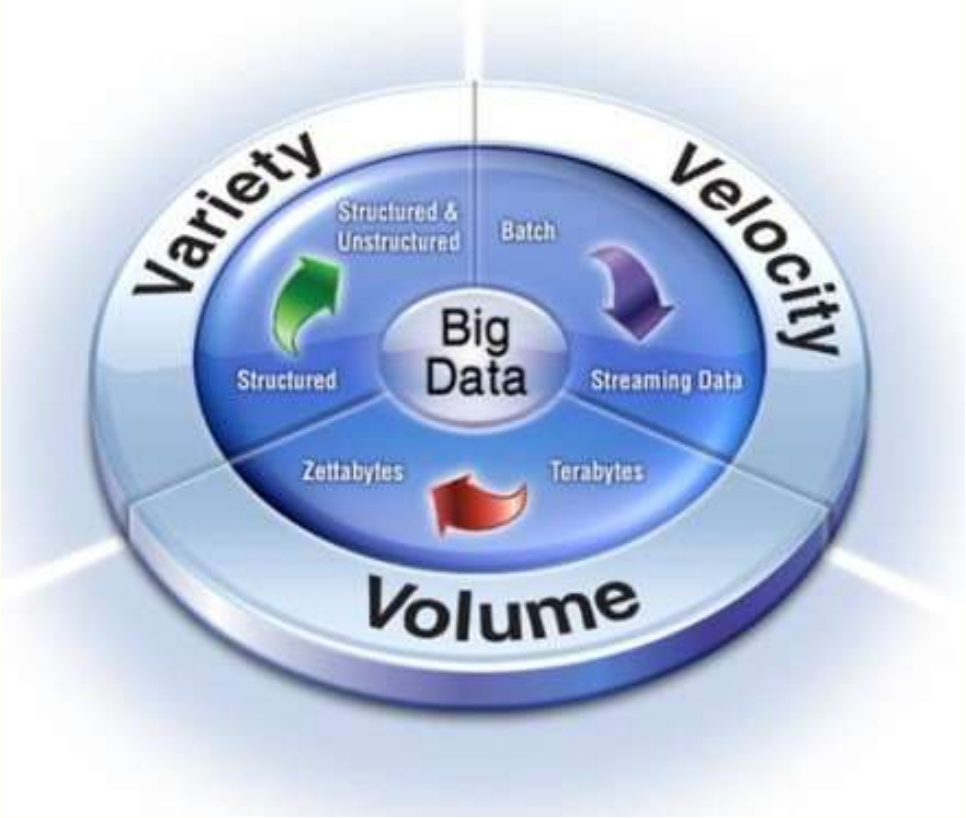
Véracité : 1 décideur sur 3 ne fait pas confiance aux données sur lesquelles il se base pour prendre ses décisions.

- Comment pouvez-vous vous appuyer sur l'information si vous n'avez pas confiance en elle?
- Etablir la confiance dans les Big Data représente un défi d'autant plus important que la variété et le nombre de sources augmentent.

Comment faire face à de telles quantités de données ?

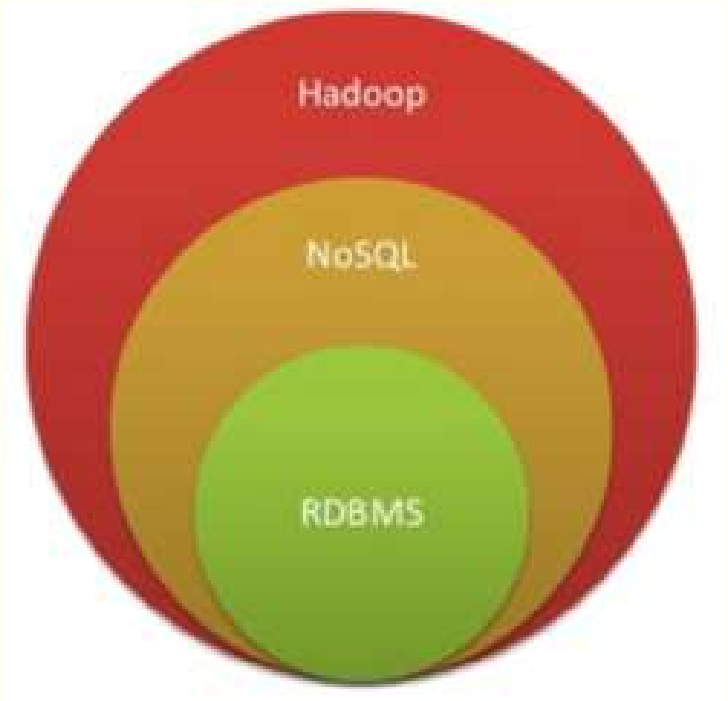
Décisionnel vs 3V

- L'approche classique incompatible avec les 3V du BigData

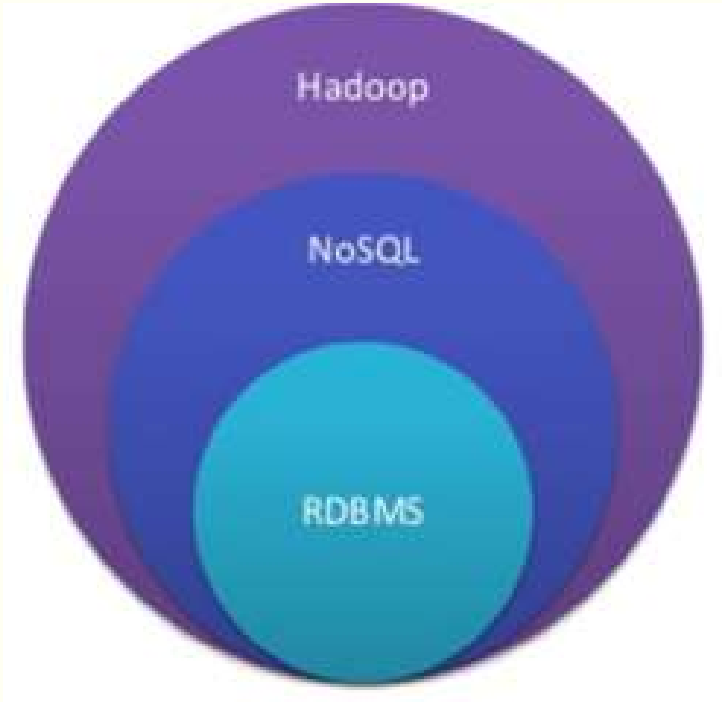


Problématique de Stockage de données

En interne ?

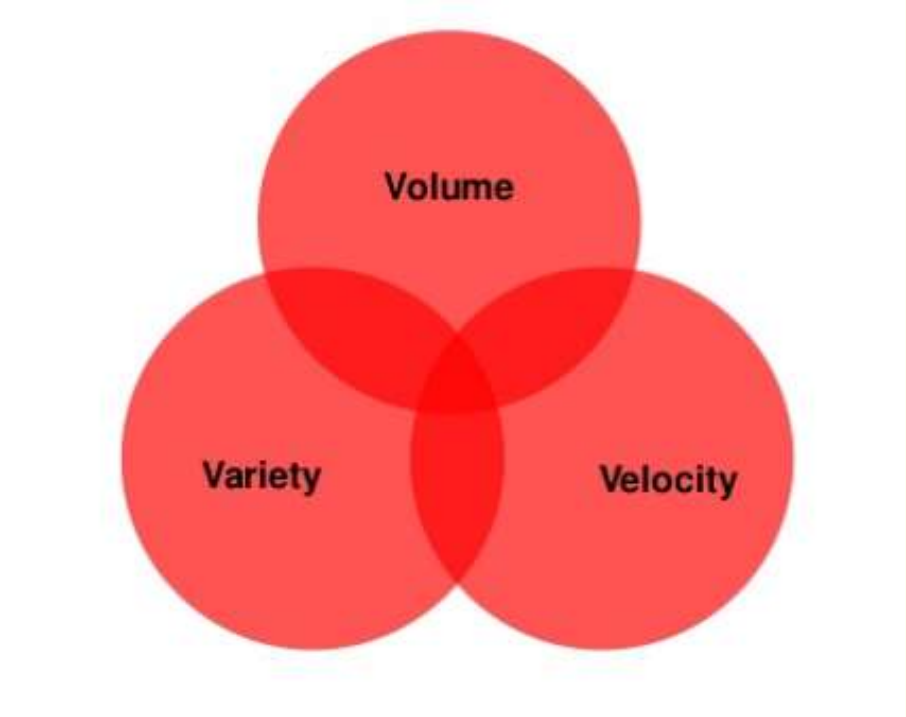


Sur le cloud ?

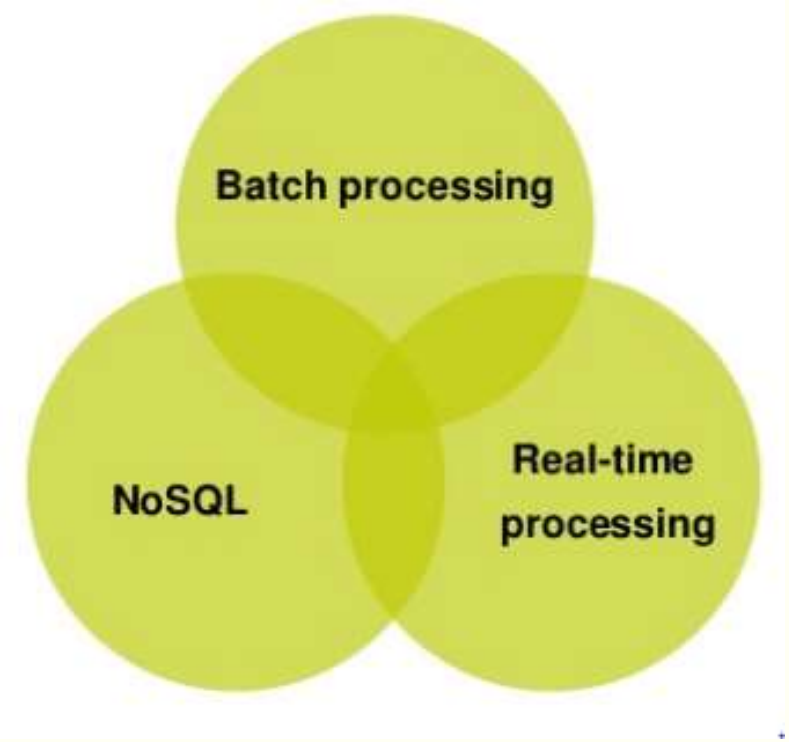


Problématique de traitement de données

- 3 Problèmes



- 3 Solutions



Le NoSql 1

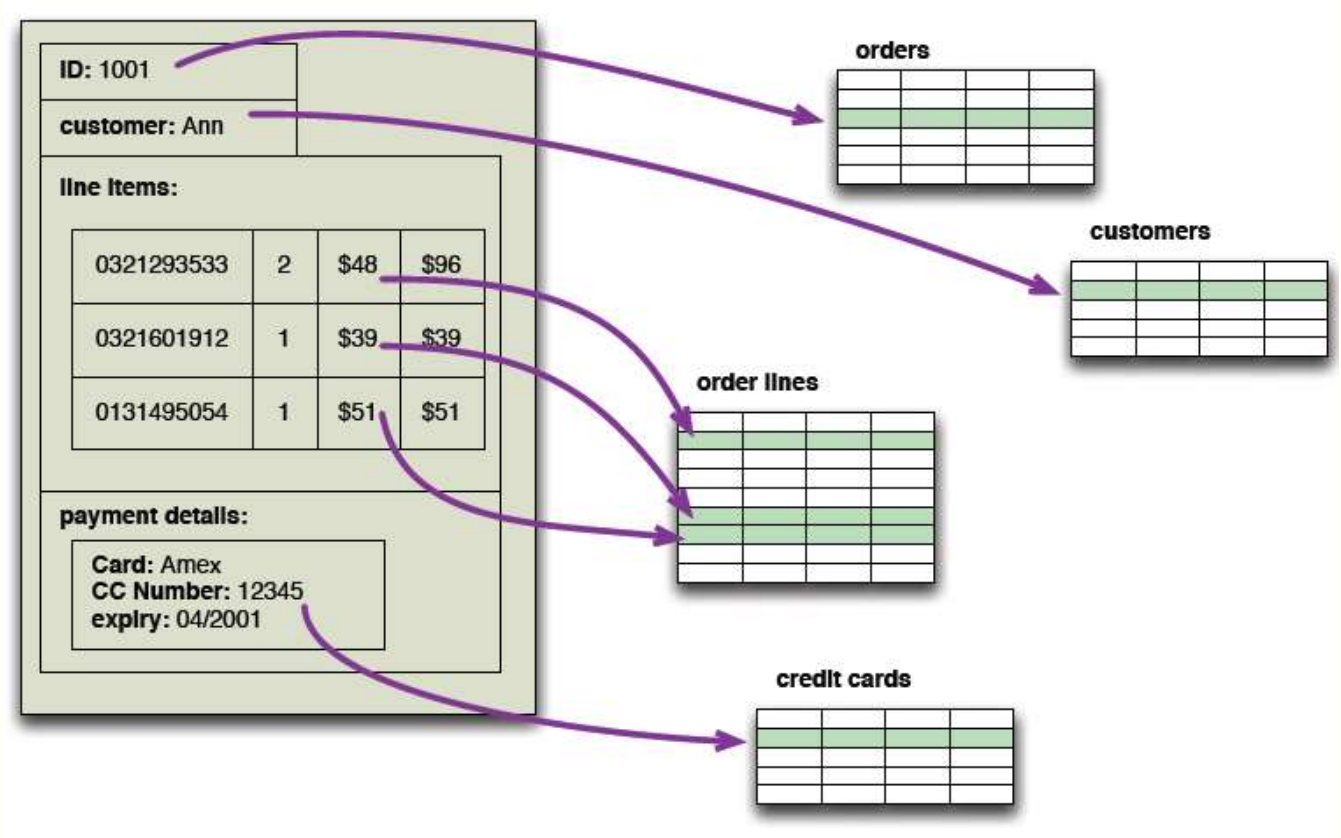
Le monde Relationnel

Caractéristiques :

- Données structurées (Table/Schéma),
- Données Normalisées (Formes Normales),
- Standard (SQL),
- Transactionnel (ACID),
- Requête Complexe (Jointure),
- Des contraintes(Intégrité des données),
- …

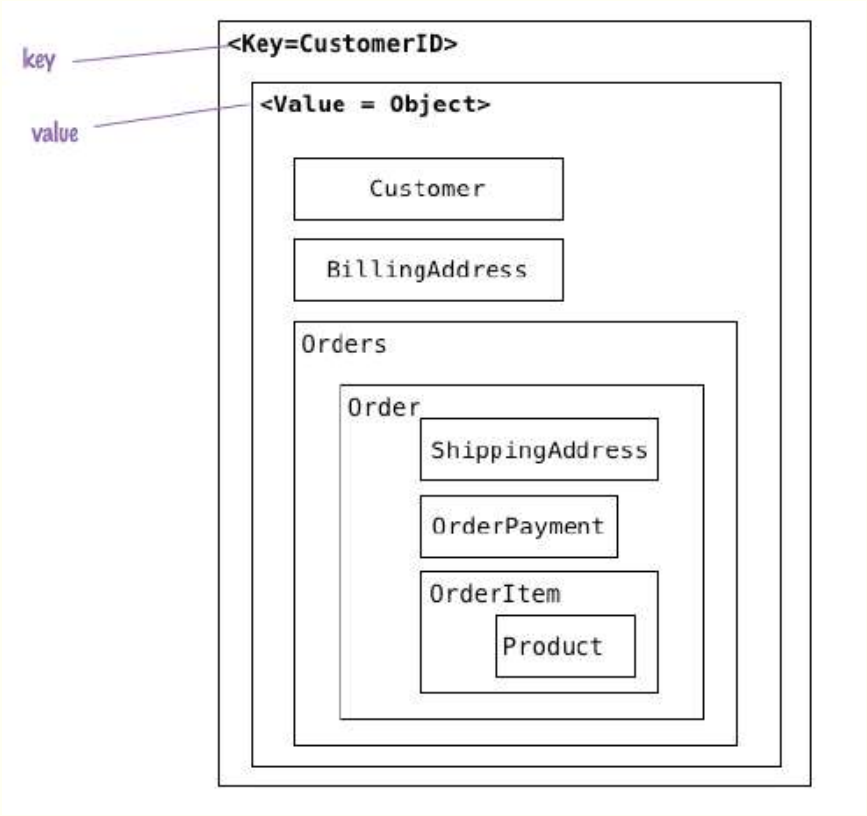
Le NoSql 2

Le monde Relationnel



Le NoSql 3

Aggregate model



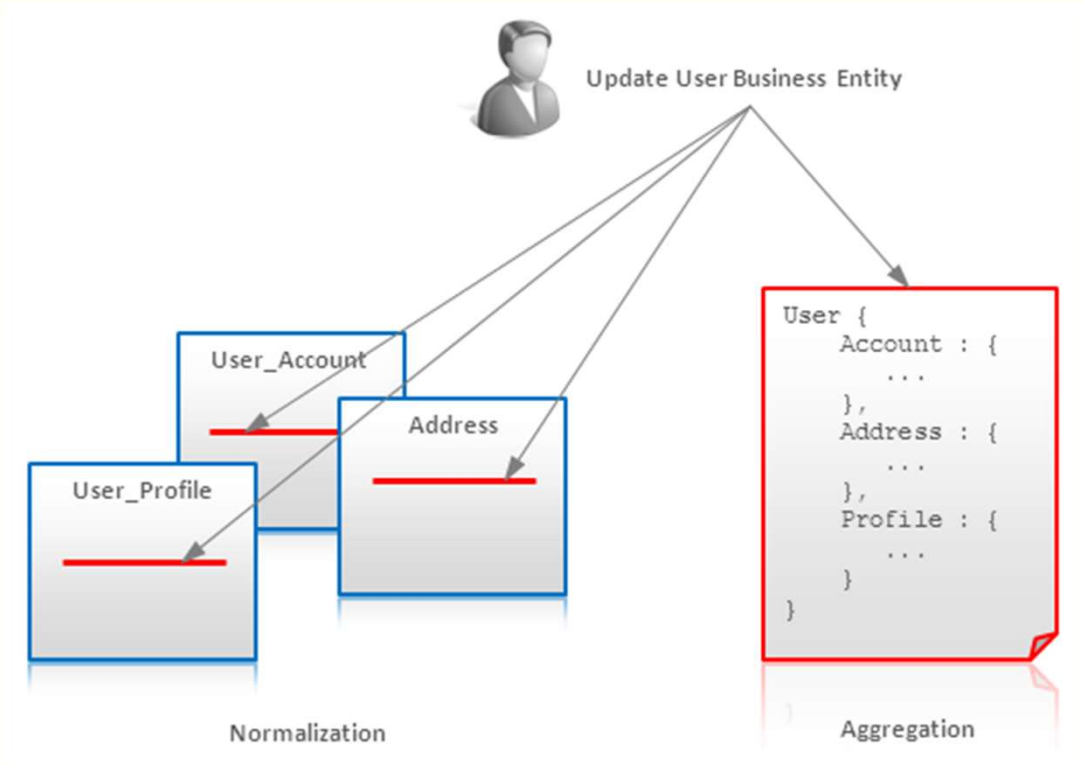
Le NoSql 4

Example

```
   // in customers
{
  "customer": {
    "id": 1,
    "name": "Martin",
    "billingAddress": [{"city": "Chicago"}],
    "orders": [
        {
          "id":99,
          "orderItems":[
            {
              "productId":27,
              "price": 32.45,
              "productName": "NoSQL Distilled"
            }
          ],
          "shippingAddress":[{"city":"Chicago"}]
          "orderPayment":[
            {
              "ccinfo":"1000-1000-1000-1000",
              "txnId":"abelif879rft",
              "billingAddress": {"city": "Chicago"}
            }
          ],
        }
    ]
  }}
```

Le NoSql 5

Normalisation vs Agrégation

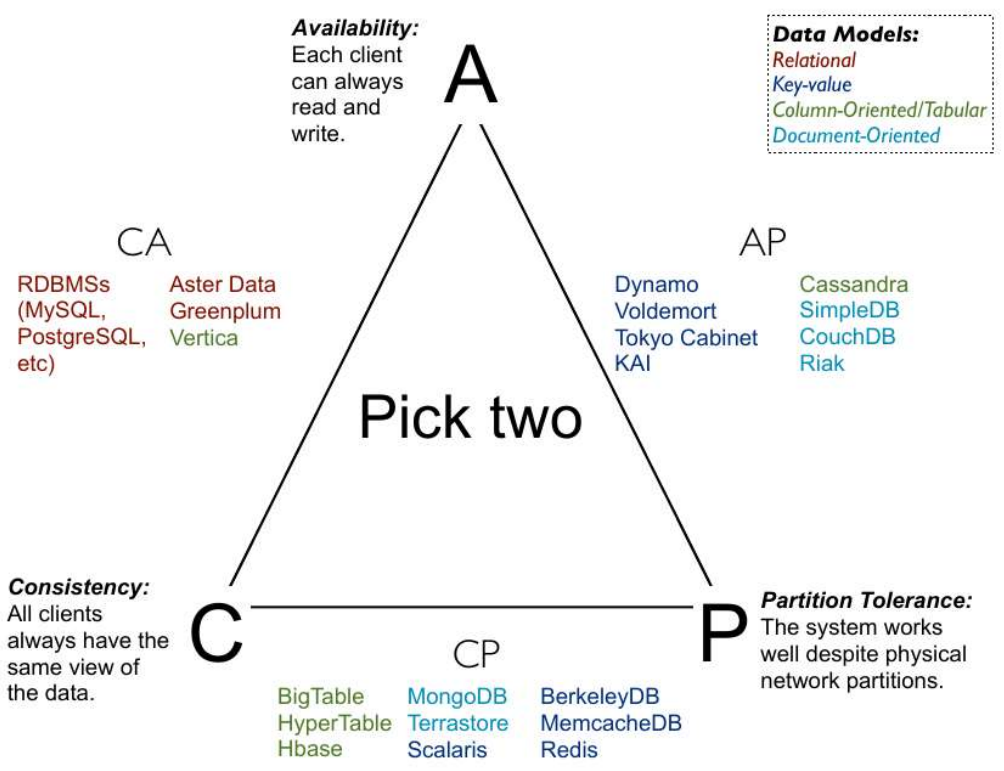


Le NoSql 6

Les bases NoSQL (comprendre Not Only SQL) répondant aux problématiques:

- Pas de relations
  - Pas de schéma physiques ou dynamiques
  - Notion de “collections”
- Données éventuellement complexes
  - Imbrication, tableaux
- Distribution de données (milliers de serveurs)
  - Parallélisation des traitements (Map/Reduce)
- Replication des données
  - Disponibilité vs Cohérence (pas de transactions)
  - Peu d’écritures, beaucoup de lectures

Le NoSql 7



Théorème CAP (proposé par Brewer, 2000 et démontré par Gilbert et Lynch 2002)

Dans un environnement distribué, il n'est pas possible de respecter simultanément :

C : Cohérence

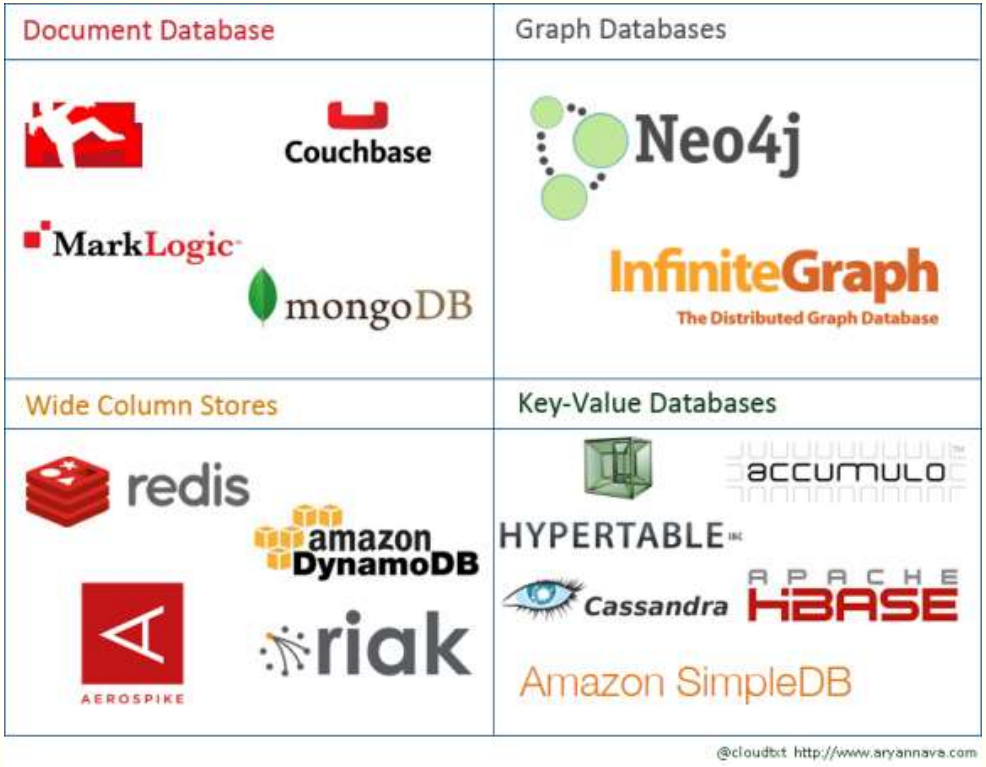
A : Disponibilité

P : Résistance au morcellement

On peut, en revanche, respecter deux contraintes.

Le NoSql 8

Les solutions NoSQL



Le NoSql 9

- **Clé/valeur**. à la manière des tableaux associatifs des langages de programmation. A une clé correspond une valeur
  - DynamoDB, Voldemort, Redis, Riak, MemcacheDB

- **Orienté colonne** : à une clé correspond un ensemble de colonne, chacune ayant une valeur.
  - HBase, Hypertable

- **Orienté document** : à une clé correspond des ensembles champs/valeurs qui peuvent être hiérarchisés
  - MongoDB, CouchDB, Terrastore, Cassandra

- **Orienté graphe** : les données sont modélisées sous forme de nœuds qui ont des liaisons entre eux.
  - Neo4j, OrientDB, FlockDB

NoSQL & Clé-Valeurs

Clé / valeur : Ce modèle peut être assimilé à une hashmap distribuée.

- Les données sont, donc, simplement représentées par un couple clé/valeur.
- La valeur peut être une simple chaîne de caractères, un objet sérialisé…
- Cette absence de structure ou de typage ont un impact important sur le requêtage.
  - En effet, toute l'intelligence portée auparavant par les requêtes SQL devra être portée par l'applicatif qui interroge la BD.
  - Néanmoins, la communication avec la BD se résumera aux opérations PUT, GET et DELETE.
- Les solutions les plus connues sont Redis, Riak et Voldemort créé par LinkedIn.

NoSQL & Orienté colonne

Orienté colonne : Ce modèle ressemble à première vue à une table dans un SGBDR à la différence qu'avec une BD NoSQL orientée colonne, le nombre de colonnes est dynamique. En effet, dans une table relationnelle, le nombre de colonnes est fixé dés la création du schéma de la table et ce nombre reste le même pour tous les enregistrements dans cette table. Par contre, avec ce modèle, le nombre de colonnes peut varier d'un enregistrement à un autre ce qui évite de retrouver des colonnes ayant des valeurs NULL. Comme solutions, on retrouve principalement HBase (implémentation Open Source du modèle BigTable publié par Google) ainsi que Cassandra (projet Apache qui respecte l'architecture distribuée de Dynamo d'Amazon et le modèle BigTable de Google).

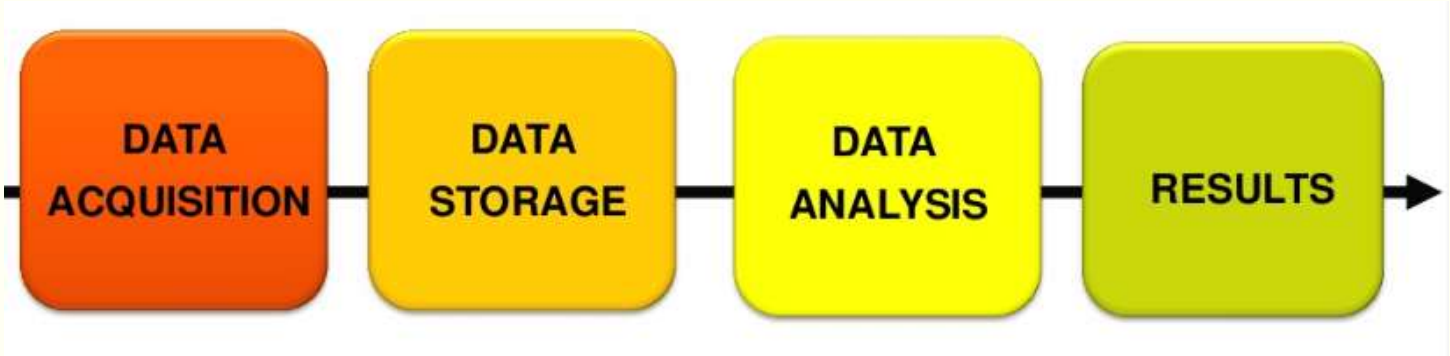
NoSQL & Orienté document

- Orienté document : Ce modèle se base sur le paradigme clé valeur. La valeur, dans ce cas, est un document de type JSON ou XML. L'avantage est de pouvoir récupérer, via une seule clé, un ensemble d'informations structurées de manière hiérarchique. La même opération dans le monde relationnel impliquerait plusieurs jointures. Pour ce modèle, les implémentations les plus populaires sont CouchDB d'Apache, RavenDB (destiné aux plateformes .NET/Windows avec la possibilité d'interrogation via LINQ) et MongoDB.

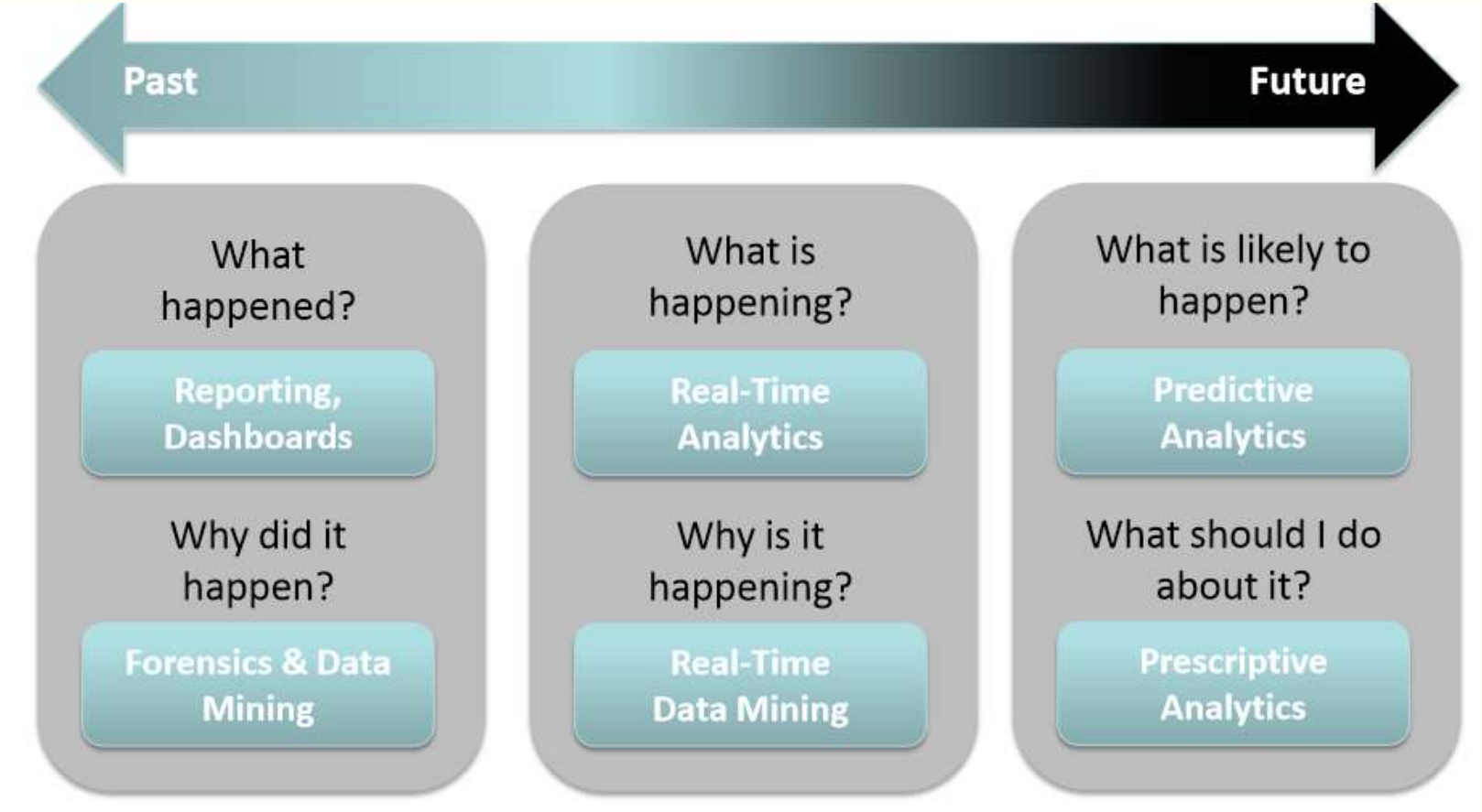
NoSQL & Orienté graphe

- Orienté graphe : Ce modèle de représentation des données se base sur la théorie des graphes. Il s'appuie sur la notion de noeuds, de relations et de propriétés qui leur sont rattachées. Ce modèle facilite la représentation du monde réel, ce qui le rend adapté au traitement des données des réseaux sociaux. La principale solution est Neo4J.

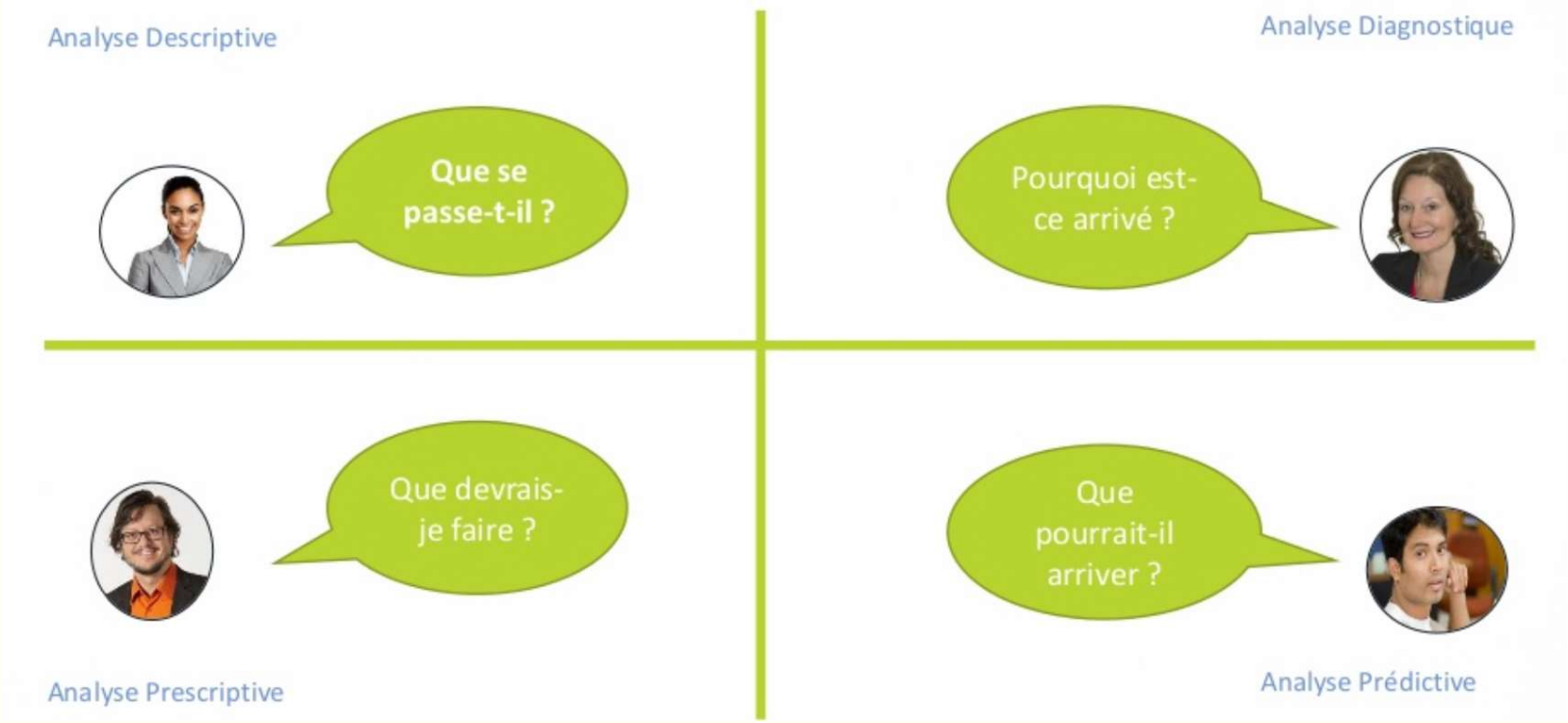
Data Processing Pipeline



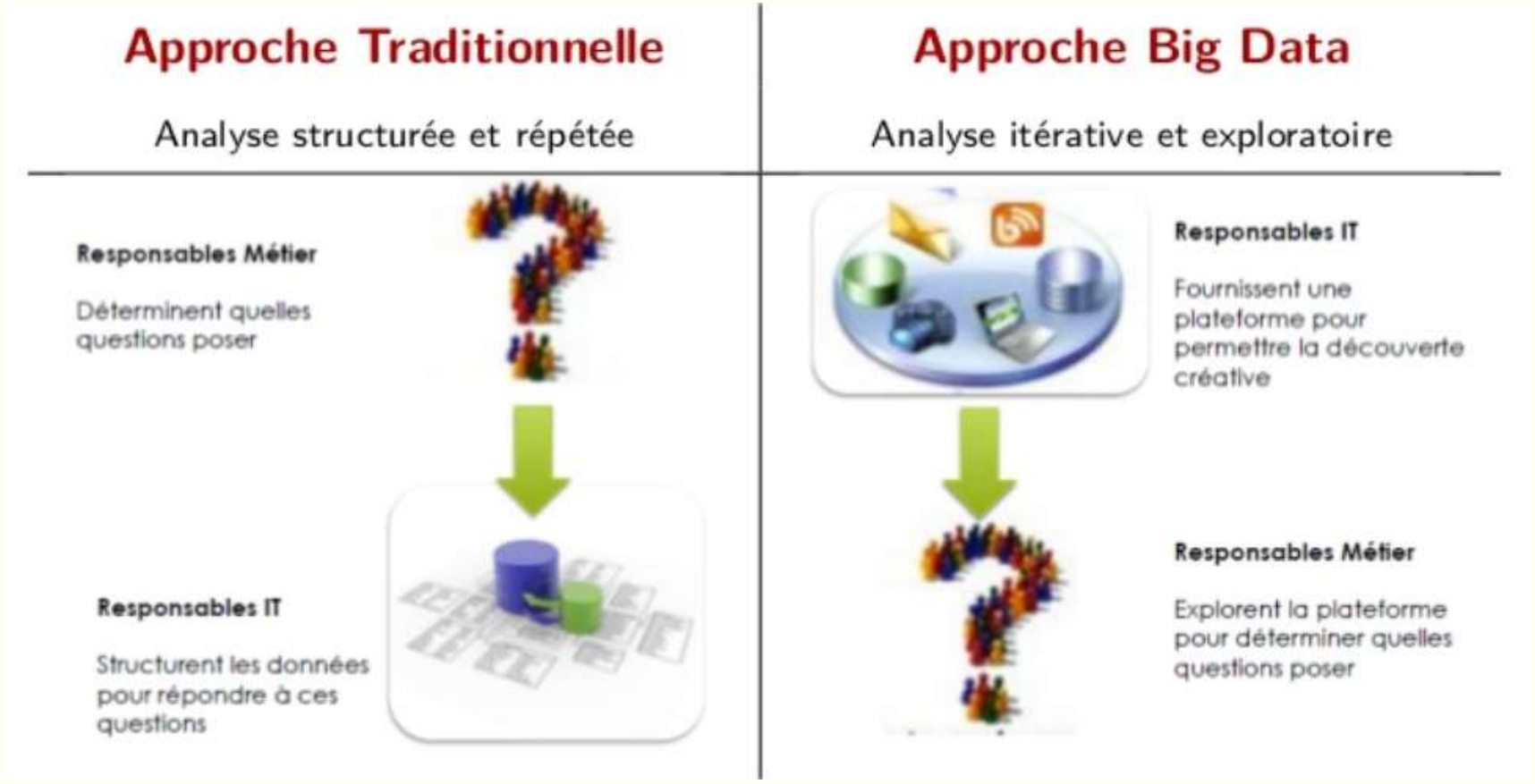
Types d'analyse 1



Types d'analyse 2



Business Intelligence VS Big Data Technology

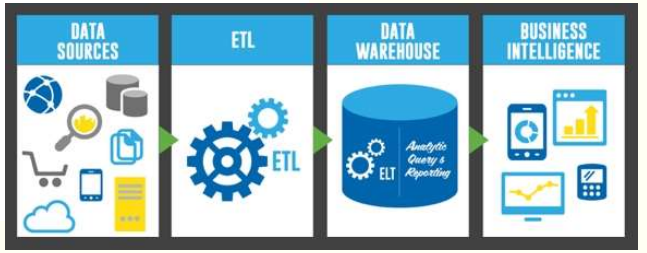


Approche traditionnelle 1

Analyse structurée et répétée

- Les besoins métiers guident la conception de la solution.
- Appropriée pour les données structurées.
- Les opérations et les processus répétitifs, les sources relativement stables et les besoins sont bien compris et cadrés.

Approche traditionnelle 2

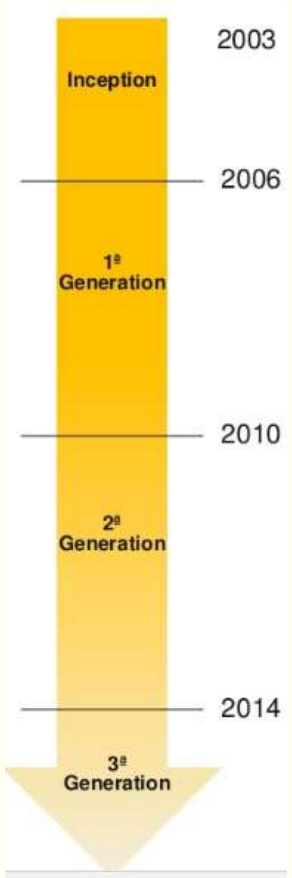


Approche Big Data

Les sources d'information guident la découverte créative.

Historique du traitement de données

- **Batch processing**
  - Large amount of statics data
  - Scalable solution
  - *Volume*
- **Real-time processing**
  - Computing streaming data
  - Low latency
  - *Velocity*
- **Hybrid computation**
  - Lambda Architecture
  - *Volume + Velocity*



Traitement par lot: Batch Processing 1

Caractéristiques

- A accès à toutes les données,
- Peut réaliser des traitements lourds et complexes,
- Est en général plus concerné par le débit (nombre d'actions réalisées en une unité de temps) que par la latence (temps requis pour réaliser l'action) des différents composants du traitement,
- Sa latence est calculée en minutes (voire plus),
- Cible les caractéristiques **volume** et **variété.**

Traitement par lot: Batch Processing 2

Inconvénients

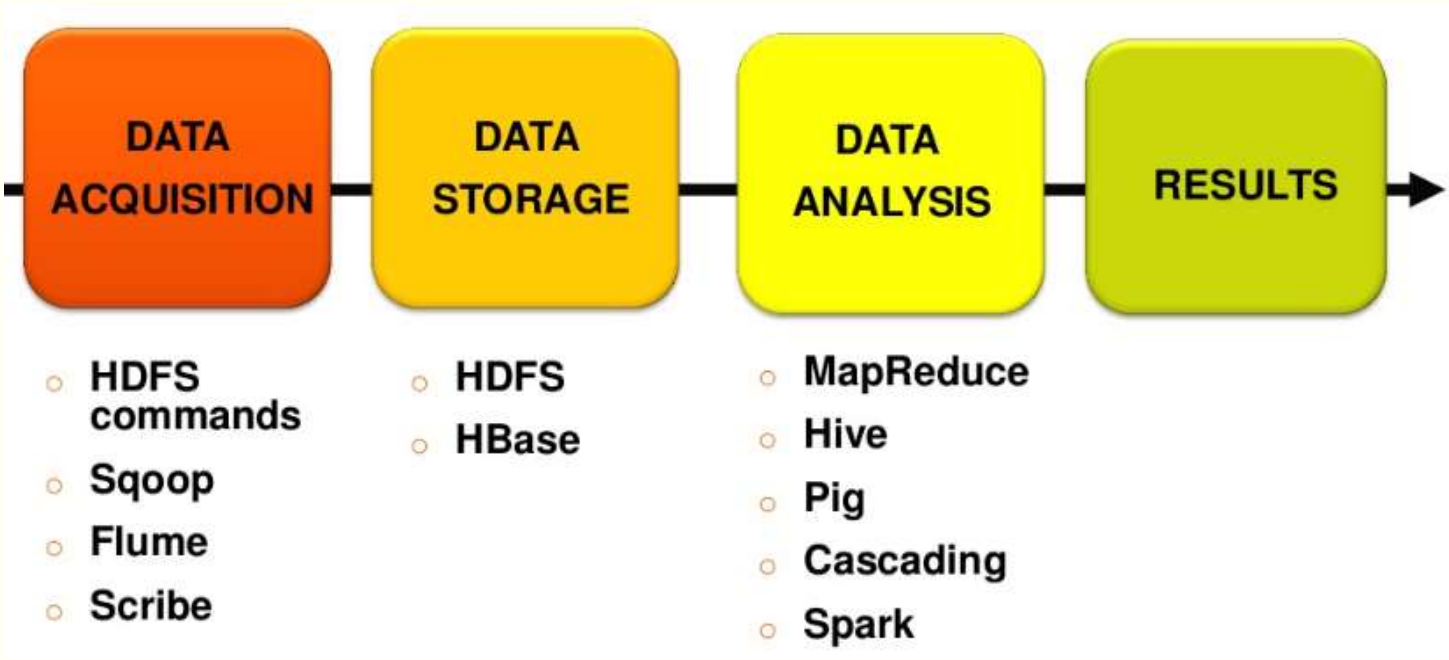
- Toutes les données doivent être prêtes avant le début du job
  - N'est pas approprié pour des traitements en ligne ou temps réel
- Latence d'exécution élevée
  - Produit des résultats sur des données relativement anciennes

Cas d'utilisation

- Les chèques de dépôt dans une banque sont accumulés et traités chaque jour
- Les statistiques par mois/jour/année
- Factures générées pour les cartes de crédit (en général mensuelles)

Traitement par lot: Batch Processing 3

Technologies



Traitement par Streaming 1

Caractéristiques

- Les traitements se font sur un élément ou un petit nombre de données récentes
- Le traitement est relativement simple
- Doit compléter chaque traitement en un temps proche du temps-réel
- Les traitements sont généralement indépendants
- Asynchrone: les sources de données n'interagissent pas directement avec l'unité de traitement en streaming, en attendant une réponse par exemple
- La latence de traitement est estimée en secondes

Traitement par Streaming 2

Inconvénients

- Pas de visibilité sur l'ensemble des données
  - Certains types de traitements ne peuvent pas être réalisés
- Complexité opérationnelle élevée
  - Plus complexes à maintenir que les traitements batch
  - Le système doit être toujours connecté, toujours prêt, avoir des temps de réponses courts, et gérer les données à l'arrivée
- Risque de perte de données

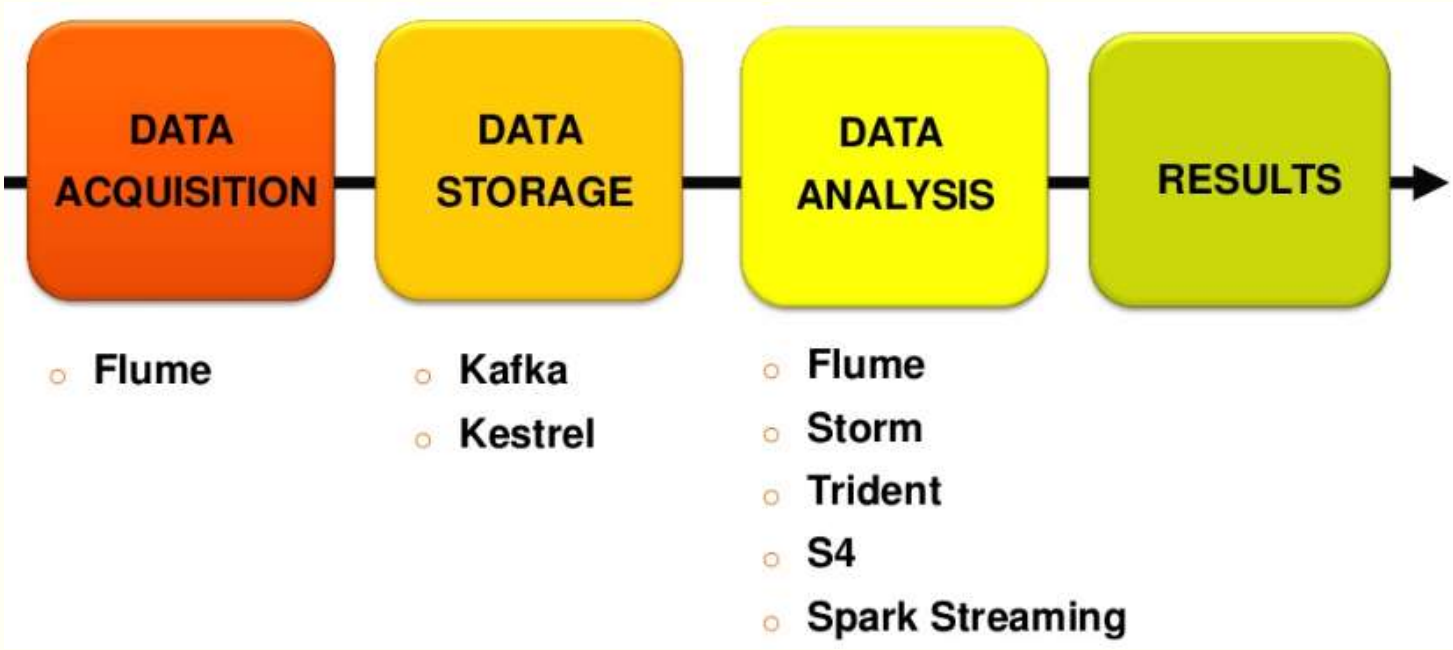
Traitement par Streaming 3

Cas d'utilisation

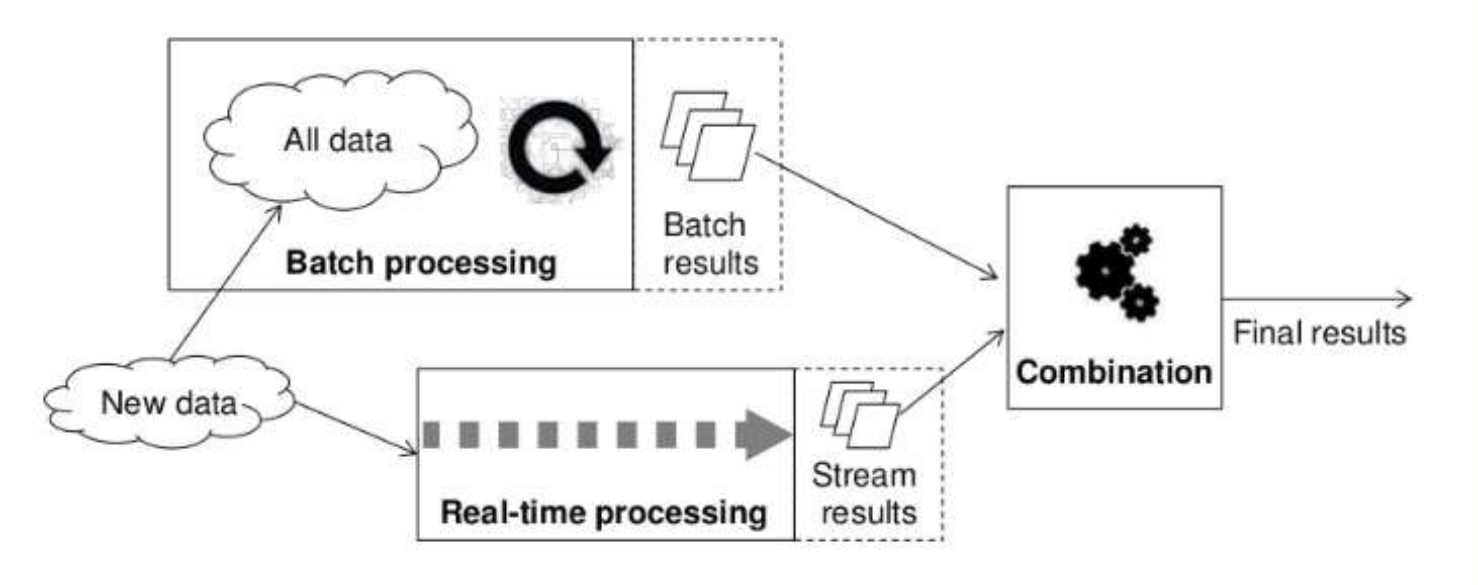
- Recommandations en temps réel
  - Prise en compte de navigation récente, géolocalisation
  - Publicité en ligne, re-marketing
- Surveillance de larges infrastructures
- Agrégation de données financières à l'échelle d'une banque
- Internet des objets
- ...

Traitement par Streaming 4

Technologies



Lambda Architecture 1



Merci,
Avez-vous des Questions ?

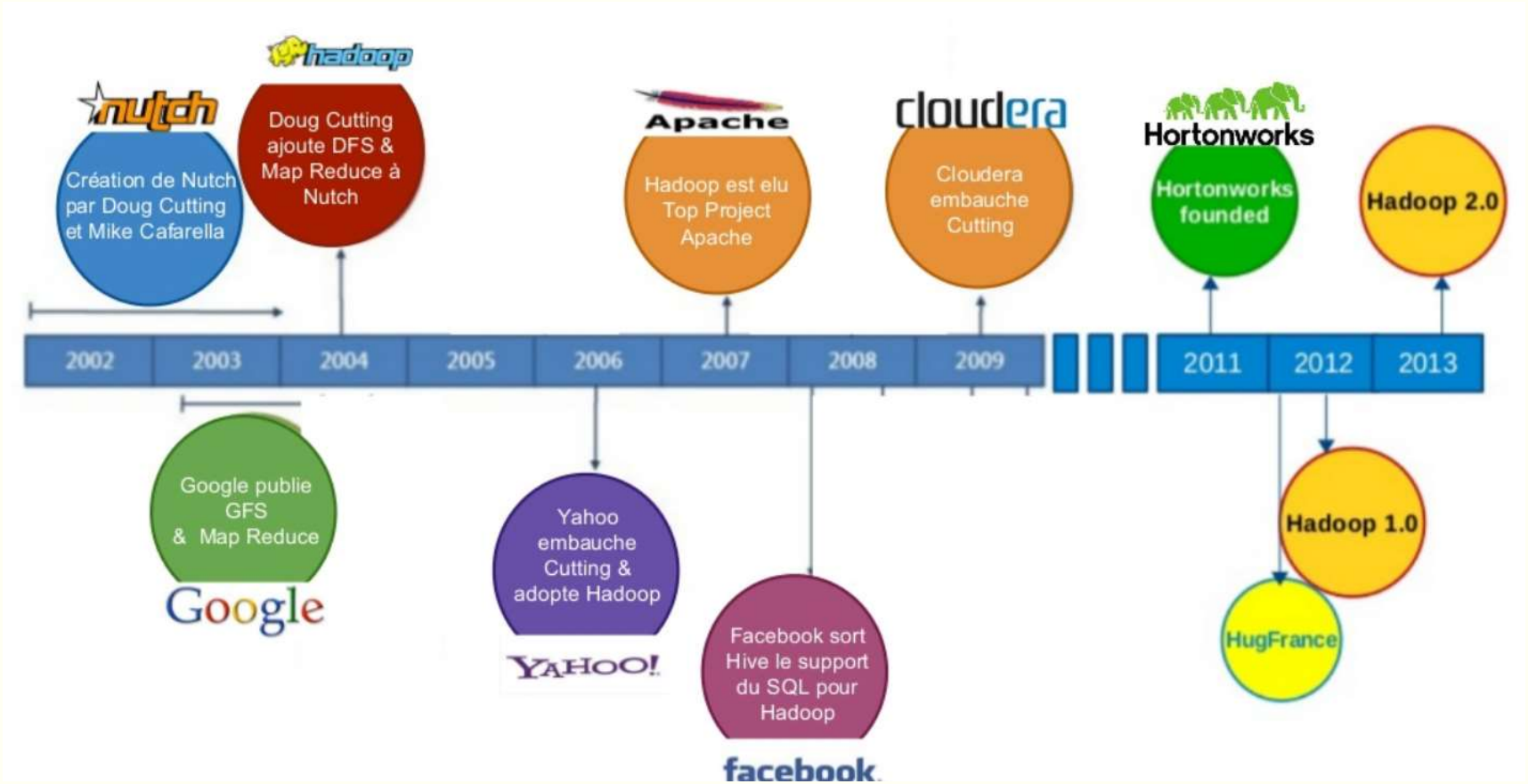
MODULE 2

Hadoop: présentation

Apache Hadoop

- Projet Open Source – fondation Apache. http://hadoop.apache.org/
- Développé en Java; portable.
- Assure l'exécution de tâches **map/reduce;** fourni le framework, en Java, pour leur développement.
- Autres langages supportés par le biais d'utilitaires internes.

Historique


nutch
Création de Nutch par Doug Cutting et Mike Cafarella
hadoop
Doug Cutting ajoute DFS & Map Reduce à Nutch
Apache
Hadoop est elu Top Project Apache
cloudera
Cloudera embauche Cutting
Hortonworks
Hortonworks founded
Hadoop 2.0
2002
2003
2004
2005
2006
2007
2008
2009
2011
2012
2013
Google publie GFS & Map Reduce
Google
Yahoo embauche Cutting & adopte Hadoop
YAHOO!
Facebook sort Hive le support du SQL pour Hadoop
facebook
Hadoop 1.0
HugFrance

Qui utilise Hadoop

Hadoop est le framework le plus utilisé actuellement pour manipuler et faire du Big Data.

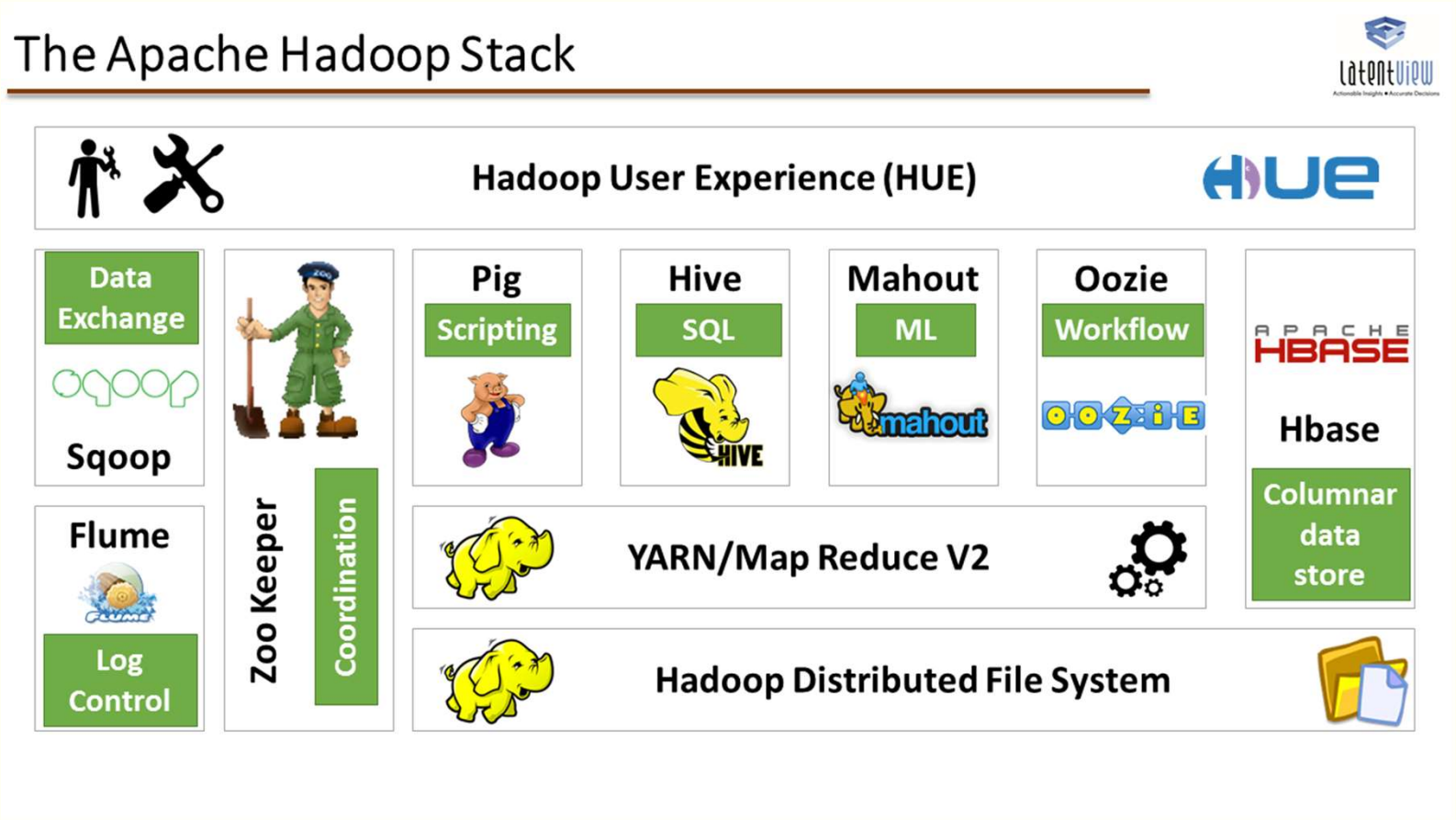
c'est un projet open source (Apache v2 licence).

Hadoop – Composants

Hadoop est constitué de plusieurs composants couvrant:

- le stockage,
- la répartition des données,
- les traitements distribués,
- l'entrepôt de données,
- le workflow,
- la programmation,
- sans oublier la coordination de l'ensemble des composants.

Hadoop – L'Ecosystème



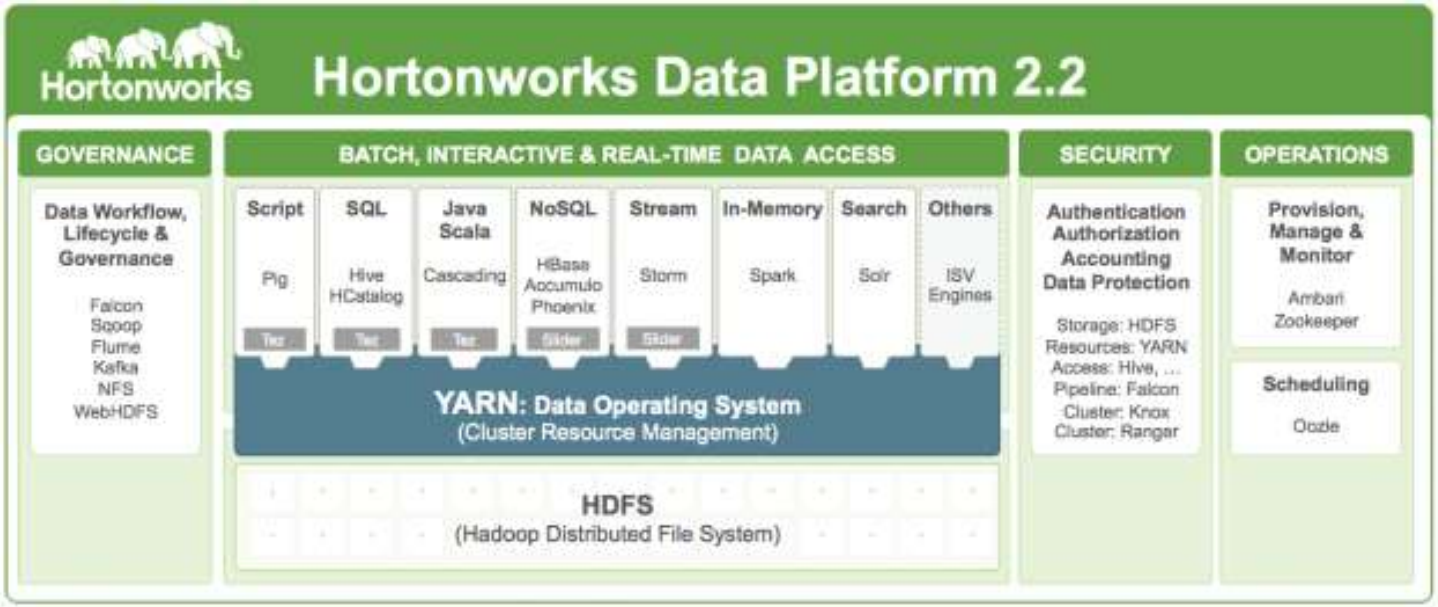
Hadoop – Les distributions 1

HortonWork : Présentation

- HortonWorks a été formé en juin 2011 par des membres de l'équipe Yahoo en charge du projet Hadoop.
- Leur but est de faciliter l'adoption de la plate forme Hadoop d'Apache, c'est pourquoi tous les composants sont open source et sous licence Apache.
- Le modèle économique d'HortonWorks est de ne pas vendre de licence mais uniquement du support et des formations.
- Cette distribution est la plus conforme à la plate forme Hadoop d'Apache et HortonWorks est un gros contributeur Hadoop.

Hadoop – Les distributions 2

HortonWork : Composants de la plate forme HD



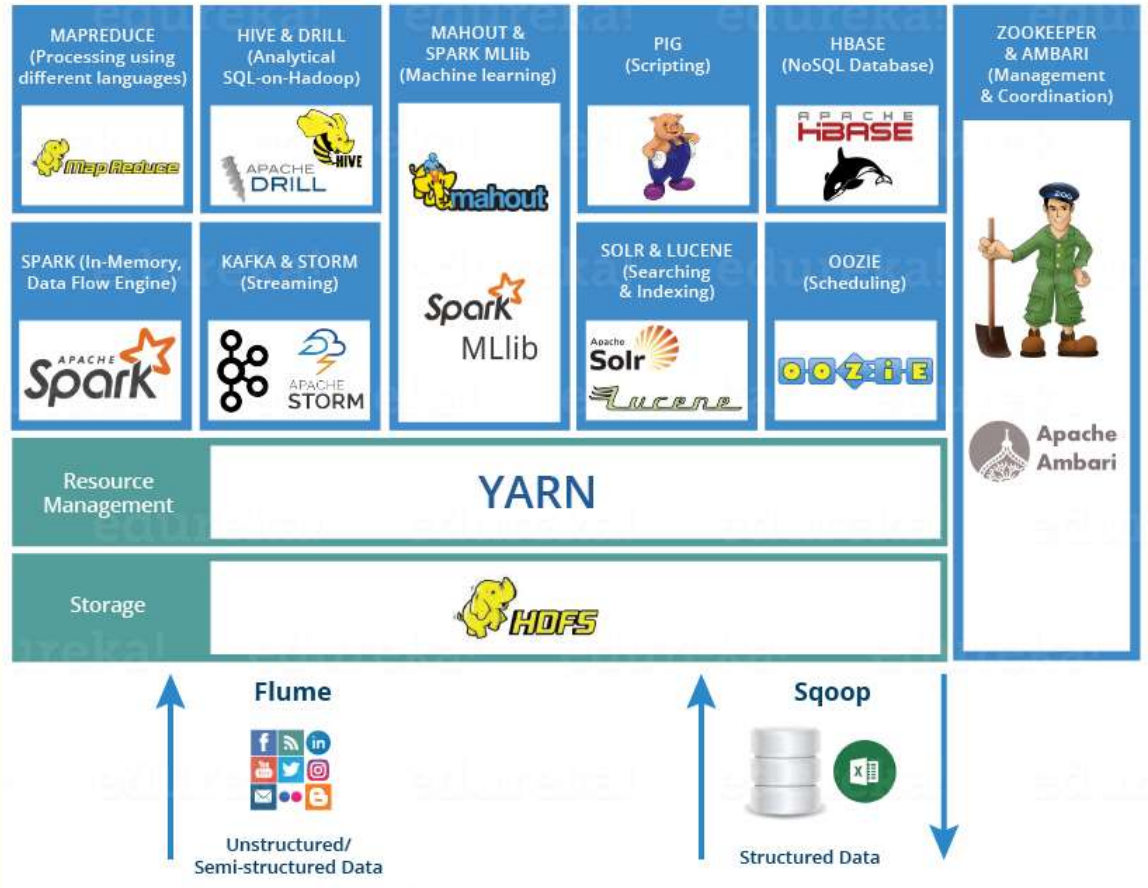
Hadoop – Les distributions 3

Cloudera: Présentation

- Fondée par des experts Hadoop en provenance de Facebook, Google, Oracle et Yahoo.
- Si leur plate forme est en grande partie basée sur Hadoop d'Apache, elle est complétée avec des composants maison essentiellement pour la gestion du cluster.
- A noter aussi que la version d'Apache Hadoop distribuée est la dernière version stable complétée de patchs critiques ainsi que de quelques fonctionnalités de la version de développement.
- Le modèle économique de Cloudera est la vente de licences mais aussi du support et des formations.
- Cloudera propose une version entièrement open source (Licence Apache 2.0).

Hadoop – Les distributions 4

Cloudera : Composants de la plate forme

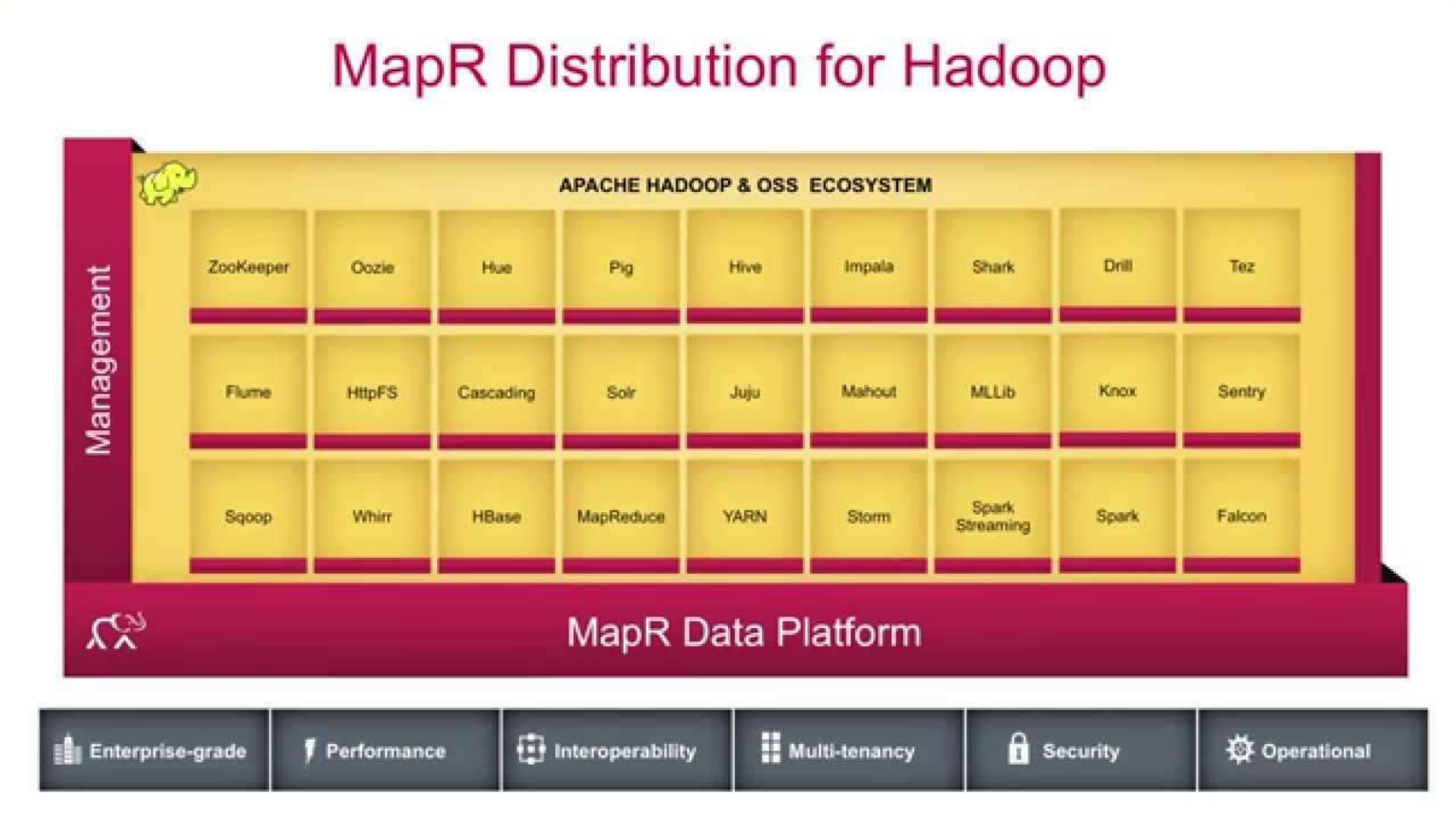


Hadoop – Les distributions 5

MapR: Présentation

- MapR a été fondée en 2009 par d'anciens membres de Google.
- Bien que son approche soit commerciale, MapR contribue à des projets Hadoop comme HBase, Pig, Hive, ZooKeeper et surtout Drill.
- MapR se distingue surtout de la version d'Apache Hadoop par sa prise de distance avec le cœur de la plate-forme. Ils proposent ainsi leur propre système de fichier distribué ainsi que leur propre version de MapReduce : MapR FS et MapR MR.
- Trois versions de leur solution sont disponibles :
  - M3 : version open source.
  - M5 : Ajoute des fonctions de haute disponibilité et du support.
  - M7 : Environnement HBase optimisé.

Hadoop – Les distributions 6



Merci,
Avez-vous des Questions ?

MODULE 3

L'architecture Hadoop

Hadoop – Core

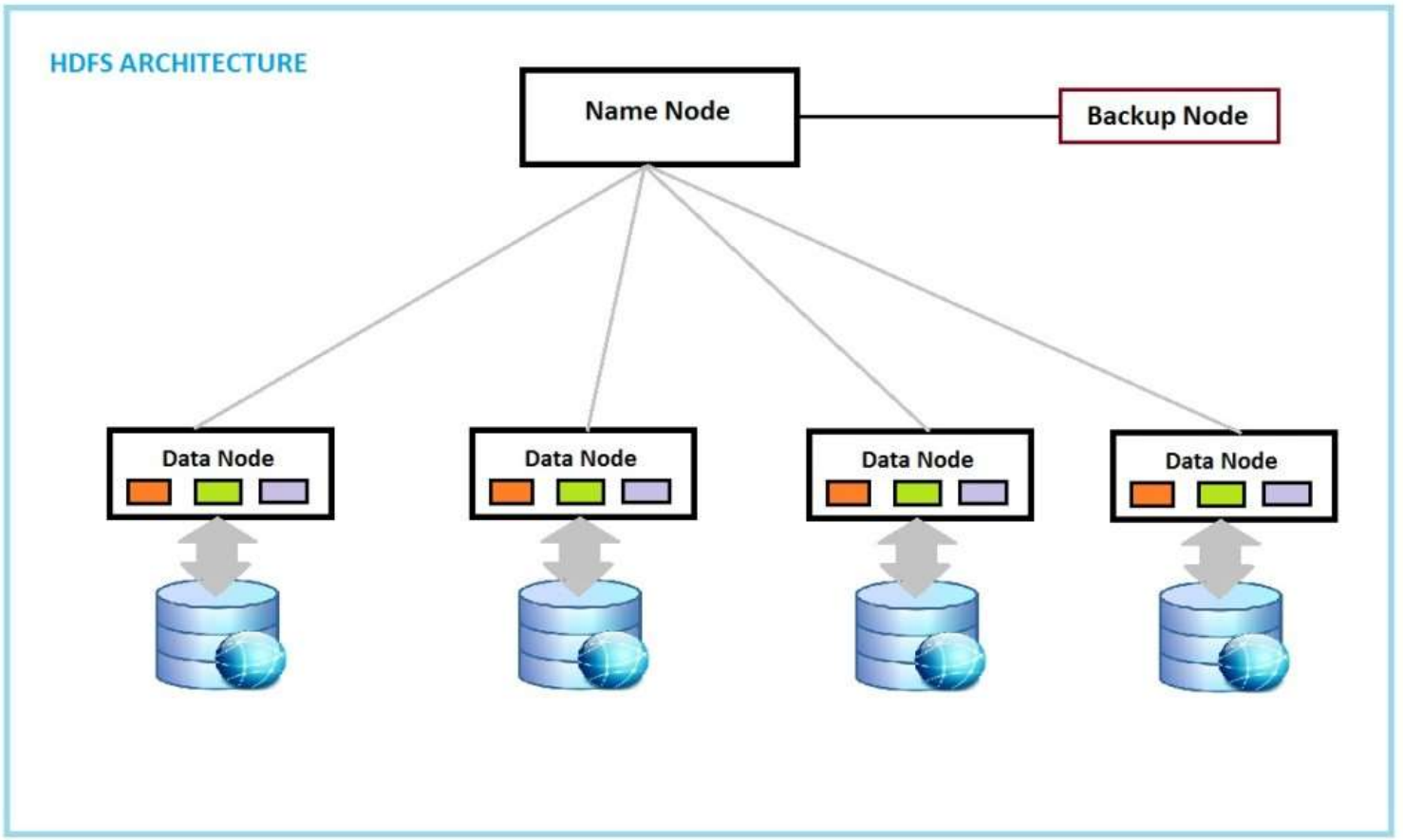
MapReduce

- Répartit la tâche entre les processeurs les plus proches des données & assemble les résultats
- Distribué entre les « nodes »

HDFS

- Système de Stockage tolérant aux pannes, organisé en cluster
- Assure automatiquement la redondance des données
- NameNode enregistre la position des données

Hadoop – HDFS


HDFS ARCHITECTURE
Name Node
Backup Node
Data Node
Data Node
Data Node
Data Node

HDFS - Présentation

- **HDFS**: Hadoop Distributed File System.

- Système de fichiers distribué associé à Hadoop. C'est là qu'on stocke données d'entrée, de sortie, etc.

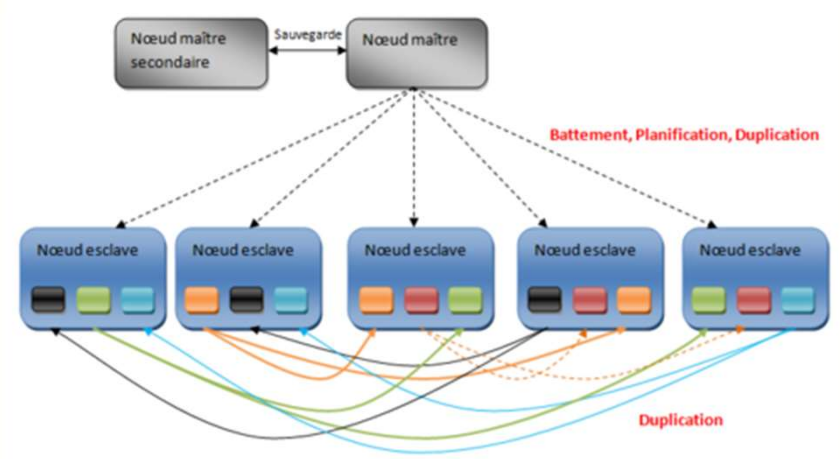
- Caractéristiques:
  - **Distribué**
  - **Tolérance aux fautes**
  - **Conscient des caractéristiques physiques de l'emplacement des serveurs (racks) pour l'optimisation.**

- HDFS utilise des tailles de blocs largement supérieures à ceux des systèmes classiques.
  - Par défaut, la taille est fixée à 64 Mo.
- HDFS fournit un système de réplication des blocs dont le nombre de réplications est configurable.

HDFS - Architecture

Repose sur :

- Le NameNode, unique sur le cluster. Stocke les informations relative aux noms de fichiers et à leurs caractéristiques de manière centralisée.
- Le Secondary NameNode (Noeud maître secondaire), vérifie périodiquement l'état du Namenode principal et copie les métadonnées. Si le Namenode principal est indisponible, le Namenode secondaire prend sa place.
- Le DataNode(Noeud de données), plusieurs par cluster. Stocke le contenu des fichiers eux-même, fragmentés en blocs (64KB par défaut).



HDFS - NameNode

- Le Namenode a une connaissance des Datanodes dans lesquels les blocs sont stockés.
- Ainsi, quand un client sollicite Hadoop pour récupérer un fichier, c'est via le Namenode que l'information est extraite.
- Ce Namenode va indiquer au client quels sont les Datanodes qui contiennent les blocs. Il ne reste plus au client qu'à récupérer les blocs souhaités.
- Toutes ces métadonnées, hormis la position des blocs dans les Datanodes, sont stockées physiquement sur le disque système dans deux types de fichiers spécifiques edits_xxx et fsimage_xxx.
- Les différents types de métadonnées :
  - liste des fichiers
  - liste des blocs pour chaque fichier
  - liste des DataNodes pour chaque bloc
  - Attributs de fichiers, dernier accès, facteur de réplication.

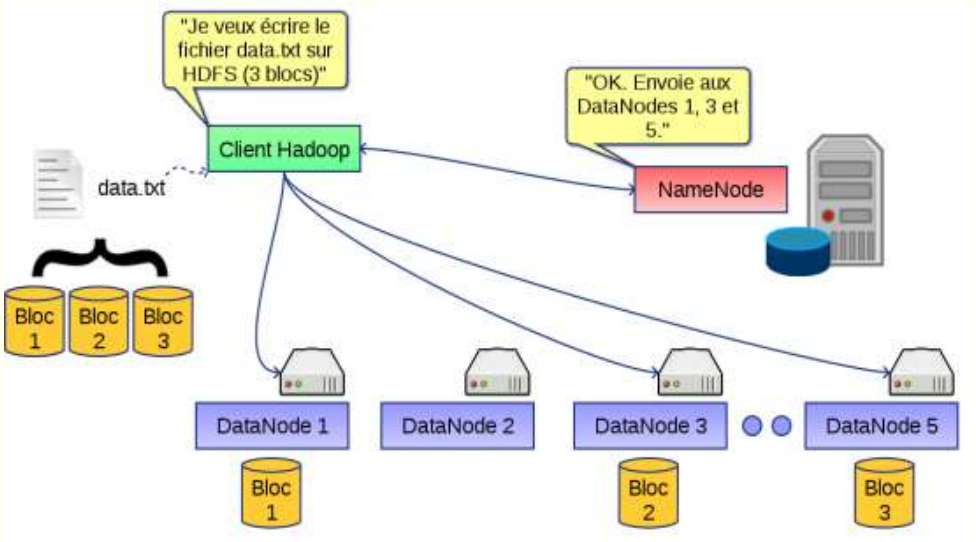
HDFS - Secondary NameNode

- Le Namenode dans l'architecture Hadoop est un point unique de défaillance (Single Point of Failure en anglais).
- Si ce service est arrêté, il n'y a pas moyen de pouvoir extraire les blocs d'un fichier donné.
- Pour répondre à cette problématique, un Namenode secondaire appelé Secondary Namenode a été mis en place dans l'architecture Hadoop.
- Son fonctionnement est relativement simple puisque le Namenode secondaire vérifie périodiquement l'état du Namenode principal et copie les métadonnées via les fichiers edits_xxx et fsimage_xxx.
- Si le Namenode principal est indisponible, le Namenode secondaire prend sa place.

HDFS - DataNode

- Un Datanode contient les blocs de données.
- Les Datanodes sont sous les ordres du Namenode et sont surnommés les Workers.
- Ils sont donc sollicités par les Namenodes lors des opérations de lecture et d'écriture.
- En lecture, les Datenodes vont transmettre au client les blocs correspondant au fichier à transmettre.
- En écriture, les Datanodes vont retourner l'emplacement des blocs fraîchement créés.
- Les Datanodes sont également sollicités lors de l'initialisation du Namenode et aussi de manière périodique, afin de retourner la liste des blocs stockés.

HDFS - Écriture d'un fichier



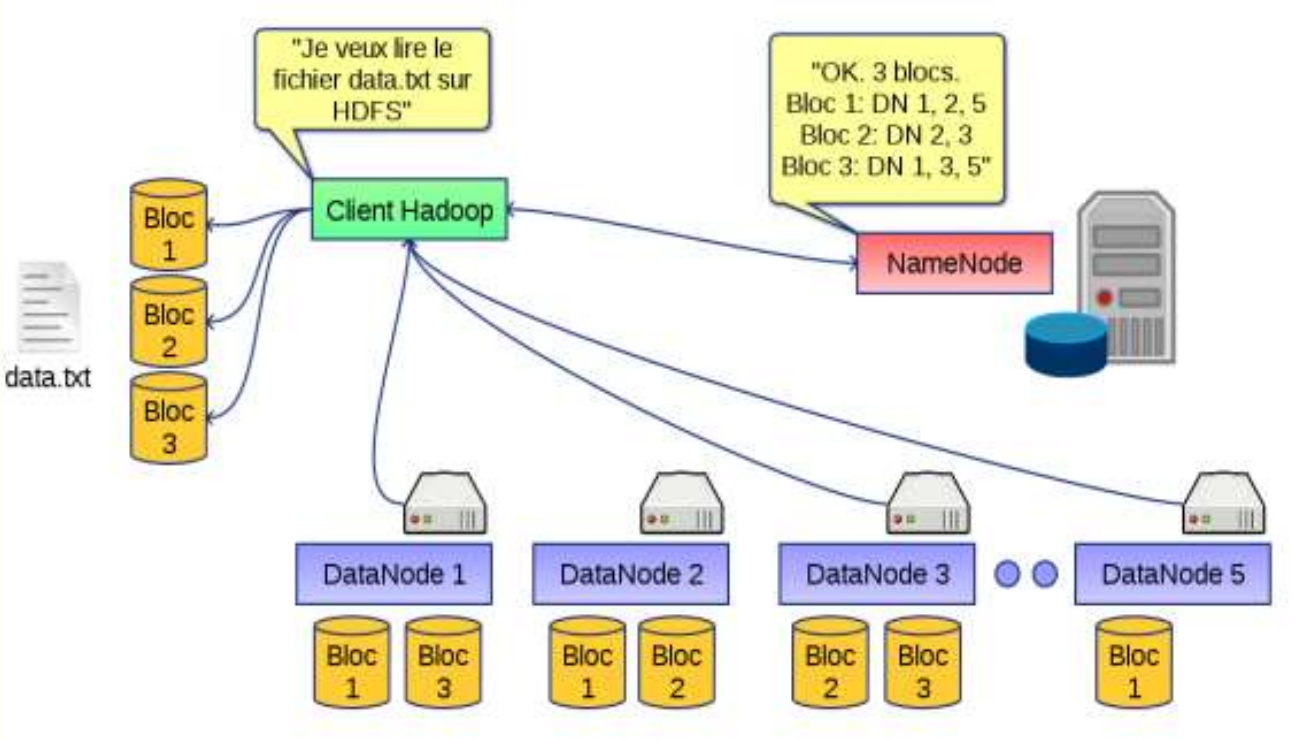
- Le client indique au NameNode qu'il souhaite écrire un bloc.
- Celui-ci lui indique le DataNode à contacter.
- Le client envoie le bloc au DataNode.
- Les DataNodes répliquent le bloc entre eux.
- Le cycle se répète pour le bloc suivant.

HDFS - Lecture d'un fichier

Pour lire un fichier:

- Le client contacte le NameNode du cluster, indiquant le fichier qu'il souhaite obtenir.
- Le NameNode lui indique la taille, en blocs, du fichier, et pour chaque bloc une liste de DataNodes susceptibles de lui fournir.
- Le client contacte les DataNodes en question pour obtenir les blocs, qu'il reconstitue sous la forme du fichier.
- En cas de DataNode inaccessible/autre erreur pour un bloc, le client contacte un DataNode alternatif de la liste pour l'obtenir.

HDFS - Lecture d'un fichier



HDFS - En pratique

- On utilise la commande console hadoop, depuis n'importe quelle machine du cluster.
- Sa syntaxe reprend celle des commandes Unix habituelles:

```
$ hadoop fs -put data.txt /input/files/data.txt
$ hadoop fs -get /input/files/data.txt data.txt
$ hadoop fs -mkdir /output
$ hadoop fs -rm /input/files/data.txt
… etc …
```

MapReduce - Architecture

Repose sur deux serveurs:

- Le JobTracker, unique sur le cluster. Reçoit les tâches map/reduce à exécuter (sous la forme d'une archive Java .jar), organise leur exécution sur le cluster.

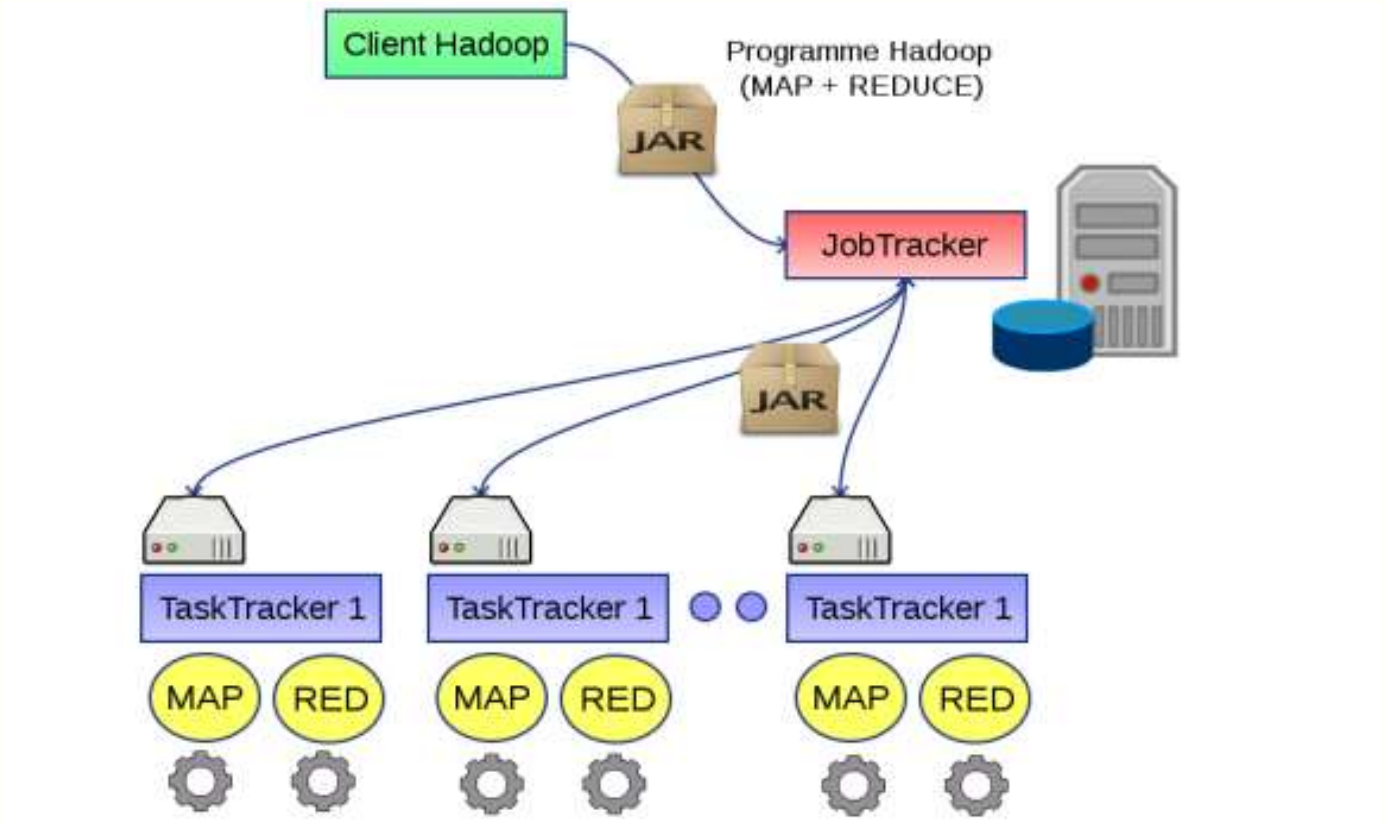
- Le TaskTracker, plusieurs par cluster. Exécute le travail map/reduce lui-même (sous la forme de tâches map et reduce ponctuelles avec les données d'entrée associées).

- Chacun des TaskTrackers constitue une unité de calcul du cluster.

MapReduce - Architecture

- Le serveur JobTracker est en communication avec HDFS; il sait où sont les données d'entrée du programme map/reduce et où doivent être stockées les données de sortie.
- Il peut ainsi optimiser la distribution des tâches selon les données associées.
- Pour exécuter un programme map/reduce, on devra donc:
  - Écrire les données d'entrée sur HDFS.
  - Soumettre le programme au JobTracker du cluster.
  - Récupérer les données de sortie depuis HDFS.

MapReduce - Architecture



MapReduce - Exécution d'une tâche

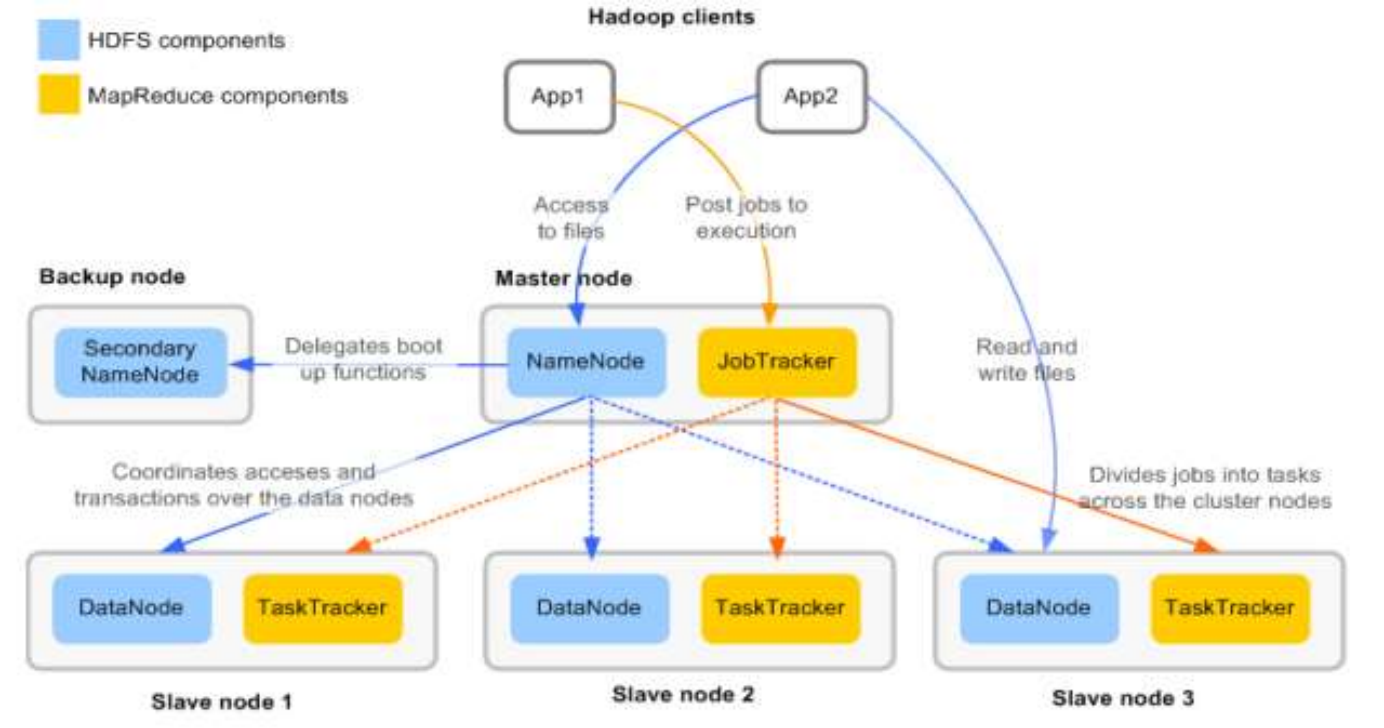
- Tous les TaskTrackers signalent leur statut continuellement par le biais de paquets heartbeat.
- En cas de défaillance d'un TaskTracker (heartbeat manquant ou tâche échouée), le JobTracker avise en conséquence: redistribution de la tâche à un autre nœud, etc.
- Au cours de l'exécution d'une tâche, on peut obtenir des statistiques détaillées sur son évolution (étape actuelle, avancement, temps estimé avant completion, etc.), toujours par le biais du client console hadoop.

MapReduce - Autres considérations

- Un seul JobTracker sur le serveur: point de défaillance unique. Là aussi, compensé par des architectures serveurs adaptées.

- Les deux serveurs « uniques » NameNode et JobTracker sont souvent actifs au sein d'une seule et même machine: le nœud maître du cluster.
- Tout changement dans la configuration du cluster est répliqué depuis le nœud maître sur l'intégralité du cluster.

Architecture générale

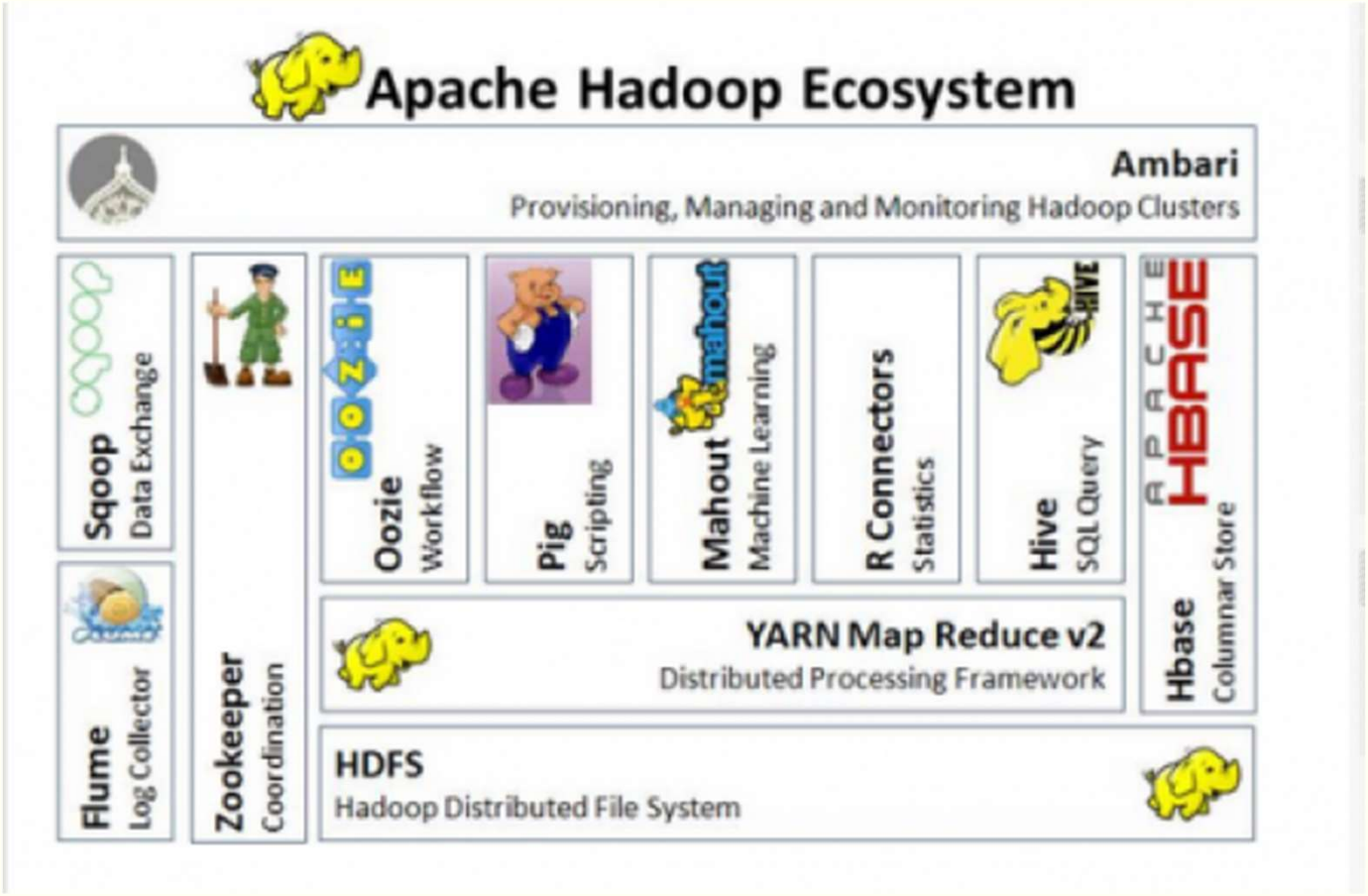


Merci,
Avez-vous des Questions ?

MODULE 4

L’Ecosystème

Hadoop – L'Ecosystème



Hadoop – L'Ecosystème

Hadoop est constitué de plusieurs composants couvrant :

- le stockage,
- la répartition des données,
- les traitements distribués,
- l'entrepôt de données,
- le workflow,
- la programmation,
- sans oublier la coordination de l'ensemble des composants.

Hadoop – L'Ecosystème

HDFS (Hadoop Distributed File System) est le système de stockage primaire utilisé par les applications Hadoop. HDFS permet de gérer la réplication de multiples blocs de données et leur distribution sur les nœuds de calcul à travers un cluster pour permettre des calculs fiables et extrêmement rapides.

MapReduce est une plate-forme de programmation conçue pour écrire des applications permettant le traitement rapide et parallélisé de vastes quantités de données réparties sur plusieurs clusters de nœuds de calcul.

Hadoop – L'Ecosystème

- Hbase (Apache) est un système de gestion de base de données non relationnelle, distribuée.
- HBase est un sous-projet d'Hadoop.
- HBase est inspirée des publications de Google sur BigTable. Comme BigTable, c'est une base de données orientée colonnes.
- HBase est souvent utilisé conjointement au système de fichiers HDFS, ce dernier facilitant la distribution des données de HBase sur plusieurs noeuds.
- Contrairement à HDFS, HBase permet de gérer les accès aléatoires read/write pour des applications de type temps réel.

Hadoop – L'Ecosystème

Cassandra(Facebook)

Cassandra est une base de données orientée colonnes développée sous l'impulsion de Facebook.

Cassandra supporte l'exécution de jobs MapReduce qui peuvent y puiser les données en entrée et y stocker les résultats en retour (ou bien dans un système de fichiers).

Cassandra comparativement à HBase est meilleur pour les écritures alors que ce dernier est plus performant pour les lectures.

Hadoop – L'Ecosystème

Hive(Facebook)

Hive est à l'origine un projet Facebook qui permet de faire le lien entre le monde SQL et Hadoop.

Il permet l'exécution de requêtes SQL sur un cluster Hadoop en vue d'analyser et d'agréger les données.

Le langage SQL est nommé HiveQL. C'est un langage de visualisation uniquement, c'est pourquoi seules les instructions de type "Select" sont supportées pour la manipulation des données.

Dans certains cas, les développeurs doivent faire le mapping entre les structures de données et Hive.

Hive utilise un connecteur jdbc/odbc.

Hadoop – L'Ecosystème

Pig (Yahoo) : Scripting sur les données

Pig est à l'origine un projet Yahoo qui permet le requêtage des données Hadoop à partir d'un langage de script.

Contrairement à Hive, Pig est basé sur un langage de haut niveau PigLatin qui permet de créer des programmes de type MapReduce.

Contrairement à Hive, Pig ne dispose pas d'interface web

Hadoop – L'Ecosystème

Sqoop(Cloudera) : Intégration SGBD-R

Sqoop permet le transfert des données entre un cluster Hadoop et des bases de données relationnelles.

C'est un produit développé par Cloudera.

Il permet d'importer/exporter des données depuis/vers Hadoop et Hive.

Pour la manipulation des données Sqoop utilise MapReduce et des drivers JDBC.

Hadoop – L'Ecosystème

Apache Oozie (Yahoo) : Ordonnanceur

Oozie est une solution de workflow (au sens scheduler d'exploitation) utilisée pour gérer et coordonner les tâches de traitement de données à destination de Hadoop.

Oozie s'intègre parfaitement avec l'écosystème Hadoop puisqu'il supporte les types de jobs suivant :

- MapReduce (Java et Streaming),
- Pig,
- Hive,
- Sqoop,

Autres tels que programmes Java ou scripts de type Shell.

Hadoop – L'Ecosystème

Flume(Cloudera) : Log

Flume est une solution de collecte et d'agrégation de fichiers logs, destinés à être stockés et traités par Hadoop.

Il a été conçu pour s'interfacer directement avec HDFS au travers d'une API native.

Flume est à l'origine un projet Cloudera, reversé depuis à la fondation Apache.

Alternatives : Apache Chukwa.

Hadoop – L'Ecosystème

Apache ZooKeeper : Clustering

ZooKeeper est un service de coordination des services d'un cluster Hadoop.

En particulier, le rôle de ZooKeeper est de fournir aux composants Hadoop les fonctionnalités de distribution.

Pour cela il centralise les éléments de configuration du cluster Hadoop, propose des services de clusterisation et gère la synchronisation des différents éléments (événements).

ZooKeeper est un élément indispensable au bon fonctionnement de HBase.

Hadoop – L'Ecosystème

Apache Ambari (HortonWorks): Supervision

- Ambari est un projet d'incubation Apache initié par HortonWorks et destiné à la supervision et à l'administration de clusters Hadoop.
- C'est un outil web qui propose un tableau de bord. Cela permet de visualiser rapidement l'état d'un cluster.
- Ambari dispose d'un tableau de bord dont le rôle est de fournir une représentation :
  - De l'état des services.
  - De la configuration du cluster et des services.
  - Des informations issues de Ganglia et de Nagios.
  - De l'exécution des jobs.
  - Des métriques de chaque machine et du cluster.

Merci,
Avez-vous des Questions ?

MODULE 5

Demos

Demos

- Installation de Cloudera dans VMWare,
- Demo HDFS,
- Demo MapReduce,
- Demo Sqoop,
- Demo Hive,
- Demo Pig,
- Demo Flume,
- Demo Hbase.

Merci,
Avez-vous des Questions ?