

Framework & Technologies Big Data

Licence Computing System

–Parcours Informatique Multimédia–



DR. Rania MKHININI GAHAR

Institut Supérieur d'informatique de Mahdia
Département Informatique

Année Universitaire: 2023-2024

Objectifs du Module



À l'issue de ce cours, l'étudiant doit avoir les compétences suivantes :

- ▶ Disposer des connaissances nécessaires en Big Data
- ▶ Comprendre les fondamentaux de la programmation parallèle distribuée via MapReduce sous Hadoop.
- ▶ S'appropriier le panel d'outils des SGBD NoSQL.
- ▶ Élarger ses connaissances en Cloud Computing et les plateformes disponibles.

Plan

- 1 Introduction au Big Data
- 2 Hadoop, MapReduce et le Big Data
- 3 Big Data et NoSQL
- 4 Bibliographie et Logistique Recommandées

Plan

- 1 Introduction au Big Data
- 2 Hadoop, MapReduce et le Big Data
- 3 Big Data et NoSQL
- 4 Bibliographie et Logistique Recommandées

Big Data : Faits

- ▶ Les big data (données volumineuses) désignent des **ensembles de données** qui deviennent tellement volumineux qu'ils en deviennent **difficiles à travailler avec des outils classiques** de gestion de base de données ou de gestion de l'information ;
- ▶ Chaque jour, nous générons 2,5 trillions d'octets de données ;
- ▶ **90%** des données dans le monde ont été créées au cours des deux dernières années seulement (selon IBM) ;
- ▶ **Sources :**
 - ▶ Capteurs utilisés pour collecter les informations climatiques ;
 - ▶ Messages sur les médias sociaux ;
 - ▶ Images numériques et vidéos publiées en ligne ;
 - ▶ Enregistrements transactionnels d'achat en ligne ;
 - ▶ Signaux GPS de téléphones mobiles ;
 - ▶ ...
- ▶ Données appelées **Big Data** ou **Données Massives**

Big Data : Intérêts

1 / 3

- Chefs d'entreprise prennent fréquemment des décisions basées sur des informations en lesquelles ils n'ont pas confiance, ou qu'ils n'ont pas

1 / 2

- Chefs d'entreprise disent qu'ils n'ont pas accès aux informations dont ils ont besoin pour faire leur travail

83 %

- Des DSI (Directeurs des SI) citent : « L'informatique décisionnelle et analytique » comme faisant partie de leurs plans pour améliorer leur compétitivité

60 %

- Des PDG ont besoin d'améliorer la capture et la compréhension des informations pour prendre des décisions plus rapidement

Big Data : Sources (1)

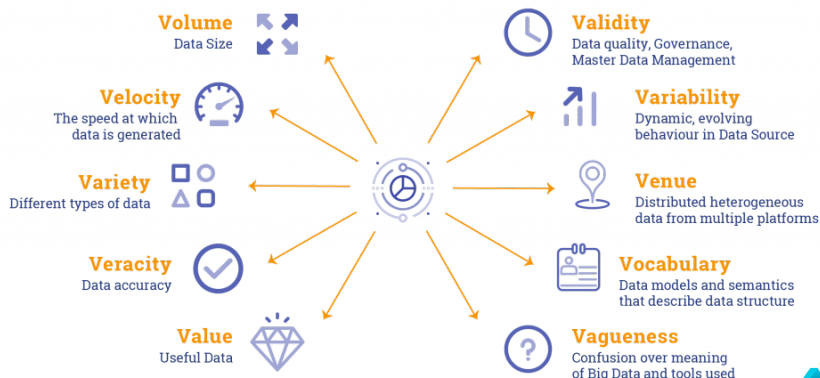
- ▶ Sources multiples : sites, Bases de Données, téléphones, serveurs :
 - ▶ Détecter les sentiments et les réactions des clients ;
 - ▶ Détecter les conditions critiques ou potentiellement mortelles dans les hôpitaux, et à temps pour intervenir ;
 - ▶ Prédire des modèles météorologiques pour planifier l'usage optimal des éoliennes ;
 - ▶ Prendre des décisions risquées basées sur des données transactionnelles en temps réel ;
 - ▶ Identifier les criminels et les menaces à partir de vidéos, sons et flux de données ;
 - ▶ Étudier les réactions des étudiants pendant un cours, prédire ceux qui vont réussir, d'après les statistiques et modèles réunis au long des années (domaine Big Data in Education).

Big Data : Sources (2)

- ▶ Les données massives sont le résultat de la rencontre de trois éléments essentiels qui sont :
 - ▶ Internet ;
 - ▶ Les réseaux sociaux ;
 - ▶ Les appareils intelligents : les ordinateurs, les tablettes, les smartphones, les objets connectés...
- ▶ L'Internet permet la transmission de l'information quelle que soit sa forme sur les appareils intelligents :
 - ▶ Appareil intelligent : création de données ;
 - ▶ Utilisateur des réseaux sociaux : consommateur ;
 - ▶ Internet : vecteur de transmission.

Les 10 Vs de Big Data

THE 10 Vs OF BIG DATA



Source: xenonstack.com

Les 10 Vs de Big Data : Quésaquo

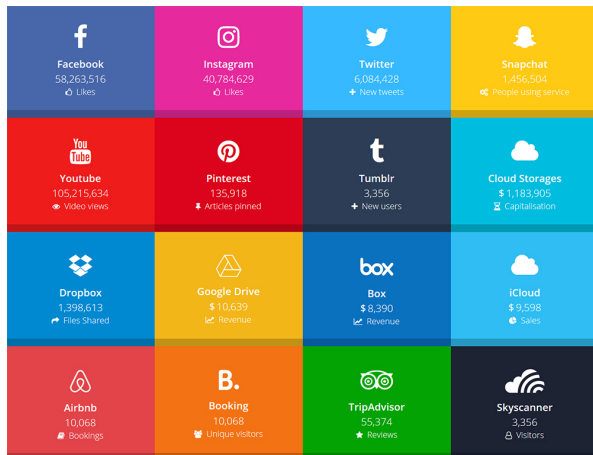
[1]Volume

- ▶ **Volume** : les organisations sont submergées de volumes de données croissants de tous types
 - ▶ Les ordres de grandeurs sont énormes!!!
 - ▶ Toujours plus exponentiel ;
 - ▶ 281 milliards de mail envoyés par jour en 2018 (hors spam). Prévision pour 2022 : 333 milliards mails
 - ▶ 40 000 recherches sont analysée sur Google chaque seconde, soit plus de 3,5 milliards par jour ! (Source : Google Search Statistics)
 - ▶ 100 heures de vidéo sont en moyenne téléchargées sur YouTube chaque minute (Source : YouTube)
 - ▶ Le nombre d'objets connectés passerait à 17,8 milliards en 2018 et 34,2 milliards en 2025

Les 10 Vs de Big Data : Quésaquo

[1] Volume

La Capture est prise le 11 Février 2019¹.



1. <https://visual.ly/community/infographic/how/internet-real-time>

Les 10 Vs de Big Data : Quésaquo

[2]Variété

► Des données très hétérogènes

- Médias sociaux (Facebook, tweeter, Instagram, Youtube,...)
- Données de téléphonie mobiles (SMS, GPS)
- Données géographiques (producteurs institutionnels, OSM,...)
- Documents (pdf, word, excel,...)
- Wikis, forums, pages Web,...
- Données transactionnelles (connexion,validation TC, badge,...)
- Photos (en milliards, ex. FB 50 milliards), vidéos
- Open data (ouverture de données publiques)
- Capteurs (trafic, température, pollution,...)
- 51% des données sont structurées, 27% sont non structurées et 21% semi-structurées. (Source : Tata Consultancy Services)

Les 10 Vs de Big Data : Quésaquo

[3]Vélocité/Vitesse

- ▶ Des données statiques aux données en temps réel
 - ▶ Parfois, 2 minutes c'est trop
 - ▶ Pour les processus chronosensibles 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.
 - ▶ Parfois 2 secondes c'est trop (trading haute fréquence)
 - ▶ 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

Les 10 Vs de Big Data : Quésako

[4]Véracité

► La **confiance** dans la données : le point le plus important

- Des données très hétérogènes, pas toujours structurées
- Pas toujours de documentation, de traçabilité sur ces données
- La clef pour une bonne utilisation : la confiance dans les données
- 1 décideur sur 3 ne fait pas confiance aux données sur lesquelles il se base pour prendre ses décisions
 - Comment peut-on se baser sur des informations si on a pas confiance en elle ?
- Établir la confiance dans les Big Data représente un défi d'autant plus important que la variété et le nombre de sources augmentent
- L'un des grands défis de demain

Les 10 Vs de Big Data : Quésaquo

[5] Variabilité

- ▶ La variabilité dans le contexte des données volumineuses fait référence à plusieurs choses différentes.
- ▶ L'un est le **nombre d'incohérences** dans les données.
- ▶ Celles-ci doivent être détectées par des méthodes de **détection d'anomalies** et de **valeurs aberrantes** afin de permettre toute analyse significative.
- ▶ Les mégadonnées sont également **variables** en raison de la multitude de dimensions résultant de multiples types et sources de données disparates.
- ▶ La variabilité peut également faire référence à la vitesse incohérente à laquelle les **données volumineuses** sont chargées dans votre base de données.

Les 10 Vs de Big Data : Quésaquo

[6] Validité

- ▶ Semblable à la véracité, la validité fait référence à la précision et à la correction des données pour l'utilisation prévue.
- ▶ Selon **Forbes**, environ 60% du temps d'un scientifique est consacré au nettoyage de ses données avant de pouvoir effectuer une analyse.

Les 10 Vs de Big Data : Quésaquo

[7] Vulnérabilité

- ▶ Les données volumineuses ont entraîné de nouvelles préoccupations en matière de **sécurité**.
- ▶ Certes, une violation de données avec le Big Data est une **violation colossale**
- ▶ Malheureusement, il y a eu de nombreuses violations du Big Data. Un autre exemple, comme l'a rapporté CRN : en mai 2016, «un pirate informatique appelé Peace a posté des données sur le Web sombre pour les vendre, qui auraient inclus des informations sur 167 millions de comptes LinkedIn et 360 millions d'e-mails et de mots de passe pour les utilisateurs de MySpace».

Les 10 Vs de Big Data : Quésaquo

[8] Volatilité

- ▶ Combien de temps faut-il avoir pour que les données soient considérées comme non pertinentes, historiques ou inutiles ? Pendant combien de temps les données doivent-elles être conservées ?
- ▶ En raison de la rapidité et du volume des mégadonnées, il convient toutefois d'examiner attentivement la volatilité des données.
- ▶ Il sera nécessaire d'établir des règles relatives à la crédibilité et à la disponibilité des données et d'assurer une récupération rapide des informations chaque fois que cela est nécessaire.
- ▶ Il faut veiller à ce qu'elles soient clairement liées aux besoins et aux processus de l'entreprise.
- ▶ Avec le big data, la complexité et les coûts d'un processus de stockage et de récupération sont amplifiés.

Les 10 Vs de Big Data : Quésaquo

[9] Valeur

- ▶ Les autres caractéristiques du Big Data n'ont pas de sens si l'on ne tire pas de valeur commerciale des données.
- ▶ Le Big Data permet de dégager une **valeur substantielle** : comprendre mieux les clients, les cibler en conséquence, optimiser les processus et améliorer les performances de la machine ou de l'entreprise.
- ▶ Avant de se lancer dans une stratégie Big Data, il convient de comprendre le potentiel et les caractéristiques plus complexes du big data.

Les 10 Vs de Big Data : Quésako

[10] Visualisation

Une autre caractéristique des données massives est leur difficulté à visualiser.

- ▶ En raison des limitations de la technologie en mémoire et de la faible **évolutivité** (passage à l'échelle), des fonctionnalités et du temps de réponse, les outils de visualisation, les données massives actuelles sont confrontées à des défis techniques.
- ▶ On ne peut pas s'appuyer sur les **graphiques traditionnels** pour tenter de tracer un milliard de points de données.
- ▶ Il faut donc différentes manières de représenter des données (à voir plus tard)
- ▶ Si l'on tient compte de la multitude de variables résultant de la variété et de la rapidité du big data et des relations complexes qui les unissent, **on peut voir que développer une visualisation significative n'est pas si facile.**

Chiffres clés et forces majeures

Chiffres clés

Des chiffres importants qui s'ajoutent aux nouveaux concepts et notions soulevées par le Big Data :

- ▶ Plus de 2 milliards de vidéos regardées sur Youtube chaque jour et 220 milliards de recherche sur Google chaque mois ;
- ▶ 30 milliards de contenus statut, photo, vidéo, événement, etc. sont ajoutés sur Facebook par mois par plus de 600 millions d'utilisateurs actifs ;
- ▶ Le nombre d'appareils connectés à Internet a dépassé le nombre d'humains en 2008 ;
- ▶ La compagnie de Social Games traite 1 Petabyte (1 million de GB) de données chaque jour ;
- ▶ Le marché du Big Data et des Big Analytics (ou broyage de données) pourraient représenter près de 250 milliards de dollars sur 4 ans ;
- ▶ On estime que des données de mauvaise qualité coûtent plus de 600 milliard de dollars par année aux entreprises américaines.

Chiffres clés et forces majeures

Forces majeurs

D'une manière générale, on identifie cinq forces majeures à l'origine de l'accélération et l'augmentation du mouvement Big Data à savoir :

- ▶ La révolution du stockage grâce au Cloud ;
- ▶ L'avènement d'une nouvelle science des données : les Analytics avancés ;
- ▶ Les nouvelles possibilités de monétisation ;
- ▶ L'automatisation des échanges de données et les objets connectés ;
- ▶ *Les progrès de la visualisation de données.*

Importance et défis du Big Data

Importance du Big Data

- ▶ Dans la technologie de l'information : améliorer la sécurité, diagnostiquer les anomalies et le dépannage en analysant les structures dans les logs existants ;
- ▶ Au service chargé de la clientèle : en utilisant des informations des centres d'appels afin d'obtenir les modèles de clientèle et donc d'améliorer la satisfaction du client par la personnalisation des services ;
- ▶ Dans l'amélioration des services et des produits : à travers l'utilisation du contenu des médias sociaux. En connaissant les préférences des clients potentiels, l'entreprise peut modifier son produit afin de répondre à une plus large gamme de personnes ;
- ▶ Dans la détection de la fraude : dans les transactions en ligne pour toute type d'industrie ;
- ▶ Dans l'évaluation des risques en analysant les informations provenant des transactions sur le marché financier.

Importance et défis du Big Data

Défis du Big Data

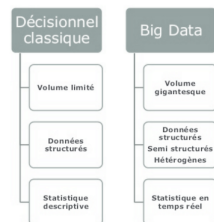
- ▶ Afin de déterminer la meilleure stratégie pour une entreprise, il est essentiel que les données qu'on compte sur soient correctement analysées ;
- ▶ Le laps de temps de cette analyse est important parce que certaines d'entre elles doivent être effectuées fréquemment afin de déterminer rapidement tout changement dans l'environnement des affaires ;
- ▶ Nouvelles technologies \Rightarrow problème organisationnel ;
- ▶ La nécessité des spécialistes de l'informatique : pour qu'une entreprise prend l'initiative du Big Data, elle doit soit engager des experts ou former les employés existants dans ce nouveau domaine ;
- ▶ **La confidentialité et la sécurité** : Comme le Big Data englobe une grande quantité de données complexes, il est très difficile pour une entreprise de trier ces données selon des niveaux privés et d'appliquer la sécurité adéquate.

Big Data et décisionnel

- ▶ Le modèle OLAP, ou traitement analytique en ligne, est considéré l'ancêtre du Big Data ;
- ▶ Il s'agit de volumes importants de données historiques qui représentent toutes les données de l'entreprise, et qui sont requêtées afin d'obtenir des informations agrégées et statistiques de l'activité de l'entreprise (décisionnel, ou Business Intelligence), ou pour extraire des informations nouvelles de ces données existantes à l'aide d'algorithmes de traitement des données (Data Mining).

▶ Mais...

- Modélisation des données préliminaires : Fait et Dimensions ;
- Structuration de données / à des besoins spécifiques ;
- Application à une analyse multi-dimensionnelle des données, mais pas pour DM.



Récapitulatif

Pour résumer :

- ▶ Des données exponentielles :
 - ▶ Volumineuses
 - ▶ Variées
 - ▶ En temps réel
 - ▶ De sources très hétérogènes
 - ▶ Mais qui encore « indomptées »
 - ▶ Encore que peu structurées

Récapitulatif

Pour résumer :

- ▶ Des données exponentielles :
 - ▶ Volumineuses
 - ▶ Variées
 - ▶ En temps réel
 - ▶ De sources très hétérogènes
 - ▶ Mais qui encore **« indomptées »**
 - ▶ Encore que peu structurées
- ▶ Mais que faire avec ces données ?

Récapitulatif

Pour résumer :

- ▶ Des données exponentielles :
 - ▶ Volumineuses
 - ▶ Variées
 - ▶ En temps réel
 - ▶ De sources très hétérogènes
 - ▶ Mais qui encore « indomptées »
 - ▶ Encore que peu structurées
- ▶ Mais que faire avec ces données ?
 - ▶ Une des pistes : **la visualisation** (exploration, fouille, analyse)



- 1 Introduction au Big Data
- 2 Hadoop, MapReduce et le Big Data**
- 3 Big Data et NoSQL
- 4 Bibliographie et Logistique Recommandées

Objectifs

- ▶ Découvrir diverses solutions complémentaires liées à Hadoop ;
- ▶ Apprendre à installer et utiliser Hadoop ;
- ▶ Découvrir la méthodologie MapReduce ;
- ▶ Apprendre à rédiger et exécuter des programmes MapReduce sous Hadoop

Introduction

- ▶ Hadoop est un framework 100% open source, écrit en Java et géré par la fondation Apache
- ▶ Hadoop est capable de stocker et traiter de manière efficace un grand nombre de données, en reliant plusieurs serveurs banalisés entre eux pour travailler en parallèle.
- ▶ Hadoop est utilisé par des entreprises ayant de très fortes volumétries de données à traiter. Parmi elles, des géants du web comme *Facebook*, *Twitter*, *LinkedIn*, ou encore les géants de l'e-commerce à l'instar de *eBay* et *Amazon*

Qui utilise Hadoop ?



Hadoop : Aperçu

- ▶ Plate forme MapReduce open-source écrite en java(Doug Cutting 2009)
- ▶ projet de la fondation apache
- ▶ Un système de fichier distribué (HDFS)
- ▶ Un ordonnanceur du programmes MapReduce
- ▶ Une API MapReduce en Java,R, Python C++,...
- ▶ Une interface utilisateur
- ▶ Une interface administrateur

Hadoop : Caractéristiques

- ▶ **Economique** : Permet aux entreprises de libérer toute la valeur de leurs données en utilisant des serveurs peu onéreux.
- ▶ **Flexible** : Hadoop permet de stocker de manière extensible tous types de données. Grâce à son système de fichier HTDFS, les utilisateurs peuvent transférer leurs données vers Hadoop sans avoir besoin de les reformater.
- ▶ **Tolère les pannes** : les données sont répliquées à travers le cluster afin qu'elles soient facilement récupérables suite à une défaillance du disque, du nœud ou du bloc.

Apache Hadoop : Une technologie en plein essor

- ▶ De plus en plus de données produites par des SI deviennent très nombreuses. Ces données doivent toutes être analysées, corrélées, etc
- ▶ Hadoop offre une solution idéale et facile à implémenter au problème.
 - ▶ Le projet Hadoop consiste en deux grandes parties :
 - Stockage des données : HDFS (Hadoop Distributed File System) ;
 - Traitement des données : MapReduce.
 - ▶ Principe :
 - Diviser les données ;
 - Les sauvegarder sur une collection de machines, appelées cluster ;
 - Traiter les données directement là où elles sont stockées, plutôt que de les copier à partir d'un serveur distribué.
 - **Note** : Il est possible d'ajouter des machines à votre cluster, au fur et à mesure que les données augmentent.

Hadoop : Distributions

Quatre solutions leaders sur le marché se partagent Hadoop :
Cloudera, Hortonworks, MapR, Amazon Elastic Map Reduce (EMR).



Distributions



Hadoop : Distribution

- 1 **Cloudera** : est une plateforme Big Data mature aujourd'hui. Elle est composée de 2 éditions : l'offre express et l'offre entreprise (la version commerciale). Son grand atout reste son interface unifiée de gestion.
- 2 **Hortonworks** est la solution dont vous aurez besoin pour bénéficier d'un support d'entreprise tout en bénéficiant d'une technologie 100% open source.
- 3 **MapR** : son système de fichiers MapR-F décuple la vitesse d'écriture et de lecture des données.
- 4 **Amazon Elastic MapReduce (EMR)** : Amazon offre même une solution hébergée, pré-configurée dans son cloud

Hadoop : Distribution

Pourquoi des distributions ?

- ▶ Pour regrouper de façon homogène les différentes extensions ;
- ▶ Pour faciliter l'installation, la diffusion, le support ;
- ▶ Pour permettre d'incuber des Business Model pour les éditeurs qui contribuent largement à Hadoop ;

Comment choisir une solution Hadoop ?

- ▶ Modèle économique (Open Source, Commercial..).
- ▶ Les composants ;
- ▶ Maturité de la solution, le support, la documentation, le retour d'expériences ;
- ▶ Le rapport avec Hadoop, la rapidité des évolutions ;
- ▶ Partenariats (hébergeurs...), compatibilité avec les produits satellites.

Hadoop : Domaines d'application

- ▶ Dans le domaine de la gestion de clientèle (Anticipation des désabonnements)
- ▶ Dans le domaine de la publicité (Ciblage de la clientèle)
- ▶ Dans le domaine de la lutte contre la fraude
- ▶ Etc. . . .

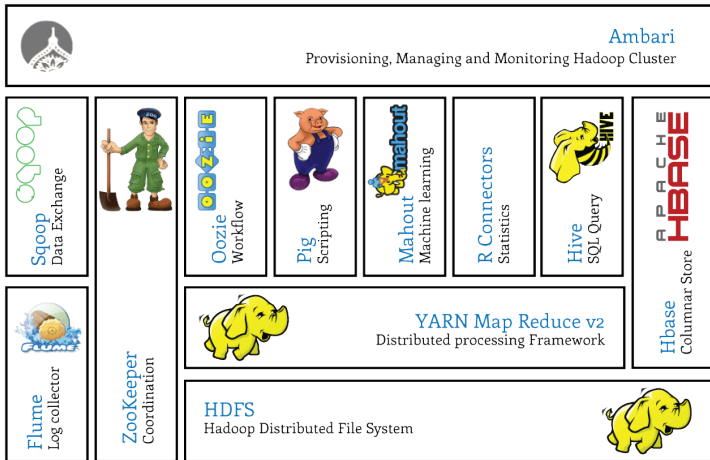
Hadoop : Fonctionnement

- ▶ Hadoop fonctionne sur le principe des grilles de calcul consistant à répartir l'exécution d'un traitement intensif de données sur plusieurs nœuds ou grappes de serveurs.
- ▶ Les machines sont regroupés par rack

Hadoop : Écosystème



Apache Hadoop Ecosystem



/TheOpentuto



/TheOpentuto



/TheOpentuto



/Opentutos

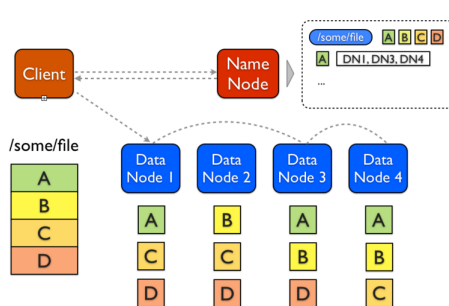
Hadoop : Écosystème

- ▶ **HDFS** : Système de fichiers distribué ;
- ▶ **MapReduce** : Framework de traitement parallélisé ;
- ▶ **Ambari** :
 - ▶ Destiné à la supervision et à l'administration de clusters Hadoop ;
 - ▶ Outil Web qui propose un tableau de bord (visualisation de l'état d'un cluster, état des services, configuration, supervision, exécution des jobs, métriques.)
- ▶ **HBase** : Système de gestion de base de données non-relationnelles distribué de type orientée colonnes.
- ▶ **Pig** : Requête des données Hadoop à partir d'un langage de script.
- ▶ **Hive** : Requête de type SQL.
- ▶ ...

HDFS : Présentation

- ▶ Pour stocker les données en entrée de nos tâches Hadoop, ainsi que les résultats de nos traitements, on va utiliser HDFS : Hadoop Distributed FileSystem ;
- ▶ HDFS est inspiré de GFS, un système de fichiers distribué conçu par Google ;
- ▶ L'implémentation de HDFS a son origine dans un whitepaper issu du département de recherche de Google (*The Google File System, 2003*)
- ▶ Il s'agit du système de fichier standard de Hadoop - au même sens que les systèmes de fichiers FAT32, NTFS ou encore Ext3FS, mais :
 - ▶ Systèmes de fichiers : Les données sont écrites dans des blocs gérés par le FileSystem ;
 - ▶ HDFS : Les données sont écrites dans des blocs gérés par le HDFS, qu'il est évidemment distribué.
- ▶ Permet :
 - ▶ La réplication (les blocs sont répliqués) ;
 - ▶ La scalabilité (les blocs ne sont pas tous sur la même machine) ;
 - ▶ Le stockage des données sur un ensemble de serveurs distribués ;

HDFS : Architecture (1)



► **Node (Master/Slave)** : Dans une architecture Hadoop, chaque membre pouvant traiter des données est appelé Node (Nœud) ;

► Un seul d'entre eux peut être Master même s'il peut changer au cours de la vie du cluster, il s'agit du NameNode (NN). Il est responsable de la localisation des données dans le cluster.

- Les autres nœuds, stockant les données, sont des slaves appelés DataNode (DN) ;
- Le NN est donc un point unique de défaillance (Single Point Of Failure (SPOF))

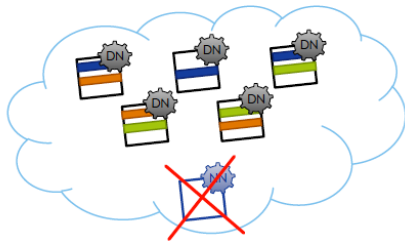
HDFS : Architecture (2)

- ▶ NN : stocke les informations relatives aux noms de fichiers. C'est ce serveur qui, par exemple, va savoir qu'un fichier dans le répertoire *Data_Input*, créé par le programmeur, comporte 58 blocs de données, et qui sait où ils se trouvent. Il y a un seul NN dans tout le cluster Hadoop ;
- ▶ DN : stocke les blocs de données eux-mêmes. Il y a un DN pour chaque machine au sein du cluster, et ils sont en communication constante avec le NN pour recevoir de nouveaux blocs, indiquer quels blocs sont contenus sur le DN, signaler des erreurs, etc...
- ▶ BlockSize : Taille unitaire de stockage (généralement 64 Mo ou 128 Mo). C'est à dire qu'un fichier de 1 Go et une taille de block de 128 Mo sera divisé en 8 blocks.
- ▶ Replication Factor
 - ▶ C'est le nombre de copies d'une donnée devant être réparties sur les différents nœuds du cluster (souvent 3, c'est à dire une primaire et deux secondaires).
- ▶ Processus de lecture HDFS :
 - ▶ Interrogation du NN pour localiser les adresses des nœuds hébergeant les blocs sous-jacents les plus proches ;
- ▶ Processus d'écriture
 - ▶ Écriture sur le DN ;
 - ▶ DN communique ses blocs au NN ;
 - ▶ Replication.

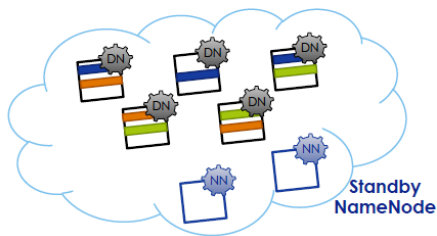
Dysfonctionnement !!!

- ▶ La gestion du stockage est assurée par les daemons Hadoop. On a pas à se soucier d'où sont stockées les données ;
- ▶ Hadoop réplique lui-même les données : les fichiers sont disponibles à tout moment sur plusieurs DN, et si une machine tombe en panne, on a toujours accès aux données grâce à la replication ;
- ▶ Si l'un des nœuds a un problème, les données seront perdues :
 - ▶ Hadoop réplique chaque bloc 3 fois ;
 - ▶ Il choisit 3 nœuds au hasard, et place une copie du bloc dans chacun d'eux ;
 - ▶ Si le nœud est en panne, le NN le détecte, et s'occupe de répliquer encore les blocs qui y étaient hébergés pour avoir toujours 3 copies stockées.
- ▶ Si le NN a un problème ?

Dysfonctionnement !!!



- ▶ Si c'est un problème d'accès (réseau), les données sont temporairement inaccessibles ;
- ▶ Si le disque du NN est défaillant, les données seront perdues à jamais!!!!



- ▶ Pour éviter cela, le NN sera dupliqué, non seulement sur son propre disque, mais également quelque part sur le système de fichiers du réseau ;
- ▶ Définition d'un autre NN (standby NN) pour reprendre le travail si le NN actif est défaillant.

Présentation (1)

- ▶ Pour exécuter un problème large de manière distribuée, il faut pouvoir découper le problème en plusieurs problèmes de taille réduite à exécuter sur chaque machine du cluster (stratégie algorithmique dite du divide and conquer / diviser pour régner) ;
- ▶ De multiples approches de division d'un problème en plusieurs sous-tâches existent : Open MP, MPI, etc... ;
- ▶ MapReduce est un paradigme visant à généraliser les approches existantes pour produire une approche unique applicable à tous les problèmes ;
- ▶ MapReduce, écrit en C++, existait déjà depuis longtemps, notamment dans les langages fonctionnels (Lisp, Scheme), mais la présentation du paradigme sous une forme rigoureuse, généralisable à tous les problèmes et orientée calcul distribué est attribuable à un whitepaper issu du département de recherche de Google publié en 2004 (*MapReduce : Simplified Data Processing on Large Clusters*) ;
- ▶ Un framework pour l'analyse de Big Data :
 - Pour données non structurées, sans schéma, etc. ;
 - Pour de très grands clusters ;
 - Des milliers de nœuds ;
 - Partitionnement et parallélisation automatiques.
- ▶ SQL ou Xquery trop lourd : Jointure, Group By, etc.. ;
- ▶ De nombreuses variations : Hadoop (Apache), Hadoop++ (projet qui vise à améliorer la performance de Hadoop pour les requêtes analytiques), Amazon Elastic MapReduce (Amazon EMR : est un service Web utilisant Hadoop), etc.

Présentation (2)

MapReduce définit deux opérations distinctes à effectuer sur les données d'entrée :

► Map :

- Transforme les données d'entrée en une série de couples (key, value) ;
- Regroupe les données en les associant à des clés, choisies de telle sorte que les couples (key, value) aient un sens par rapport au problème à résoudre.

Note Cette opération doit être parallélisable. On doit pouvoir découper les données d'entrée en plusieurs fragments, et faire exécuter l'opération Map à chaque machine du cluster sur un fragment distinct.

► Reduce :

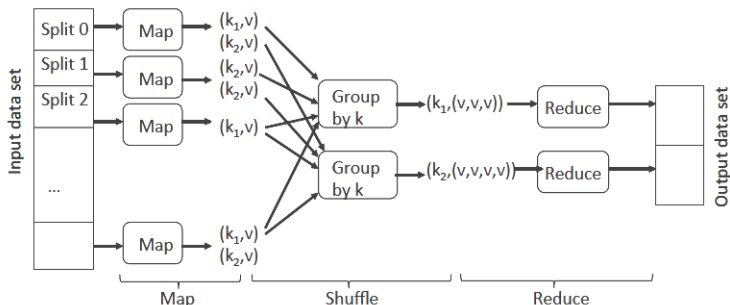
- Applique un traitement à toutes les valeurs de chacune des clés distinctes produite par l'opération Map ;
- Au terme de l'opération Reduce, on aura un résultat pour chacune des clés distinctes.

Ici, on attribuera à chacune des machines du cluster une des clés uniques produites par Map, en lui donnant la liste des valeurs associées à la clé. Chacune des machines effectuera alors l'opération Reduce pour cette clé.

Modèle de programmation

- ▶ Données sous forme de paires (key, value).
 - ▶ Ex. (doc-id, content), (word, count), etc. ;
- ▶ Le programmeur fournit le code de deux fonctions :
 - ▶ Map (key, value) \rightarrow list(ikey, ivalue) : Permet de faire le même traitement en parallèle sur des données partitionnées ;
 - ▶ Reduce(ikey, list(ivalue)) \rightarrow list(ikey, fvalue) : Permet d'agréger les données traitées par Map.
- ▶ Traitement parallèle des étapes Map et Reduce.
 - ▶ Partitionnement des données ;
 - ▶ Tolérance aux fautes ;
 - ▶ Ordonnancement des accès disques : méthodes qu'un SE utilise pour décider de l'ordre dans lequel les opérations d'E/S seront transmises aux disques.

Fonctionnement MapReduce



On distingue donc 4 étapes dans un traitement MapReduce :

1. **Découper** (split) les données d'entrée en plusieurs fragments ;
2. **Mapper** chacun de ces fragments pour obtenir des couples (key, value) ;
3. **Grouper** (shuffle) ces couples (key, value) par clé (key) ;
4. **Réduire** (reduce) les groupes indexés par clé en une forme finale, avec une valeur pour chacune des clés distinctes.

Architecture fonctionnelle

Principe de programmation

Pour résoudre un problème via la méthodologie MapReduce avec Hadoop, on doit :

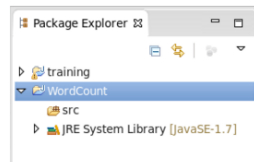
- ▶ Choisir une manière de découper les données d'entrée de telle sorte que l'opération Map soit parallélisable ;
- ▶ Définir quelle clé utiliser pour notre problème ;
- ▶ Écrire le programme pour l'opération Map ;
- ▶ Écrire le programme pour l'opération Reduce.

...et Hadoop se chargera du reste (problématiques calcul distribué, groupement par clé distincte entre Map et Reduce, etc.).

Création d'un projet Java Wordcount sous Eclipse

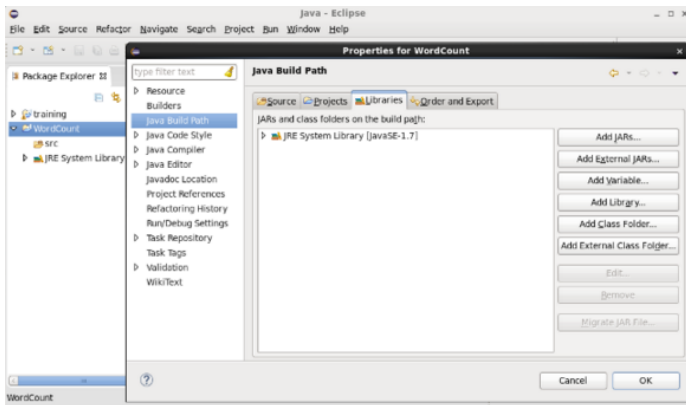
Création d'un nouveau projet

Lien : <http://fr.slideshare.net/andreaiacono/mapreduce-34478449?related=1>
<http://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/>



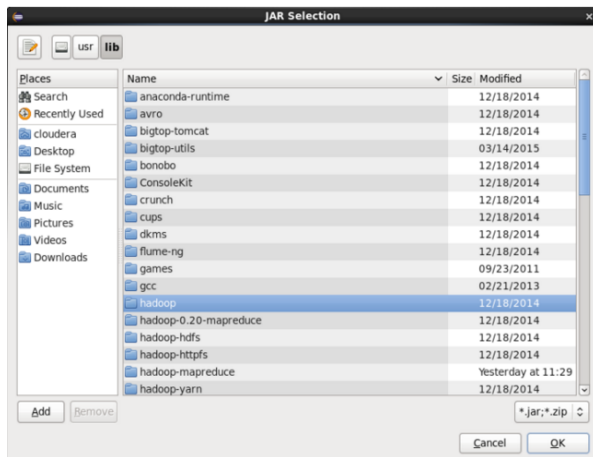
Création d'un projet Java Wordcount sous Eclipse

Importation des libraires externes (1)

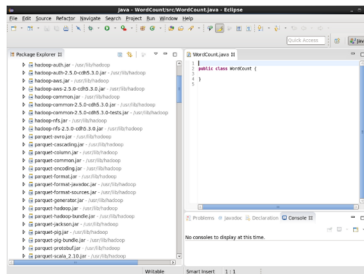


Création d'un projet Java Wordcount sous Eclipse

Importation des librairies externes (2)



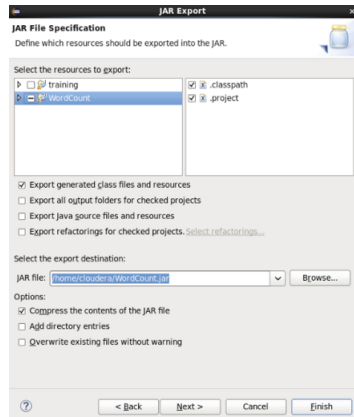
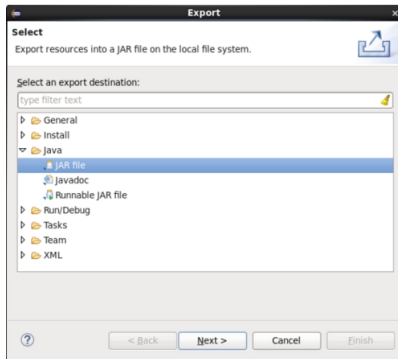
Création d'une nouvelle classe WordCount



Pour plus de détail consultez : <http://kickstarthadoop.blogspot.com/2011/04/word-count-hadoop-map-reduce-example.html>

Création d'un projet Java Wordcount sous Eclipse

Exportation du Jar



Exécution du Jar sous Hadoop avec la distribution Cloudera

- ▶ Notre entrée

```
[clouderaquickstart ~]$hadoop fs -cat /home/cloudera/wordcount.txt  
Hello Hadoop Goodbye Hadoop
```
- ▶ Affichage du contenu du système de fichiers :

```
[cloudera@quickstart ~]$ hadoop fs -ls  
Found 3 items  
drwx----- - cloudera cloudera 0 2015-03-24 09 :41 .Trash  
drwx----- - cloudera cloudera 0 2015-03-24 19 :01 .staging  
drwxr-xr-x - cloudera cloudera 0 2015-03-24 19 :01 oozie-oozi
```
- ▶ Copie du fichier dans HDFS

```
[clouderaquickstart ~]$ hadoop fs -mkdir Data_Input  
[clouderaquickstart ~]$ hadoop fs -put /home/cloudera/wordcount.txt Data_Input/  
[clouderaquickstart ~]$ hadoop fs -ls Data_Input  
Found 1 items  
-rw-r--r-- 1 cloudera cloudera 30 2015-03-24 20 :23 input/wordcount.txt  
[clouderaquickstart ~]$ hadoop fs -cat input/wordcount.txt  
Hello Hadoop, Goodbye Hadoop.  
[clouderaquickstart ~]$
```
- ▶ Exécution

```
[clouderaquickstart ~]$ hadoop jar /home/cloudera/WCount.jar WCount Data_Input/wordcount.txt Output
```
- ▶ Affichage du contenu du répertoire Output

```
[clouderaquickstart ~]$ hadoop fs -ls output
```
- ▶ Affichage du résultat

```
[clouderaquickstart ~]$ hadoop fs -cat Output/part-00000  
Goodbye 1 Hadoop 2 Hello 1
```

- 1 Introduction au Big Data
- 2 Hadoop, MapReduce et le Big Data
- 3 Big Data et NoSQL**
- 4 Bibliographie et Logistique Recommandées

Nouveaux besoins en gestion de données

Système distribué

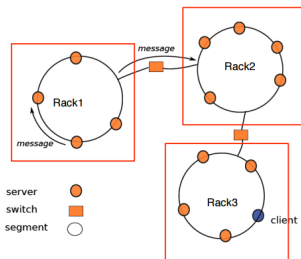
- ▶ Système distribué = système logiciel qui permet de coordonner plusieurs ordinateurs. Généralement, cette coordination se fait par envoi de messages via un réseau auquel sont reliés ces ordinateurs.
- ▶ **Pourquoi** ? parce qu'on manipule un très grand volume de données. Sans distribution, on n'a pas d'application **scalable**.
- ▶ 2 scénarios de traitement des données :
 - ▶ 1. **Par distribution des traitements** (scaling des traitements) :
 - on distribue les traitements sur un nombre de machines important afin d'absorber des charges très importantes ;
 - on envoie les données aux endroits appropriés, et on enchaîne les exécutions distantes (scénario type Workflow implémentable avec des Web services).
 - ▶ 2. **Par distribution des données** (scaling des données) :
 - on distribue les données sur un nombre important de serveurs afin de stocker de très grands volumes de données ;
 - on pousse les programmes vers ces serveurs (plus efficace de transférer un petit programme sur le réseau plutôt qu'un grand volume de données - Ex : algorithme MapReduce).

Système distribué

Exemple : Data Centers

- ▶ Utilise des LANs (Local Area Networks). On distingue 3 niveaux de communication :

- ▶ 1. Les serveurs sont regroupés en racks. Liaison réseau rapide, environ 1Go/sec.
- ▶ 2. Un data center consiste en un grand nombre de racks, interconnectés par des routeurs (switches). Liaison à 100 Mo/sec.
- ▶ 3. Entre différents data centers, il y a aussi une possibilité de communication mais par liaison assez lente (internet - 2-3 Mo/sec).



- ▶ Les serveurs communiquent par envoi de messages, ils ne partagent pas de disque ni de ressource de traitement \Rightarrow Architecture shared nothing.
- ▶ Début 2010 : 1 data center Google contient entre 100 et 200 racks, chacun contenant 40 serveurs. Environ 5000 serveurs par data-center pour un total de 1 millions de serveurs (estimation d'après la consommation électrique).
- ▶ Data center de Facebook (2010) : 2500 CPU (serveurs), 1 PetaByte d'espace disque (= milleTerabytes = 1 million de Gigabytes), plus de 250 Gigabytes données compressées (plus de 2 Terabytes non compressés).

Limites des systèmes SGBD classiques

SGBD Relationnels offrent :

- ▶ Un système de jointure entre les tables permettant de construire des requêtes complexes impliquant plusieurs entités ;
- ▶ Un système d'intégrité référentielle permettant de s'assurer que les liens entre les entités sont valides.

Contexte fortement distribué : Ces mécanismes ont un coût considérable :

- ▶ Avec la plupart des SGBD relationnels, les données d'une BD liées entre elles sont placées sur le même nœud du serveur ;
- ▶ Si le nombre de liens important, il est de plus en plus difficile de placer les données sur des nœuds différents.

Limites des systèmes SGBD classiques

- ▶ **SGBD Relationnels sont généralement transactionnels** \Rightarrow Gestion de transactions respectant les contraintes ACID (Atomicity, Consistency, Isolation, Durability) ;
- ▶ Le modèle ACID est un des plus anciens et des plus importants concepts à l'origine des SGBD actuels. Il définit quatre principes que toute BDR doit atteindre :
 - ▶ **Atomicité** : Tout ou rien ;
 - ▶ **Cohérence** : Les transactions qui modifient l'état de la base font en sorte que les contraintes d'intégrité référentielle sont respectées ;
 - ▶ **Isolation** : Une transaction ne peut pas accéder aux données mise à jour par une autre transaction avant qu'elle ne soit validée par son propriétaire ;
 - ▶ **Durabilité** : Toute transaction validée (commitée) est assurée d'être prise en compte quelque soit la panne (transaction logs permettant de rejouer les modifications en cas de restauration).

Limites des systèmes SGBD classiques

Constat :

- ▶ Essor des très grandes plate-formes et applications Web (Google, Facebook, Twitter, LinkedIn, Amazon,...) ;
- ▶ Volume considérable de données à gérer par ces applications nécessitant une distribution des données et leur traitement sur de nombreux serveurs ;
- ▶ Ces données sont souvent associées à des **objets complexes** et **hétérogènes**.
⇒ Limites des SGBD traditionnels (relationnels et transactionnels) basés sur SQL.

D'où, nouvelles approches de stockage et de gestion des données :

- ▶ Permettant une meilleure scalabilité dans des contextes fortement distribués ;
- ▶ Permettant une gestion d'objets complexes et hétérogènes sans avoir à déclarer au préalable l'ensemble des champs représentant un objet ;
- ▶ Regroupées derrière le terme NoSQL (Not Only SQL), proposé par Carl Strozzi, ne se substituant pas aux SGBD Relationnels mais les complétant en comblant leurs faiblesses.

Limites des systèmes SGBD classiques

- ▶ Nécessaire de distribuer les traitements de données entre différents serveurs ;
- ▶ Difficile de maintenir les contraintes ACID à l'échelle du système distribué entier tout en maintenant des performances correctes.
⇒ La plupart des SGBD NoSQL relâchent les contraintes ACID, ou même ne proposent pas de gestion de transactions.
- ▶ Bases de données NoSQL :
 - Ce n'est pas (comme son nom le laisse supposer) NoSQL (pas de SQL) ;
 - Privilégier donc NOSQL plutôt que NoSQL.
- ▶ Bases de données non-relationnelles et largement distribuées ;
- ▶ Permet une analyse et une organisation rapides des données de très grands volumes et de types de données disparates ;
- ▶ Appelées également :
 - Cloud Databases ;
 - Non-Relational Databases ;
 - Big Data Databases...

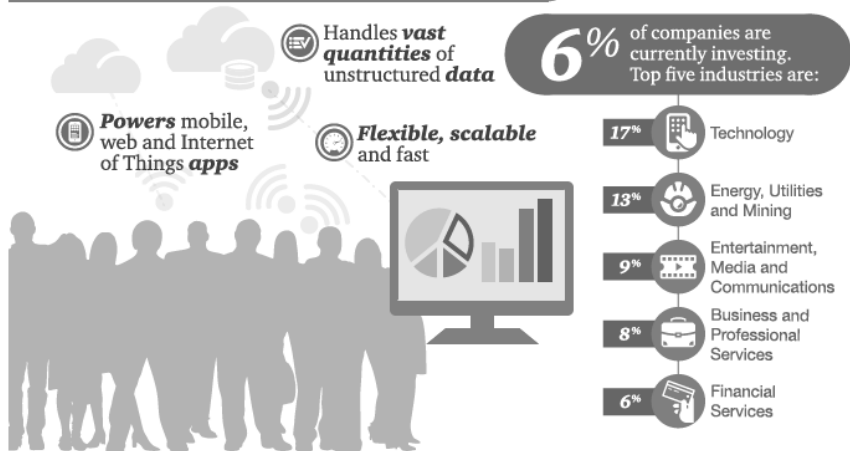
Genèse du NoSQL

- ▶ **1988** : Le terme a été inventé par Carlo Strozzi qui présentait le NoSQL comme un modèle de BD plus léger et sans interface SQL ;
- ▶ **Juin 2009** - Meetup NoSQL de San Francisco : Le mouvement NoSQL a prit un essor important. Plus de 100 développeurs de logiciels ont assisté à des présentations de solutions telles que Project Voldemort, Cassandra Project, Dymomite, HBase, Hypertable, CouchDB et MongoDB ;
- ▶ **2011** : un travail de spécification pour un langage de manipulation standardisé a débuté sous le nom de UnQL (Unstructured Query Language). Il se propose de formaliser la façon dont les bases NoSQL requêtent les collections (équivalent des tables pour les BDR).

⇒ Le NoSQL est issu du travail des grands acteurs d'Internet (Google, Amazon, Facebook, LinkedIn,...) en écho aux nouvelles architectures (Cloud, Grille de calcul, etc.).

Potential des BD NoSQL dans l'entreprise

NoSQL databases



<http://www.pwc.com/us/en/advisory/business-digital-technology-trends-nosql-databases.html>

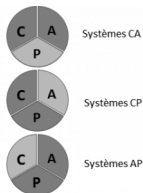
Théorème de CAP (Brewer, 2000)

- L'émergence du NoSQL est liée au théorème de CAP (Coherence (Cohérence), Availability (Disponibilité), Partition-Tolerance (Résistance au morcellement)).



Théorème de CAP:

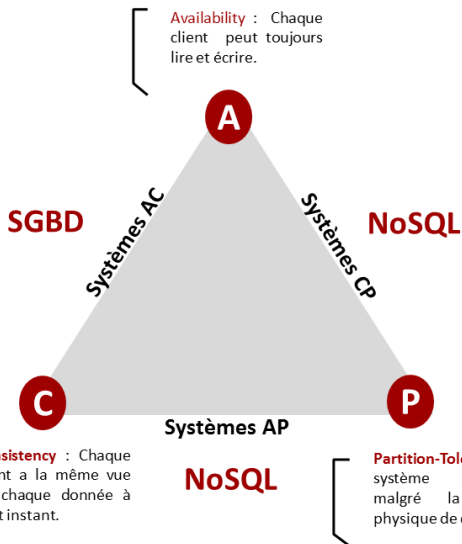
On ne peut obtenir à la fois que 2 des 3 Caractéristiques dans un système réparti



- **Cohérence** ou **Consistance** : Tous les nœuds du système voient exactement les mêmes données au même moment ;
- **Availability** ou **Disponibilité** : L'échec d'un nœud n'empêche pas les survivants de continuer à fonctionner ;
- **Partition-tolerance** ou **Résistance au partitionnement** : Le système étant partitionné, aucune panne moins importante qu'une coupure totale du réseau ne doit l'empêcher de répondre correctement (en cas de partitionnement en sous-réseaux, chacun doit pouvoir fonctionner de manière autonome).

⇒ Dans un système distribué, il est impossible d'obtenir ces 3 propriétés en même temps, il faut en choisir 2 parmi les 3.

Théorème de CAP



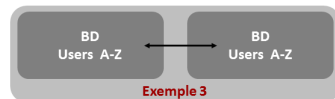
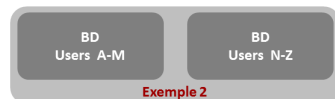
- ▶ Les **SGBDR** assurent les propriétés de **Consistance** et de **Disponibilité** (Availability) \Rightarrow Système **AC** ;
- ▶ Les **SGBD NoSQL** sont des systèmes **CP** (Cohérent et Résistant au partitionnement) ou **AP** (Disponible et Résistant au partitionnement).

\Rightarrow Dans un système distribué, il est impossible d'obtenir ces 3 propriétés en même temps, il faut en choisir 2 parmi les 3.

Théorème de CAP

Consistance (Coherence)

- ▶ **Exemple 1** : Une instance de la BD est automatiquement pleinement consistante puisqu'il n'y a qu'un seul nœud qui maintient l'état ;
- ▶ **Exemple 2** : Si 2 serveurs de BD sont impliqués, et si le système est conçu de telle sorte que toutes les clés de A à M sont conservées sur le serveur 1, et les clés N à Z sont conservées sur le serveur 2, alors le système peut encore facilement garantir la cohérence ;
- ▶ **Exemple 3** : Supposons 2 BD avec des répliques. Si une des BD fait une opération d'insertion de ligne, cette opération doit être aussi faite (committed) dans la seconde BD avant que l'opération soit considérée comme complète.
 - Pour avoir une cohérence de 100% dans un tel environnement répliqué, les nœuds doivent communiquer ;
 - Plus le nombre de répliques est grand plus les performances d'un tel système sont mauvaises.

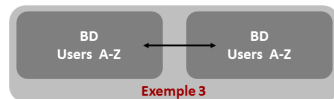


Théorème de CAP

Disponibilité (Availability)

Les BD dans Exemple 1 ou Exemple 2 ne sont pas fortement disponibles :

- ▶ **Exemple 1** : Si le nœud tombe en panne, on perd 100% des données ;
- ▶ **Exemple 2** : Si le nœud tombe en panne, on perd 50% des données ;
- ▶ **Exemple 3** : Une simple réplication de la BD sur un autre serveur assure une disponibilité de 100%.
 - Augmenter le nombre de nœuds avec des copies des données (répliques) augmente directement la disponibilité du système, en le protégeant contre les défaillances matérielles ;
 - Les répliques aident à équilibrer la charge d'opérations concurrentes, notamment en lecture.



Théorème de CAP

Résistance au partitionnement (Partition-tolerance)

- ▶ Supposons que soit assuré par réplication *consistance* et *disponibilité*. Dans le cas de l'exemple 3, supposons 2 serveurs de BD dans 2 Data-Centers différents, et que l'on perde la connexion réseau entre les 2 Data-Centers, faisant que les 2 BD sont incapables de synchroniser leurs états ;
- ▶ Si vous parvenez à gérer les opérations de lecture/écriture sur ces 2 BD, il peut être prouvé que les 2 serveurs ne seront plus consistants ;
- ▶ une application bancaire gardant à tout moment *l'état de votre compte* est l'exemple parfait du problème des enregistrements inconsistants :
 - ▶ Si un client retire 1000 euros à Marseille, cela doit être immédiatement répercuté à Paris, afin que le système sache exactement combien il peut retirer à tout ;
 - ▶ Si le système ne parvient pas à le faire, cela pourrait mécontenter de nombreux clients.
- ▶ Si les banques décident que la *consistance* est très importante, et désactivent les opérations d'écriture lors de la panne, alors la *disponibilité* du cluster sera perdu puisque tous les comptes bancaires dans les 2 villes seront désormais gelés jusqu'à ce que le réseau soit de nouveau opérationnel.

Caractéristiques générales des BD NoSQL

Les BD NoSQL :

- ▶ Adoptent une **représentation de données non relationnelle** ;
- ▶ Ne remplacent pas les BDR mais sont une **alternative**, un **complément** apportant des solutions plus intéressantes dans **certains contextes** ;
- ▶ Apportent une **plus grande performance** dans le contexte des applications Web avec des **volumétries de données exponentielle** ;
- ▶ Utilisent une **très forte distribution** de ces données et des traitements associés sur de **nombreux serveurs** font un **compromis sur le caractère ACID des SGBDR** pour plus de scalabilité horizontale et d'évolutivité ;
- ▶ **Pas de schéma** pour les données ou **schéma dynamique** ;
- ▶ Données de structures **complexes** ou **imbriquées** ;
- ▶ **Données distribuées** : partitionnement horizontal des données sur plusieurs nœuds (serveurs) généralement par utilisation d'algorithmes MapReduce ;
- ▶ **Réplication des données** sur plusieurs nœuds ;
- ▶ Privilégient la **Disponibilité** à la Cohérence (théorème de CAP) : **AP** (Disponible + Résistant au partitionnement) plutôt que CP (Cohérent + Résistant au partitionnement).
⇒ N'ont en général pas de gestion de transactions.
- ▶ Mode d'utilisation : **Peu d'écritures, beaucoup de lectures.**

Taxonomie des bases NoSQL

Stocker les informations de la façon la mieux adaptée à leur représentation

⇒ Différents types de BD NoSQL :

- ▶ Clé/Valeur (Key/value) : Basique, chaque objet est identifié par une clé unique constituant la seule manière de le requêter.
 - ▶ Voldemort, Redis, Riak,....
- ▶ Orientées Colonnes (Column Store) : Permet de disposer d'un très grand nombre de valeurs sur une même ligne, de stocker des relations *one-to-many*, d'effectuer des requêtes par clé (adaptés au stockage de listes : messages, posts, commentaires, ...).
 - ▶ HBase, Cassandra, Hypertable,....
- ▶ Orientées Documents (Document Databases) : Pour la gestion de collections de documents, composés chacun de champs et de valeurs associées, valeurs pouvant être requêtées (adaptées au stockage de profils utilisateur).
 - ▶ MongoDB, CouchDB, Couchbase,....
- ▶ Orientées Graphes (Graph Databases) : Pour gérer des relations multiples entre les objets (adaptés aux données issues de réseaux sociaux,...).
 - ▶ Neo4j, OrientDB,....

Paysage des SGBD NoSQL

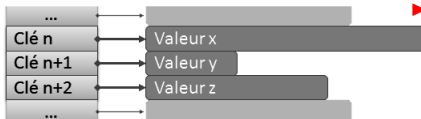


Key/Value Store

- ▶ Elles fonctionnent comme un grand tableau associatif et retourne une valeur dont elle ne connaît pas la structure ;
- ▶ Leur modèle peut être assimilé à une table de hachage (hashmap) distribuée ;
- ▶ Les données sont simplement représentées par un couple clé/valeur ;
- ▶ La valeur peut être une simple chaîne de caractères, ou un objet sérialisé ;
- ▶ Cette absence de structure ou de typage ont un impact important sur le requêtage : toute l'intelligence portée auparavant par les requêtes SQL devra être portée par l'applicatif qui interroge la BD ;
- ▶ Implémentations les plus connues :
 - Amazon Dynamo (Riak en est l'implémentation Open Source) ;
 - Redis (projet sponsorisé par VMWare) ;
 - Voldemort (développé par LinkedIn en interne puis passage en open source).
- ▶ Chaque objet est identifié par une clé unique, seule façon de le requêter.



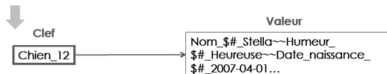
Key/Value Store



► Leur exploitation est basée sur 4 opérations (CRUD) :

- **C**reate : créer un nouvel objet avec sa clé : `create(key, value)` ;
 - **R**ead : lit un objet à partir de sa clé : `read(key)` ;
 - **U**ppdate : met à jour la valeur d'un objet à partir de sa clé : `update(key, value)` ;
 - **D**elete : supprime un objet à partir de sa clé : `delete(key)` ;
- + Auxquelles on retrouve couramment associée la fonction rechercher (Search ou LookUp).

Id	Nom	Humeur	Date_naissance	Couleur
12	Stella	Heureuse	2007-04-01	NULL
13	Wimma	Faim	NULL	Noire
9	Ninja	NULL	NULL	NULL



- Elles disposent généralement d'une simple interface de requêtage HTTP REST accessible depuis n'importe quel langage de développement ;
- Ont des performances très élevées en lecture et en écriture et une **scalabilité horizontale** considérable ;
- Le besoin en **scalabilité verticale** est faible du fait de la simplicité des opérations effectuées.

Key/Value Store

Utilisations principales

- ▶ Dépôt de données avec besoins de requêtage très simples ;
- ▶ Système de stockage de cache ou d'information de sessions distribuées (quand l'intégrité relationnelle des données est non significative) ;
- ▶ Les profils, préférences d'utilisateur ;
- ▶ Les données de panier d'achat ;
- ▶ Les données de capteurs ;
- ▶ Les logs de données ;
- ▶

Key/Value Store

Forces et faiblesses

Forces :

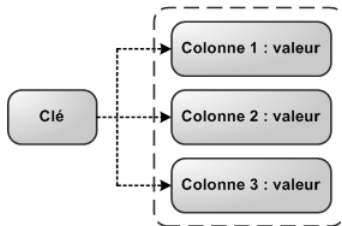
- ▶ Modèle de données **simple** ;
- ▶ Bonne mise à l'échelle horizontale pour les lectures et écritures :
 - ▶ Évolutivité (scalable) ;
 - ▶ Disponibilité ;
 - ▶ Pas de maintenances requises lors d'ajout/suppression de colonnes.

Faiblesses :

- ▶ Modèle de données TROP simple :
 - ▶ Pauvre pour les données complexes ;
 - ▶ Interrogation seulement sur clé ;
 - ▶ Déporte une grande partie de la complexité de l'application sur la couche application elle-même.

BD NoSQL orientées Colonnes

- ▶ Les données sont stockées par colonne, non par ligne ;
- ▶ On peut facilement ajouter des colonnes aux tables, par contre l'insertion d'une ligne est plus coûteuse ;
- ▶ Quand les données d'une colonne se ressemblent, on peut facilement compresser la colonne.
- ▶ Modèle proche d'une table dans un SGBDR mais ici le nombre de colonnes :
 - ▶ est dynamique ;
 - ▶ peut varier d'un enregistrement à un autre ce qui évite de retrouver des colonnes ayant des valeurs NULL.
- ▶ Implémentations les plus connues :
 - HBase (Open Source de BigTable de Google utilisé pour l'indexation des pages Web, Google Earth, Google analytics, ...)
 - Cassandra (fondation Apache qui respecte l'architecture distribuée de Dynamo d'Amazon, projet né de chez Facebook)
 - SimpleDB de Amazon.



BD NoSQL orientées Colonnes

Les principaux concepts associés sont les suivants :

► **Colonne** :

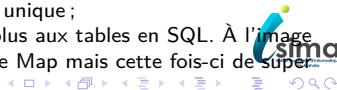
- Objet de plus bas niveau, il est composé d'une clé (le nom de la colonne), d'une valeur et d'un timestamp. Les timestamps des colonnes sont utilisés pour résoudre les conflits entre plusieurs nœuds de l'infrastructure, pour savoir si un nœud a une version plus récente. Il est donc important que tous les nœuds de l'infrastructure soient synchronisés sur la même horloge.
- Une colonne contenant d'autres colonnes est nommée super-colonne.

► **Super-colonnes** :

- Situées dans les familles de colonnes. Elle représente une colonne dont les valeurs sont d'autres colonnes ;
- Si on veut faire le parallèle avec une base SQL cela représente une ligne. On retrouve cette correspondance clé-valeur, la clé permet d'identifier la super-colonne tandis que la valeur est la liste des colonnes qui la compose.

► **Famille de colonnes** :

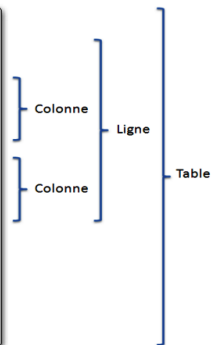
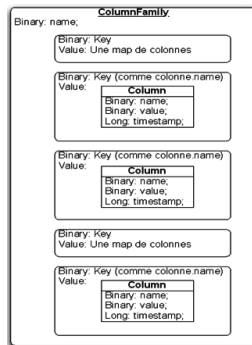
- Permettent de regrouper plusieurs colonnes (ou super-colonnes) ;
- Les colonnes sont regroupées par ligne ;
- Chaque ligne est identifiée par un identifiant unique (assimilées aux tables dans le modèle relationnel) et sont identifiées par un nom unique ;
- Les familles de colonnes sont ce qui ressemble le plus aux tables en SQL. À l'image des super colonnes, elles disposent elles aussi d'une Map mais cette fois-ci de super colonnes et non pas de colonnes.



Colonne / Super-Colonne/Famille de Colonnes

Column
Binary: name;
Binary: value;
Long: timestamp;

SuperColumns		
Key	Value	
person1	Column	
	Name	Value
	firstName	John
	lastName	Calagan
person2	Column	
	Name	Value
	firstName	George
	lastName	Truffe



BD NoSQL orientées Colonnes

	Nom	Prenom	Tel	Email
1	Dupond	Tom		d.tom@mail.com
2	Durand	Max	0102030405	
3	André	Pierre		

Organisation d'une table dans une base de données relationnelle



1	Nom	Dupond	Prenom	Tom	Email	d.tom@mail.com
2	Nom	Durand	Prenom	Max	Tel	0102030405
3	Nom	André	Prenom	Pierre		

Organisation d'un column-family dans une base de données orientée colonnes

BD NoSQL orientées Colonnes

- ▶ Elles sont les plus complexes à appréhender des BD NoSQL, même si au final on a un schéma assez proche des bases documentaires ;
- ▶ Elles sont très utilisées pour les traitements d'analyse de données et dans les traitements massifs (notamment via des opérations de type MapReduce) ;
- ▶ Elles offrent plus de flexibilité que les BDR :
 - ▶ Il est possible d'ajouter une colonne ou
 - ▶ une super colonne
 - ▶ à n'importe quelle ligne
 - ▶ d'une famille de colonnes, colonnes ou super-colonne à tout instant.

BD NoSQL orienté Colonnes

Forces et faiblesses des BD NoSQL orientées Colonnes

Forces :

- ▶ Modèle de données supportant des données semi-structurées ;
- ▶ Naturellement indexé (colonnes) ;
- ▶ Bonne mise à l'échelle à l'horizontale ;
- ▶ MapReduce souvent utilisé en scaling horizontal ;
- ▶ On peut voir les résultats de requêtes en temps réel.

Faiblesses :

- ▶ Modèle de données TROP simple : À éviter pour des données interconnectés : si les relations entre les données sont aussi importantes que les données elles-mêmes (comme la distance ou calculs de la trajectoire) ;
- ▶ À éviter pour les lectures de données complexes ;
- ▶ Exige de la maintenance - lors de l'ajout / suppression de colonnes et leurs regroupements.

BD NoSQL orientées Colonnes

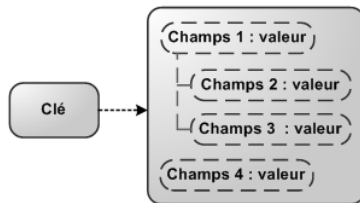
Utilisations principales des BD NoSQL orientées colonnes

Les BD NoSQL orientées colonnes sont principalement utilisées pour :

- ▶ Netflix l'utilise notamment pour le logging et l'analyse de sa clientèle ;
- ▶ Ebay l'utilise pour l'optimisation de la recherche ;
- ▶ Adobe l'utilise pour le traitement des données structurées et de Business Intelligence (BI) ;
- ▶ Des sociétés de TV l'utilisent pour cerner leur audience et gérer le vote des spectateurs (nombre élevé d'écritures rapides et analyse de base en temps réel (Cassandra) ;
- ▶ Peuvent être de bons magasins d'analyse des données semi-structurées ;
- ▶ Utilisées pour la journalisation des événements et pour des compteurs.

BD NoSQL orientées Documents

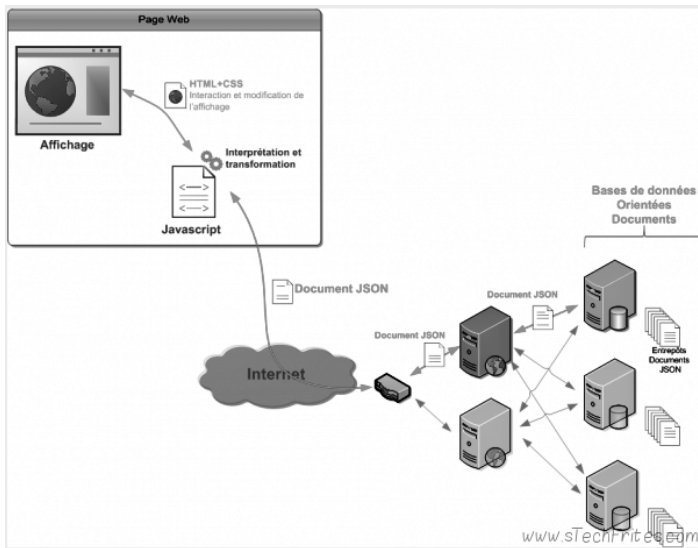
- ▶ Elles stockent une collection de **documents** ;
- ▶ Elles sont basées sur le modèle clé/valeur mais la valeur est un document en format semi-structuré hiérarchique de type JSON ((JavaScript Object Notation) ou XML.
- ▶ Les documents n'ont pas de schéma, mais une structure arborescente : ils contiennent une liste de champs, un champ a une valeur qui peut être une liste de champs, ...
- ▶ Elles ont généralement une interface d'accès HTTP REST permettant d'effectuer des requêtes (plus complexe que l'interface CRUD des BD clés/valeurs) ;
- ▶ Implémentations les plus connues :
 - MongoDB ;
 - CouchDB (fondation Apache) ;
 - RavenDB (pour plate-formes .NET/Windows - LINQ), ...



BD NoSQL orientées Documents

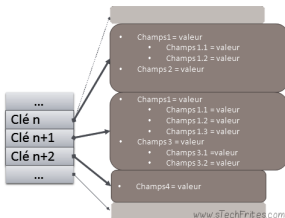
- ▶ Un document est composé de champs et des valeurs associées ;
- ▶ Ces valeurs :
 - peuvent être requêtées ;
 - sont soit d'un type simple (entier, chaîne de caractères, date, ...)
 - soit elles-mêmes composées de plusieurs couples clé/valeur.
- ▶ Bien que les documents soient structurés, ces BD sont dites ***schemaless*** : il n'est pas nécessaire de définir au préalable les champs utilisés dans un document ;
- ▶ Les documents peuvent être très hétérogènes au sein de la BD ;
- ▶ Permettent d'effectuer des requêtes sur le contenu des documents/objets : pas possible avec les BD clés/valeurs simples ;
- ▶ Elles sont principalement utilisées dans le développement de CMS (Content Management System - outils de gestion de contenus).

BD NoSQL orientées Documents



BD NoSQL orientées Documents

Instance MongoDB



```
{
  "ID":1,
  "FIRST":"Frank",
  "LAST":"Weigel",
  "ZIP":"94040",
  "CITY":"MV",
  "STATE":"CA"
}
```

JSON

USER INFO

KEY	First	Last	ZIP_id
1	Frank	Weigel	2
2	Ali	Bob	2
3	Mark	Azad	2
4	Steve	Yen	3

ADDRESS INFO

ZIP_id	City	State	ZIP
1	DEN	CO	30303
2	MV	CA	94040
3	CHI	IL	60609
4	NY	NY	10010

<http://www.stechfrites.com/book/export/html/2>
<http://changeaas.com/2014/09/14/nosql-databases-2/>

BD NoSQL orientées Documents

Avantage du format JSON :

- ▶ Format abstrait pour une représentation simplifiée dans les différents langages.
- ▶ Indépendant du langage de programmation ;
- ▶ Simple et complet pour la représentation des objets ;
- ▶ Utilise des types de données connus et simples à décrire ;
- ▶ Une bonne lisibilité de la syntaxe
- ▶ Temps de traitement très proche de celui des fichiers XML ;
- ▶ Moins volumineux en terme de stockage ;
- ▶ Pour JavaScript, un document JSON représente un objet.

Utilisations principales des BD NoSQL orientées Documents :

- ▶ Enregistrement d'événements ;
- ▶ Systèmes de gestion de contenu ;
- ▶ Web analytique ou analytique temps-réel ;
- ▶ Catalogue de produits ;
- ▶ Systèmes d'exploitation...

BD NoSQL orientées Documents

Forces et faiblesses des BD NoSQL orientées Documents

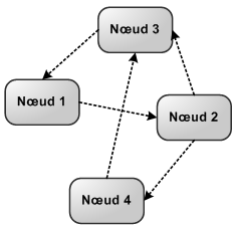
Forces :

- ▶ Modèle de données simple mais puissant (expressions de structures imbriquées) ;
- ▶ Bonne mise à l'échelle ;
- ▶ Pas de maintenance de la BD requise pour ajouter/supprimer des colonnes ;
- ▶ Forte expressivité de requêtage (requêtes assez complexes sur des structures imbriquées).

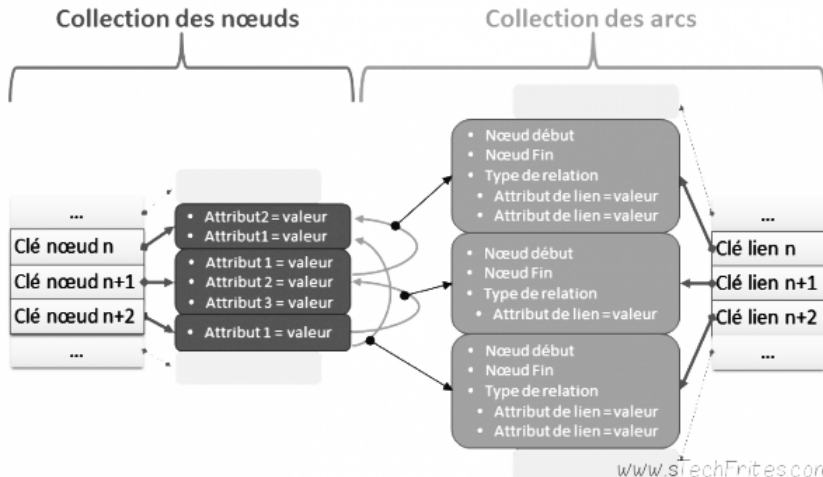
Faiblesses :

- ▶ Inadaptée pour les données interconnectées ;
- ▶ Modèle de requête limitée à des clés (et indexes) ;
- ▶ Peut alors être lent pour les grandes requêtes (avec MapReduce).

BD NoSQL orientées Graphes

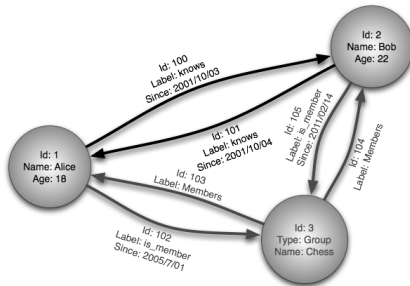
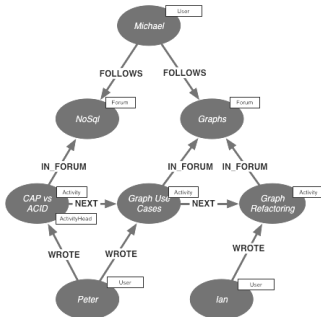
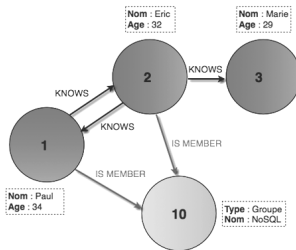
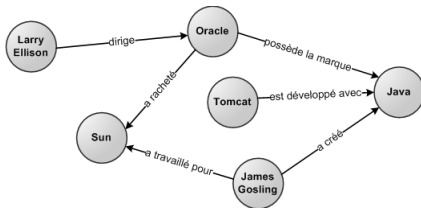
- ▶ Elles permettent la modélisation, le stockage et la manipulation de données complexes liées par des relations non-triviales ou variables ;
 - ▶ Modèle de représentation des données basé sur la théorie des graphes ;
 - ▶ S'appuie sur les notions de nœuds, de relations et de propriétés qui leur sont rattachées.
- 
- ```
graph TD; N1[Nœud 1] -.-> N2[Nœud 2]; N1 -.-> N3[Nœud 3]; N2 -.-> N3; N2 -.-> N4[Nœud 4]; N3 -.-> N4;
```
- ▶ Implémentations les plus connues :
    - Neo4J ;
    - OrientDB (fondation Apache), ...
  - ▶ Elles utilisent :
    - Un moteur de stockage pour les objets (similaire à une base documentaire, chaque entité de cette base étant nommée nœud) ;
    - Un mécanisme de description d'arcs (relations entre les objets), arcs orientés et avec propriétés (nom, date, ...)
  - ▶ Elles sont bien plus efficaces que les BDR pour traiter les problématiques liées aux réseaux (cartographie, relations entre personnes, ...) ;
  - ▶ Sont adaptées à la manipulation d'objets complexes organisés en réseaux : cartographie, réseaux sociaux,...

# BD NoSQL orientées Graphes



www.stechFrites.com

# Exemples



# BD NoSQL orientées graphes

## Forces et faiblesses des BD NoSQL orientées graphes

### Forces :

- ▶ Modèle de données puissant ;
- ▶ Rapide pour les données liées, bien plus rapide que SGBDR ;
- ▶ Modèles d'interrogation bien établis et performants : Tinkerpop pile (fournit un ensemble commun d'interfaces permettant aux différentes technologies informatiques graphiques de travailler ensemble, que le développeur utilise en cas de besoin), SPARQL et Cypher.

### Faiblesses :

- ▶ Fragmentation (sharding) :
  - Même si elles peuvent évoluer assez bien ;
  - Pour certains domaines, on peut aussi fractionner.



# BD NoSQL orientées Graphes

## Utilisations principales des BD NoSQL orientées graphes

Les BD NoSQL type orientées graphes sont principalement utilisées pour :

- ▶ Moteurs de recommandation ;
- ▶ Business Intelligence (BI) ;
- ▶ Semantic Web ;
- ▶ Social computing ;
- ▶ Données géospatiales ;
- ▶ Généalogie ;
- ▶ Web of things ;
- ▶ Catalogue des produits ;
- ▶ Sciences de la Vie et calcul scientifique (bio-informatique) ;
- ▶ Données liées, données hiérarchiques ;
- ▶ Services de routage, d'expédition et de géolocalisation ;
- ▶ Services financiers : chaîne de financement, dépendances, gestion des risques, détection des fraudes,...

# Plan

- 1 Introduction au Big Data
- 2 Hadoop, MapReduce et le Big Data
- 3 Big Data et NoSQL
- 4 Bibliographie et Logistique Recommandées

# Bibliographie et Logistique Recommandées

- ▶ <https://www.lebigdata.fr/definition-big-data>
- ▶ <https://www.lebigdata.fr/definition-big-data>  
<https://www.talend.com/fr/resources/guide-big-data/>
- ▶ <https://www.mongodb.com/>
- ▶ <https://hbase.apache.org/> <https://cassandra.apache.org/>
- ▶ <https://www.dropbox.com/>
- ▶ <https://azure.microsoft.com/fr-fr/>
- ▶ <https://aws.amazon.com/fr/>
- ▶ <https://www.hosteur.com/stockage/hdrive>