BigData : concepts et technologies

Bruno DEFUDE
Telecom SudParis

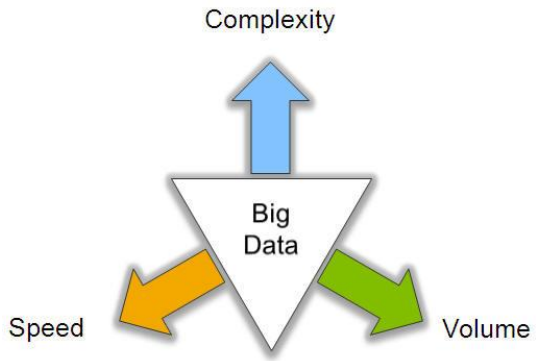
Programme

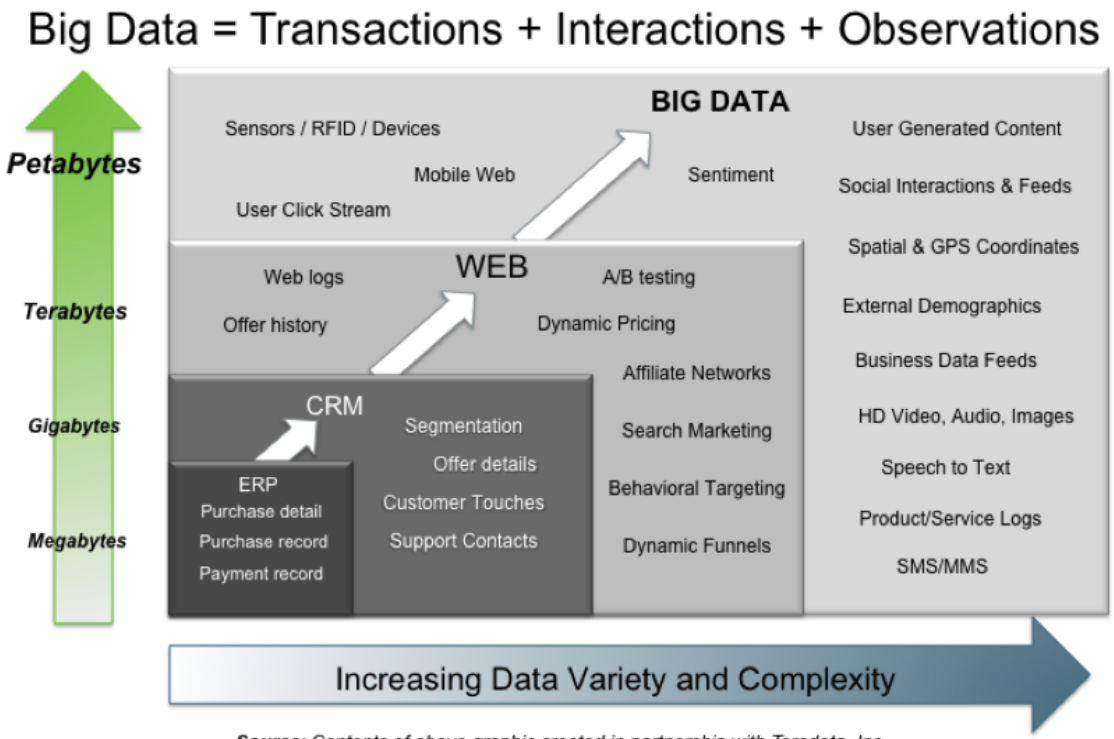
- **Introduction et contexte**
- **Entreprise data-driven**
- **Des données aux big data**
  - Bases de données NoSQL
  - Entrepôts de données
  - Data lake
- **De Hadoop à Spark**
  - Typologie des outils big data
  - Mapreduce et Hadoop
  - spark

Programme

- **Introduction et contexte**
- **Entreprise data-driven**
- **Des données aux big data**
  - Bases de données NoSQL
  - Entrepôts de données
  - Data lake
- **De Hadoop à Spark**
  - Typologie des outils big data
  - Mapreduce et Hadoop
  - spark

Big Data : Les 3 V

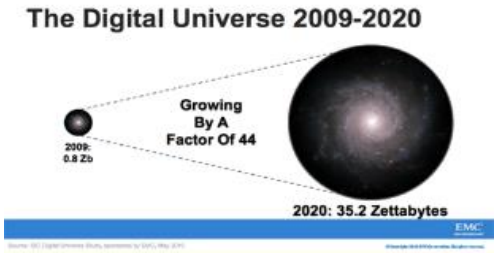


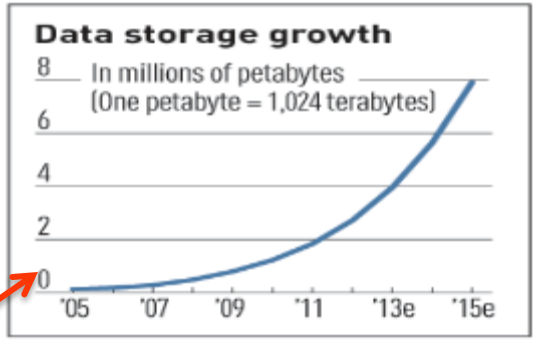


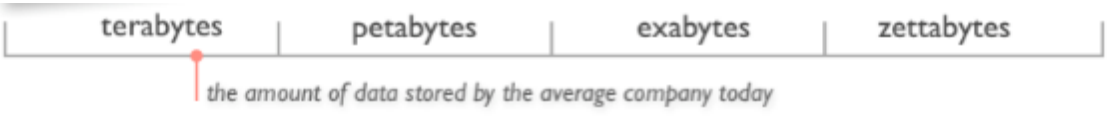
***Source**: Contents of above graphic created in partnership with Teradata, Inc.*

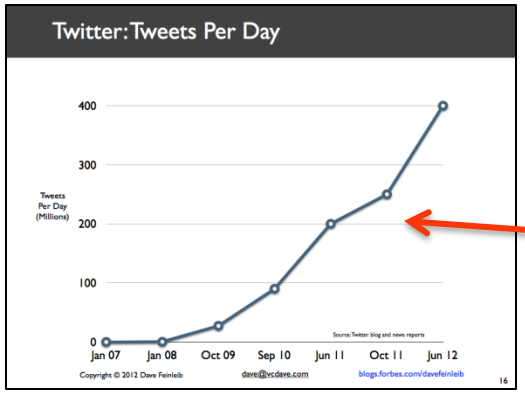
Caractéristiques du Big Data : 1-Echelle (Volume)

- **Volume des données**
  - Augmentation de 44x entre 2009 et 2020
  - De 0.8 zettabytes à 35zb
- **Augmentation exponentielle du volume**



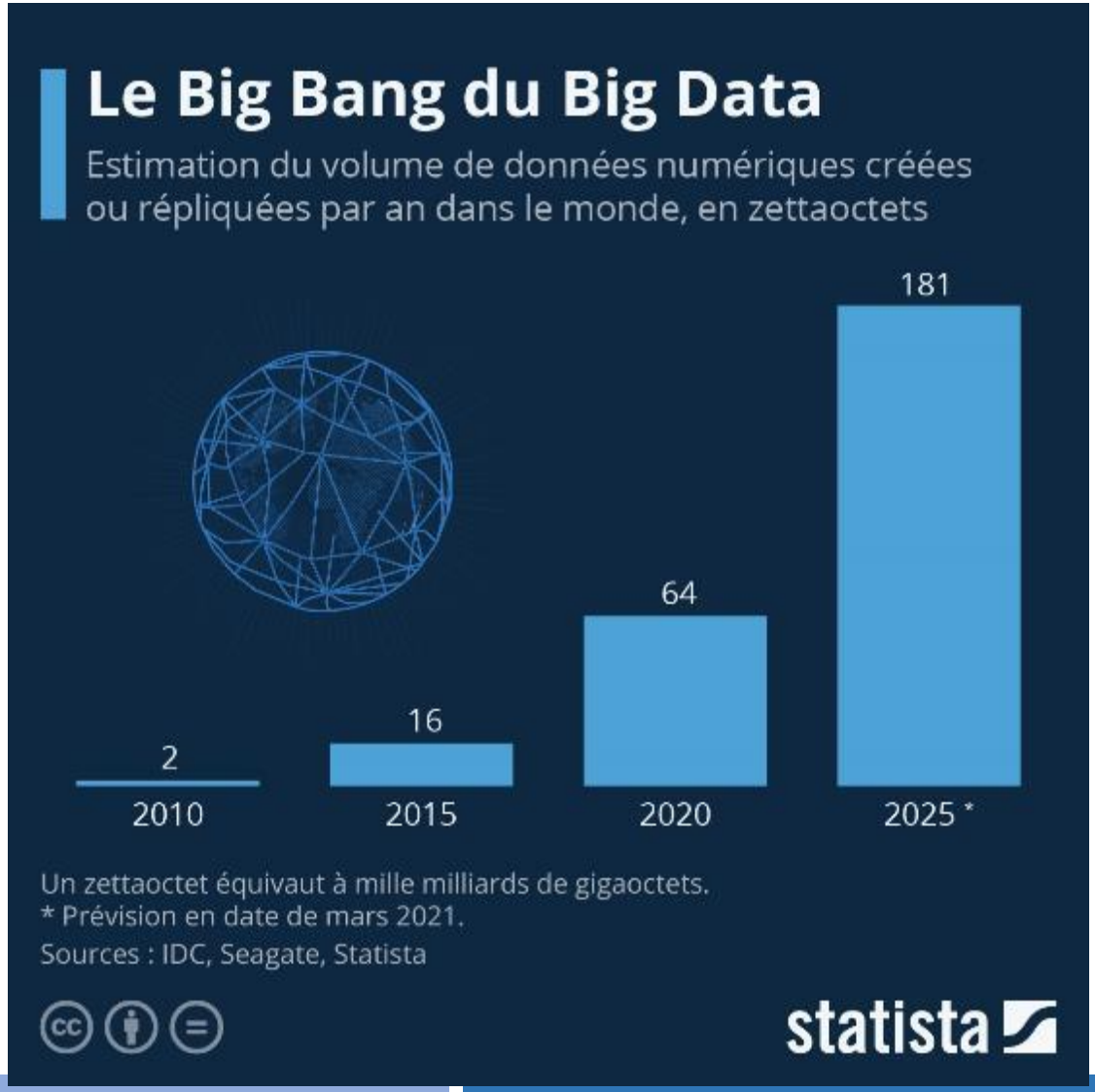






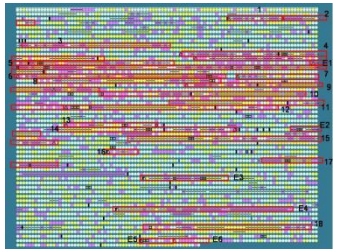
Augmentation exponentielle en données collectées/générées

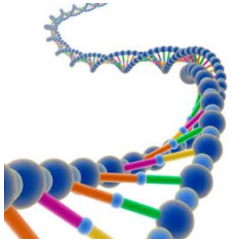
Des chiffres ...

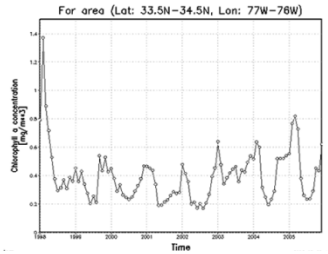


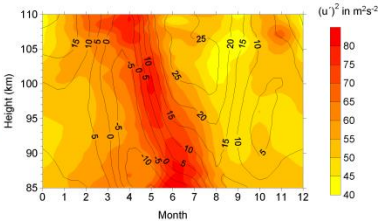
Caractéristiques du Big Data : 2-Complexité (Variété)

- **Des formats, types et structures variées**
- **Textes, nombres, images, audio, vidéo, séquences, séries temporelles, données réseaux sociaux, données multidimensionnelles etc…**
- **Données statiques vs. flux de données (streaming)**
- **Une même application peut collecter/générer de nombreux types de données**









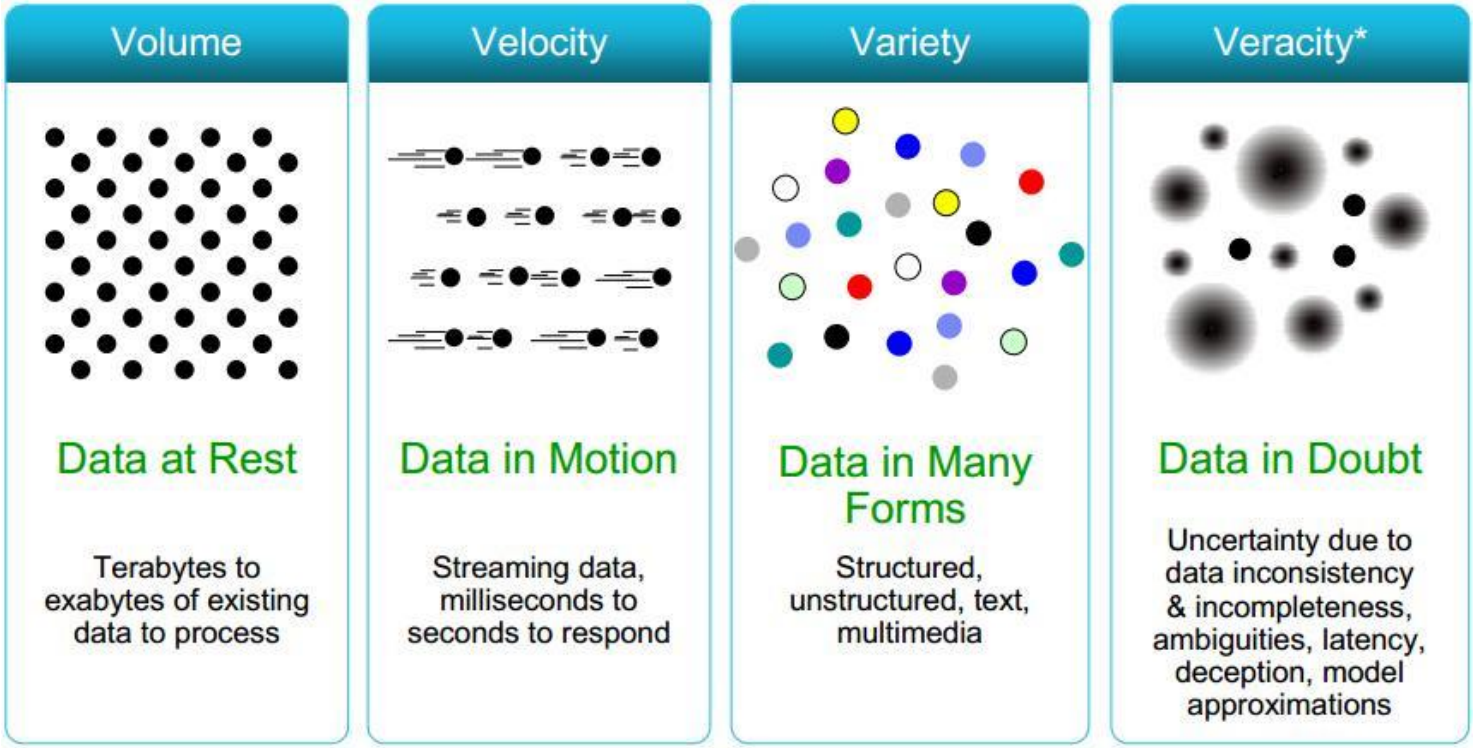
Pour extraire de la connaissance➔ tous ces types de données à relier ensemble

Caractéristiques du Big Data :
3- Vitesse (Vélocité)

- **Les données sont générées à la volée avec une grande fréquence et doivent être traitées rapidement**
- **Analyse à la volée (online)**
- **Décisions tardives ➔ pertes d'opportunitées**
- **Exemples**
  - **E-Promotions:** Basé sur votre localisation, votre historique d'achats, ce que vous aimez ➔ envoi de promotions pour les magasins proches de vous
  - **Surveillance de la santé :** des capteurs surveillent vos activités et signaux vitaux ➔ toute mesure anormale requiert une réaction immédiate

Certains rajoutent d'autres V

V : véracité, est ce que les données ont du sens ?
V : validité, est ce que les données sont cohérentes et précises ?
V : volatilité, en combien de temps dois je traiter les données ?

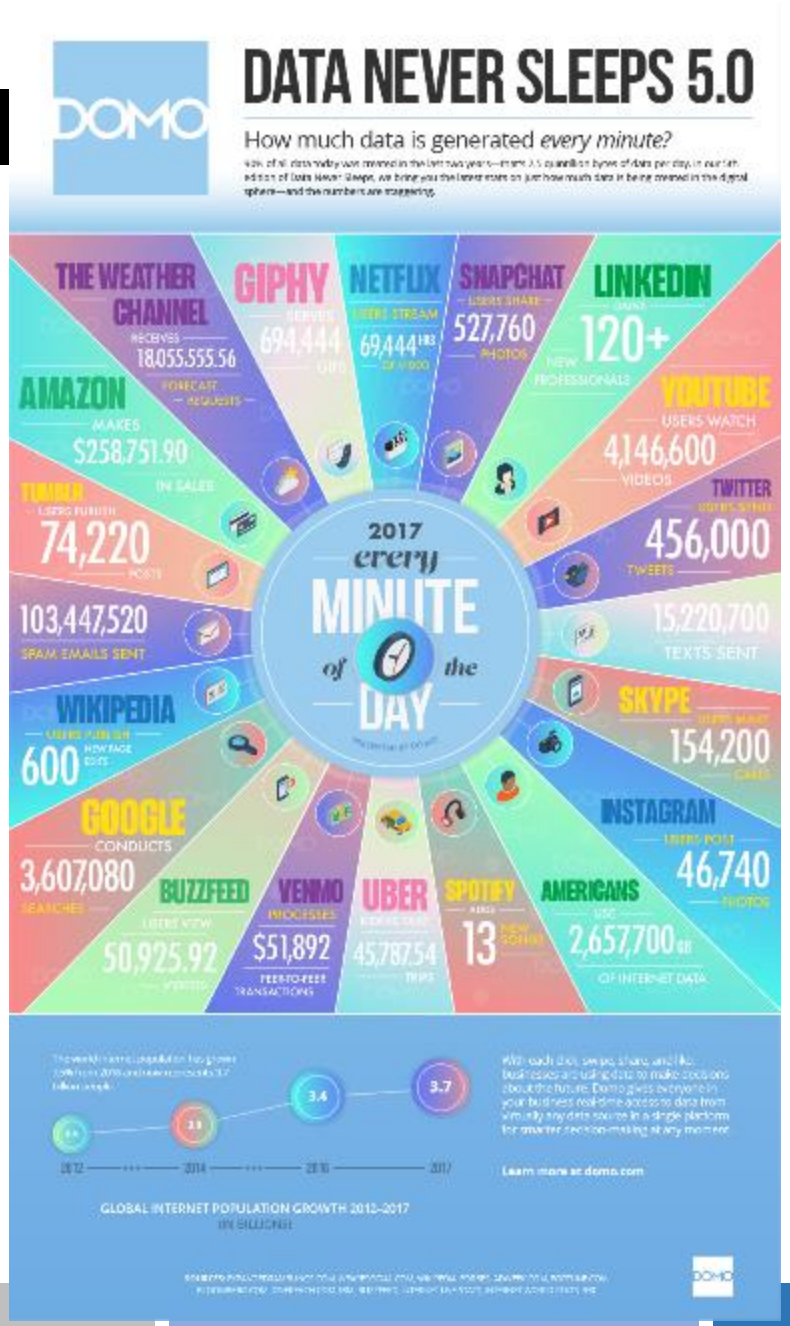


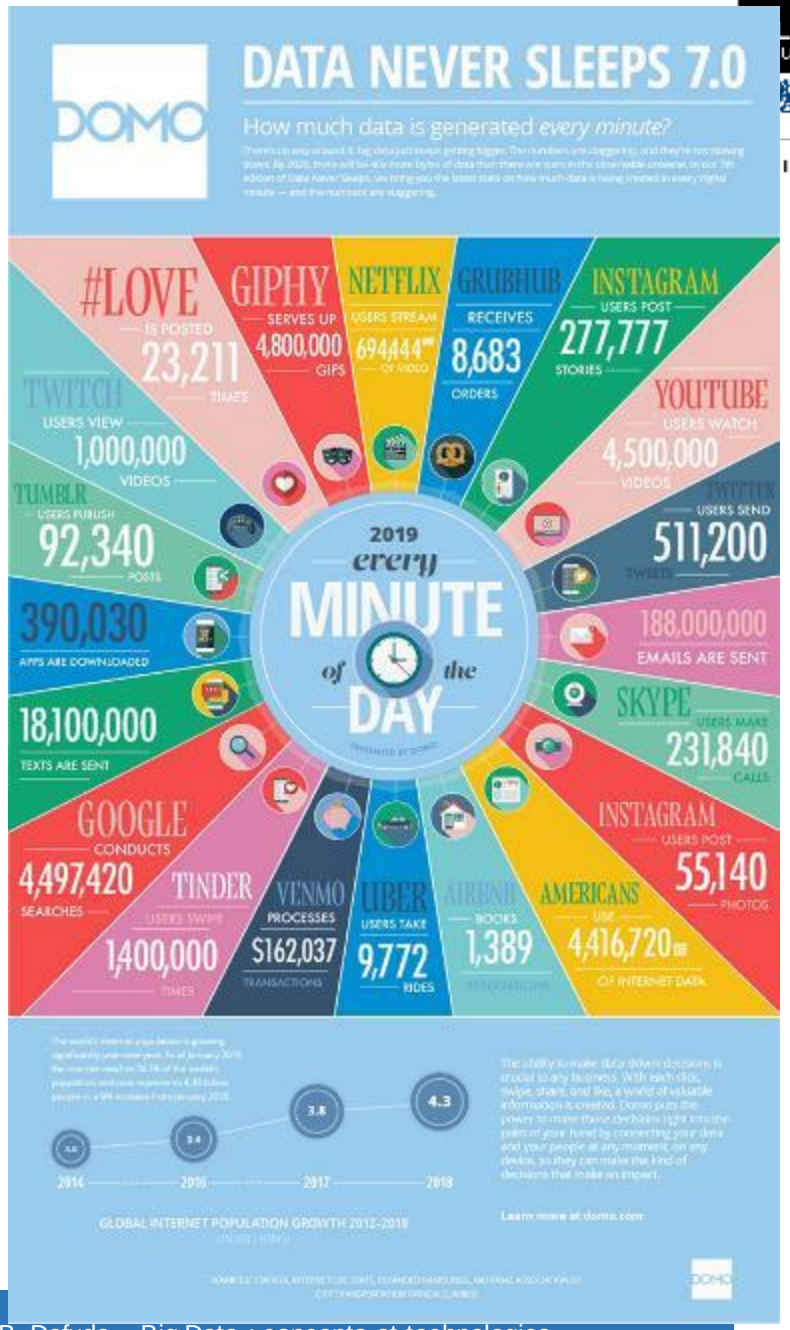
Qualité!

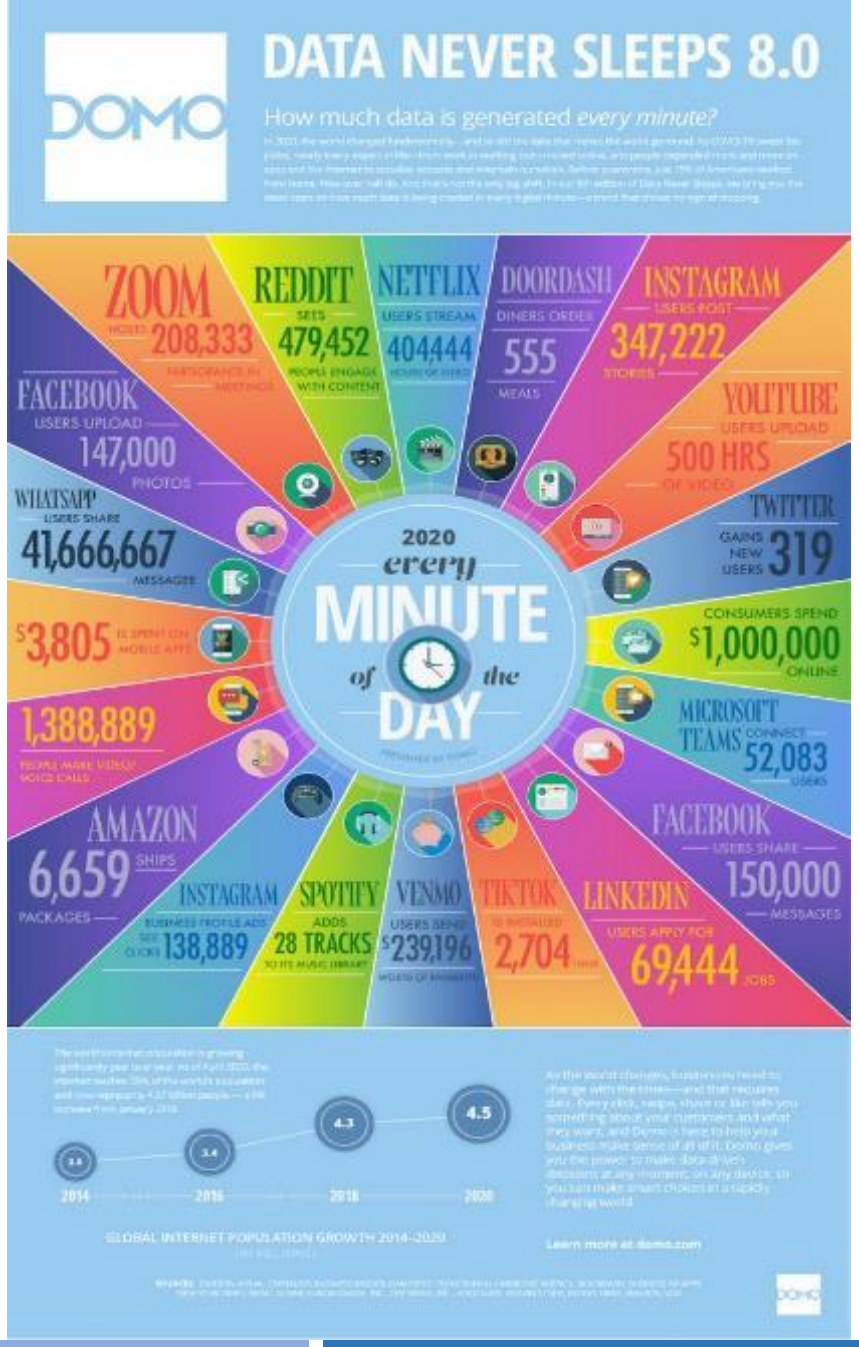
- La qualité des analyses menées sur les données depend de la qualité des données
- Complexe et couteux d'injecter la qualité dans les données (gouvernance des données, master data management)
- Le passage au big data ne simplifie pas le problème
- Il faut vivre avec la non qualité : choisir les données sur lesquelles on va faire de la qualité, qualifier la qualité des sources et la prendre en compte dans les analyses, …

D'où vient le "Big Data"?

- **Science**
  - Données d'astronomie, de génomique, environnementales, de transports, …
- **Sciences Humaines et Sociales**
  - Livres numérisés, documents historiques, données d'interaction sociales, traces GPS, …
- **Business & Commerce**
  - Ventes, données boursières, traffic aérien, …
- **Loisirs**
  - Images, vidéo, films, MP3, …
- **Médecine**
  - Imagerie médicale, dossiers médicaux, …


DOMO
DATA NEVER SLEEPS 5.0
How much data is generated every minute?
THE WEATHER CHANNEL
18,055,555.56
GIPHY
694,444
NETFLIX
69,444
SNAPCHAT
527,760
PHOTOS
LINKEDIN
120+
NEW PROFESSIONALS
AMAZON
MAKES
$258,751.90
IN SALES
YOUTUBE
USERS WATCH
4,146,600
VIDEOS
TUMBLR
74,220
TWITTER
456,000
TWEETS
2017
every
MINUTE
of the
DAY
103,447,520
SPAM EMAILS SENT
15,220,700
TEXTS SENT
WIKIPEDIA
600
SKYPE
154,200
GOOGLE
CONDUCTS
3,607,080
SEARCHES
INSTAGRAM
USERS POST
46,740
PHOTOS
BUZZFEED
50,925.92
VENMO
PROCESSES
$51,892
FEE-TO-FEE TRANSACTIONS
UBER
45,787.54
TRIPS
SPOTIFY
13
AMERICANS
2,657,700
OF INTERNET DATA
3.4
3.7
2012
2014
2016
2017
Learn more at domo.com
GLOBAL INTERNET POPULATION GROWTH 2012–2017


DOMO
DATA NEVER SLEEPS 7.0
How much data is generated every minute?
#LOVE
IS POSTED
23,211
TIMES
GIPHY
SERVES UP
4,800,000
GIFS
NETFLIX
USERS STREAM
694,444
GRUBHUB
RECEIVES
8,683
ORDERS
INSTAGRAM
USERS POST
277,777
STORIES
TWITCH
USERS VIEW
1,000,000
VIDEOS
YOUTUBE
USERS WATCH
4,500,000
VIDEOS
TUMBLR
USERS PUBLISH
92,340
POSTS
2019
every
MINUTE
of the
DAY
TWITTER
USERS SEND
511,200
TWEETS
390,030
APPS ARE DOWNLOADED
188,000,000
EMAILS ARE SENT
18,100,000
TEXTS ARE SENT
SKYPE
USERS MAKE
231,840
CALLS
GOOGLE
CONDUCTS
4,497,420
SEARCHES
INSTAGRAM
USERS POST
55,140
PHOTOS
TINDER
USERS SWIPE
1,400,000
VENMO
PROCESSES
$162,037
UBER
USERS TAKE
9,772
RIDES
AIRBNB
BOOKS
1,389
AMERICANS
USE
4,416,720
OF INTERNET DATA
3.4
3.8
4.3
2014
2016
2017
2018
GLOBAL INTERNET POPULATION GROWTH 2012–2018


DOMO
DATA NEVER SLEEPS 8.0
How much data is generated every minute?
ZOOM
208,333
REDDIT
479,452
NETFLIX
404444
DOORDASH
555
INSTAGRAM
347,222
FACEBOOK
147,000
PHOTOS
YOUTUBE
500 HRS
WHATSAPP
41,666,667
MESSAGES
TWITTER
319
2020
every
MINUTE
of the
DAY
$3,805
1,388,889
$1,000,000
MICROSOFT TEAMS
52,083
AMAZON
6,659
PACKAGES
INSTAGRAM
138,889
SPOTIFY
28 TRACKS
VENMO
$239,196
TIKTOK
2,704
LINKEDIN
69,444
FACEBOOK
150,000
MESSAGES
2014
2016
2018
2020
GLOBAL INTERNET POPULATION GROWTH 2014-2020
Learn more at domo.com

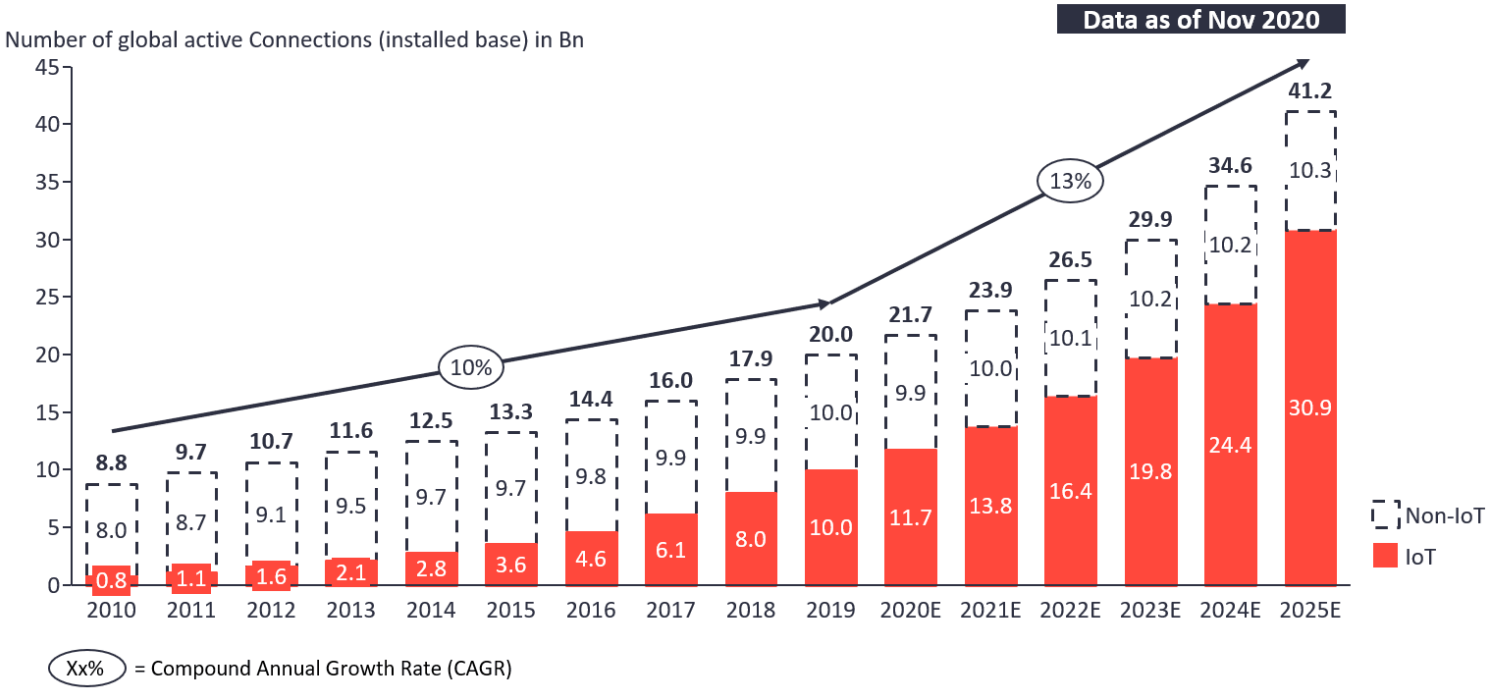
Explosion de l'IoT

IOT ANALYTICS

Insights that empower you to understand IoT markets

Total number of device connections (incl. Non-IoT)

20.0Bn in 2019– expected to grow 13% to 41.2Bn in 2025



Note: Non-IoT includes all mobile phones, tablets, PCs, laptops, and fixed line phones. IoT includes all consumer and B2B devices connected – see IoT break-down for further details

Source(s): IoT Analytics - Cellular IoT & LPWA Connectivity Market Tracker 2010-25

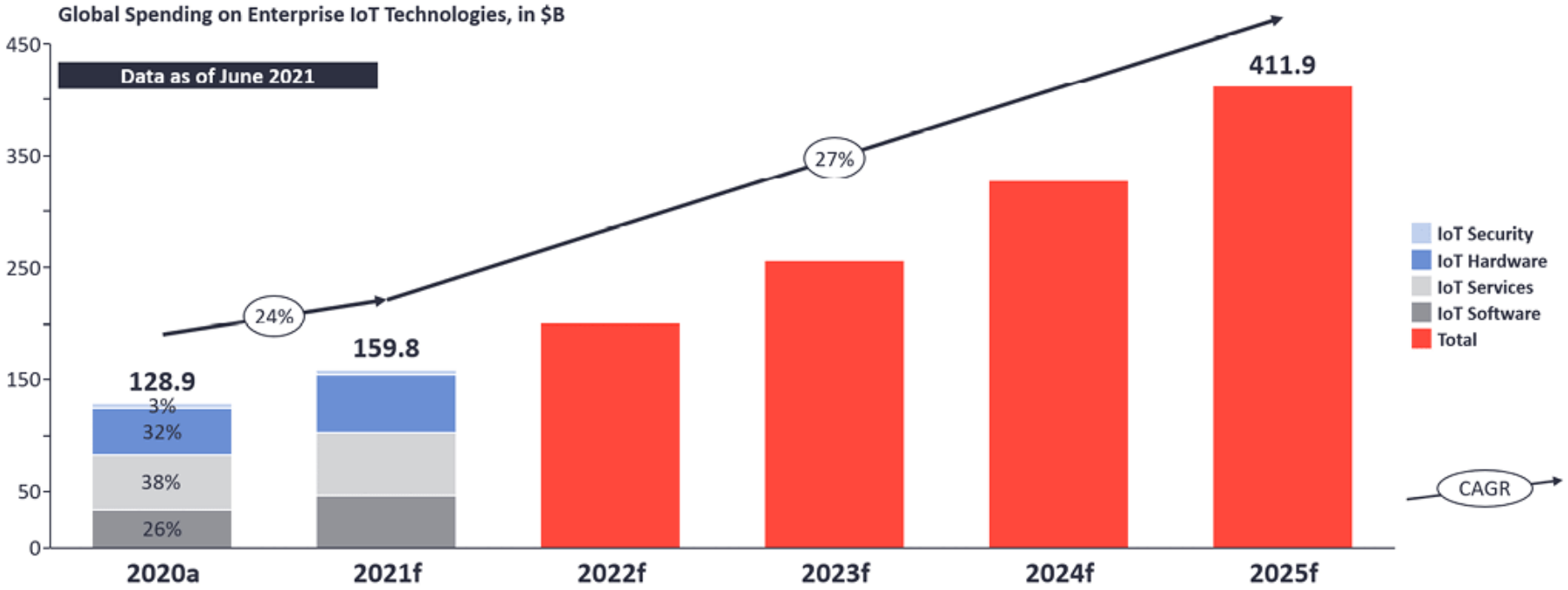
https://iot-analytics.com/state-of-the-iot-2020-12-billion-iot-connections-surpassing-non-iot-for-the-first-time/

Dépenses de l'IoT

IOT ANALYTICS

Your Global IoT Market Research Partner

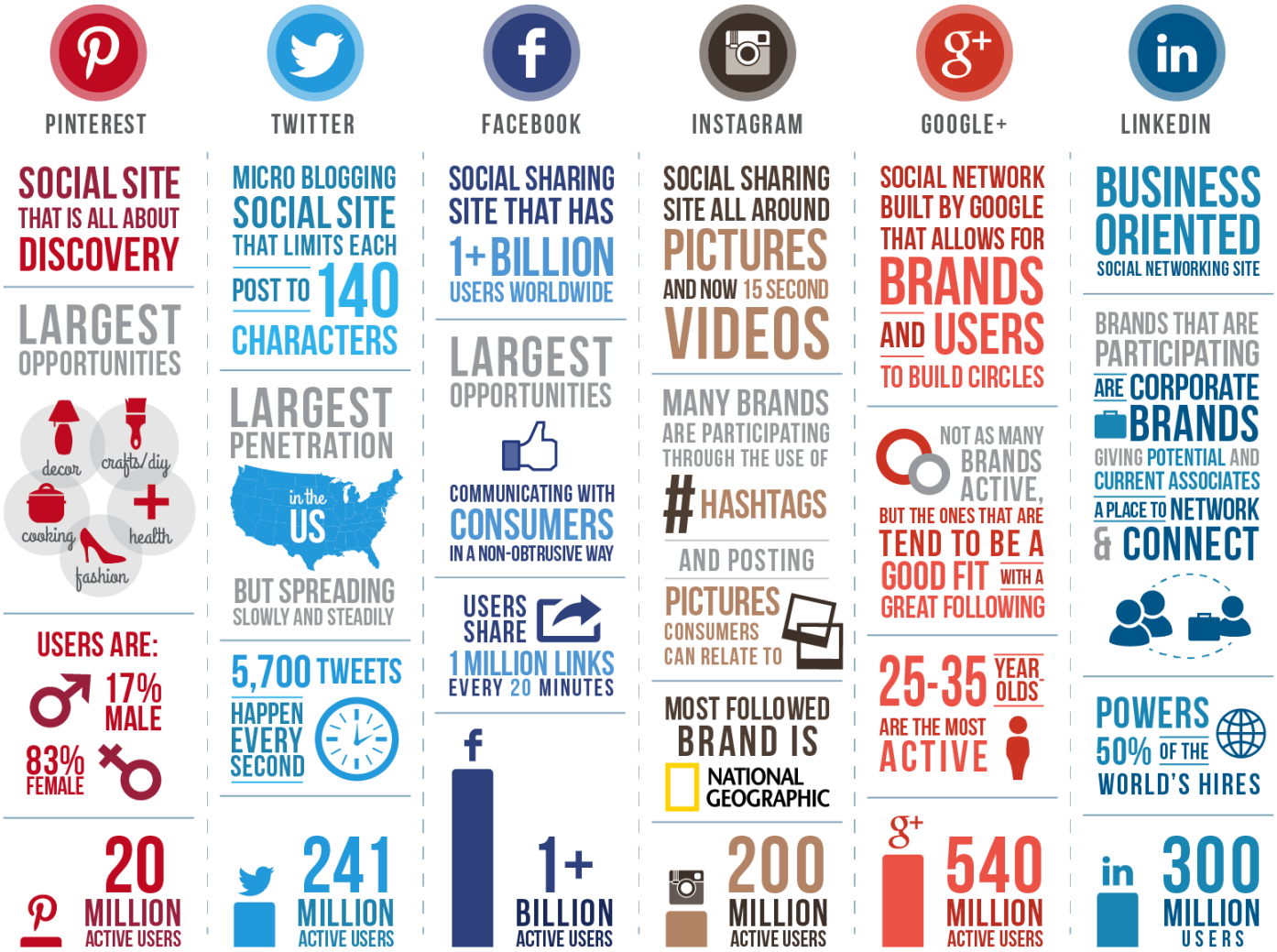
IoT Enterprise Spending 2020 – 2025



Note: IoT Analytics defines IoT as a network of internet-enabled physical objects. Objects that become internet-enabled (IoT devices) typically interact via embedded systems, some form of network communication,

https://iot-analytics.com/state-of-the-iot-2020-12-billion-iot-connections-surpassing-non-iot-for-the-first-time/

L'ampleur des réseaux sociaux



Programme du matin

- Introduction et contexte
- **Entreprise data-driven**
- Des données aux big data
  - Bases de données NoSQL
  - Entrepôts de données
  - Data lake
- De Hadoop à Spark
  - Typologie des outils big data
  - Mapreduce et Hadoop
  - spark

Omniprésence de la donnée

- **Numérisation de la société passe notamment par une production de plus en plus importante de données**
- **Énormément de données sont disponibles (stockées) mais peu sont valorisées**
- **Donnée nouvelle « semence » : plus on utilise une donnée, plus elle prend de la valeur**
- **La valeur nait généralement d'un croisement de données (éviter les silos)**

Importance des données

- **Linkedin : données massives sur les personnes avec études, postes occupés, réseau de relations**
  - devenir des diplômés
  - Changements de régions
  - Bon fonctionnement des institutions scolaires et universitaires
- **Google : données massives sur le déplacement des personnes (google location history)**
  - Prévision de trafic avec recommandations
  - Compréhension des déplacements urbains
- **Google flu trends vs réseau sentinelle**
- **Désintermédiation**
  - Booking.com leader de la réservation hôtelière sans posséder un seul hôtel!

Les données, génératrices de valeur

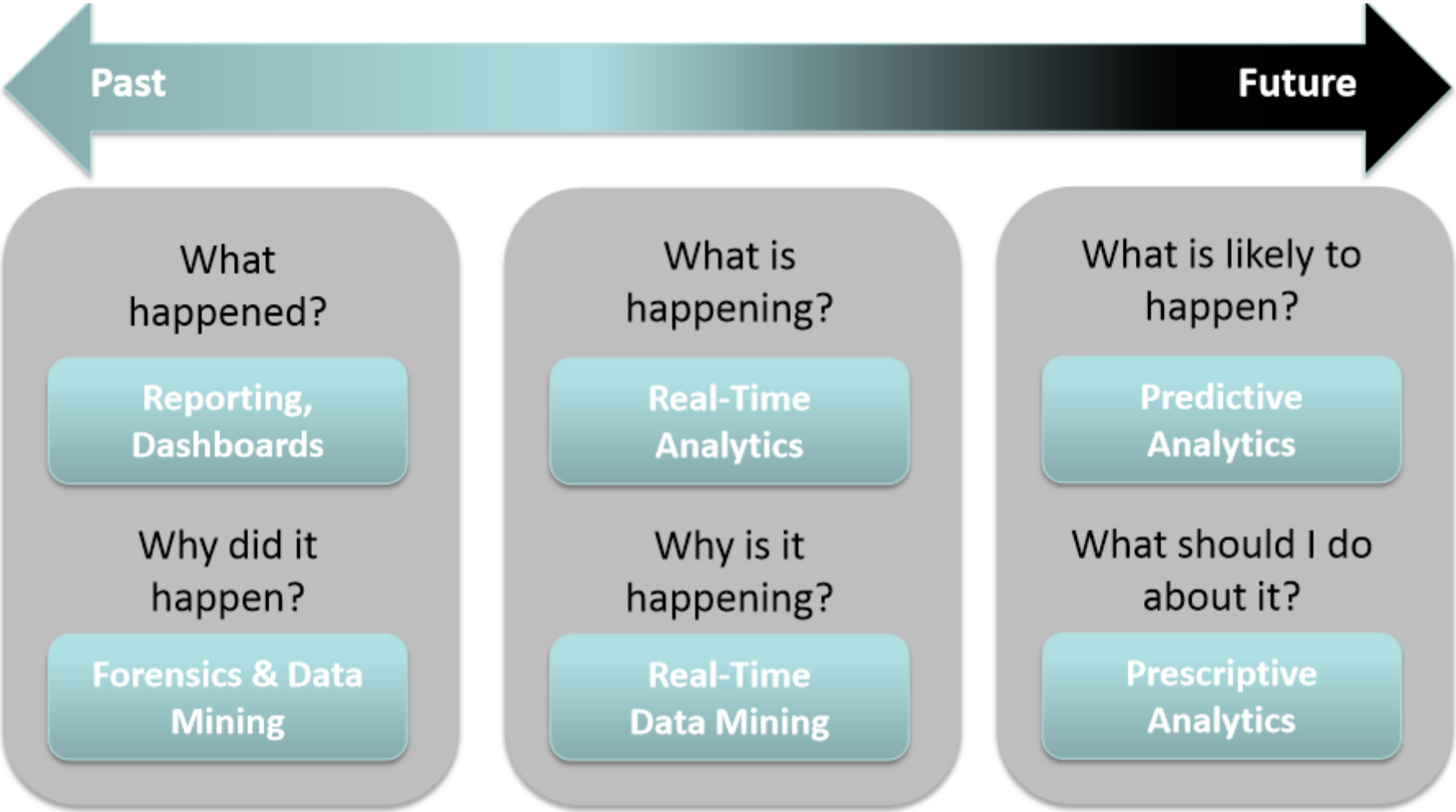
- Meilleure connaissances des clients : clients360
- Optimisation des processus : via l'analyse des usages recueillis dans les logs
- Création de nouveaux business
- Ont une valeur marchande
- Patrimoine informationnel
- Au cœur de la transformation numérique

Impact du big data sur le SI : plus de résolution!

Wolfenstein game graphics

Merci à Simon Chignard

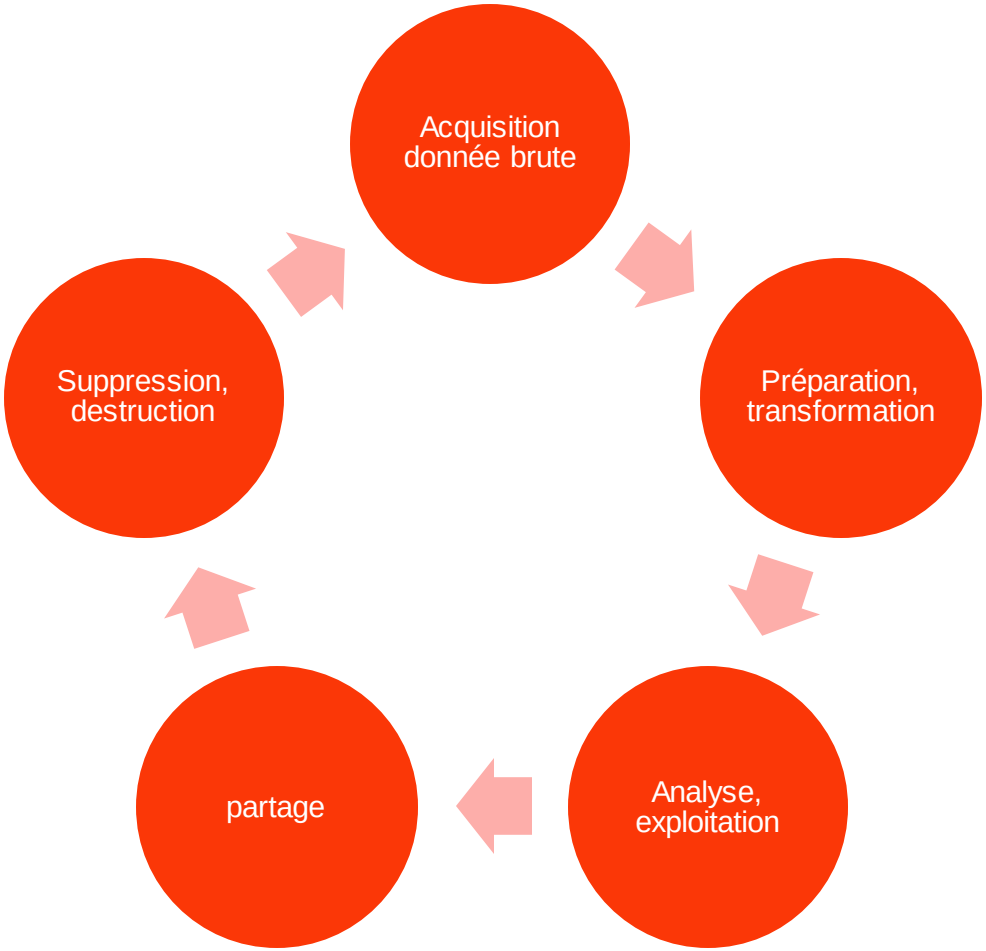
Types d'analyse



La bonne boite

- https://labonneboite.pole-emploi.fr
- **Permet d'obtenir les entreprises qui, pour un métier donné et une zone géographique, ont la probabilité la plus forte d'embaucher**
- **Permet de cibler ses candidatures spontanées**
- **Basé sur les DPAE (déclarations préalables à l'embauche) sur les 7 derniers semestres ainsi que sur les embauches effectives de personnes via pole emploi**
- **DPAE permet d'estimer le nombre de recrutements à venir, embauches effectives permettent de ventiler sur les différent métiers de l'entreprise**

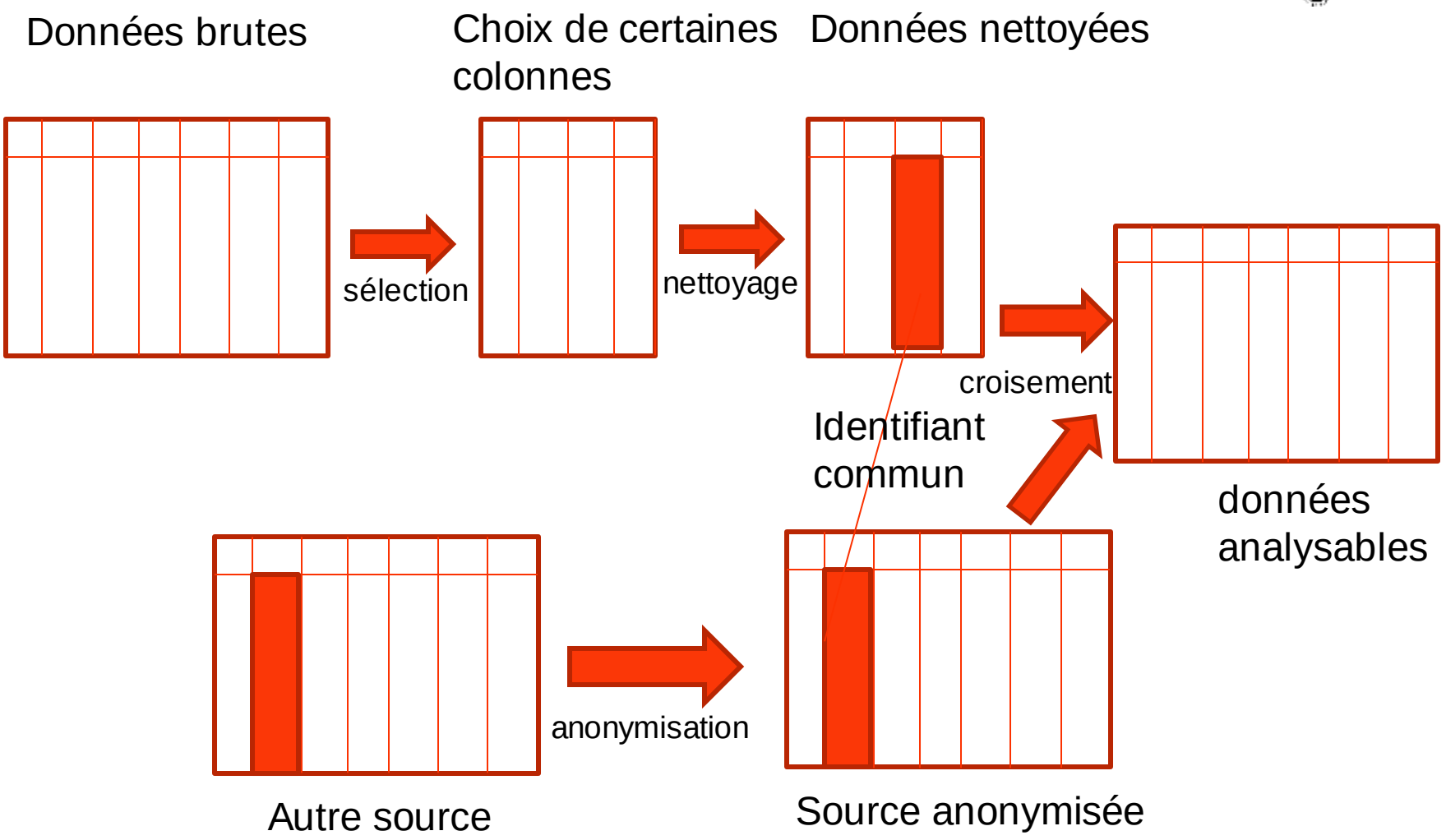
Cycle de vie de la donnée



Sources de données

- **Sources internes à l'entreprise**
- **Sources web**
  - Blog
  - Sites web concurrents
  - Etc
- **Réseaux sociaux**
  - Facebook
  - Linkedin
  - Twitter
- **Open data gouvernementales**

Préparation des données



Qualité des données

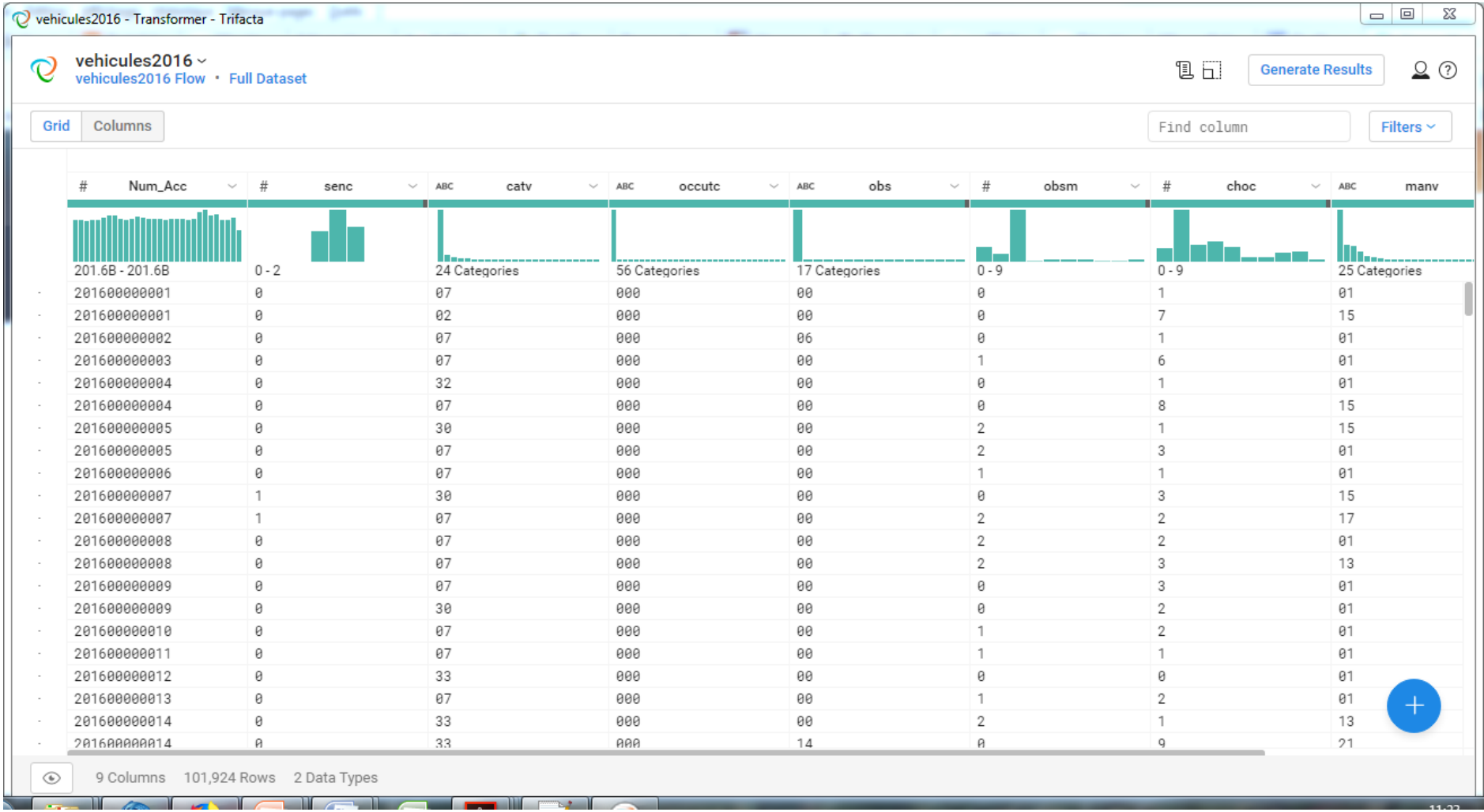
Les données du monde réel sont “sales”

- **Incomplètes** : valeurs vides
- **Bruitées** : contiennent des erreurs (typo, transpositions de mots, valeurs multiples dans un attribut unique)
- **Incohérentes :** abbréviations, sources multiples, …
- **Imprécises :** imprécision d’un capteur
- **Pas à jour :** oubli d’une mise à jour

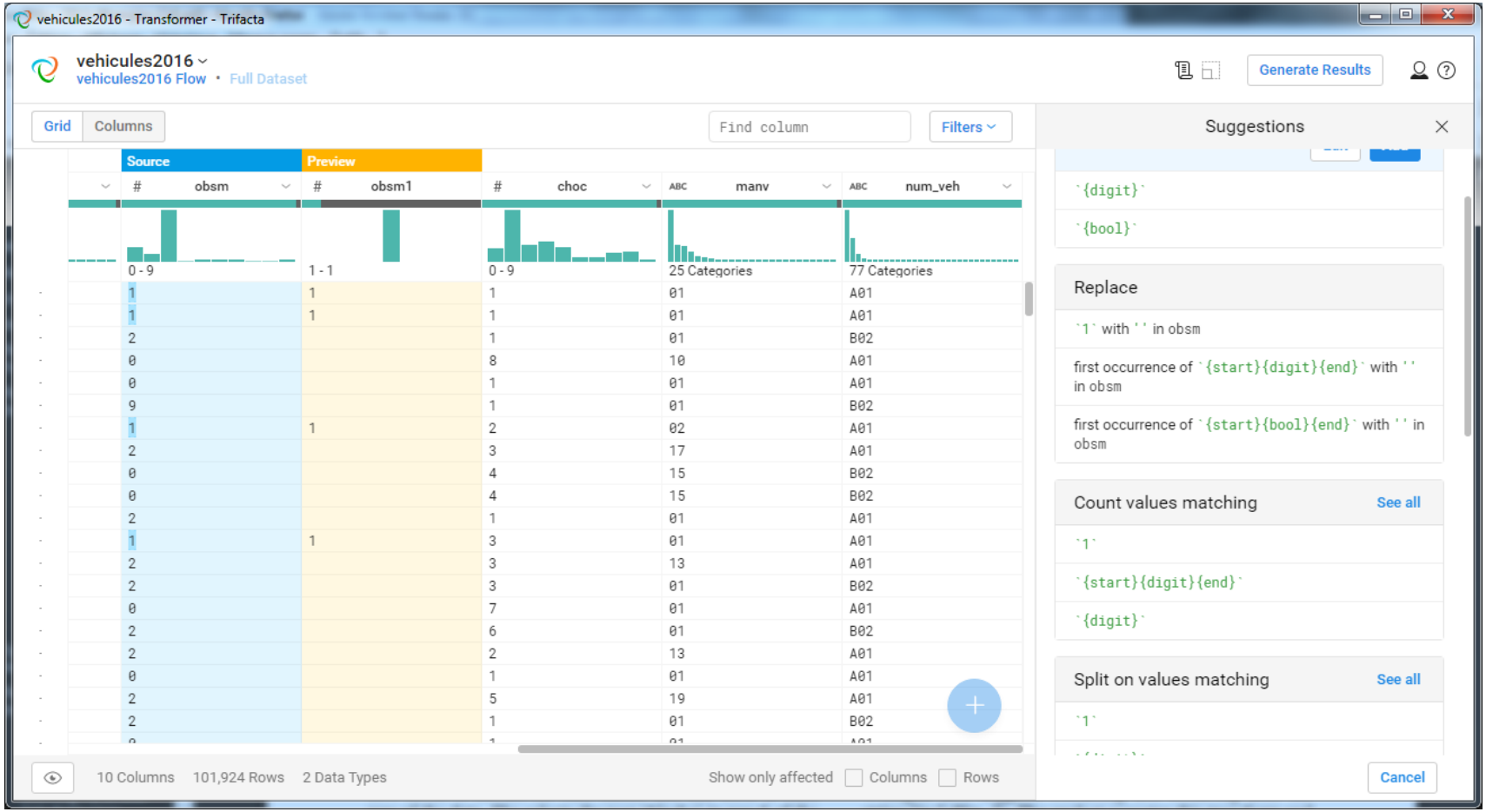
Nettoyage des données

- **Dédoublonner**
- **Gérer les valeurs nulles (suppression, mettre valeur par défaut, …)**
- **Gérer les valeurs en anomalie (outliers)**
- **Analyser les dépendances entre attributs (y compris entre plusieurs sources)**
- **…**

Exemple de trifacta



Exemple de trifacta (2)



Partage des données

- **Tension entre respect de la vie privée (privacy) et utilité des données**
- **Partager des données personnelles n'est pas possible sans respecter le RGPD**
- **Utilisateur donne son consentement pour un usage donné**
- **Besoin d'anonymiser (données ne sont plus personnelles) ou de pseudonymiser (données restent personnelles)**
- **Plus on corrèle des sources, plus on augmente les risques de révéler la vie privée**

Anonymisation (CNIL)

- **L'anonymisation est un traitement qui consiste à utiliser un ensemble de techniques de manière à rendre impossible, en pratique, toute identification de la personne par quelque moyen que ce soit et ce de manière irréversible.**
- **Une donnée personnelle anonymisée perd son caractère personnel**

Pseudonymisation (CNIL)

- La pseudonymisation est un traitement de données personnelles réalisé de manière à ce qu'on ne puisse plus attribuer les données relatives à une personne physique sans avoir recours à des informations supplémentaires. En pratique la pseudonymisation consiste à remplacer les données directement identifiantes (nom, prénom, etc.) d'un jeu de données par des données indirectement identifiantes (alias, numéro dans un classement, etc.).
- La pseudonymisation permet ainsi de traiter les données d'individus sans pouvoir identifier ceux-ci de façon directe.
- En pratique, il est toutefois bien souvent possible de retrouver l'identité de ceux-ci grâce à des données tierces.
- C'est pourquoi des données pseudonymisées demeurent des données personnelles.
- L'opération de pseudonymisation est réversible, contrairement à l'anonymisation.

Anonymisation

- **Important de respecter la vie privée et donc de ne pas permettre d'identifier les personnes sur lesquelles les données ont été collectées**
- **Trois garanties à respecter :**
  - Ne pas identifier un individu
  - Ne pas permettre de lier des enregistrements d'un même individu
  - Ne pas permettre de déduire avec une probabilité élevée les valeurs d'un attribut à partir d'autres

Comment anonymiser ?

- **Randomisation (ajouter de l'aléa)**
  - Dégrader la précision d'une mesure par exemple
  - Permutation des valeurs entre différents enregistrements (garde la distribution, difficile à faire en gardant la cohérence)
- **Généralisation (changer d'échelle, passer d'une ville à un département, …)**
  - K-anonymat : généraliser des valeurs d'attributs de manière à être sûr qu'il y ait au moins k individus avec une même valeur
- **Pseudo-anonymisation**
  - Transformer une valeur par une fonction mathématique

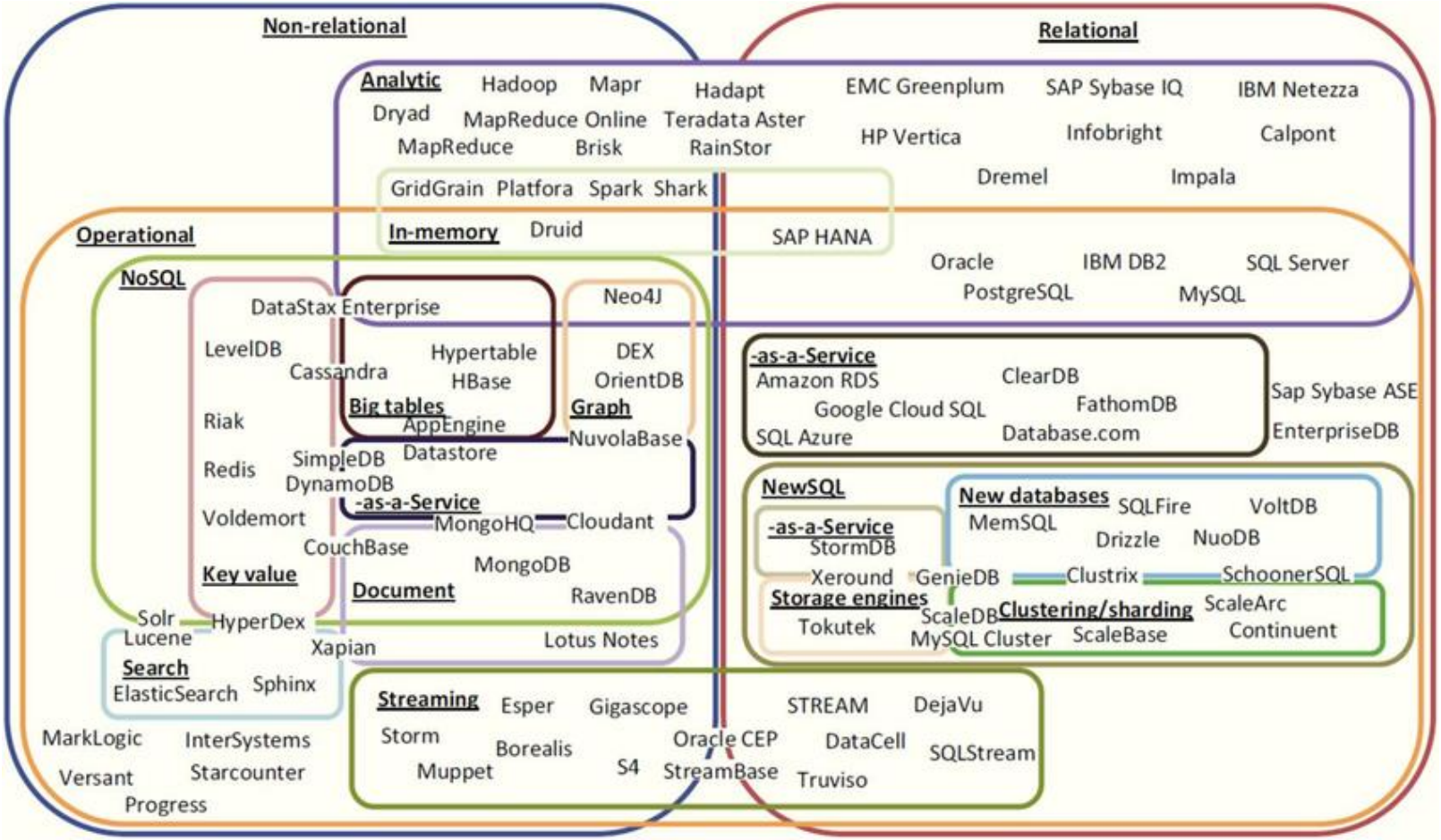
Attaques possibles

- Trouver la fonction d'anonymisation et l'inverser
- Corréler avec d'autres sources non anonymisées
- Utiliser des informations externes

Programme

- Introduction et contexte
- Infrastructures pour le big data
  - Cloud
  - Bases de données NoSQL
- **Des données aux big data**
  - Bases NoSQL
  - Entrepôts de données
  - Data lake
- De Hadoop à Spark
  - Typologie des outils big data
  - Mapreduce et Hadoop
  - spark

Paysage des SGBD



https://www.researchgate.net/publication/303562879_Setting_Up_a_Big_Data_Project_Challenges_Opportunities_Technologies_and_Optimization

SGBD : un peu d'histoire

- **1970 : systèmes hiérarchiques et réseaux, codasyl**
  - Essentiellement des enregistrements liés entre eux par des adresses physiques
  - Peu déclaratif
  - Mises à jour du schéma difficiles
- **1980 : systèmes relationnels**
  - Modèle logique indépendant du modèle physique
  - Déclaratif -> SQL
  - Mises à jour du schéma faciles

SGBD : un peu d'histoire (2)

- **1990 : systèmes objets**
  - Modèle de données plus général (combinaison de set, array et tuple) : ODMG et OQL comme langage de requêtes
  - Même système de types que dans les langages de programmation
- **2000 : systèmes XML**
  - Modèle de données incluant structuré et semi-structuré
  - Xquery comme langage de requêtes déclaratif
- **2010 : systèmes NoSQL**

Les données aujourd'hui

- **Très grande diversité des données**
  - Structurées, non structurées, séries temporelles, vidéo, capteurs, réseaux sociaux, données spatiales, …
- **Explosion de la volumétrie des données**
- **Explosion du nombre d'utilisateurs**
- **Grande variabilité des charges**
  - Remise en cause du « one size fits all » du relationnel
  - Remise en cause des architectures des SGBD
  - Une multitude de SGBD spécialisés

Un point sur le relationnel

- **Importance du schéma de données (données structurées, stockage, optimisation de requêtes)**
- **Vision universelle (« one size fits all »)**
- **Langage de requêtes déclaratif : SQL**
- **Technologie d'implémentation mature (plus de 30 ans de R&D)**
- **OLTP**
  - Transactions ACID
- **OLAP (SGBD parallèles, datawarehouse)**
  - Stockage colonne (HP Vertica, …)

Bases NoSQL

- **Bases relationnelles : technologie très mature, offre bien positionnée, bien adaptée pour des données structurées, efficacité, cohérence (transactions), simplicité d'utilisation**
- **Bases NoSQL : technologie émergente, offre pléthorique et peu lisible, bien adaptée pour des données non structurées, privilégie les performances vs cohérence, plus difficile à programmer**
- **Trois grandes catégories + SGBD colonnes**
  - Clé-valeur : Dynamo, redis
  - Documents : couchDB, MongoDB, …
  - Graphes : Neo4J
  - Colonne : Cassandra

NoSQL : objectifs prioritaires

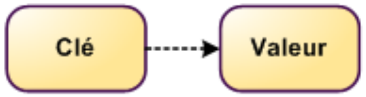
- **Montée en charge**
  - Utilisation de serveurs standard
  - Mise à jour à faible latence
  - Permet mise à jour / insertion à haute intensité
- **Elasticité– s'adapte en fonction de la charge**
- **Haute disponibilité– arrêt implique perte de revenu**
  - Réplication (en mode multi-maîtres)
  - Réplication géographique
  - Recouvrement sur panne automatique

NoSQL : objectifs faibles

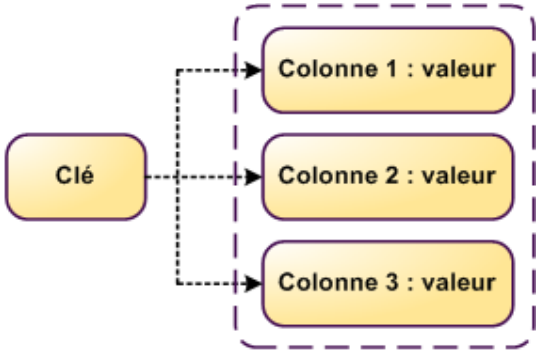
- **Pas de requêtes déclaratives**
  - SQL non supporté
  - Opérations CRUD via des API spécifiques
- **Pas de jointures**
  - Fait par programmation
  - Tendance à dénormaliser les données ?
- **Pas de transactions**
  - Transactions réduites à une seule instruction
  - Compromis cohérence disponibilité réglable (e.g., Dynamo)

➔ **Permet le passage à l'échelle**

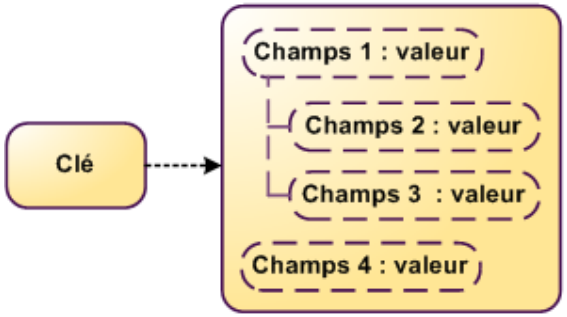
NoSQL : Les 4 catégories



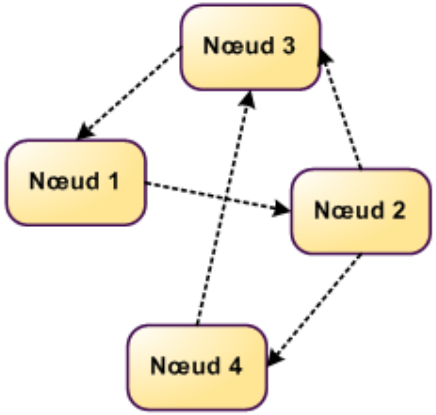
BDD Clé-Valeur



BDD Orientée colonnes



BDD Orientée document



BDD Orientée graphe

Bases clé-valeur

- **Modèle de données basé clé**
  - Clé est l'identifiant unique
  - Clé est le niveau d'accès
  - Valeur est opaque
- **De nombreux produits**
  - Propriétaires : **Bigtable** (Google), **PNUTS** (Yahoo!), **Dynamo** (Amazon)
  - Open source : **HBase, Hypertable, Cassandra, Voldemort**
- **Très utilisé pour les applis web**
- **Requêtes simples**

Bases NoSQL : clé-valeur

- **Interface très simple**
  - modèle de données : paires (clé, valeur)
  - Opérations: Insert(clé,valeur), Fetch(clé), Update(clé), Delete(clé)
- **Implémentation : efficacité, passage à l'échelle, tolérance aux fautes**
  - enregistrements distribués sur les noeuds via les clés
  - Réplication
  - Transactions mono-enregistrement, cohérence à terme ("eventual consistency")

NoSQL Clé-Valeurs – exemples d'utilisation

- **Logs de sites Web ou d'application**
- **Profils utilisateurs de site Web/réseaux sociaux**
- **Données de capteurs**
- **Cache Web ou BD**
- **Paniers sur sites de e-commerce**

- **…**

NoSQL Clé-Valeurs

Avantages :

- **Modèle de données très simple**
- **Scalabilité horizontale élevée (en lecture et en écriture)**
- **Performance de lecture / écriture**
- **Modification facile du contenu associé à une clé et par extension du « schéma » (ajout d'une colonne par ex.).**
- **Pas de maintenance requise lors de l'ajout/suppression d'une colonne**
- **Augmente la disponibilité**

Inconvénients :

- Modèle de données simple
- limité pour les données complexes
- Interrogation simple car accès par clé
- Fonctionnalités en général limitées aux opérations CRUD
- Interrogation sur la clé uniquement
- Déporte la complexité de l'application sur le client applicatif

Bases NoSQL : Documents

- **Comme les bases clé-valeur mais la valeur est un document semi-structuré**
  - modèle de données : paires (clé, document)
  - Document: JSON, XML, autres formats semi-structurés
  - Opérations de base : Insert(key,document), Fetch(key), Update(key), Delete(key)
  - aussi Fetch basé sur le contenu des documents
- **Produits**
  - CouchDB, MongoDB, SimpleDB etc

Bases documents

- Schéma non nécessaire et variable (peut varier d'un document à l'autre)
- Les documents peuvent être très hétérogènes au sein de la BD
- Imbrication de données (schéma arborescent)
- Chaque donnée du document peut être interrogée
- On peut effectuer des requêtes sur le contenu des documents (impossible avec NoSQL Clé-Valeur)

NoSQL Documents

Avantages :

- Modèle simple mais bonne puissance d'expression (structure imbriquée)
- Interrogation de tout attribut (+indexation)
- Variété de types de données et des opérations (y compris retourner seulement une partie des valeurs)
- Pas de maintenance de la BD requise pour ajouter/supprimer des « colonnes »
- Simplifie la mise à jour d'un système avec le support de champs optionnels,
  - Ajout / suppression de champs sans migration de modèle de données

Inconvénients :

- Inadaptée pour les données interconnectées (Inter-connexion de données complexes)
- Modèle de requête limité à des clés (et index)
- Opération transactionnelle mono document
- Pas de mise à jour partielle d'un document -> Mode tout ou rien
- Pas adapté à des besoins de mise à jour fréquents sur des structures de données volumineuses

NoSQL Documents : Exemples d'utilisations

- **Gestion de contenu:**
  - bibliothèques numériques,
  - catalogues de produits,
  - dépôts de logiciels « xxxStores »
  - collections multimédia
- **Collection d'événements complexes**
- **Gestion de boîtes email**
- **Gestion des historiques d'utilisateurs sur réseaux sociaux**

Systèmes NoSQL orienté colonnes

- Mixe approche clé valeur et structuration en colonnes
- Famille de colonnes ≈ table relationnelle ou collection de lignes
- Ligne = clé + ensemble de (nom colonne, valeur, timestamp)
- Lignes à l'intérieur d'une même famille de colonnes n'ont pas forcément la même structure
- Cassandra, Accumulo, HBase

Avantages des systèmes orienté colonnes

- Modèle de donnée expressif : types simples et collections (set et map)
- Peut s'utiliser avec ou sans schéma
- Reste dynamique (ajout, suppression de colonnes)
- Peut supporter un grand nombre de colonnes
- Avec Cassandra, on peut utiliser CQL à la syntaxe proche de SQL

Inconvénients des systèmes orienté colonnes

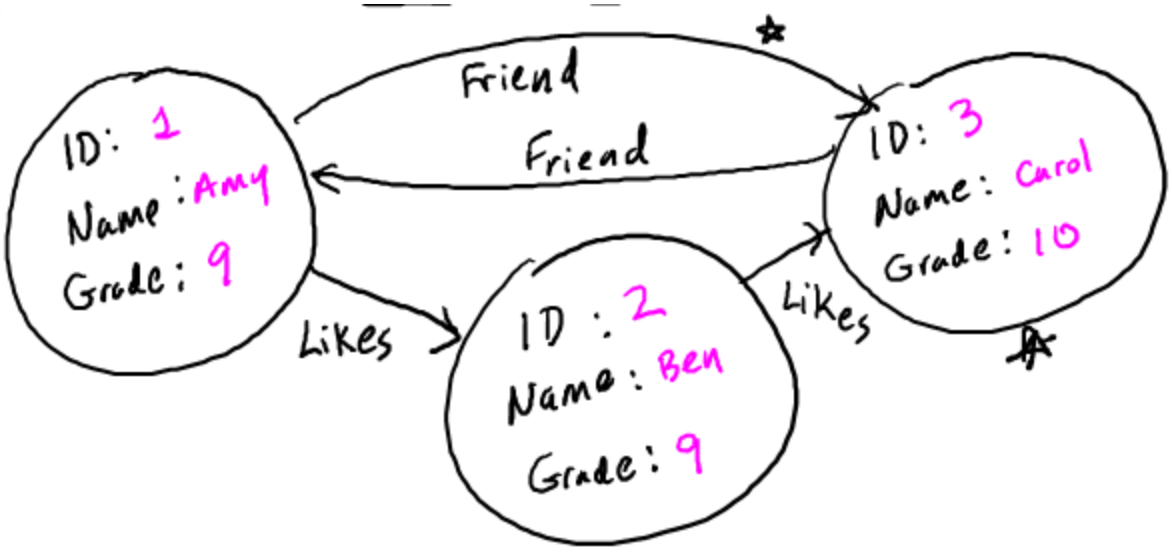
- **Modélisation de données assez complexe (beaucoup de concepts)**
- **Langage de requête reste simple (pas de jointure)**
- **Il faut penser l'organisation des données à partir des requêtes**

Exemple d'Applications pour systèmes orienté colonnes

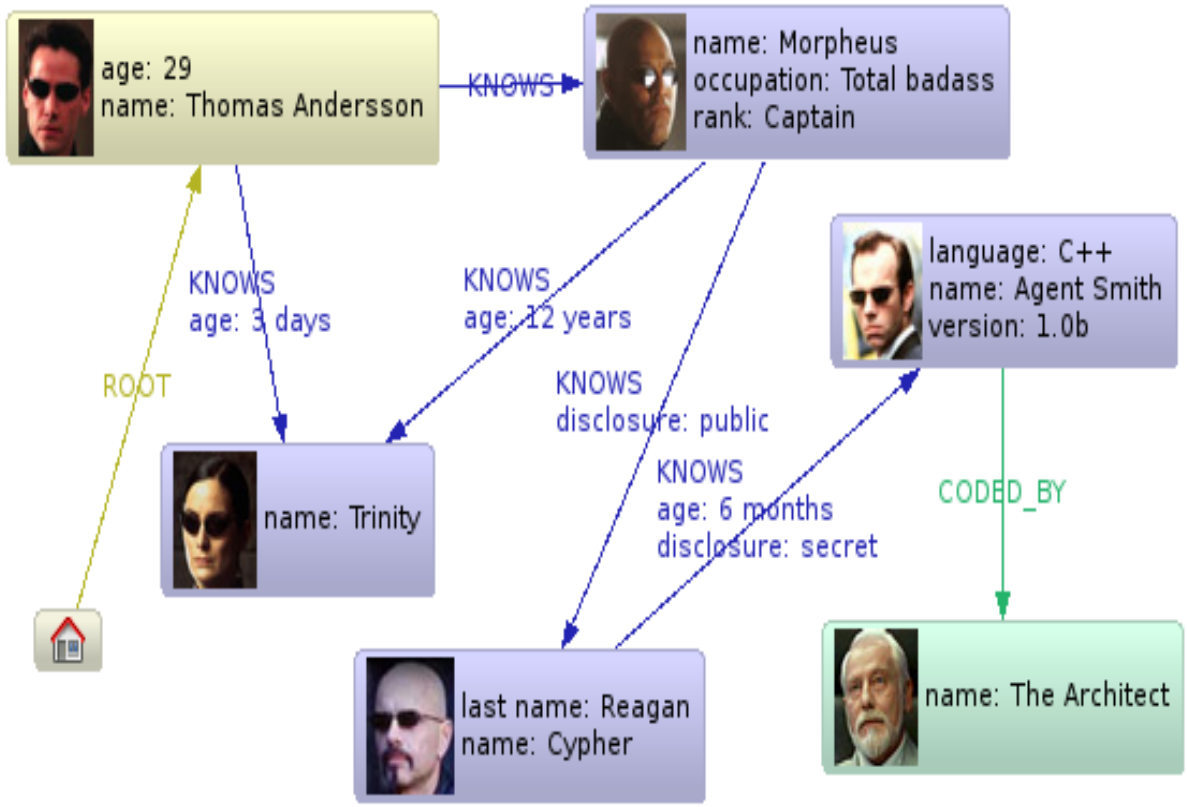
- **E-bay (enregistrement de toutes les données utilisateurs)**
- **Netflix (gestion de 95% de leurs données)**
- **IoT**
- **Séries temporelles**

Bases NoSQL : graphes

- modèle de données : noeuds et arcs
- noeuds peuvent avoir des propriétés (dont ID)
- arcs peuvent avoir des labels ou roles
- Neo4j, Titan, OrientDB



NoSQL Graphe : exemple



Cypher, un langage de requêtes pour graphes

- **Langage déclaratif pour manipuler des graphes inspiré par SQL et SPARQL**
- **Développé depuis 2011 par Neo4j**
- **« standardisé » par le projet openCypher (www.opencypher.org)**

Exemple de requête Cypher

SQL

```
with recursive
as (
    select
        parent, child as descendant,
        1 as level from source
    union all
    select
        d.parent, s.child, d.level + 1
    from descendants as d
    join source s on d.descendant = s.parent
)
select * from descendants
order by parent, level, descendant ;
```

Cypher

```
MATCH
    p=(descendant)-[:Parent*]->(ancestor)
RETURN
    (ancestor), (descendant), length(p)
ORDER BY (ancestor), (descendant), length(p)
```

Trouver toutes les paires ancêtre - descendant

NoSQL graphe – les avantages

- Modèle de données puissant et adapté pour le stockage de grands graphes
- Offre des fonctionnalités de calculs sur grands graphes
- Des langages de requêtes spécifiques aux graphes (expressions de chemin, sous-graphes, …)

NoSQL Graphe – les inconvénients

- Problèmes de performance pour calculer de grandes agrégations de données.
  - Si la requête concerne le dénombrement de nœuds (des produits par exemple), et un filtrage par catégorie (vente) pour obtenir le nombre de ventes sur un mois, le temps de calcul peut devenir très important.
  - L'emploi d'une base de donnée relationnelle classique sera alors préférable

- Très spécifique aux graphes et réseaux

NoSQL Graphe – exemples d'application

- **Calcul sur les graphes sociaux (recommandations, plus courts chemins,…)**
- **Calculs sur les réseaux des SIG : réseaux routiers, canalisations, électricité, …**
- **Web social (linked data)**
- **Moteurs de recommandation : Meetic utilise une BD graphe pour obtenir des recommandations pertinentes entre les membres, par exemple pour connaître les amis d'amis qui ont le même âge.**
- **Web sémantique (RDF)**

Comment choisir un système NoSQL

- **Attention, beaucoup de spécificités propres à chaque système**

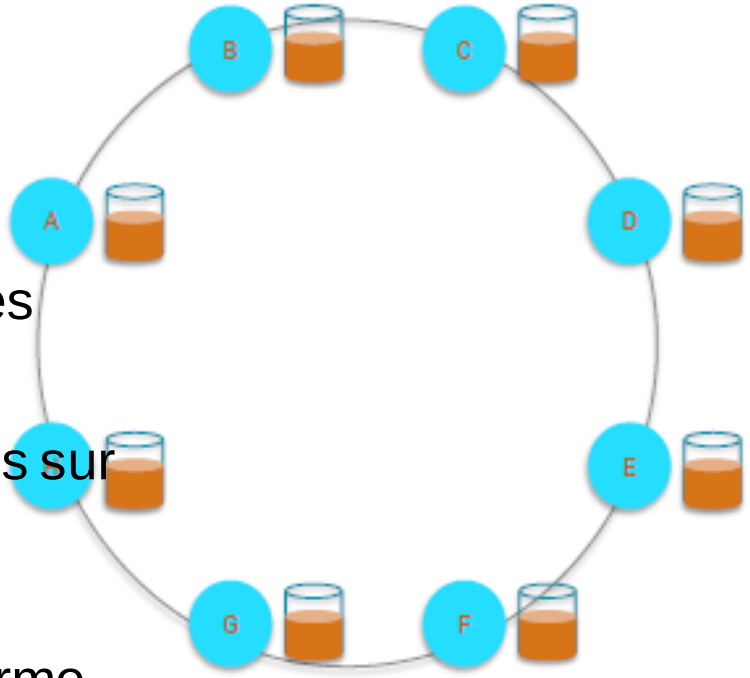
Clé-valeur	Documents	Orienté colonnes	graphe
- Pas de typage des données - Structure très dynamique - Restreint à CRUD	- Modèle de données expressif	-Modèle de données expressif -Grand nombre de colonnes	-Spécifique aux données graphes - langage de requête adapté

sharding

- **Objectifs :**
  - montée en charge linéaire en fonction de la taille des données
  - Équilibrage de charge
  - Supprimer le besoin de synchronisation entre des nœuds (pas de 2PC ou autre algo de consensus fort coûteux)
- **Partition horizontal des données sur un cluster (ou plusieurs clusters)**
- **Partitionnement se fait via une clé et une fonction de hachage**

Sharding (2)

- **Consistent hashing :**
  - Distribution uniforme
  - Assure équilibrage charge
  - Coût de migration constant
  - Requêtes d'intervalles coûteuses
- **Hashage plaçant**
  - Met les données de clés proches sur un même nœud
  - Favorise requêtes d'intervalles
  - Pas forcément distribution uniforme

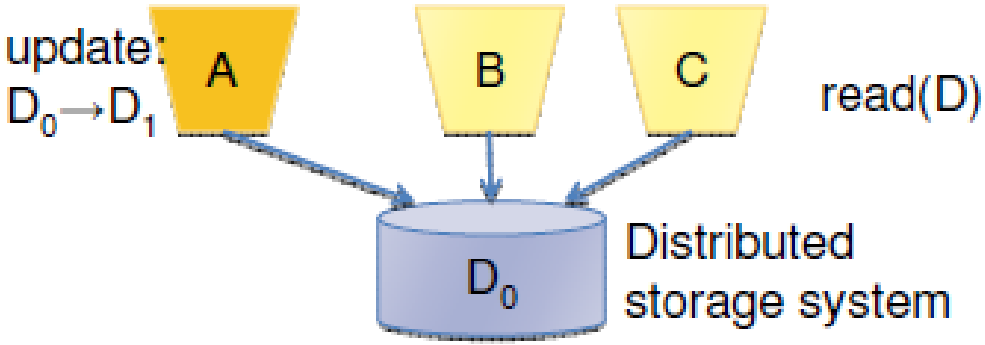


Modèle de cohérence ?

- **Systèmes distribués classiques : focus sur sémantique ACID**
  - Atomicity: une opération (e.g., write) est soit efectuée sur TOUS LES réplicas soit sur aucun d'entre eux
  - Consistency: après chaque opération tous les réplicas atteignent le même état
  - Isolation: aucune opération (e.g., read) ne peut voir les données d'une autre opération (e.g., write) dans un état intermédiaire
  - Durability: lorsqu'une écriture réussit elle est persistante

- **Systèmes web haute performance : focus sur BASE**
  - Basically Available
  - Soft-state (or scalable)
  - Eventually consistent

Modèles de cohérence



- **Cohérence forte:**
  - Après la fin de la mise à jour, chaque nouvel accès depuis A, B, C retourne D_1
- **Cohérence faible :**
  - Ne garantit pas que les nouveaux accès retournent D_1 -> un certain nombre de conditions doivent être remplies pour retourner D_1
- **Eventual consistency:** forme spécifique de cohérence faible
  - Garantit que si aucune nouvelle mise à jour ne se produit, alors à terme tous les accès retournent D_1

ACID vs BASE

ACID

- Cohérence forte pour les transactions, critère principal
- Disponibilité moins important
- Pessimiste
- Analyse rigoureuse
- Mécanismes complexes

BASE

- Disponibilité et passage à l'échelle, critères principaux
- Cohérence faible
- Optimiste
- Best effort
- Simple et rapide

Pourquoi pas ACID+BASE?

- **Quels sont les objectifs d'un système distribué?**
  - C, A, P

- **Strong Consistency: tous les clients voient la même vue des données, même en présence de mises à jour**

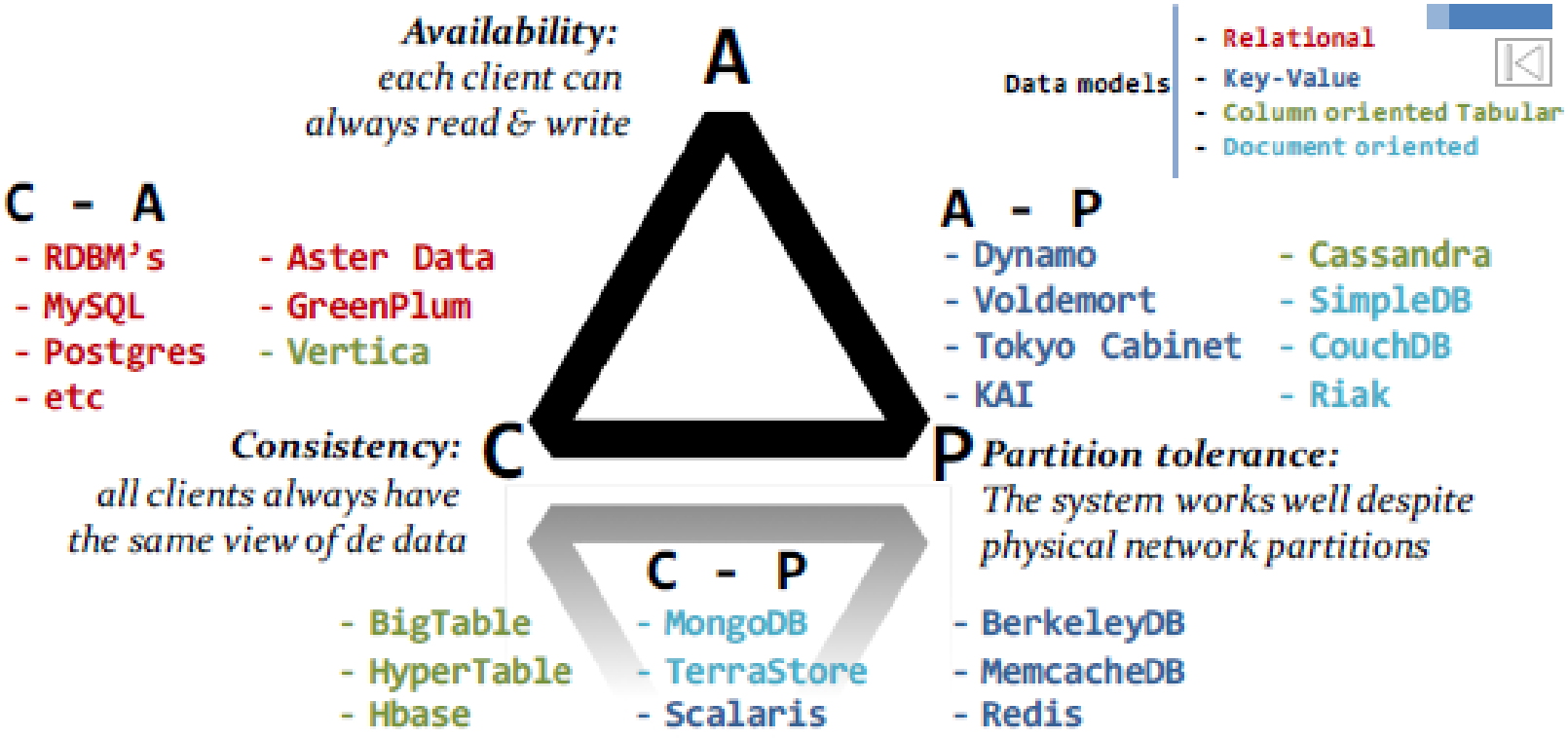
- **High Availability : tous les clients peuvent accéder à une réplique des données, même en présence de fautes**

- **Partition-tolerance: les propriétés du système sont respectées même en cas de partition**

Théorême CAP [Brewer et Lynch]

- **On ne peut avoir que deux des trois propriétés**
- **Le choix de la propriété à relacher détermine les caractéristiques de votre système**

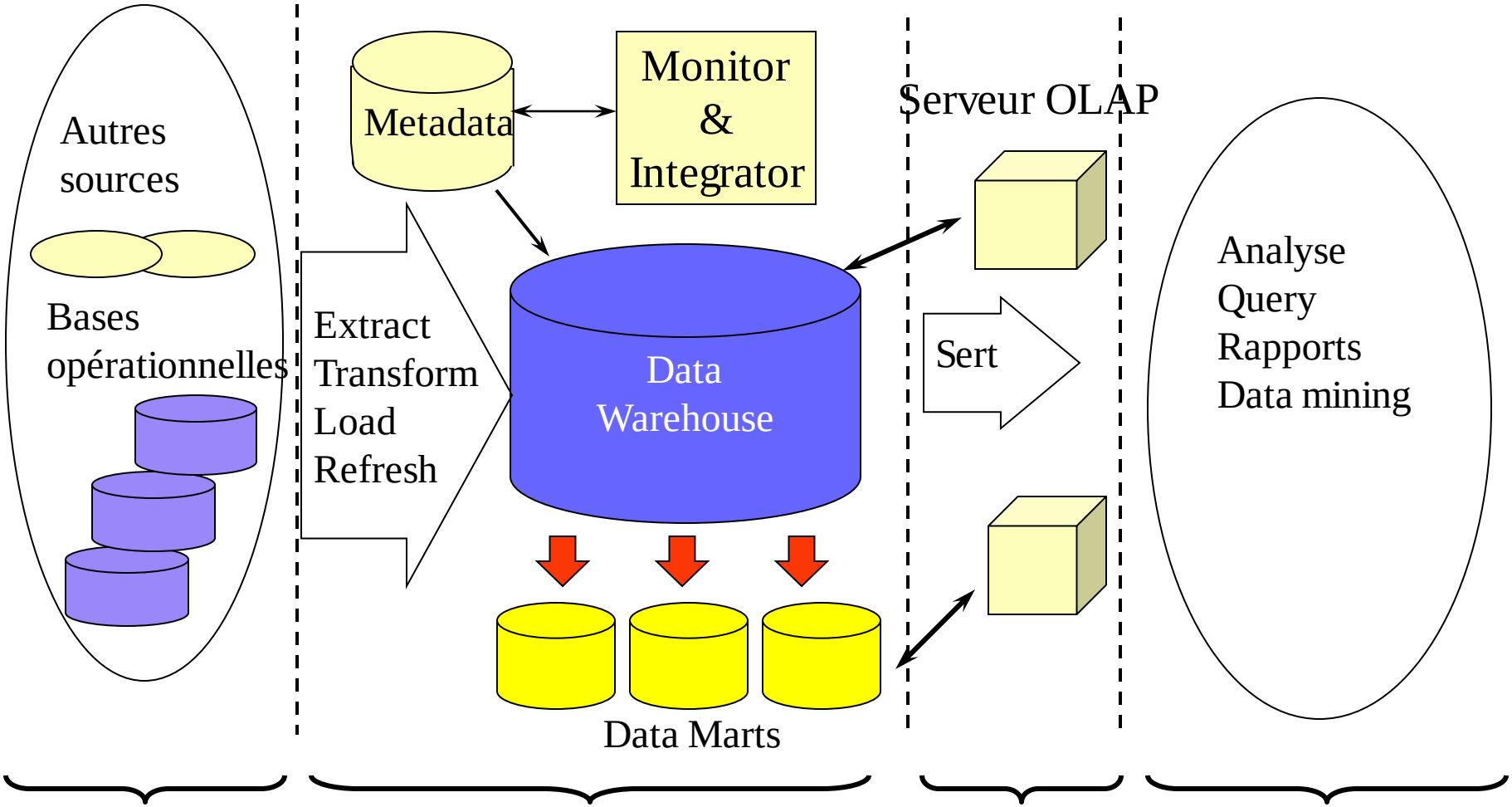
SGBD et CAP


Availability:
each client can
always read & write
A
Data models
- Relational
- Key-Value
- Column oriented Tabular
- Document oriented
C - A
- RDBM's
- MySQL
- Postgres
- etc
- Aster Data
- GreenPlum
- Vertica
A - P
- Dynamo
- Voldemort
- Tokyo Cabinet
- KAI
- Cassandra
- SimpleDB
- CouchDB
- Riak
Consistency:
all clients always have
the same view of de data
C
P
Partition tolerance:
The system works well despite
physical network partitions
C - P
- BigTable
- HyperTable
- Hbase
- MongoDB
- TerraStore
- Scalaris
- BerkeleyDB
- MemcacheDB
- Redis

Cas d'usage

- **Analyser la market place d'amazon**
- **Volumétrie des données**
- **Opérations principales**
  - Vue des clients
  - Vue d'amazon
- **Prioriser les opérations**
- **Définir les objectifs principaux de la solution de stockage des données**

Le processus de DW



Transformations

- Uniformiser : mettre les dates dans un même format, même unité de valeur
- Gérer les formes multiples d'une même valeur (e.g AF234 ou Air France 234)
- Homogénéiser les structures (e.g adresse représentée avec un seul attribut ou une structure de plusieurs attributs rue, ville, …)
- Corriger des erreurs (nettoyer)
- Plein de scripts à écrire, doit être modifié à chaque nouvelle source ajoutée, à chaque modification du schéma de l'entrepôt (intérêt des outils ETL comme Talend Studio)

Solutions DW

IBM Netezza	Teradata	EMC Greenplum	HP Vertica	Oracle Exadata	SAP HANA	SAS HP Analytics
Solution MPP *(traitements massivement parallèles)*	Solution MPP *(traitements massivement parallèles)*	Solution MPP *(traitements massivement parallèles)*	Solution MPP *(traitements massivement parallèles)*	Appliance Data WareHouse	Solution MPP *(in-memory)*	Solution MPP *In memory*
Installation Appliance seule	Installation Appliance seule	Appliance et Software, (stockage colonne et hybride)	Base de données en mode colonne	Stockage mixte (flash et disque), mode colonne et compression	Configuration mixte appliance et software	Solution MPP *In memory*
Offre initiale depuis 2000 ; version actuelle depuis 2004	Offre initiale depuis 1983 ; version actuelle depuis 2009	Offre initiale depuis 2005	Offre initiale depuis 2005	Version 11g Exadata depuis 2008	Offre initiale depuis 2011	Offre initiale depuis 2011

Solutions OLAP

- SAS OLAP server
- Oracle OLAP server option
- Clickhouse
- IBM cognos
- …

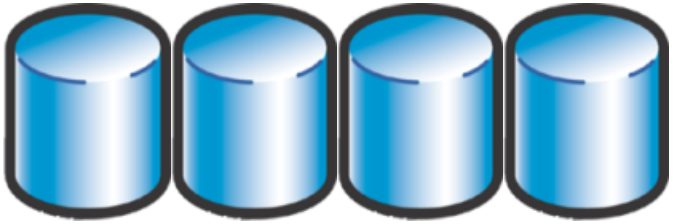
Impacts : Les différences entre Data warehouse traditionnel et Analyses Big data

DATA WAREHOUSE (BI traditionnelle)

- Sources de données essentiellement internes, connues et structurées
- Modèles de données stables
- La majorité des données sont des données historiques
- De nombreux rapports produits de manière récurrente

PLATEFORME BIG DATA

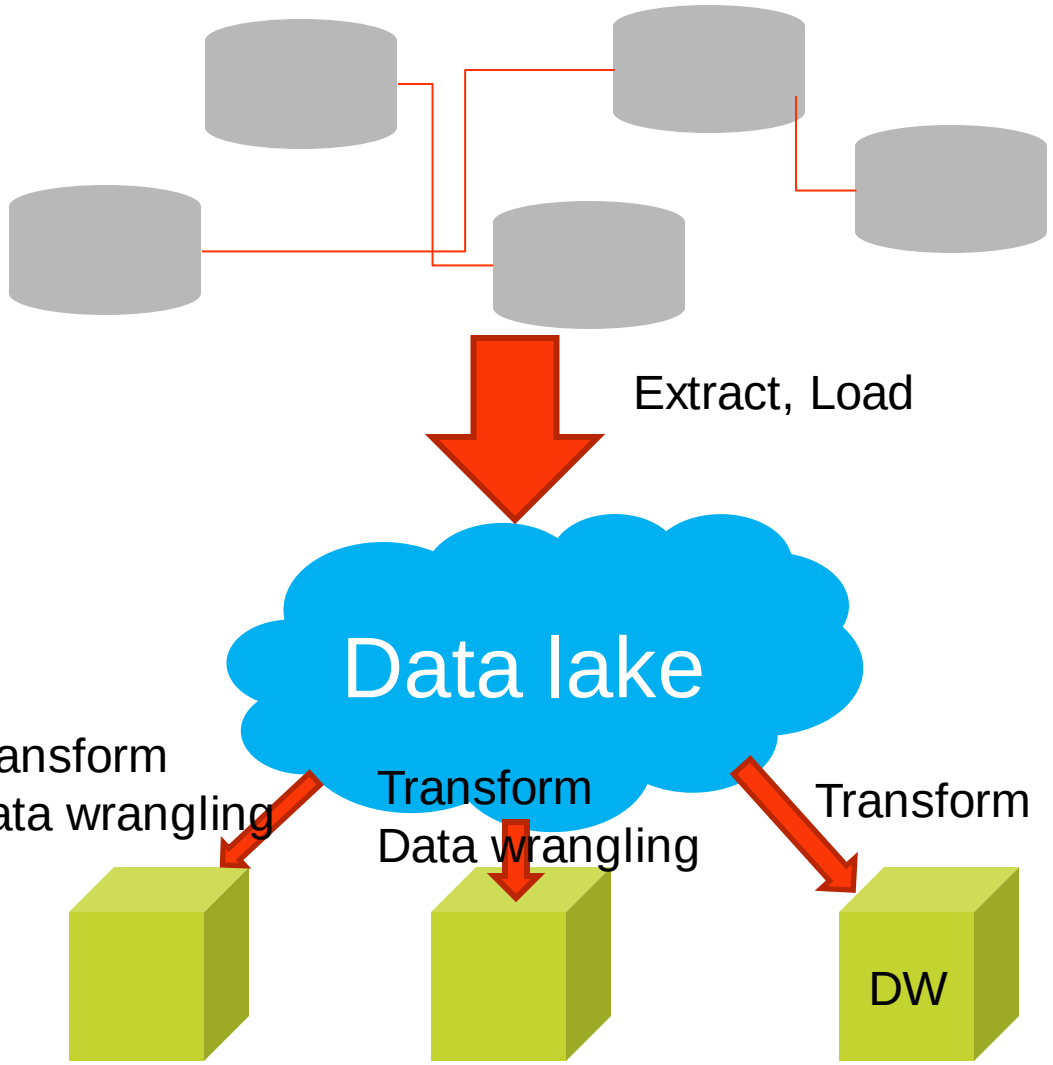
- Nombreuses sources externes
- Importants volumes de données non-structurées
- Besoin d'itérations rapides pour expérimenter des hypothèses
- L'analyse est faite sur des données qui restent dans leur état brut



Sources traditionnelles

Nouvelles sources

Data lake



Sources de Données opérationnelles

Analytiques

Data warehouse vs datalake

Critère	Datawarehouse	Data lake
Données	Structurées	(semi)-Structurées, non structurées, brutes
alimentation	ETL	EL
qualité	en amont	En aval
Stockage	Coûteux pour grosses volumétries	Peu cher
Intégration	Schéma unifié	Pas de schéma unifié
Agilité	Faible	forte
Utilisateurs	Métiers	Data scientists

Organisation d'un data lake

Objectives
- ✓ Plan the structure based on optimal data retrieval
- ✓ Avoid a chaotic, unorganized data swamp

Common ways to organize the data:

Time Partitioning
Year/Month/Day/Hour/Minute

Subject Area

Security Boundaries
Department
Business unit
etc...

Downstream App/Purpose

Data Retention Policy
Temporary data
Permanent data
Applicable period (ex: project lifetime)
etc...

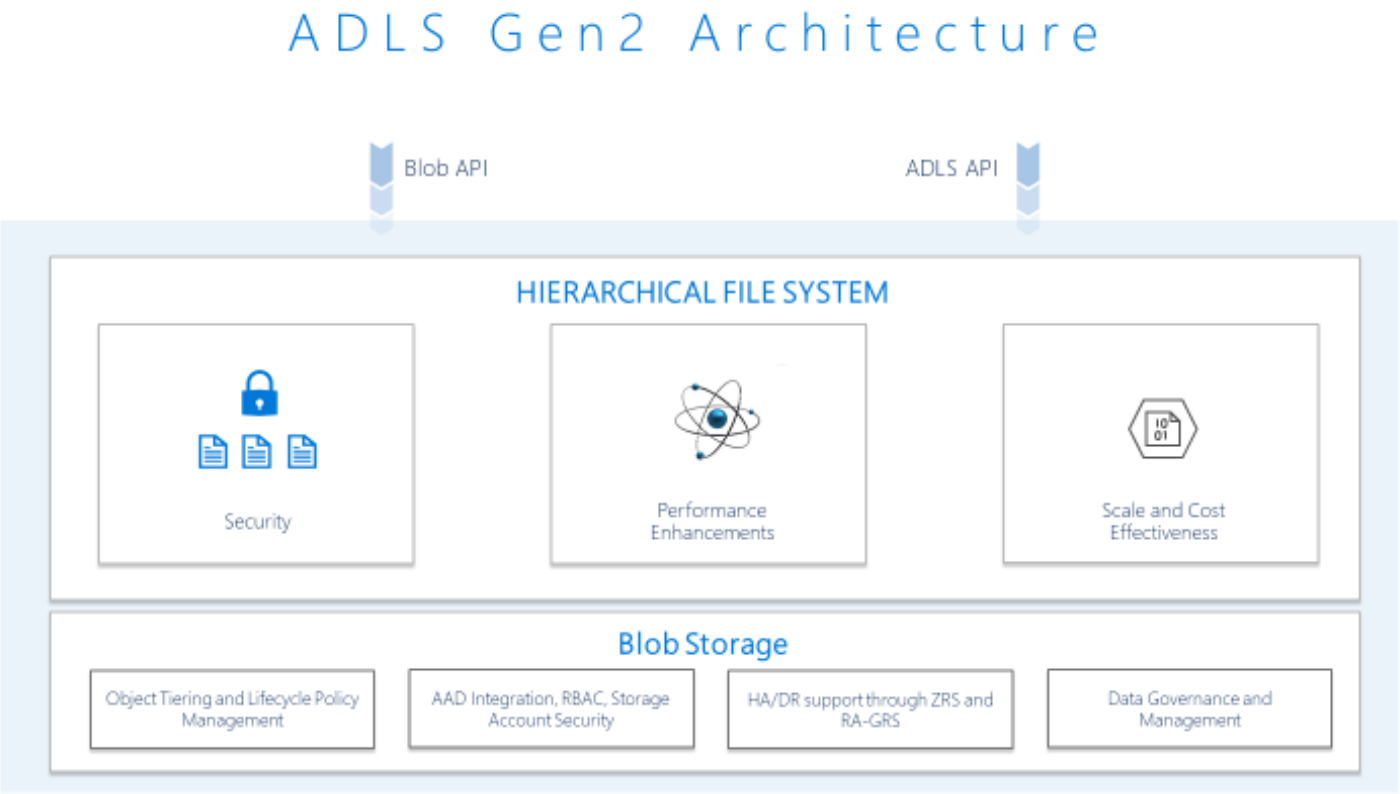
Business Impact / Criticality
High (HBI)
Medium (MBI)
Low (LBI)
etc...

Owner / Steward / SME

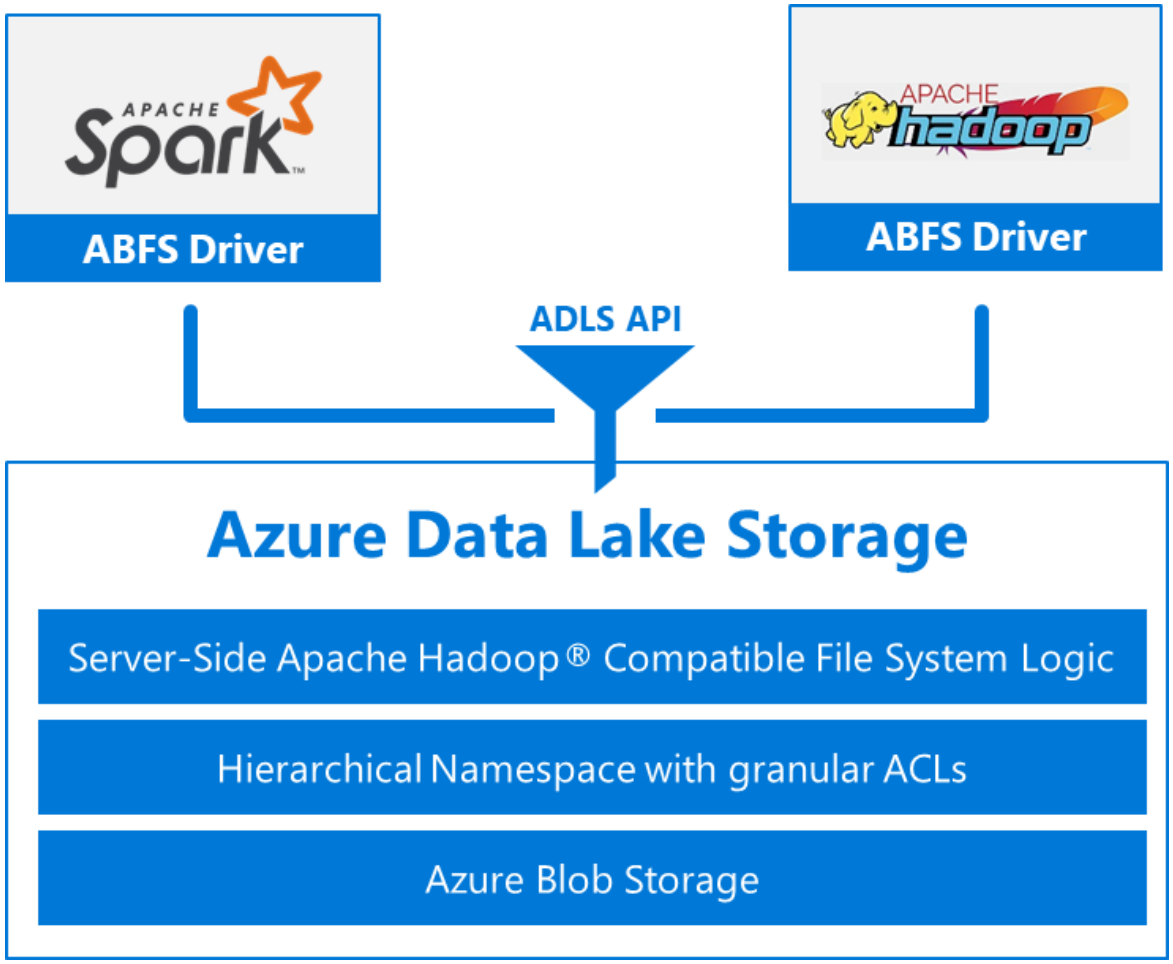
Probability of Data Access
Recent/current data
Historical data
etc...

Confidential Classification
Public information
Internal use only
Supplier/partner confidential
Personally identifiable information (PII)
Sensitive – financial
Sensitive – intellectual property
etc...

Azure Data Lake Store Gen2



Azure Data Lake Store Gen2 (2)



Quelques fournisseurs

- **AWS data lake**
- **Azure data lake**
- **Delta lake de databricks**
- **…**

Programme

- Introduction et contexte
- Infrastructures pour le big data
  - Cloud
  - Bases de données NoSQL
- Du BI au big data
  - Bases NoSQL
  - Entrepôts de données
  - Data lake
- **De Hadoop à Spark**
  - Typologie des outils big data
  - Mapreduce et Hadoop
  - spark

Pile logicielle Big data

Dataviz

Machine learning

Requêtage SQL, DW

Approches déclaratives

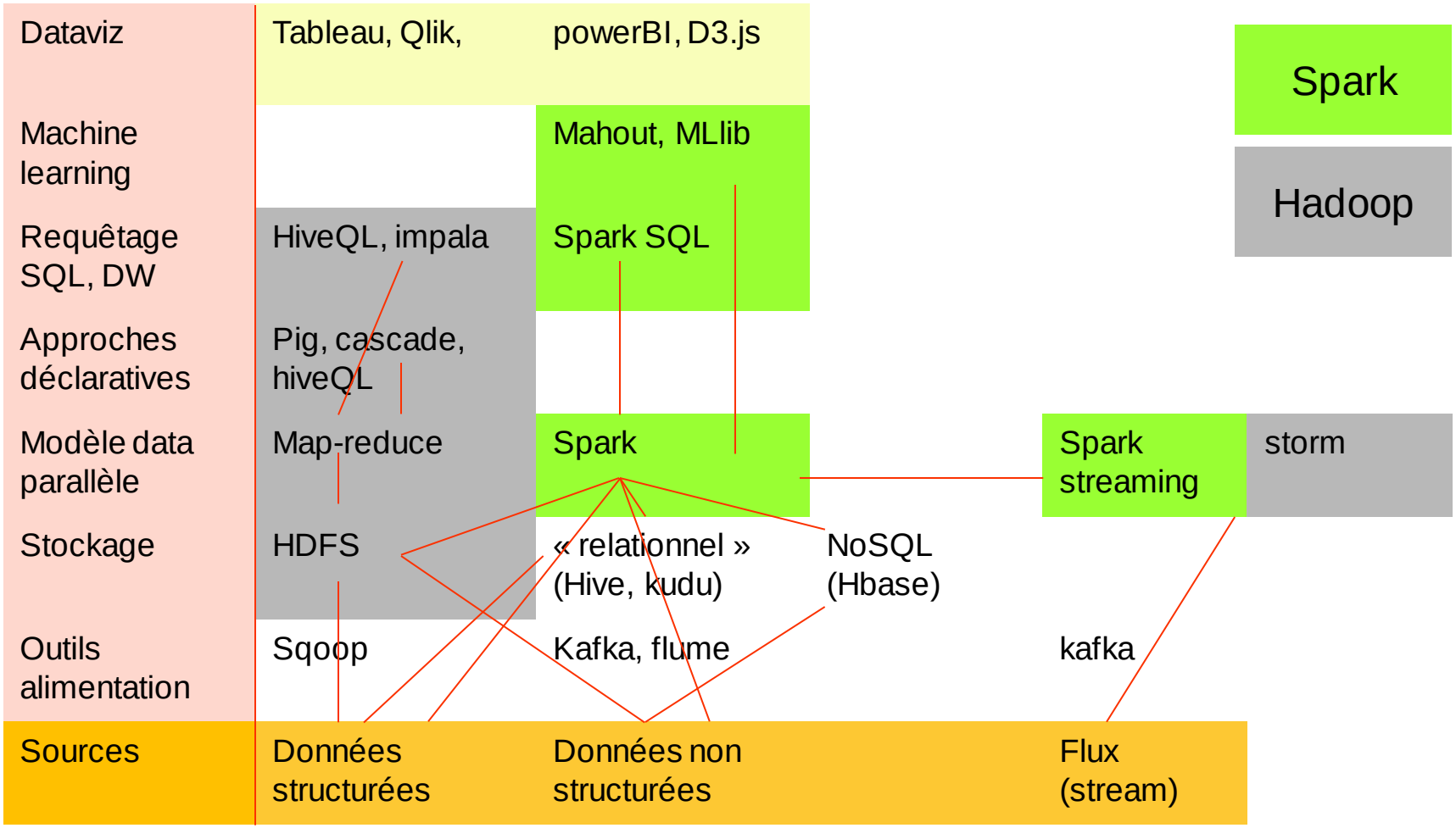
Modèle data parallèle

Stockage

Outils alimentation

Sources

Piles logicielles Big data



Batch Processing - Définition

Définition

- Exécution d'un programme sans intervention humaine
- Opère sur des données disponibles dans des fichiers collectées sur une période donnée
- Résultat en temps différé

Avantages

- Très efficace pour traiter de grandes quantités de données
- Très robuste pour des traitements complexes

Inconvénients

- Des latences de quelques minutes, voire des heures
- Pas de résultats partiels

Quand et Comment l'utiliser ?

- Le temps d'exécution ne doit pas être une contrainte
- Lancer en tâches périodiques pour des traitements approfondis

Batch – Cas d'utilisation fonctionnel

Le contexte

- The New York Times a besoin de convertir l'ensemble des articles publié entre 1851 et 1980 en PDF
- Chaque article est composé de plusieurs fichiers TIFF
- Le code pour générer un PDF est relativement simple

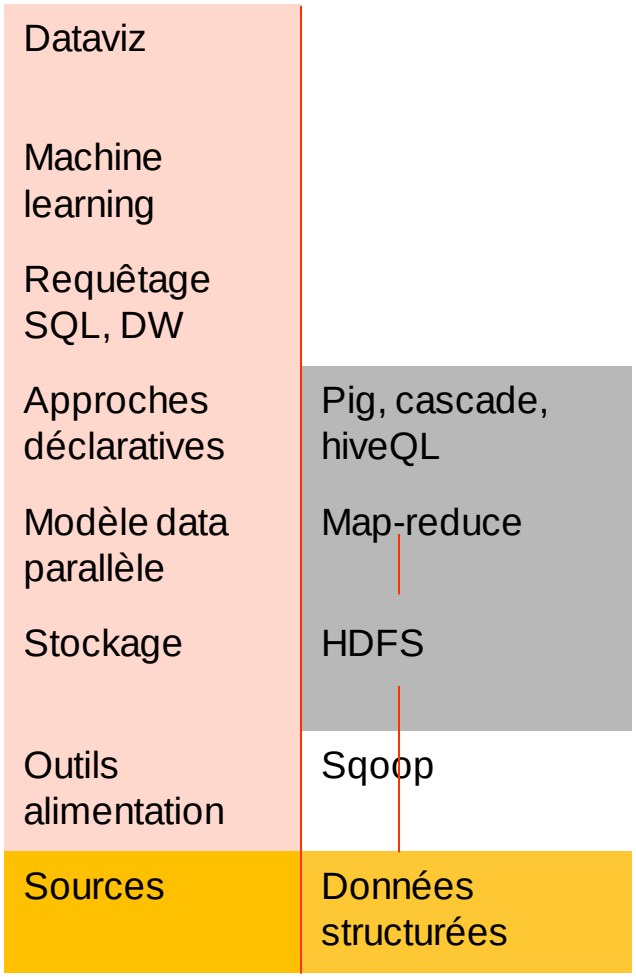
La Technologie

- Amazon Simple Storage Service (S3)
- Amazon Elastic Compute Cloud (EC2)
- Amazon Elastic MapReduce (EMR)

Le Résultat

- 4To de fichiers scannés envoyés à S3
- 100 instances EC2 utilisées
- Conversion en 24h de 4To de fichiers en 1,5To de PDF

Cas d'usage 1 : batch processing



Hadoop

Stream Processing - Définition

Définition

- En entrée un flux continu
- Traitement en flux tendu
- En sortie un flux continu

Avantages

- Quasi temps réel
- Besoin de moins de ressources car les données sont traitées petit à petit

Inconvénients

- Traitements relativement simples
- Peut avoir besoin d'une file d'attente

Quand et Comment l'utiliser ?

- Besoin de traiter les données en continu
- Besoin d'un minimum de latence

Streaming – Cas d'utilisation fonctionnel

Le contexte

- Analyser les réseaux sociaux en temps réel pour détecter des signaux positifs ou négatifs
- Limiter la perte d'informations (tweets produits mais non traités en temps réel)

La Technologie

- Storm ou spark streaming : traiter à la volée
- Kafka : récupérer les tweets de manière fiable

Le Résultat

- Capacité à détecter en temps réel des signaux négatifs lors de la sortie d'un produit et contrer au plus tôt

Cas d'usage 2 streaming : tweets

Dataviz

Machine learning

Requêtage SQL, DW

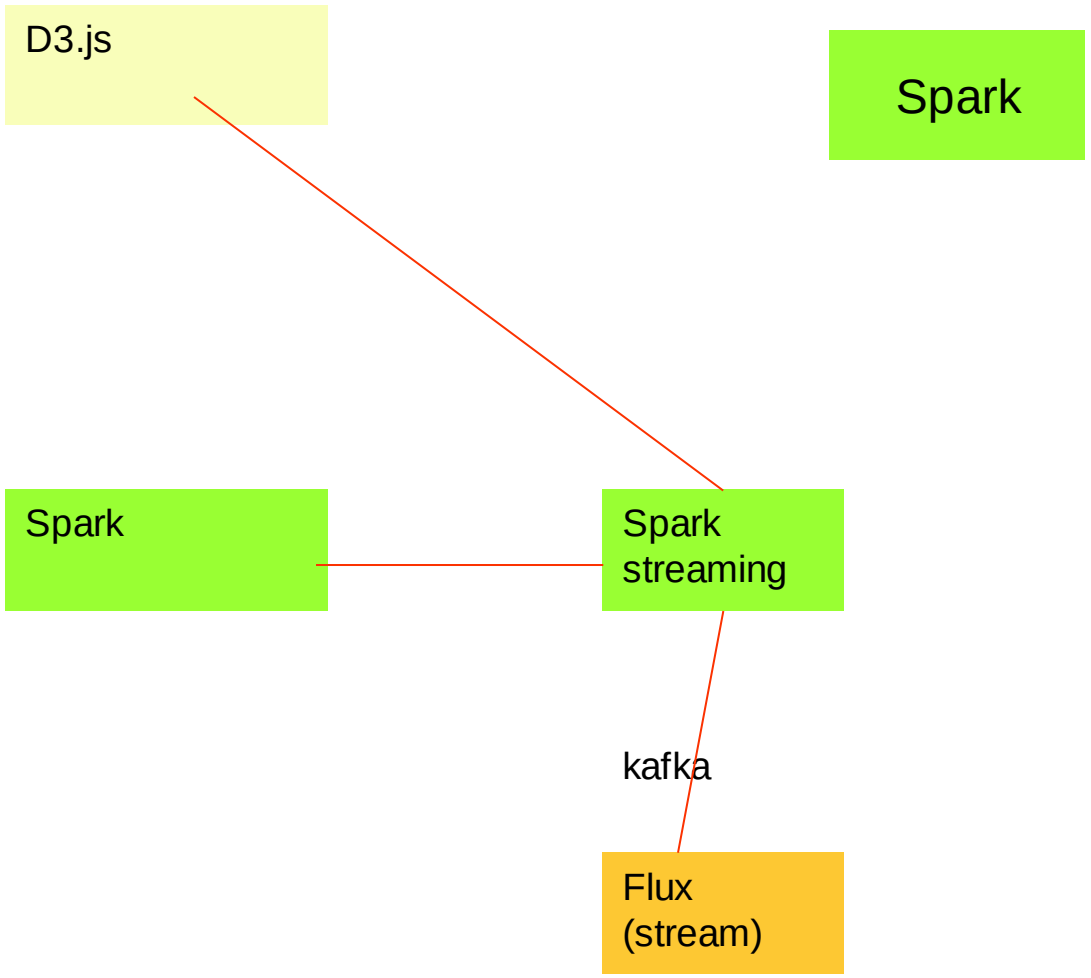
Approches déclaratives

Modèle data parallèle

Stockage

Outils alimentation

Sources



Machine learning - Définition

Définition

- En entrée un historique de données multidimensionnelles
- Traitement en deux phases : construction d'un modèle par apprentissage (phase très lourde) et application du modèle sur de nouvelles données (phase beaucoup plus légère)
- En sortie une prédiction (classification, scoring, recommandation, …)

Avantages

- Automatisation de traitements complexes pas facile à modéliser
- Une fois le modèle construit, le traitement est simple

Inconvénients

- Phase d'apprentissage nécessite beaucoup de données (éventuellement étiquetées) et est coûteuse en temps de calcul
- Peut évoluer dans le temps et perdre en précision

Quand et Comment l'utiliser ?

- Problème difficilement modélisable à priori
- Si on dispose de données d'entrainement

Machine learning – classification de dossiers clients

LE CONTEXTE

- Analyser des dossiers clients pour déterminer si on peut leur accorder un crédit consommation
- Diminuer le temps de réponse sur des demandes de crédit (le ML permet aux analystes de passer du temps sur les dossiers plus difficiles à traiter)

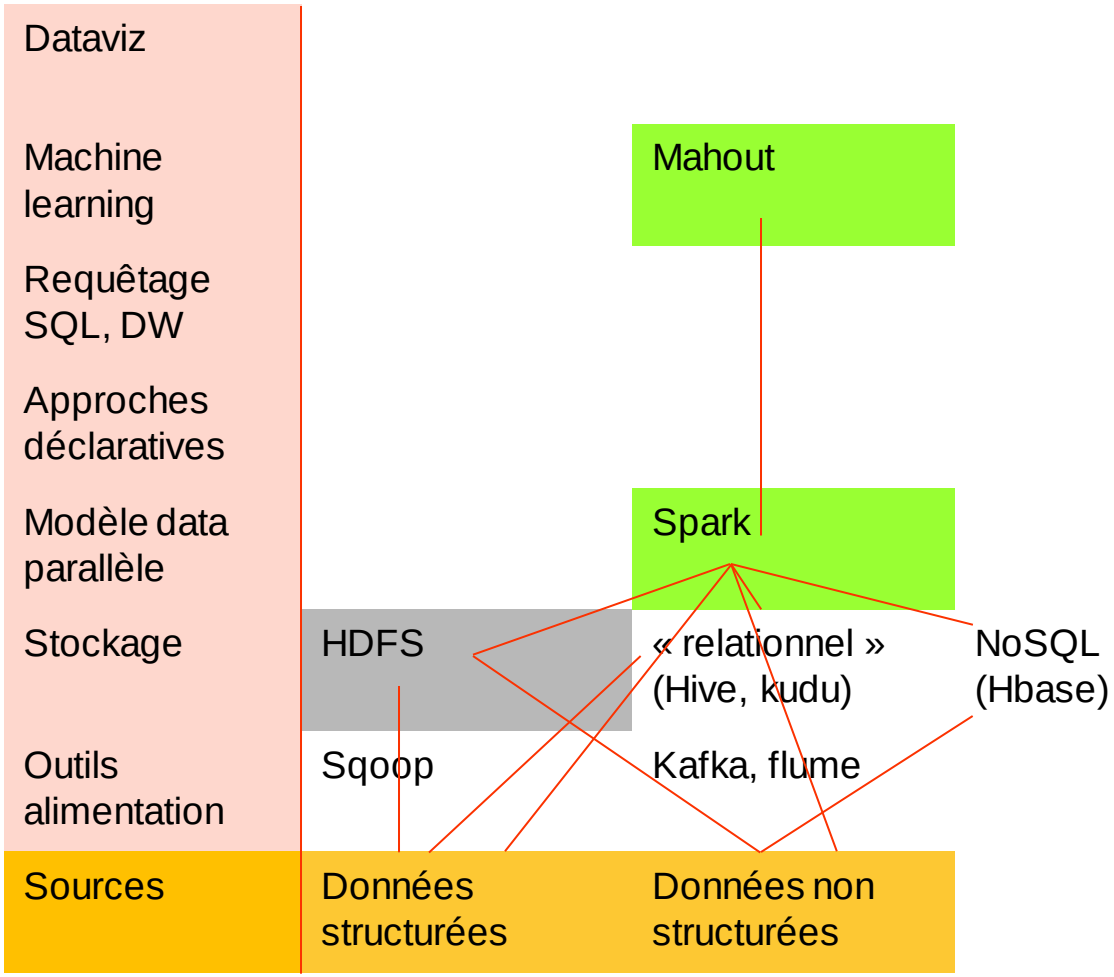
LA TECHNOLOGIE

- Construire un jeu de données de dossiers clients avec la décision de crédit associée (historique des demandes précédentes)
- Avec Mllib construire un modèle de clustering à partir du jeu de données
- Une fois le modèle construit il peut être appliqué sur un nouveau dossier

LE RÉSULTAT

- Étiquetage d'un dossier de demande de crédit avec très favorable, réservé, défavorable

Cas d'usage ML : classification de dossiers clients (modèle)



Infrastructure matérielle pour le bigdata

- **Clusters spécifiques**
- **Cloud**
- **BD parallèles (hard et soft)**
  - Oracle Exadata
  - Teradata
  - Microsoft SqlServer PDW

- **Engouement pour les GPU (composants de calcul pour le graphique) et leur adaptation à l'IA/DL : nvidia, amd notamment**
- **Autre hardware spécifique :**
  - TPU de google (tensor process unit)
  - Huawei Atlas 500 AI : carte dédiée basse consommation et faible coût

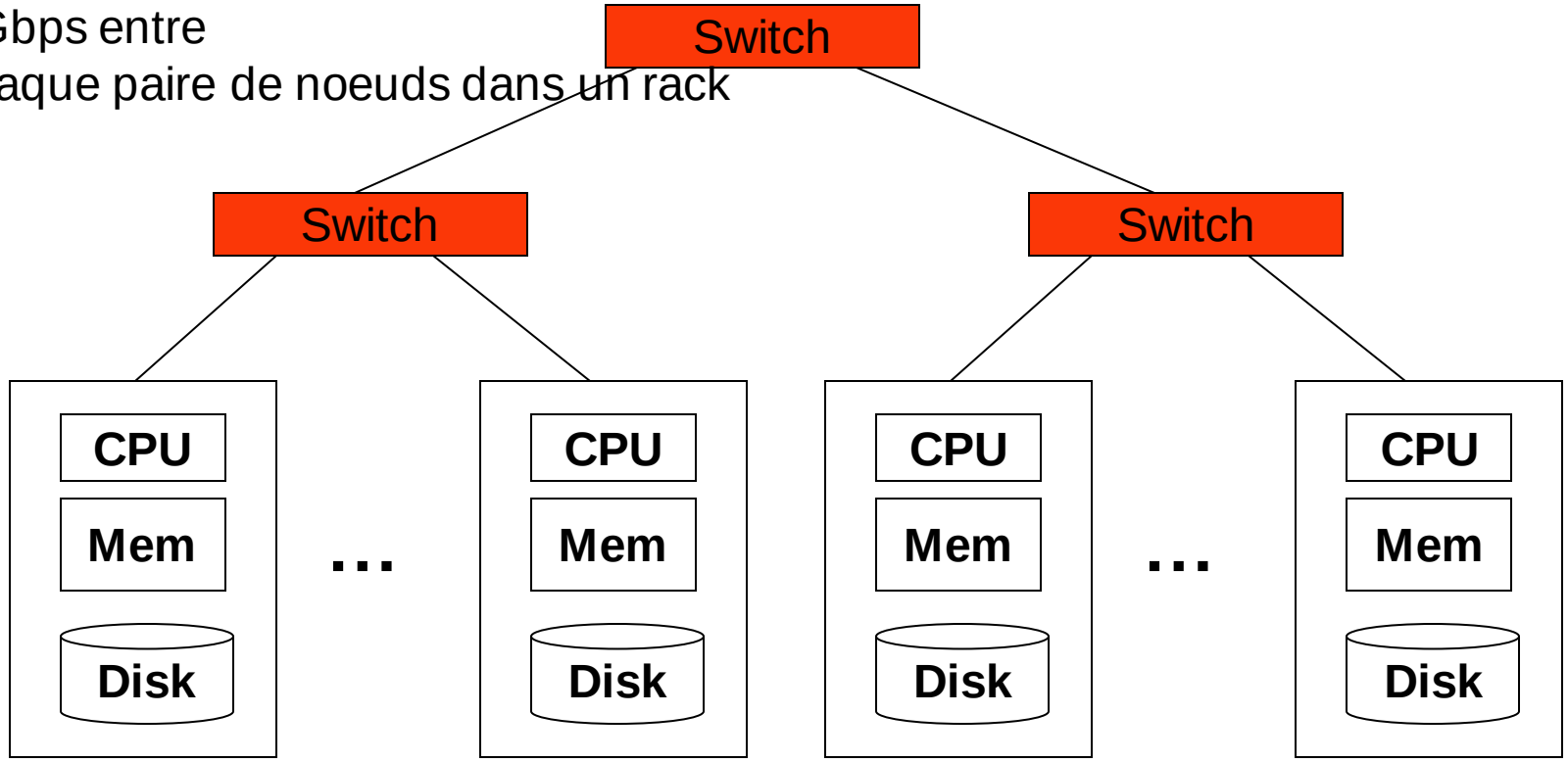
Programme

- Introduction et contexte
- Infrastructures pour le big data
  - Cloud
  - Bases de données NoSQL
- Du BI au big data
  - Bases NoSQL
  - Entrepôts de données
  - Data lake
- **De Hadoop à Spark**
  - Typologie des outils big data
  - Mapreduce et Hadoop
  - spark

Architecture d'un cluster

2-10 Gbps backbone entre racks

1 Gbps entre
Chaque paire de noeuds dans un rack



Chaque rack contient 16-64 noeuds

En 2011 il était estimé que Google avait 1M machines, http://bit.ly/Shh0RO

Calcul à large échelle

- **Calcul large échelle pour du "filtrage" de données sur des machines standards**
- **Défis :**
  - **Comment distribuer le calcul?**
  - **Comment la rendre la programmation distribuée facile?**
  - **Prendre en compte les fautes :**
    - Un serveur a un MTBF de 3 ans (1,000 jours)
    - avec 1000 serveurs, une panne par jour
    - ~1M machines en 2011 pour Google
      - 1,000 machines en faute chaque jour!

Les fautes sont fréquentes …

Typical first year for a new cluster (Jeff Dean, Google):

- ~0.5 overheating (power down most machines in <5 mins, ~1-2 days to recover)
- ~1 PDU failure (~500-1000 machines suddenly disappear, ~6 hours to come back)
- ~1 rack-move (plenty of warning, ~500-1000 machines powered down, ~6 hours)
- ~1 network rewiring (rolling ~5% of machines down over 2-day span)
- ~20 rack failures (40-80 machines instantly disappear, 1-6 hours to get back)
- ~5 racks go wonky (40-80 machines see 50% packet loss)
- ~8 network maintenances (4 might cause ~30-minute random connectivity losses)
- ~12 router reloads (takes out DNS and external vips for a couple minutes)
- ~3 router failures (have to immediately pull traffic for an hour)
- ~dozens of minor 30-second blips for DNS
- ~1000 individual machine failures
- ~thousands of hard drive failures
- Slow disks, bad memory, misconfigured machines, flaky machines, etc.

- Long distance links: wild dogs, sharks, dead horses, drunken hunters, etc.

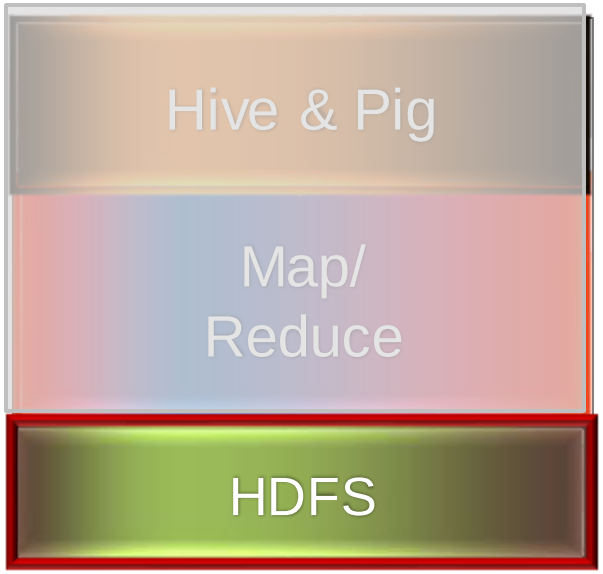
Idée et Solution

- **Problème : copier des données sur un réseau prend beaucoup de temps**
- **Idée :**
  - Garder les calculs proches des données
  - Stocker les fichiers de manière redondante pour la fiabilité
- **Map-reduce s'attaque à ces problèmes**
  - Modèle de programmation data parallèle de Google
  - Manière élégante de travailler sur des big data
  - **Système de stockage distribué**
    - Google: GFS. Hadoop: HDFS
  - **Modèle de programmation**
    - Map-Reduce

Leskovec, A. Rajaraman, J. Ullman:

HDFS – Hadoop Distributed File System

- **Base de la pile Hadoop**
- **Objectifs de HDFS :**
  - Scalable sur des 1000 de noeuds
  - Tolérant aux fautes (matérielles et logicielles)
  - Cible un petit nombre de très gros fichiers
  - Écrire une fois, lire de nombreuses fois
- **Organisation hiérarchique classique en répertoires et fichiers**
- **Très portable**



MapReduce

Use case :

- **Un très grand document**
- **Compter le nombre de fois que chaque mot apparait dans le texte**
- **Exemple d'application :**
  - Analyser les logs des serveurs web pour déterminer les URL les plus populaires

Leskovec, A. Rajaraman, J. Ullman:

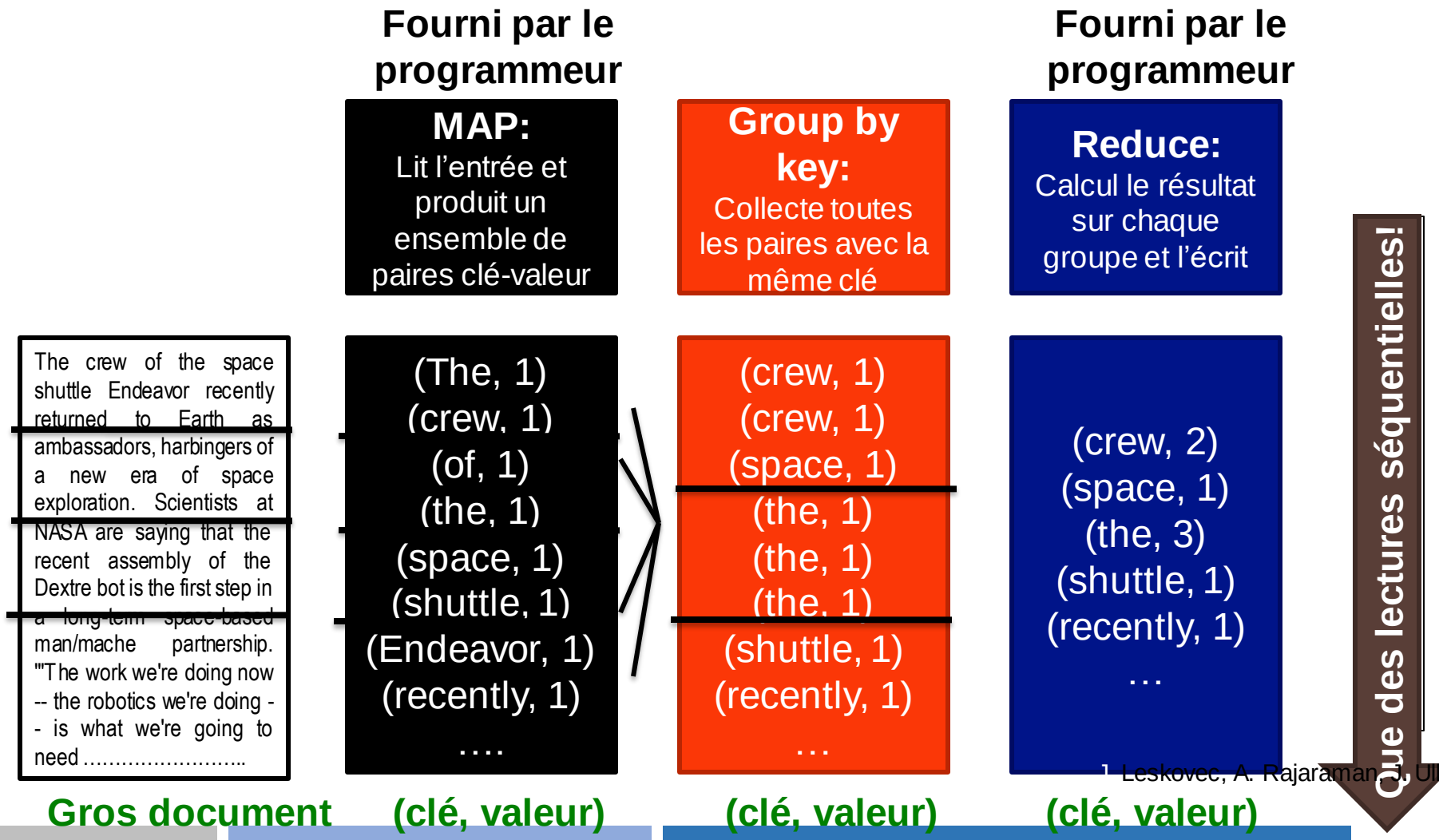
MapReduce : principes

- **Lire séquentiellement les données**
- **Map (local à un bloc) :**
  - Extraire des données ce qui nous intéresse
- **Group by key : trier et distribuer**
- **Reduce (global aux blocs initiaux) :**
  - Agréger, résumer, filtrer ou transformer
- **Ecrire le résultat**

Principe identique, seuls **Map** et **Reduce** à changer pour un problème spécifique

Leskovec, A. Rajaraman, J. Ullman:

MapReduce : compter les mots



[1] Leskovec, A. Rajaraman, J. Ullman:

Compter les mots avec MapReduce

```
map(key, value):
// key: nom du document; value: texte du document
   for each word w in value:
       emit(w, 1)
```

```
reduce(key, values):
// key: un mot; values: un ensemble de valeurs
       result = 0
       for each count v in values:
             result += v
       emit(key, result)
```

Leskovec, A. Rajaraman, J. Ullman:

Map-Reduce : l'environment

L'environnement de programmation Map-Reduce s'occupe de :

- **partitionner les données d'entrée**
- **ordonnancer l'exécution du programme sur un ensemble de serveurs**
- **Exécuter l'étape group by key**
- **Gérer les fautes**
- **Gérer la communication entre les serveurs**

Leskovec, A. Rajaraman, J. Ullman:

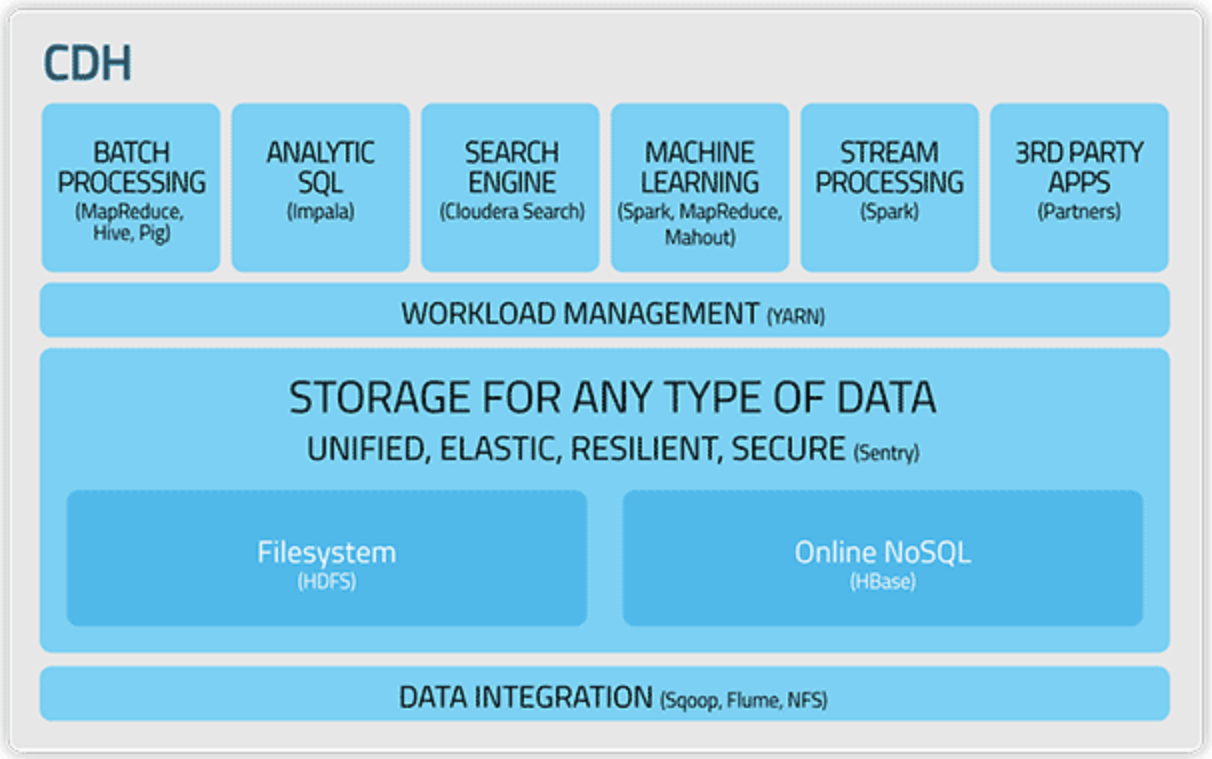
Faiblesses de Map-reduce

- Résultats intermédiaires stockés sur disque
- Pas de contrôle sur le placement des données
- Modèle d'exécution (pas d'optimisation basé sur des statistiques sur les données traitées)
- Ordonnancement des taches et lancement coûteux
- Modèle trop général et pas adapté au traitement itératif

Exercice

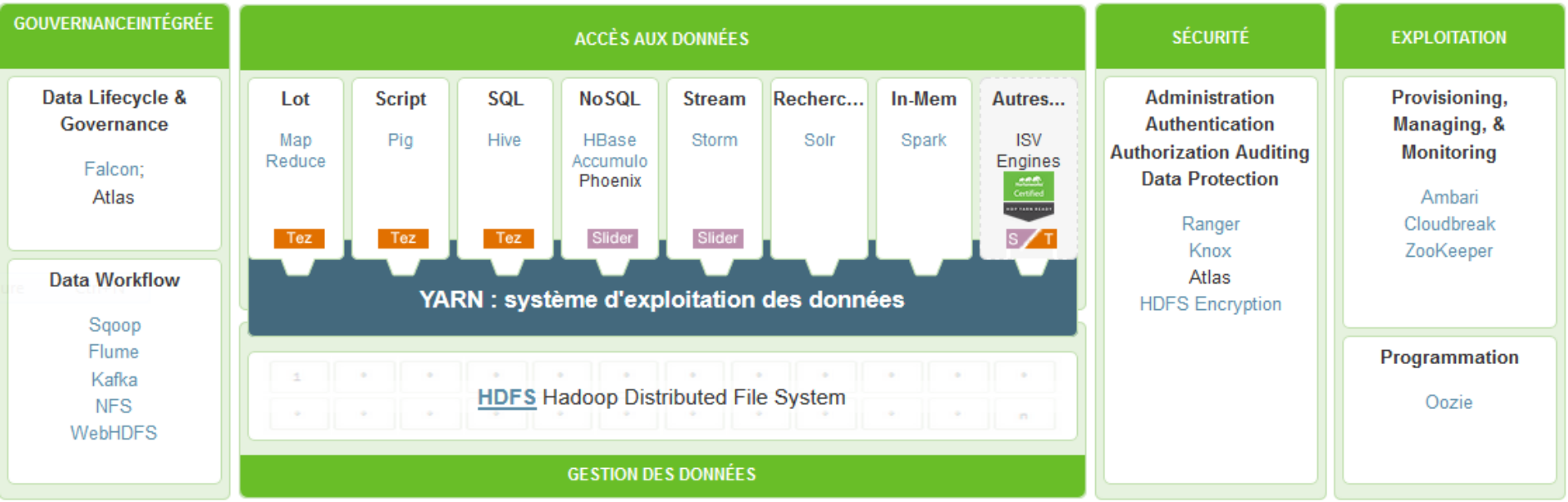
- **Donner le max d'un ensemble de nombres**
  - Entrée : un très gros fichier (distribué sur les nœuds du cluster Hadoop) constitué de lignes de nombres entiers (le séparateur est le blanc)
  - Sortie : valeur maximale rencontrée dans le fichier d'entrée
- **Anagrammes : on veut rechercher les anagrammes**
  - Entrée : un dictionnaire de mots (un mot par ligne)
  - Sortie : chaque mot du dictionnaire suivi de la liste (éventuellement vide) de ses anagrammes ou bien un mot suivi de la liste non vide de ses anagrammes (on ne donne pas les mots sans anagrammes et on donne une seule fois la liste des anagrammes)

Cloudera (www.cloudera.com)



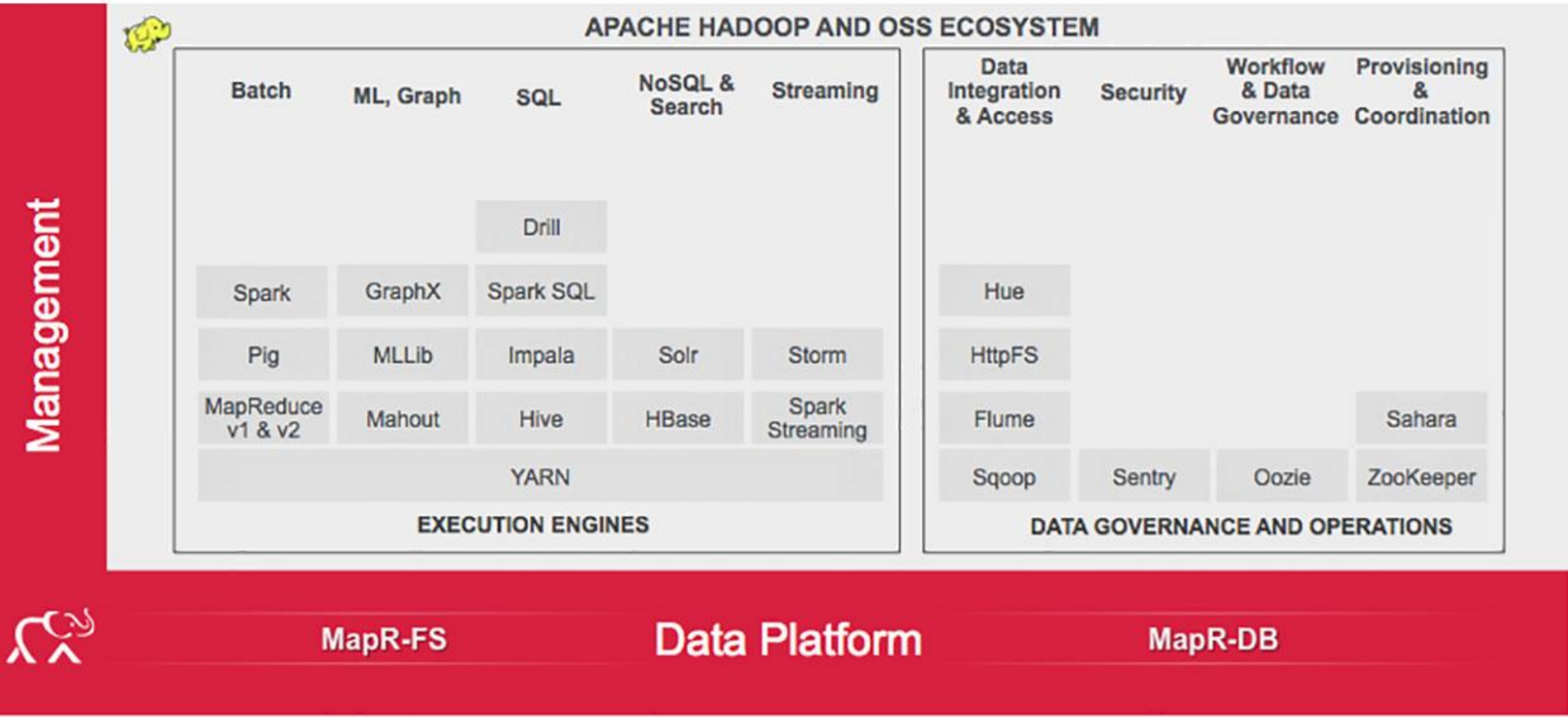
https://www.youtube.com/user/clouderahadoop

Plateforme Hortonworks HDP2.3



Encore maintenue mais fusionnée avec cloudera début 2019

MapR (www.mapr.com)



https://www.youtube.com/user/maprtech

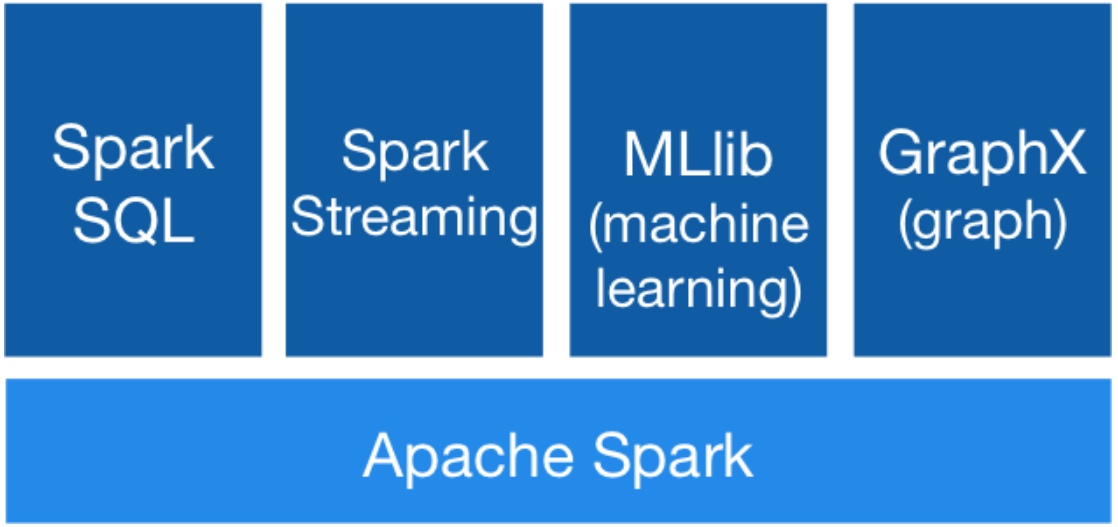
Programme

- Introduction et contexte
- Infrastructures pour le big data
  - Cloud
  - Bases de données NoSQL
- Du BI au big data
  - Data lake
  - Sources de données, anonymisation des données
  - Qualité des données
- **De Hadoop à Spark**
  - Typologie des outils big data
  - Mapreduce et Hadoop
  - spark

Historique de spark

- 2009 : lancement du projet à Stanford (AMPLab) comme alternative à map/reduce
- 2010 : version open source du code
- 2013 : devient projet Apache
- 2014 : création de databricks (développe une plateforme autour de spark)
- 2016 : lancement de spark 2.0
- 2020 : lancement de spark 3.0
- octobre 2022 : spark 3.3 (https://spark.apache.org/docs/latest/)

Apache Spark

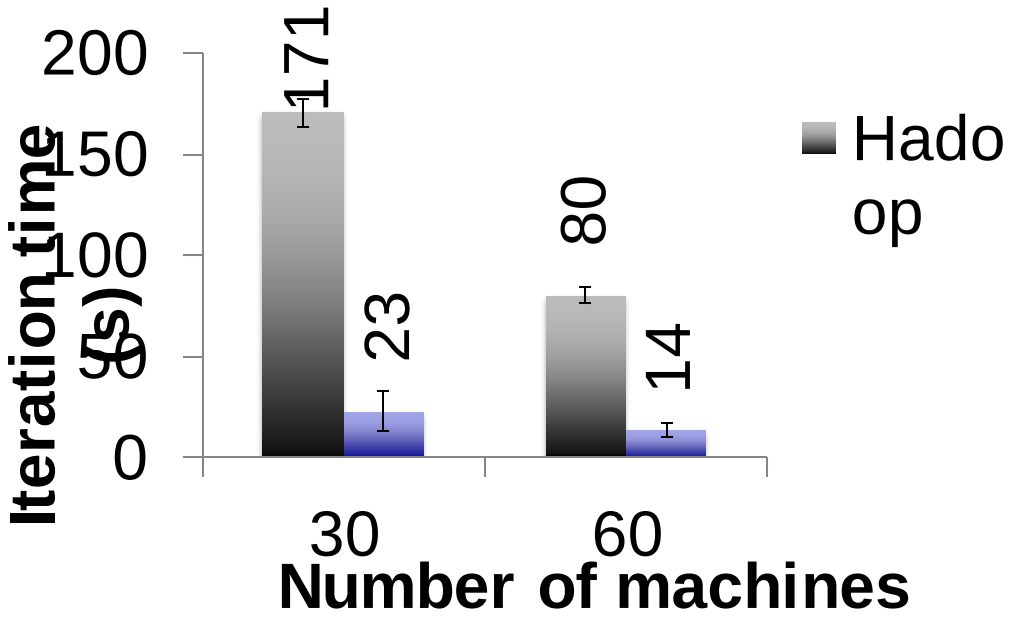


From databricks

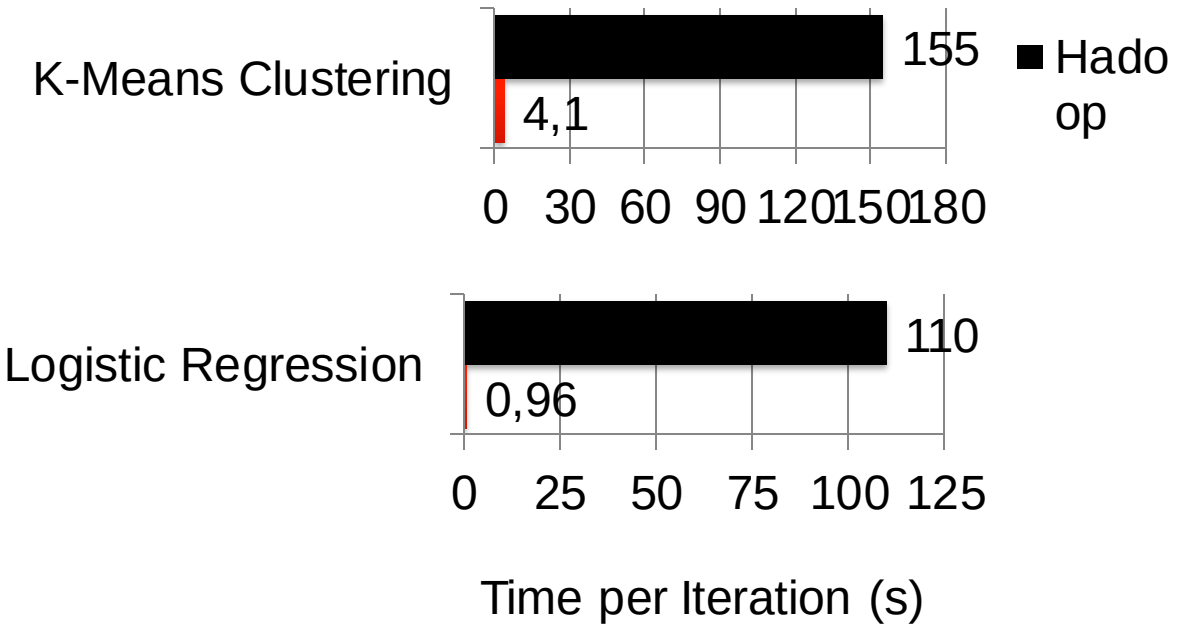
Spark

- Spark implante une abstraction simplifiée de mémoire partagée distribuée : Resilient Distributed Datasets (RDDs)
- RDDs maintiennent les données en mémoire et les reconstruisent automatiquement en cas de fautes
- Spark supporte un modèle général de calcul (DAGs) et pas seulement une topologie à deux niveaux comme map-reduce
- optimisé pour les faibles latences (capable de gérer de petites taches, contrairement aux moteurs Hadoop)
- Bien adapté aux algorithmes itératifs

Performance de SPARK-RDD pour PageRank



Perf. SPARK-RDD sur des algorithmes itératifs



SPARK : plusieurs structures de données et APIs associées

- **RDDs, DataFrames, and Datasets**
  - RDD – dans Spark 1.0, “bas niveau”
  - DataFrames – introduit dans Spark 1.3
  - Dataset – introduit dans Spark 1.6
- **Chacune avec des pros/cons/limitations**

DataFrame & Dataset

- **DataFrame :**
  - À la difference des RDD, données sont structurées avec des colonnes nommées, e.g. une relation ou un tableau xls
  - Impose une structure sur une collection distribuée de données
- **Dataset :**
  - Extension de DataFrame où les colonnes sont fortement typées (détection d'erreur à la compilation)

Wordcount : vue simplifiée

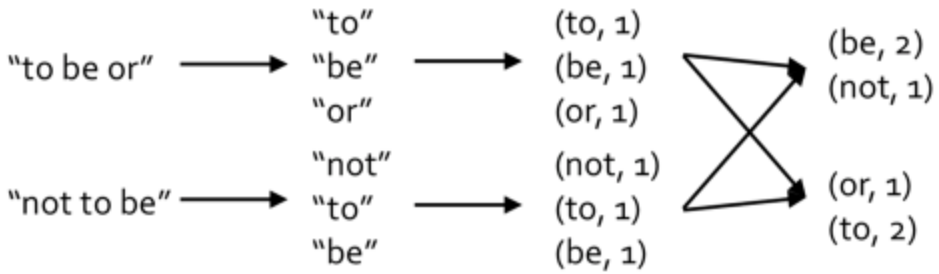
```
val file = sc.textFile("hdfs://...")

val counts = file.flatMap(line => line.split(" "))
   .map(word => (word,1))
   .reduceByKey(_ + _)

counts.saveAsTextFile("hdfs://...")
```

Transformation

Action



https://jingxuan.li/2018/05/05/COMP9313-Week-6-Spark-I/

Flots de données

- **Flots continus, non bornés, “rapides”, liés au temps de données élémentaires**
- **Présents dans de nombreuses applications**
  - Surveillance de réseau et ingénierie de trafic
  - Réseaux de capteurs, tags RFID
  - Logs des opérateurs télécoms
  - applications financières
  - logs Web
  - E-sciences
  - …
- **DSMS = Data Stream Management System**

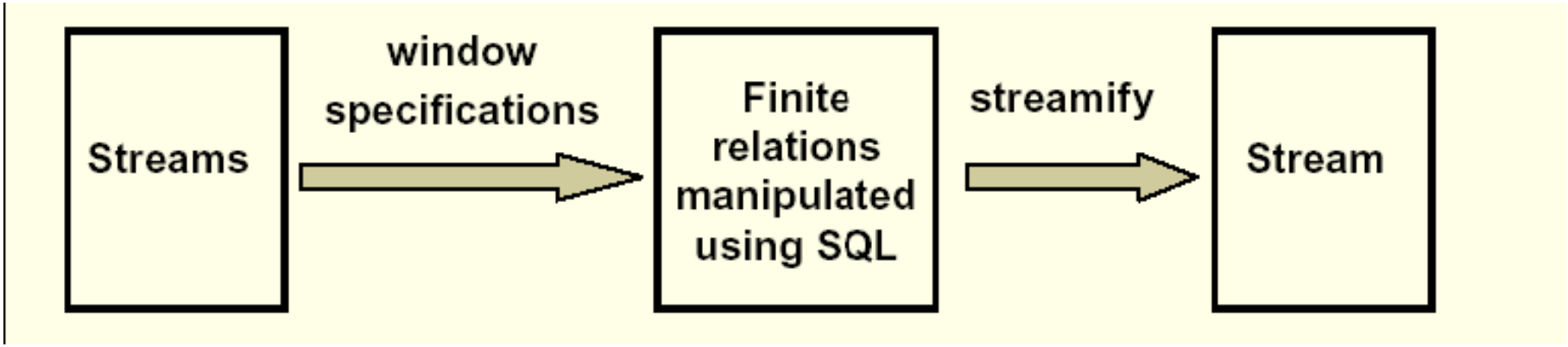
DBMS versus DSMS

- Relations persistantes
- Requêtes immédiates
- Accès aléatoire
- Plan d'accès déterminé par le gestionnaire de requêtes et par l'organisation physique de la BD

- Flots temporaires (& relations persistantes)
- Requêtes Continues
- Accès Séquentiel
- Caractéristiques des données et formats des arrivées imprévisibles

Windows

- **Méchanisme pour extraire une relation finie d'un flot infini**
- **De nombreuses variations**
  - Windows définie sur un attribut ordonné (e.g., temps)
  - Windows définie par un nombre de tuples
  - Windows définie par des marqueurs explicites

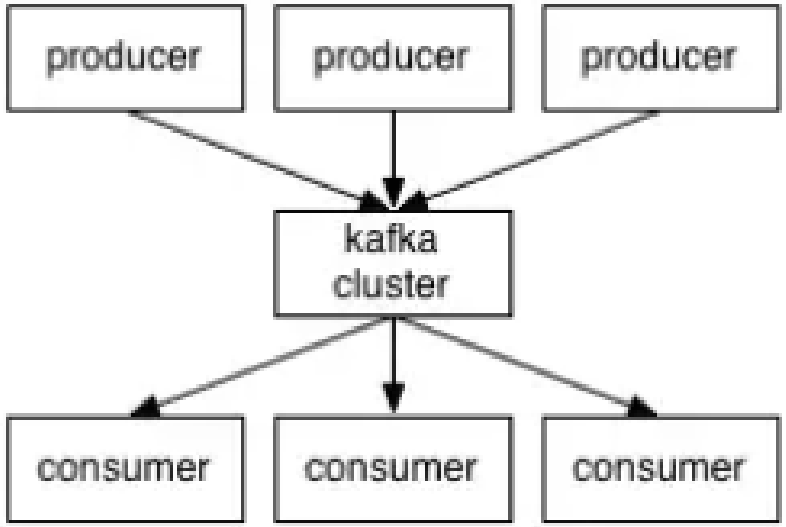


Gestion de flots de données

- **Collecter les flots**
  - À la volée
  - Stockage intermédiaire
- **Délivrer les flots (broker)**
  - Apache kafka
  - Flume
- **Traiter les flots**
  - Apache storm
  - Spark streaming

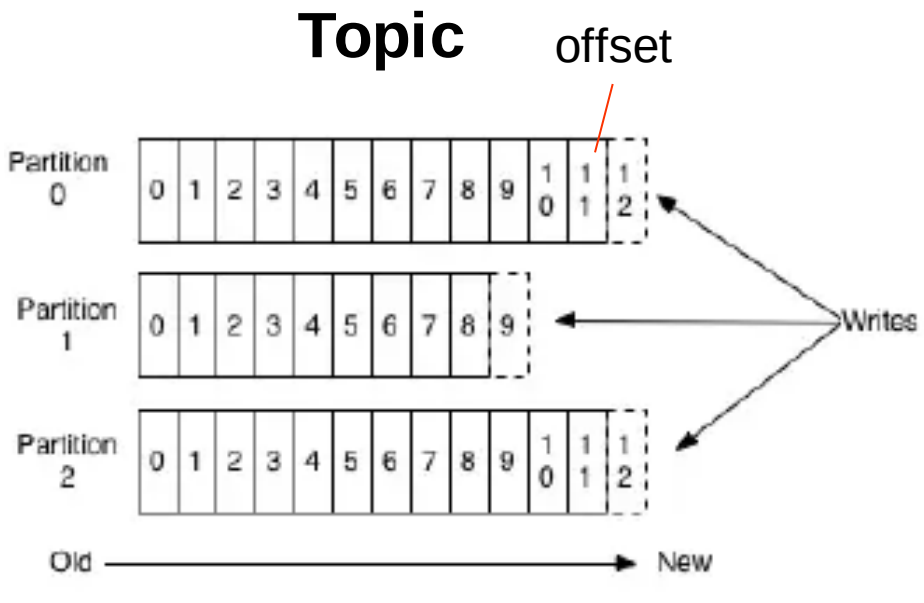
Apache Kafka

- **Service de log distribué, partitionné avec réplication**
- **Débit très élevé en lecture/écriture**
- **Faible latence**
- **Principaux concepts**
  - Cluster est un ensemble de broke
  - Topic
  - Partition : les messages d'un topic sont répartis dans plusieurs partitions sur plusieurs brokers (répartition en round robin ou via une clé du message)
  - Découple Producteur et consommateur

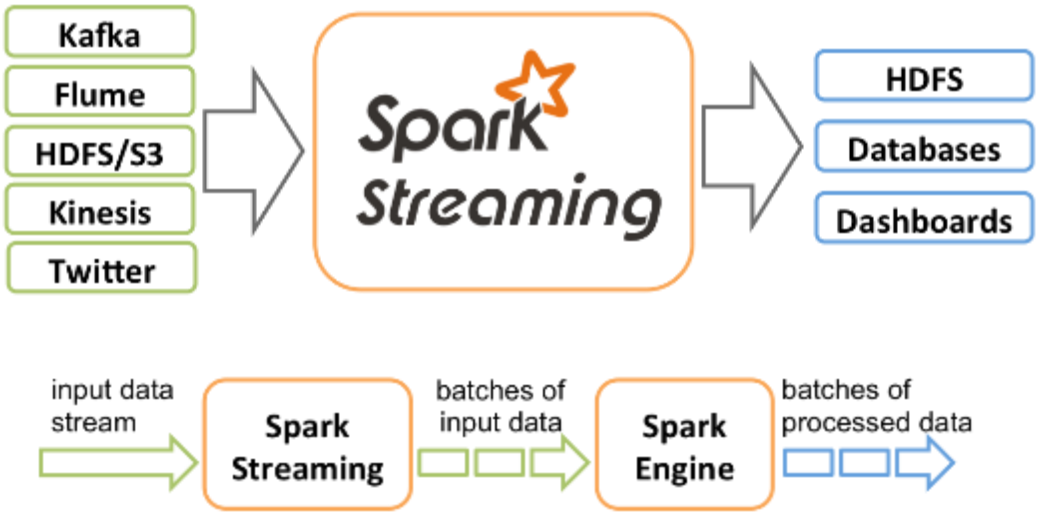


Apache Kafka (2)

- **Consommateur lit des messages dans une partition (maintien du dernier offset lu)**
- **Garanties**
  - Ordre total des messages (sur une partition)
  - Diffusion au moins une fois
  - Diffusion exactement une fois peut être implémentée par l'application
- **Pour aller plus loin https://www.youtube.com/watch?v=JalUUBKdcA0**



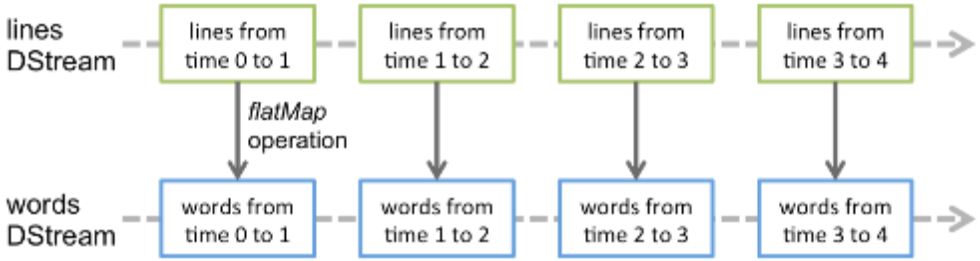
Spark streaming



DStreams

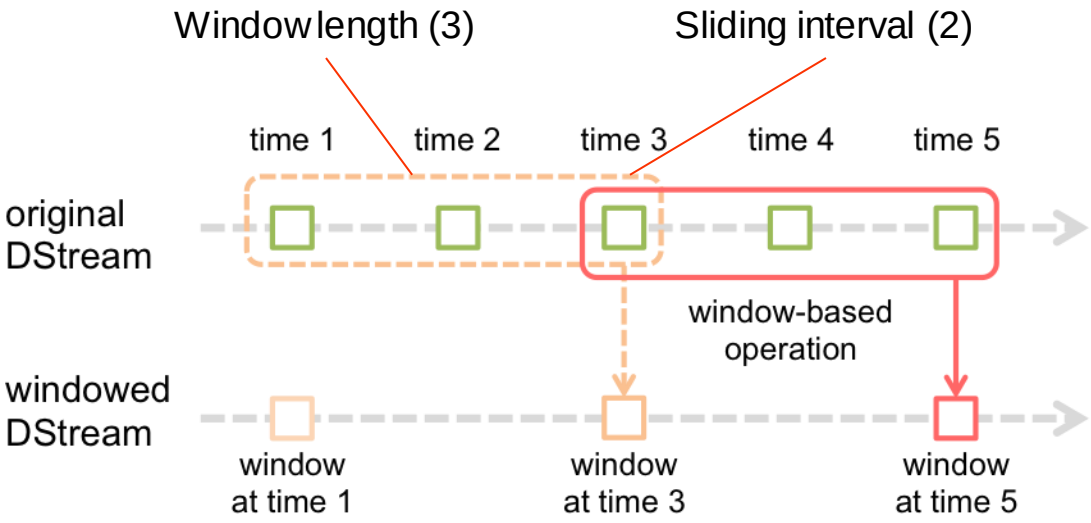
Abstraction de spark pour manipuler des flots de données infinis

Séquence continue de RDDs (discrétisation liée au temps, pas de temps fixé par le programmeur)



Se manipule ensuite avec des transformations analogues à celles des RDD

Windowed DStream



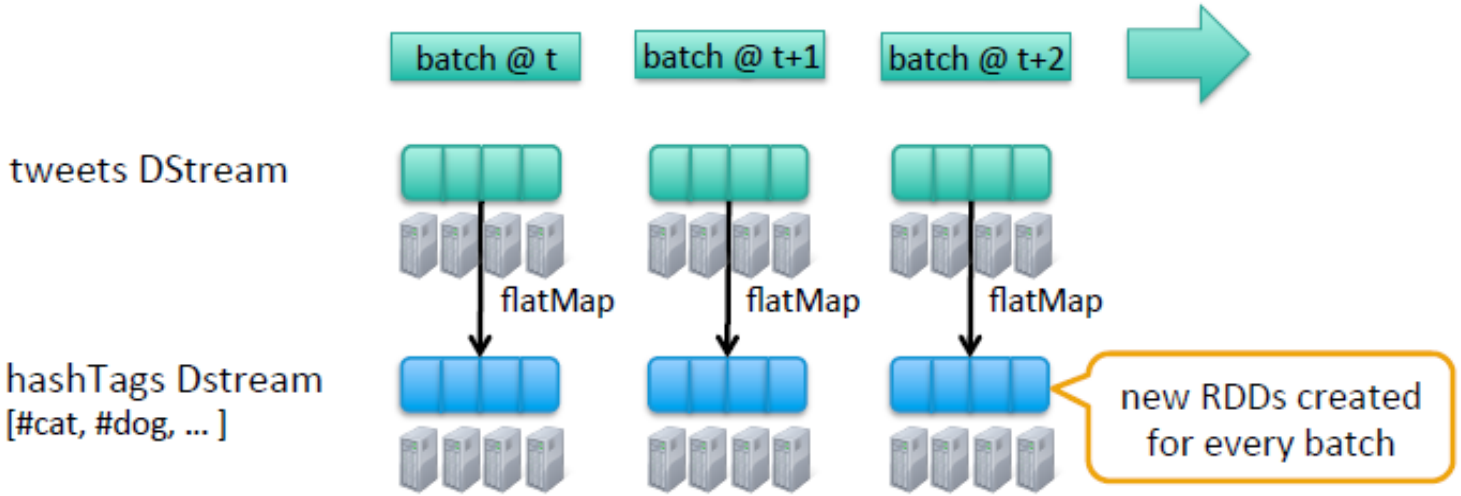
- Permet de définir des fenêtres sur les Dstreams
- Fenêtre = regroupement de Dstreams sur laquelle on va appliquer des transformations
- Fenêtre spécifiée par 2 paramètres : WL (nombre de Dstreams) et SI (chevauchement éventuel entre 2 fenêtres)

Exemple de traitement de flux twitter

```
val ssc = new StreamingContext(sparkContext, Seconds(1))
val tweets = TwitterUtils.createStream(ssc, None)
val hashTags = tweets.flatMap(status => getTags(status))
```

transformed DStream

transformation: modify data in one DStream to create another DStream



Exemple de traitement d'un flux twitter (2)

```
val sortie=
        hashtags.map(h -> h.getText().toLowerCase())
        .countByValue()
        .print();

// donne le nombre d'occurrences de chaque hashtag
```

MLlib

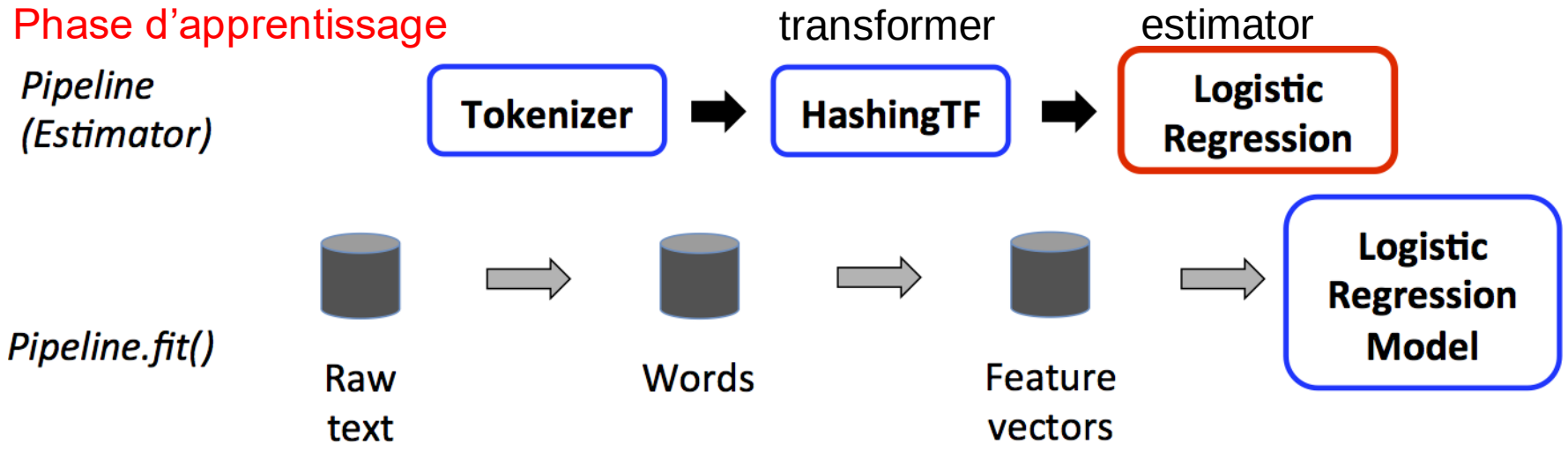
- **Encapsulation des algorithmes classiques de ML via des librairies utilisables soit sur des structures de données centralisées (Panda DataFrame en python) ou décentralisées (RDD/dataframe)**
- **Pipeline pour modéliser l'enchainement des taches d'apprentissage**
- **Deep learning en spark : databricks fournit une librairie (utilisant les pipelines) intégrant des algos classiques de DL ainsi que des interfaces vers Keras ou TensorFlow**
- **Parallélisation « automatique » si on utilise des RDD/dataframes**

Mllib (2)

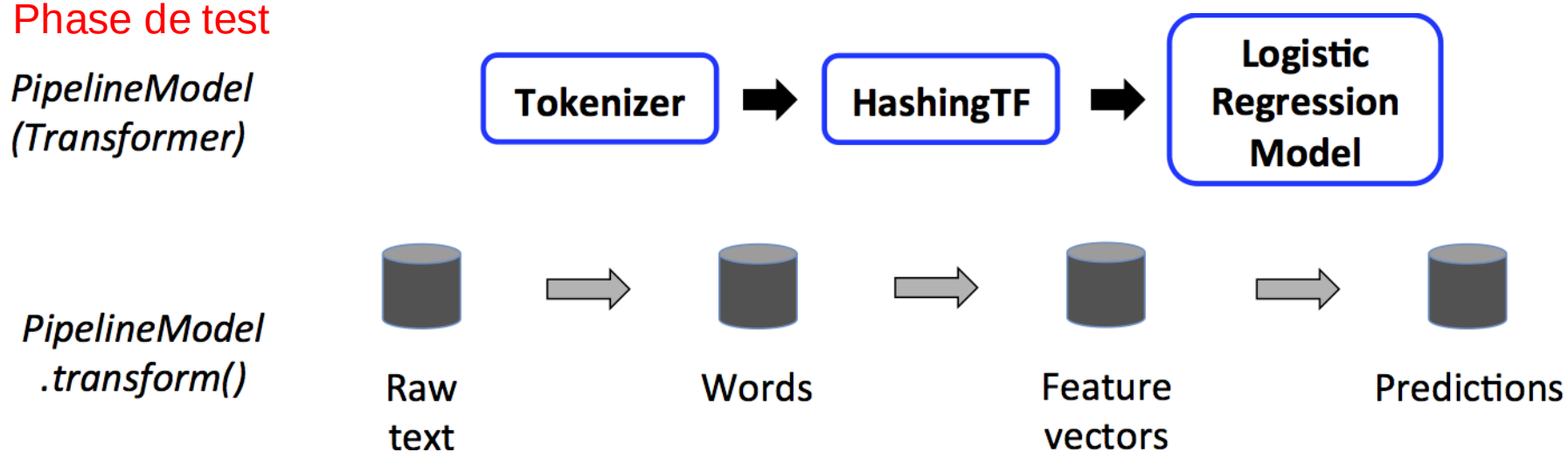
- https://spark.apache.org/docs/2.3.2/ml-guide.html
- ML Algorithms : classification, régression, clustering, filtrage collaboratif
- Featurization : extraction de « feature », transformation, réduction de dimensionalité et sélection
- Pipelines : outils pour construire, évaluer et optimiser des ML Pipelines
- Persistence : sauvegarde et chargement des algos, modèles et Pipelines
- Utilities: algèbre linéaire, statistiques, gestion données, etc.

ML pipeline

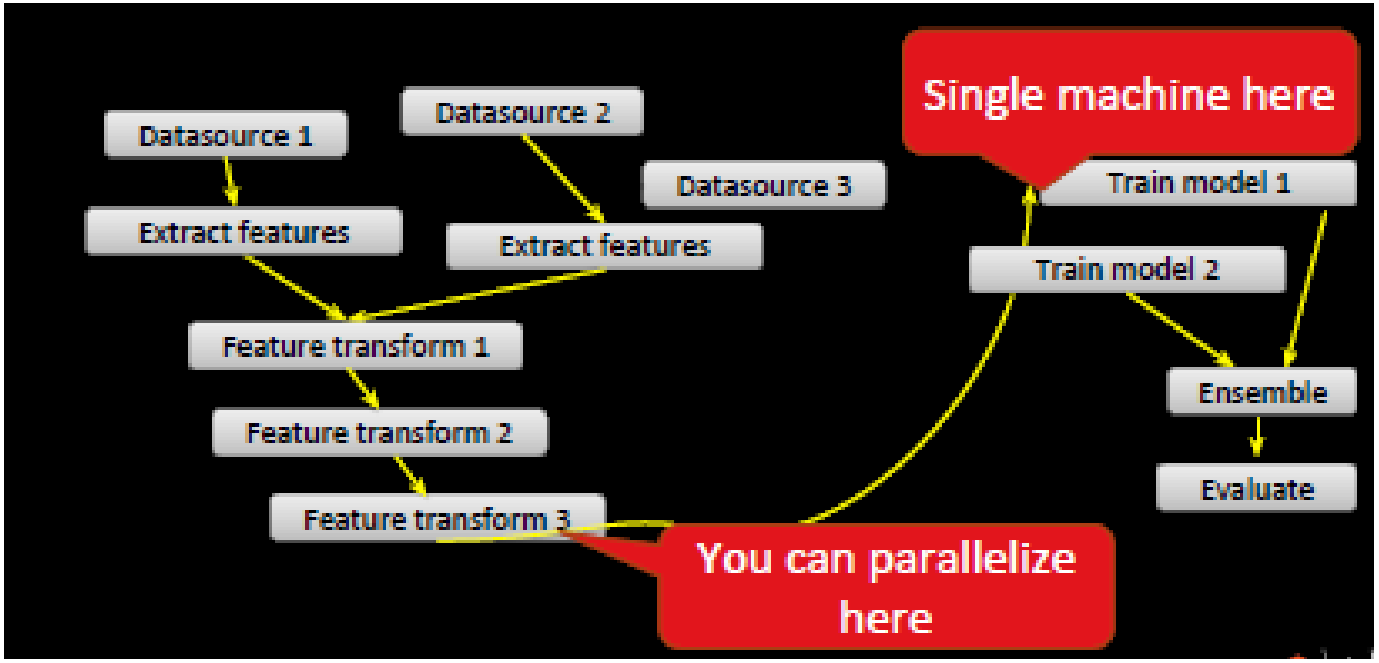
Phase d'apprentissage



Phase de test



Mllib et parallélisation

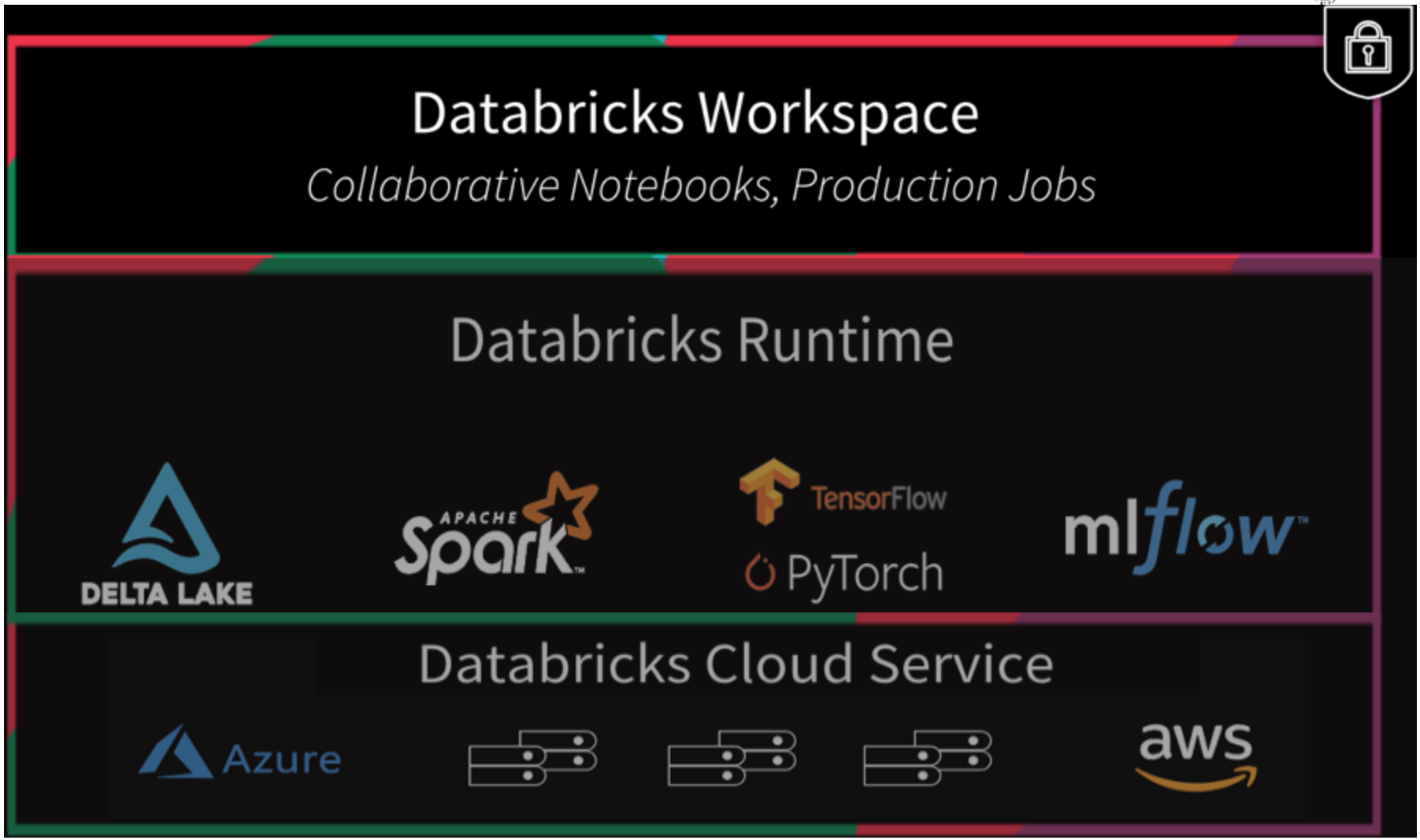


https://fr.slideshare.net/databricks/understanding-parallelization-of-machine-learning-algorithms-in-apache-spark

Synthèse sur Spark

- **Avantages**
  - Concept de RDD fournit une abstraction simple pour la programmation distribuée
  - Pile big data très complète et très unifiée au niveau programmation (RDD, librairies, pipelines)
  - Laisse au programmeur la possibilité de contrôler le partitionnement des données et la gestion mémoire (cache)
  - Efficace pour des traitements itératifs notamment
- **Inconvénients**
  - Plus difficile à programmer que map/reduce

Databricks (https://databricks.com)



Tester databricks

- https://community.cloud.databricks.com

Synthèse

- **Big data vs DW : grand nombre de sources, données non structurées, signaux faibles, vers le temps réel**
- **Architecture à base de data lake très utilisée aujourd'hui**
- **Map-reduce : modèle adapté aux grosses volumétries mais pas aux algorithmes complexes**
- **Hadoop : éco-système complet autour de map-reduce mais pas que**
- **Spark : modèle de programmation moins simple que map-reduce mais plus efficace et plus adapté aux algos complexes**
- **Plateforme Spark : ensemble cohérent autour du modèle Spark**

Références

- Opinion 05/2014 on Anonymisation Techniques: 0829/14/EN WP216, 10 april 2014
- NoSQL, parallel DBMS, Hadoop : D. DeWitt at http://gsl.azurewebsites.net/People/dewitt.aspx
- http://eric.univ-lyon2.fr/~ricco/cours/supports_data_mining.html
- Apprentissage machine - Clé de l'intelligence artificielle, Rémi Gilleron, Ed. Ellipses, février 2019
- Big Data et Machine Learning - 3e éd. - Les concepts et les outils de la data science, P. Lembergin et al., Dunod, 2019
- Spark: the definitive guide, M. Zaharia et B. Chambers, O'Reilly, 2018

Ressources utiles

- https://spark.apache.org/docs/latest/ documentation spark assez bien faite
- https://databricks.gitbooks.io/databricks-spark-reference-applications/ des exemples d'applications en spark
- http://cedric.cnam.fr/vertigo/Cours/RCP216/ un très bon cours de big data utilisant Spark comme outil de développement
- https://www.youtube.com/watch?v=65HFENCP9tw&feature=youtu.be comparaison mapr-reduce vs spark sur l'exemple word count
- https://youtu.be/v7hmsmckiDs exemple de streaming avec spark
- https://www.data-automaton.com/2019/01/03/predictive-data-analytics-with-apache-spark-part-1-introduction : analyse de données de réacteurs avec spark