Big Data et traitement des masses de données

Ghazi Aziz Sup'Management
2022

sommaire

Introduction

Les entreprise sont confrontées actuellement à des données de volumes considérables à traiter et présentant un fort enjeux commercial et marketing

Quelques chiffres

- Chaque jour, nous générons 2,5 trillions d'octets de données
- 90% des données dans le monde ont été créées au cours des dernières années.

Ces gros volumes de données deviennent de plus en plus difficiles à travailler avec des outils classiques de gestion de données.
Il faut donc un ensemble de technologies, d'outils et de procédures pour capter, traiter et analyser ces données hétérogènes afin d'extraire des informations utiles à un coût accessible.

Quelques chiffres

- La capacité de stockage double tous les 13 mois alors que le coût est divisé par deux (Loi de Kryder)
  - **1890 : la carte perforée, une mémoire à 80 trous**
  - **1928 : 50 octets par centimètre de bande magnétique**
  - **1956 : Premier disque dur (5Mo , 7000 €/Mo (env. 100 000 € actuels !))**

Quelques chiffres

- La capacité de stockage double tous les 13 mois alors que le coût est divisé par deux (Loi de Kryder)
  - **1890 : la carte perforée, une mémoire à 80 trous**
  - **1928 : 50 octets par centimètre de bande magnétique**
  - **1956 : Premier disque dur 5Mo (≈ 100 000€/Mo)**
  - **1979 : Seagate ST-506 : 5Mo (≈ 1000€/Mo)**
  - **1982 : une disquette (1,44 Mo) de la taille d'une poche de chemise**
  - **1984 : 80 minutes de musique sur les premiers CD-Rom (600 Mo)**
  - **1994 : 700 Mo sur un seul disque Zip (10 minutes de vidéo)**
  - **2000 : plus de support magnétique pour les cartes mémoire flash**
    - **1991 : 1 Go à 40 000€ ; 2015 : 512Go à 1 500€ ; 2017 : 512 Go à 300€**
  - **2011 : Seagate barracuda : 1 To (≈ 0,00012 €/Mo)**
  - **2014 : Cloud à1.000 milliards d'octets pour 7,2 euros (offre Google)**

Quelques chiffres

- Années 70 : 256 octets à 8 Ko
- Années 80 : 8 Ko à 1 Mo
- Années 90 : 1 Mo à 512 Mo
- Années 2000 : 512 Mo à 8 Go
- Années 2010 : 8 Go à ...

I've said some stupid things and some wrong things, but not that. No one involved in computers would ever say that a certain amount of memory is enough for all time.

J'ai dit des choses stupides et des choses fausses, mais pas cela. Une personne impliquer dans les ordinateurs ne dira jamais qu'une certaine quantité de mémoire est suffisante pour toujours.

Quelques chiffres

Date	Nom	Nombre de transistors	Gravure (micron)	Fréq. max horloge	Largeur données	MIPS
1971	4004	2 300	10	108 kHz	4 bits	0,06
1974	8080	6 000	6	2 MHz	8 bits	0,64
1979	8088	29 000	3	5 MHz	16 bits	0,33
1982	80286	134 000	1,5	20 MHz	16 bits	1
1985	80386	275 000	1,5	40 MHz	32 bits	5
1989	80486	1 200 000	1	100 MHz	32 bits	20
1993	Pentium	3 100 000	0,8 à 0,28	233 MHz	32 bits	100
1997	Pentium II	7 500 000	0,35/0,25	450 MHz	32 bits	300
1999	Pentium III	9 500 000	0,25/0,13	1,4 GHz	32 bits	510
2000	Pentium 4	42 000 000	0,18/0,065	3,8 GHz	32 bits	1 700
2004	Pentium 4D	125 000 000	0,09/0,065	3,6 GHz	32 bits	9 000
2006	Core 2 Duo	291 000 000	0,065	2,4 GHz	64 bits	22 000
2007	Core 2 Quad	2*2 1 000 000	0,065	3 GHz	64 bits	2*22 000
2008	Core 2 Duo	410 000 000	0,045	3,33 GHz	64 bits	24 200
2008	Core 2 Quad	2*410 000 000	0,045	3,2 GHz	64 bits	2*24 200
2009	Intel Core i5	774 000 000	0,045	2,93 GHz	64 bits	76 383
2010	Intel Core i7	1 170 000 000	0,032	3,33 GHz	64 bits	147 600

Quelques chiffres

- Réseaux sociaux
  - Facebook génère 500 Tera-octets (To) par jour.
- Données scientifiques
  - « Large Hadron Collider » (LHC) : 5 Péta-octets de données scientifiques/ jour
  - European Sentinel Program (Télédétection): 2 To par jour ...
- Internet :
  - env. 20 Zeta-octet par an (40 en 2020 ?)
- Données ouvertes :
  - Le site data.gouv.fr offre plus de 20 000 bases de données publiques

- NB

Tera : 10^{12} Peta : 10^{15} *Zetta : 10^{21}* *Yotabytes : 10^{24}*

Nombre d'atomes dans l'Univers : $\approx 10^{80}$

Quelques chiffres

Trillion : Un milliard de milliards (soit 10^{18})
Billion : Un milliard en Anglais

Avant les Big Data

Traitement massif des données

- Calcul réparti et HPC(High Performance Computer)

✓ Effort concentré sur les problèmes de "calcul intensif"

✓ Les réseaux n'étaient pas si performants, on évitait de faire des transferts de données.

- Bases de données

✓ Le modèle relationnel s'est imposé dans les années 80

✓ Données structurées (tableaux), formes normales

✓ Très forte optimisation des SGBD

Big Data ?

- Deux acceptations (frontières assez floues …)
  - Utilisation plus exhaustive, plus rapide, avec une **valorisation transversale plus systématique** de plus gros volumes de données en améliorant et automatisant les **outils existants**
  - Utilisation de **nouveaux outils** d'analyse de gros volumes de données pour obtenir des **informations autrefois impossibles à obtenir.**

Big Data ?

Frontière du BigData
On considère du BigData quand le traitement devient trop long pour une seule machine.

Définition de Big Data : (ISACA, 2013)
Le Big Data désigne des ensembles de données qui sont devenus trop volumineux ou qui évoluent trop vite pour être analysés dans un laps de temps raisonnable par le biais des techniques traditionnelles de bases de données multidimensionnelles ou relationnelles, ou encore des outils logiciels habituellement utilisés pour la saisie, la gestion et le traitement des données.

L'expression « Big Data » date de 1997 selon l'Association for Computing Machinery. En 2001, l'analyste du cabinet Meta Group (devenu Gartner) Doug Laney décrivait les big data d'après le principe des « trois V » :

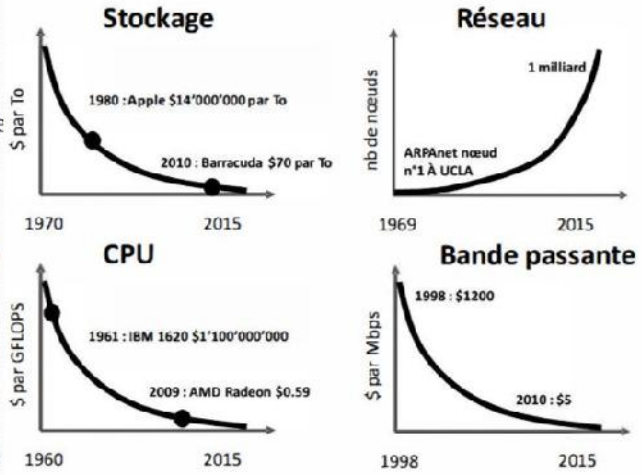
Avènement du Big Data

DES CAUSES ECONOMIQUES ET TECHNOLOGIQUES

1 Une baisse des prix exponentielle

Durant ces vingt dernières années le prix des ressources IT a chuté de manière exponentielle en accord avec la célèbre loi de Moore (selon Moore tous les 18 mois il y a doublement du nombre de transistors)

Qu'il s'agisse de la capacité de stockage, du nombreux de nœuds que l'on peut mettre en parallèle dans un datacenter, de la fréquence des CPU ou encore de la bande passante disponible (qui a favorisé l'émergence des services cloud).



Avènement du Big Data

2 Des progrès initiés par les géants du web

Pour bénéficier de ces ressources de stockage colossales, les géants du web ont du développer pour leurs propres besoins de nouvelles technologies notamment en matière de parallélisation des traitements opérant sur des volumes de données se chiffrant en plusieurs centaines de téraoctets.

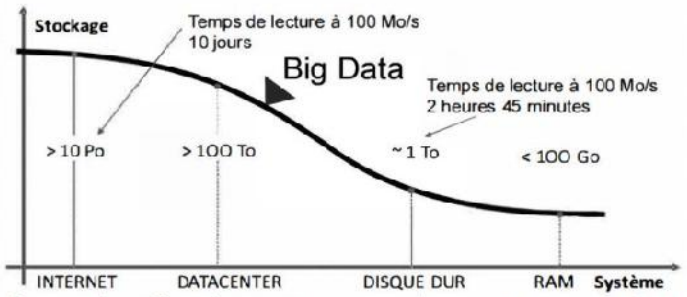
L'un des principaux progrès en la matière est venu de Google qui, pour traiter les données récupérées par les *crawlers de son moteur de recherche et indexer la* totalité du web, a mis au point un modèle de conception qui permet d'automatiser la parallélisation d'une grande classe de traitements. C'est le célèbre modèle **MapReduce**

De nombreuses avancées en génie logiciel développées à cette occasion ont par la suite essaimées dans la communauté open source qui en a proposé des systèmes équivalents gratuits. Le système de traitement parallèle **Hadoop Apache** est le principal exemple de ce transfert de technologie vers le monde open source.

Avènement du Big Data

3 Où se trouve la frontière du Big Data ?

Quelques ordres de grandeurs d'espaces de stockage ainsi que la frontière approximative du Big Data.



À titre d'exemple si l'on doit traiter ou analyser un téraoctet de données, qui correspond à la taille d'un disque dur, en quelques minutes il faudra impérativement recourir à une mise en parallèle des traitements et du stockage sur plusieurs nœuds. Selon cette définition le Big Dat a n'est pas uniquement lié à la taille mais à la vitesse des traitements

Avènement du Big Data

1998 Première base NoSQL (et utilisation du terme)
2000 Graph database Neo4j
2001 Première définition du Big Data (Volume / Variété /Vélocité) par Gartner
2005 Naissance Hadoop, Naissance de CouchDB (base NoSQL orientée documents)
2006 Publication Google BigTable
2007 Naissance de Hbase (base orientée documents)
2007 Base de MongoDB (base orientée documents)
2008 Naissance de Cassandra (base orientée documents)
2008 Création société Cloudera
2008 Hadoop bat le record « Terabyte sort Benchmark »
2009 Base clé/valeur Redis
2010 Création société DataStax (Cassandra)
2011 Création société Hortonworks
2012 YARN en remplacement de Hadoop v1
2013 Création société DataBricks (Spark)
2014 Spark bat le record «Terabyte sort Benchmark»

Avènement du Big Data

Quelques exemples qui ne relèvent pas du Big Data

- Un volume de données que l'on peut traiter au moyen d'une fiche Excel.
- Des données que l'on peut héberger dans un seul nœud d'une base de données relationnelle.
- Les données qui sont « chères » à produire telles que celles qui sont collectées par sondage ou par recensement ou produite par un organisme.
- Les données issues de capteurs physiques comme ceux de l'internet des objets
- Les données publiques librement disponibles au téléchargement. Là encore on se place dans une perspective où ce qui relève du Big Data ne pourra être téléchargé au moyen d'une seule connexion Internet même à très haut débit.

Avènement du Big Data

Quelques exemples qui relèvent du Big Data

- Les volumes de données qu'il n'est pas possible de stocker ou de traiter avec les technologies traditionnelles que les SGBDR ou pour lesquelles le coût d'un tel traitement serait prohibitif.
- Les données de logs transactionnelles d'un site web d'une grande enseigne de distribution.
- Le trafic d'un gros site web.
- Les données de localisation GSM d'un opérateur téléphonique sur une journée.
- Les données boursières échangées quotidiennement sur une grande place financière.

Récapitulatif

1. **Citez la loi de Kryder**
2. **Quelles sont les causes économiques et techniques qui ont permet l'avènement du Big Data?**
3. **Citez la loi de Moore**
4. **Donnez la définition du Big Data**
5. **Où se trouve la frontière du Big Data ?**
6. **Donnez Quelques exemples qui ne relèvent pas du Big Data ?**
7. **Donnez Quelques exemples qui relèvent du Big Data ?**

LES «5V » DU BIG DATA

Dimensions (5V)

Le *big data s'accompagne du développement d'applications à visée analytique, qui traitent* les données pour en tirer du sens. Ces analyses sont appelées Big Analytics. Elles portent sur des données quantitatives complexes à l'aide de méthodes de calcul distribué et des statistiques.

Gartner 2001 : 3V (Volume, Variété, Vélocité)
IBM 2012 : 4V (+Véracité)
2015 : 5V (+ Valeur)

LES «5V » DU BIG DATA

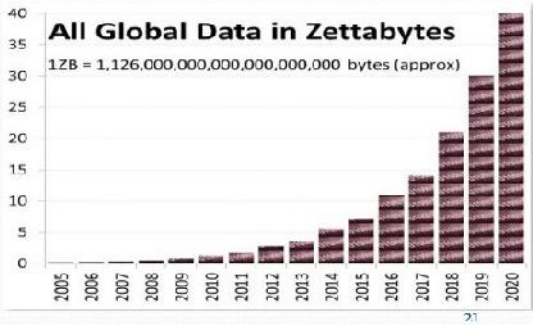
Volume

Le volume des données stockées est en pleine croissance: les données numériques de tous types créées dans le monde seraient passées de 1,2 zo par an en 2010 à 40 zo en 2020.

Chaque jour 2.5 trillions octets de données sont générées.
Prévision d'une croissance de 800% de données pour les 5 prochaines années.

Le prix de stockage a beaucoup diminué
Des solutions de stockage fiables sont nombreuses

- ➢ SAN (Storage Area Networks)
- ➢ Stockage sur le cloud



LES «5V » DU BIG DATA

Volume

Comment déterminer les données qui méritent d'être stockées?

- ➢ Transactions? Logs? Métier? Utilisateur? Capteurs? Médicales? Sociales?
- ➢ **Aucune donnée n'est inutile (juste pas encore servies)**

Accélération matérielle

- **Nombreux serveurs/clusters**

Un serveur unique ne peut stocker cette quantité d'information, garantir des temps d'accès pour un grand nombre d'utilisateur, faire des calculs rapides, etc

- **Besoin de distribuer les calculs et les données**

Comme il y a plusieurs serveurs/clusters, on a besoin d'algorithmes permettant le calcul et la distribution des données à large échelle.

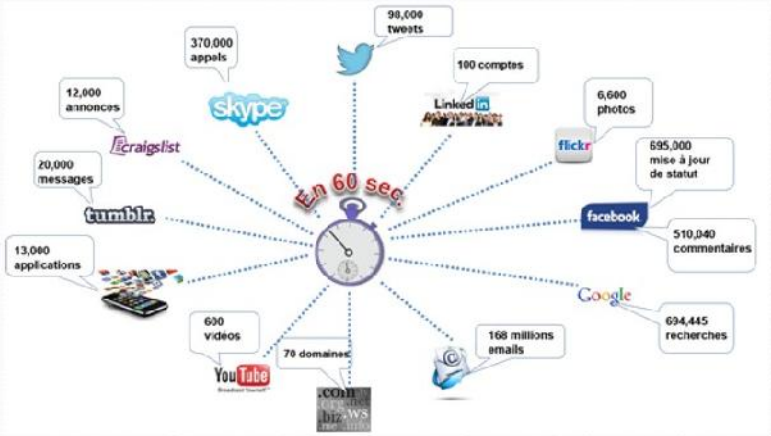
Data centers de quelques grands acteurs du Big Data :

- Google DataCenter : 70 000 servers/data center et 16 data centers, ~1M de serveurs
- Facebook : 5 data centers
- Amazon : 7 data centers, 450 000 severs
- Microsoft :environ 1M serveurs

LES «5V » DU BIG DATA

Variété

Le volume des big data met les data center face à un réel défi : la variété des données. Il ne s'agit pas de données relationnelles traditionnelles, mais des données brutes, semi-structurées ou non structurées.
Ce qui les rend difficilement utilisables avec les outils traditionnels.



LES «5V » DU BIG DATA

Vélocité (vitesse)

La vélocité représente la fréquence à laquelle les données sont générées, capturées, partagées et mises à jour.
Parfois, les données doivent être saisies et traitées au fur et à mesure de leur collection en temps réel (fouille de flots de données) pour répondre aux besoins des processus chrono-sensibles.

Par exemple, les systèmes mis en place par la bourse et les entreprises doivent être capables de traiter les données avant qu'un nouveau cycle de génération n'ait commencé, avec le risque pour l'Homme de perdre une grande partie de la maîtrise du système.

Ceci parce que les principaux opérateurs deviennent des "robots" capables de lancer des ordres d'achat ou de vente à la nanoseconde (Trading haute fréquence) sans disposer de tous les critères pertinents d'analyse pour le moyen et long terme.

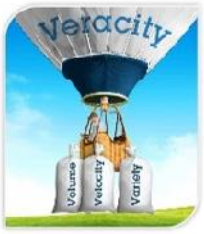
LES «5V » DU BIG DATA

Vélocité (vitesse)

- **Données créées plus rapidement.**
- **Arrivent plus rapidement aux organisations.**
- **Doivent être traitées plus vite (en temps réel).**
- **Décision rapide sur les données que l'on désire garder.**
- **Pression pour convertir rapidement les données en décision d'affaires.**
- **Les résultats livrés sont consommés plus rapidement.**

LES «5V » DU BIG DATA

Le paradoxe du Big Data : Plus de données, moins de confiance



Véracité (Veracity)

- ➢ La qualité de la fiabilité et la confidence des données
- ➢ Comment se trouver dans un déluge de hashtags ?
- ➢ Comment gérer les données partielles ou incomplètes ?
- ➢ Données bruitées, imprécises, prédictives...

La véracité répond à la question : « Est-ce qu'on peut faire confiance à la donnée disponible ?
Contient-elle suffisamment d'information ? »

Exemple

- Génération des données par spambots
- Elections présidentielles de 2012 au Mexique avec de faux comptes twitter.

Conséquences

1/3 chefs d'entreprises ne font pas confiance aux données qu'ils utilisent.

1 sur 3 Prennent des décisions basées sur des information non fiables

1 sur 2 N'ont pas l'information dont ils ont besoin

60% Ont plus que ce qu'ils peuvent utiliser

40% Temps passé sur chaque projet lié au Big data pour comprendre l'information

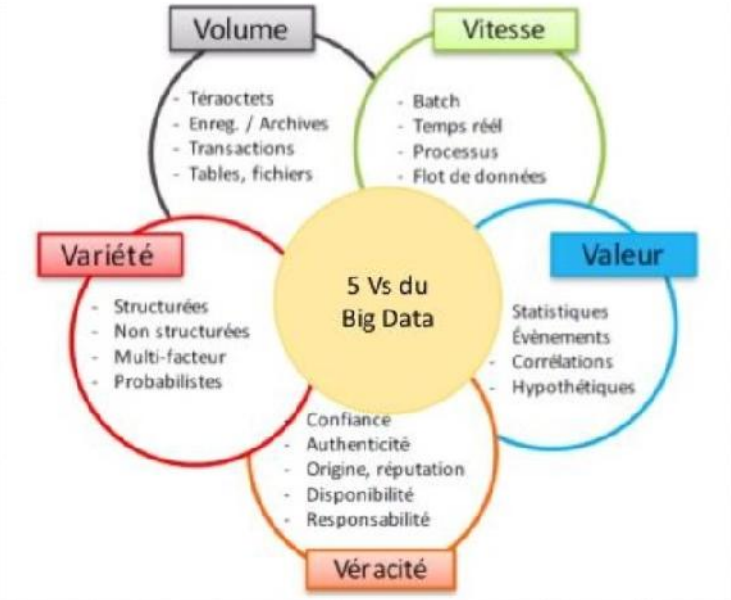
LES «5V » DU BIG DATA

Valeur (Value)

➢ La valeur ajoutée des données ou des informations extraites

➢ Sans une réelle valeur ajoutée, ce n'est qu'un gaspillage de ressources

LES «5V » DU BIG DATA



Les niveaux des structuration des données

Les différents niveaux de structuration des données

Niveau de structuration	Modèle	Exemple	Facilité de traitement
Structuré	Relationnel, objet, colonne	Base de données d'entreprise, Linkedin, …	Facile Indexé
Semi-structuré	XML, JSON, CSV, log	Google API, Twitter API, web logs, …	Facile Non indexé
Non structuré	texte, image, vidéo	Web, mail, document, …	Complexe « Brut force »

UN CHAMPS IMMENSE D'APPLICATIONS

Voici quelques exemples d'applications du Big Data en entreprise. La liste est loin d'être exhaustive :

Secteurs qui manipulent quotidiennement des volumes de données très important :

Les Banques : la sanctuarisation de données anciennes dues à des contraintes réglementaires ;

La Télécommunication : l'analyse de l'état du réseau en temps réel ;

Les Médias Numériques : le ciblage publicitaire et l'analyse de sites web ;

Les Marchés Financier : l'analyse des transactions pour la gestion des risques et la gestion des fraudes, ainsi que pour l'analyse des clients.

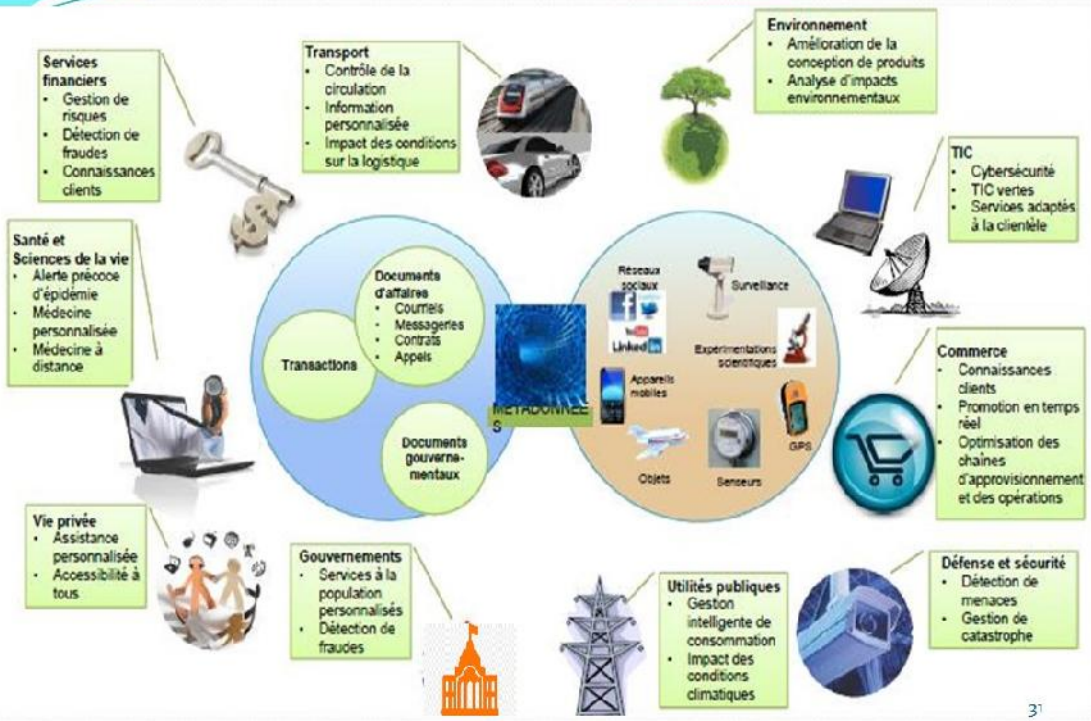
Secteurs plus hétérogène, les besoins, mais aussi l'utilisation qui est faite du Big Data, peuvent être très différents :

Les Services Publics : l'analyse des compteurs (gaz, électricité, etc.) et la gestion des équipements ;

Le Marketing : le ciblage publicitaire et l'analyse de tendance ;

La Santé : l'analyse des dossiers médicaux et l'analyse génomique.

UN CHAMPS IMMENSE D'APPLICATIONS



Election de Barack Obama 2012

• Les moyens :
– Maillage géographique très précis des électeurs
– Utilisation de données sociologiques et des réseaux sociaux
– Données récoltées en amont par le porte à porte et croisées avec des données internet
• Actions sur la Campagne :
– Les bénévoles de la campagne Obama savent quels arguments mettre en avant vis-à-vis des femmes célibataires, ou des jeunes, ou des hispaniques
– Ciblage précis des pubs, coups de fils et appels à donations
• Actions le jour du scrutin :
– Suivi en temps réel et à un niveau de détail important du niveau de participation
– Mobilisation des bénévoles sur les quartiers qui votent le moins

De nouveaux métiers du Big Data

Le Big Data a besoin de nouvelles compétences, il est donc normal de voir apparaître de nouveaux rôles :

Data Engineer : c'est l'informaticien, spécialiste du Big Data, qui va mettre en œuvre tous les outils et solutions à destination des utilisateurs (utilisateur final, data scientist, ...).

Data Analyst (Statisticien) : lorsque les analyses sur les données sont plus complexes, il faut alors faire appel à des statisticiens qui sont capables d'implémenter de nouveaux algorithmes et définir de nouveau modèles.

Chief Data Officer : Dans les grandes entreprises on nomme parfois un directeur des données. Il est en charge des données de l'entreprise, quelles soient internes ou externes :

-- gouvernance des données.

-- acquisition de nouvelles sources de données.

De nouveaux métiers du Big Data

Data Scientist Sa mission Construire des algorithmes pour extraire des informations pertinentes et utiles à partir des masses de données non structurées. *Un data scientist a une triple compétence : en statistiques et en mathématiques, en informatique et en développement, ainsi qu'une compétence business. Il est capable de créer des modèles de données, des algorithmes, pour par exemple imaginer un nouveau produit pour une nouvelle cible*

Architecte Big Data, qui élabore l'infrastructure technique permettant de gérer d'énormes volumes de données.

Data Visualizer, ont pour tâche de présenter l'information, sous forme de graphiques ou d'images, de façon compréhensible et efficace et fournir ainsi un outil précieux, notamment aux décideurs.

Data Miner Le rôle du Data Miner**, ou fouilleur de données**, est d'explorer les données à sa disposition pour trouver celles qui peuvent aider l'entreprise.

...

DATA

Data Scientist: The Sexiest Job of the 21st Century

by Thomas H. Davenport and D.J. Patil

FROM THE OCTOBER 2012 ISSUE

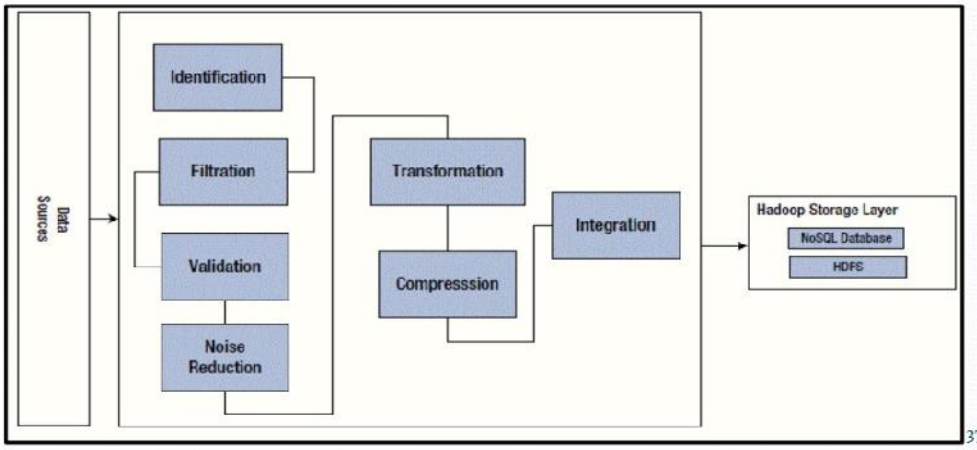
Harvard Business Review

Récapitulatif

1. Citez les 5V du Big Data
2. Quelle est la quantité des données produite chaque jour ?
3. Que représente la vélocité en Big Data ?
4. A quelle question répond la véracité du Big Data ?
5. Donnez les niveaux des structuration des données et citez qq exemples
6. Donnez qq applications du Big Data en dans les secteurs qui manipulent quotidiennement des volumes de données très important
7. Donnez qq applications du Big Data en commerce
8. Citez 5 nouveaux métiers du Big Data ?
9. Expliquez le métier du Data Scientist ?
10. Expliquez le métier du Data Visualizer

Processus de chargement et de collecte de données dans Big Data

La couche responsable du chargement de données dans Big Data, devrait être capable de gérer d'énorme volume de données, avec une haute vitesse, et une grande variété de données. Cette couche devrait avoir la capacité de valider, nettoyer, transformer, réduire (compression), et d'intégrer les données dans la grande pile de données en vue de son traitement.



Processus de chargement et de collecte de données dans Big Data

Identification des différents formats de données connues, par défaut Big Data cible les données non structurées ;

Filtration et sélection de l'information entrante pertinente pour l'entreprise ;

Validation et analyse des données en permanence ;

Réduction de bruit implique le nettoyage des données en supprimant le bruit ;

La transformation peut entrainer le découpage, la convergence, la normalisation ou la synthèse des données ;

Compression consiste à réduire la taille des données, mais sans perdre de la pertinence des données ;

Intégration consiste à intégrer l'ensemble des données dans le stockage de données de Big Data (**HDFS** ou **NoSQL** base).

Différence entre BI (Business intelligence) et Big Data

La méthodologie BI traditionnel fonctionne sur le principe de regrouper toutes les données de l'entreprise dans un serveur central (Datawarehouse ou entrepôt de données). Les données sont généralement analysées en mode déconnecté. Les données sont généralement structurées en SGBDR avec très peu de données non structurées.

Une solution Big Data, est différente d'une BI traditionnel dans les aspects suivants :

Les données sont conservées dans un système de fichiers distribué et scalable plutôt que sur un serveur central ;
Les données sont de formats différents, à la fois structurées ainsi que non structurées ;
Les données sont analysées en temps réel ;
La technologie Big Data s'appuie sur un traitement massivement parallèle (concept MPP -Massively Parallel Processing-).

Architecture Big Data

On distingue principalement les couches suivantes :

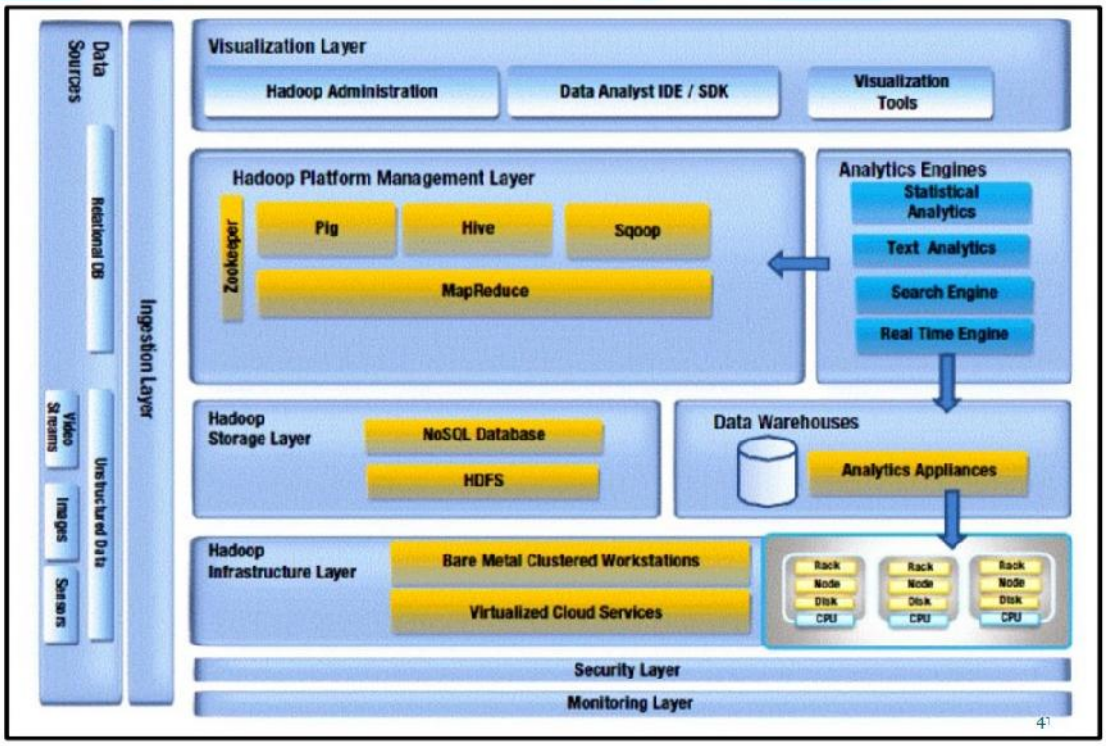
Couche matériel (infrastructure Layer) : peut-être des serveurs virtuels VMware, ou des serveurs Lame blade ;

Couche stockage (Storage layer) : les données seront stockées soit dans une base NoSQL, ou bien directement dans le système de fichier distribué ou les Datawarehouse ;

Couche management et traitement : on trouve dans cette couche les outils de traitement et analyse des données comme MapReduce ou Pig ;

Couche visualisation : pour la visualisation du résultat du traitement

Architecture Big Data



41

Les principales technologies de Big Data

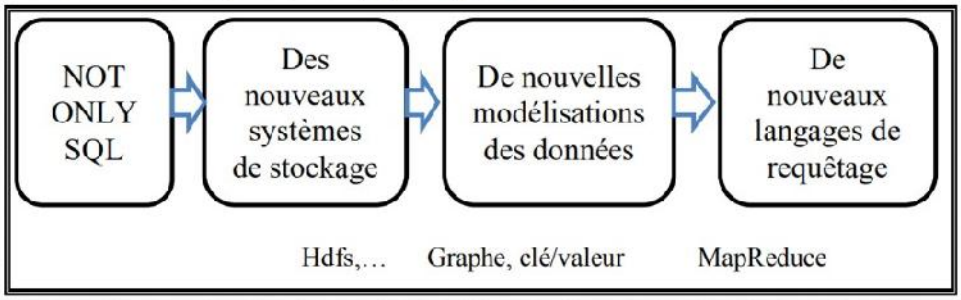
Elles sont nombreuses. Pour optimiser les temps de traitement sur des bases de données géantes, plusieurs solutions peuvent entrer en jeu :

- **Des bases de données NoSQL** (comme MongoDB, Cassandra ou Redis) qui implémentent des systèmes de stockage considérés comme plus performants que le traditionnel SQL pour l'analyse de données en masse (orienté clé/valeur, document, colonne ou graphe).
- **Des infrastructures de serveurs** pour distribuer les traitements sur des dizaines, centaines, voire milliers de nœuds. C'est ce qu'on appelle le traitement massivement parallèle. Le Framework Hadoop est sans doute le plus connu d'entre eux. Il combine le système de fichiers distribué HDFS, la base NoSQL, HBase et l'algorithme MapReduce.

42

Les principales technologies de Big Data

- **Le stockage des données en mémoire** : On parle de traitement in-memory pour évoquer les traitements qui sont effectués dans la mémoire vive de l'équipement informatique, plutôt que sur des serveurs externes.
  L'avantage du traitement in-memory est celui de la vitesse puisque les données sont immédiatement accessibles. En revanche, ces données ne sont pas stockées sur le long terme, ce qui peut poser des problèmes d'historisation.



Bases données NoSQL

Les bases de données NoSQL (No-SQL ou Not Only SQL) . Le terme NoSQL désigne une catégorie de systèmes de gestion de base de données destinés à manipuler des bases de données volumineuses pour des sites de grande audience. Les bases données NoSQL sont scalables, elles permettent de traiter les données d'une façon distribuée. Parmi les avantages du NoSQL on trouve :

- Leurs performances ne s'écroulent jamais quel que soit le volume traité. Leur temps de réponse est proportionnel au volume ;
- Elles se migrent facilement. En effet, contrairement aux SGBDR classiques, il n'est pas nécessaire de procéder à une interruption de service pour effectuer le déploiement d'une fonctionnalité impactant les modèles des données ;
- Elles sont facilement scalable. A titre d'exemple, le plus gros cluster de NoSQL fait 400 To, tandis qu'Oracle sait traiter jusqu'à une vingtaine de Téraoctet (pour des temps de réponse raisonnable).

Bases données NoSQL

Les contraintes des applications WEB à très grande échelle

Avec Les nouveaux systèmes WEB, sont apparus de nouveaux besoins métiers et de nouvelles exigences techniques. Via l'isolation des transactions et l'application de contraintes d'intégrité les SGBDR garantissent des données avec un haut niveau de cohérence.

Dans les applications e-business à grande échelle comme Amazon ou dans les réseaux sociaux comme Twitter, les priorités sont cependant très différentes. Pour une entreprise comme Amazon, par exemple, ne serait-ce que quelques minutes, est inenvisageable car elle causerait la perte de dizaine de milliers de commandes.

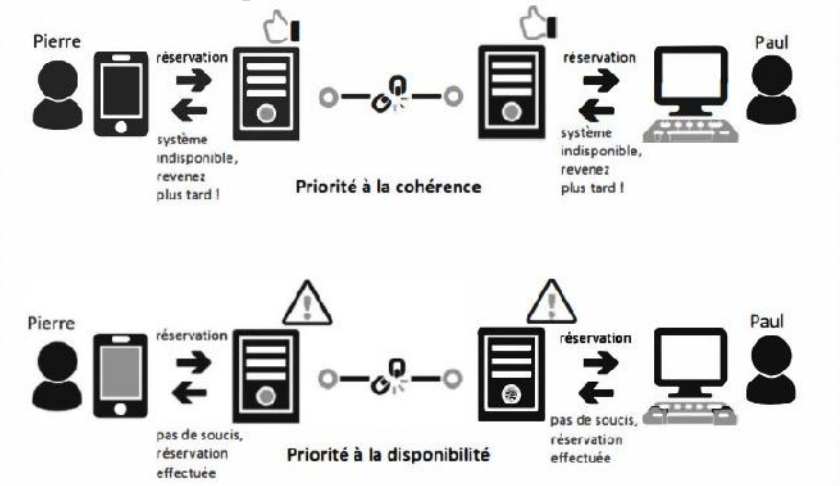
Priorité absolue est donc donnée aux performances et à la disponibilité. Celles- ci sont assurées aujourd'hui par la mise en œuvre d'architectures distribuées sur des centaines voir des milliers de machines.

l'ancienne exigence de cohérence des données ne correspond souvent plus aux priorités du business mais correspond plutôt à la disponibilité.

Bases données NoSQL

- **Sacrifier la cohèrence pour la disponibilité**

Lors d'une réservation effectuée sur un système distribué il faut choisir entre un système **disponible**, au risque de créer des **incohérences** de données, et un système dont les données sont cohérentes au risque cette fois de créer de l'attente et de perdre des clients



Un haut niveau de cohérence des données pénalise les performances ou la disponibilité d'une base de donnée

Bases données NoSQL

Sacrifier la flexibilité pour la vélocité

L'un des grands atouts des SGBDR est le découplage logique qu' ils autorisent entre les applications clientes et la structure des données que celles-ci utilisent.
Tous les usages sont en principe possibles, puisque chaque requête SQL est automatiquement convertie par le SGBDR en plan d'exécution. Cette flexibilité a toutefois un coût, car un schéma de données n' est évidemment optimal que pour certaines requêtes.
Dans les contextes Big Data où la vélocité prime souvent sur toute autre considération, ce postulat de flexibilité des SGBDR, devra donc lui aussi remis en question

ID_client	nom	prénom	commande_1		commande_2		commande_3	
			produit	quantité	produit	quantité	produit	quantité

La colocation des données impose de choisir un chemin d'accès aux données.
Récupérer la liste des produits commandés par un client est aisé mais récupérer la liste des clients qui ont commandé un produit exige de parcourir toute la table

Bases données NoSQL

Définir ce qu'est une base de données NoSQL

- **Distribuer les traitements et le stockage** sur des centaines voire des milliers de nœuds constitués de serveurs banalisés.
- **Donner la priorité aux performances** et à la **disponibilité** sur **l'intégrité** des données.
- **Traiter efficacement les données non structurées ou seulement partiellement** structurées.
- **Ils sont non relationnels** dans le sens ou ils n' offrent pas de jointures.

Bases données NoSQL

LES DIFFERENTES CATEGORIES DE SOLUTIONS NoSQL

- les entrepôts clé-valeur,
- les bases de données orientées documents
- les bases de données orientées colonnes
- Les bases de données orientées graphes.

Bases données NoSQL

Il en existe 4 types distincts qui s'utilisent différemment et qui se prêtent mieux selon le type données que l'on souhaite y stocker.

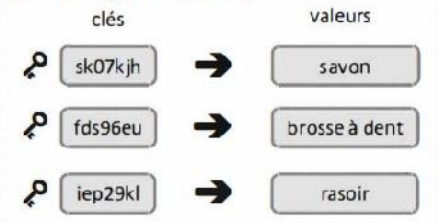
1) Les entrepôts clé-valeur ECV

Un entrepôt clé-valeur (ECV) peut être envisagé comme une collection de tables de hachage persistantes c'est-à-dire comme une collection de couples clé-valeur sur disque.

Les BD NoSQL fonctionnant sur le principe Clé-Valeur sont les plus basiques que l'on peut trouver.

- 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.

Exemples : Redis, Amazon SimpleDB, Riak, ...



Bases données NoSQL

2) les bases de données orientées documents BDOD

Une base de données orientées document (BDOD) peut schématiquement se décrire comme un ECV dont les valeurs sont des documents semi structurés. Par ce terme on entend des documents auto descriptifs généralement écrits avec un format de type XML, JSON ou similaire.
Par contraste avec les SGBDR qui exigent que chaque enregistrement d'une table ait les mêmes colonnes, spécifiées par un schéma, rien n'exige que les document s stockés dans une BDOD aient tous le même format.

BDOD	SGBDR
base de données	schéma
collection (de documents)	table
document	enregistrement
Id de document	Id d'enregistrement
DBRef (référence entre documents)	jointure (entre tables)

Equivalences entre SGBDR et BDOD

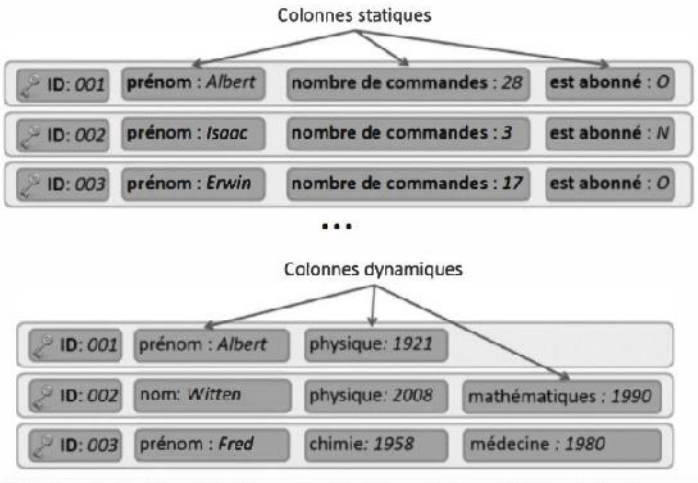
qq implémentations de BDOD : *MongoDB, CouchDB, RavenDB et Lotus Notes.*

Bases données NoSQL

3) Les bases orientées colonnes BDOC

cassandra

une BDOC comme *Cassandra est qu'il s'agit d'un ECV dont les valeurs possèdent une structure bien particulière.* Cette structure en l'occurrence est celle de colonnes dont les noms peuvent être soit *statiques*, *ils sont alors partagés par tous les enregistrements d'une collection de* colonnes, soit *dynamiques, c'est-à-dire qu'ils peuvent varier d'un enregistrement à* l'autre au sein d'une telle collection.



APACHE HBASE

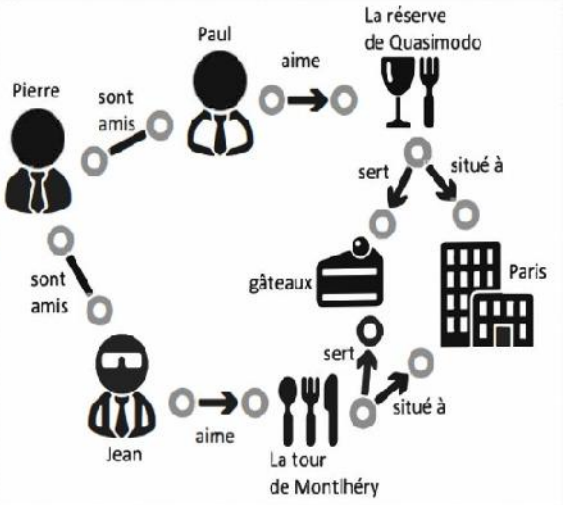
Bases données NoSQL

4) Les bases de données orientées graphes BDOG

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.

01/02/2021



- Un exemple de graphe qui relie des individus des services et des lieux géographiques

Parmi les BDOG citons: *Neo4j, Infinite Graph et OrientDB*

Les principales bases de données NoSQL

mongoDB

MongoDB

La plus populaire des bases NoSQL documentaires est écrite en C et n'utilise pas de machine virtuelle JAVA. Cette base est idéale pour débuter car elle est à la fois polyvalente et simple. Aussi à l'aise pour le stockage massif de données que pour le développement rapide orienté Web. Elle possède également une documentation de premier ordre

cassandra

Cassandra

Cassandra est le projet open source qui découlent de la technologie de stockage Facebook. À l'origine, elle a été écrite spécifiquement pour répondre à la croissance explosive de cette entreprise. Elle est assez complexe à configurer, mais elle permet d'adresser toutes les situations ou la performance et le traitement de la volumétrie est critique. Cassandra est une base de données en colonnes écrite en JAVA.

APACHE HBASE

HBase

Hbase est inspirée des publications de Google sur BigTable. Comme BigTable, elle est une base de données oriente colonne. Basée sur une architecture maitre/esclave, les bases de données HBase sont capables de gérer d'énormes quantités d'informations (plusieurs milliards de lignes par table).

RECAPITULATIF

1. Dessinez le Processus de chargement et de collecte de données dans Big Data
2. Donnez la différence entre BI (Business intelligence) et Big Data
3. Quelles sont les couches principales de l'architecture Big Data ?
4. Quelles sont les principales technologies de Big Data
5. Définir ce qu'est une base de données NoSQL
6. Donnez les différentes catégories de solutions NoSQL
7. Qu'est ce que Les entrepôts clé-valeur ECV ?
8. Qu'est ce que les bases de données orientées documents BDOD
9. Qu'est ce que les bases orientées colonnes BDOC
10. Qu'est ce que les bases orientées graphe BDOG
11. Donnez 3 exemples des principles base de données NoSQL

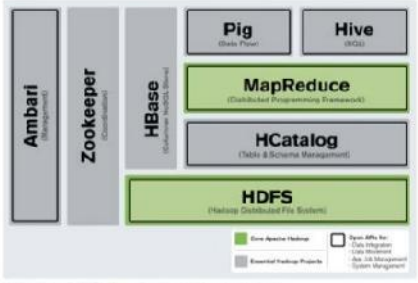
Hadoop et ses composants

Hadoop, le fer de lance des technologies Big Data

Hadoop permet de constituer une plateforme Big Data complète

- ✓ Scalable sur des volumes
- ✓ Tolérant aux pannes
- ✓ Open source
- ✓ Conçu pour un hardware standard

CORE HADOOP COMPONENTS



Hortonworks Data Platform, Powered by Apache Hadoop

Principales Distributions Hadoop

cloudera hortonworks MAPR TECHNOLOGIES EASY. DEPENDABLE. FAST. IBM

Présentation d'Hadoop

Hadoop est un Framework Java open source d'Apache pour réaliser des traitements sur des volumes de données massifs, de l'ordre de plusieurs pétaoctets (soit plusieurs milliers de To).
Hadoop a été conçu par Doug Cutting diplômé de l'Université de Stanford en 2004, également à l'origine du moteur Open Source Nutch. Doug Cutting cherchait une solution pour accroître la taille de l'index de son moteur. Il eut l'idée de créer un Framework de gestion de fichiers distribués.
Yahoo! en est devenu ensuite le principal contributeur, le portail utilisait notamment l'infrastructure pour supporter son moteur de recherche historique. Comptant plus de 10 000 clusters Linux en 2008, il s'agissait d'une des premières architectures Hadoop digne de ce nom. Crée spécialement pour les gros volumes.
Facebook pour l'analyse des logs, Google pour l'analyse des requêtes, etc...

YAHOO!

Google

Présentation d'Hadoop

Il est caractérisé par :

- **Robuste** : si un nœud de calcul tombe, ses tâches sont automatiquement réparties sur d'autres nœuds. Les blocs de données sont également répliqués;
- **Coût** : il optimise les coûts via une meilleure utilisation des ressources présentées;
- **Souple** : car il répond à la caractéristique de variété des données en étant capable de traiter différents types de données;
- **Virtualisation** : ne plus se reposer directement sur l'infrastructure physique (baie de stockage coûteuse), mais choisir la virtualisation de ses clusters Hadoop.

Trois principales distributions Hadoop sont aujourd'hui disponibles : Cloudera, Hortonworks, MapR.
Nous allons présenter deux concepts fondamentaux d'Hadoop : Sa propre version de l'algorithme MapReduce à savoir Hadoop MapReduce et son système de fichiers distribué HDFS.

Le système de fichier distribué d'Hadoop HDFS

Hadoop utilise un système de fichiers virtuel qui lui est propre : le HDFS (Hadoop Distributed File System).
HDFS est un système de fichier distribué, extensible et portable inspiré par le Google File System (GFS).
Il a été conçu pour stocker de très gros volumes de données sur un grand nombre de machine équipées de disques durs banalisés, il permet de l'abstraction de l'architecture physique de stockage, afin de manipuler un système de fichier distribué comme s'il s'agissait d'un disque dur unique.
Toutefois, HDFS se démarque d'un système de fichiers classique pour les principales raisons suivantes :

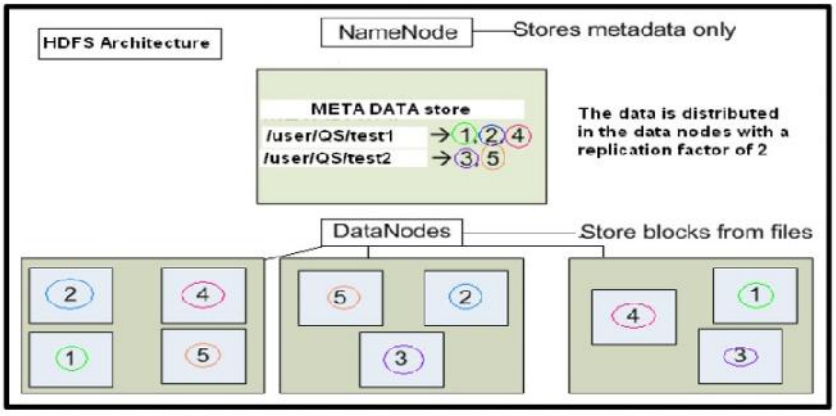
HDFS n'est pas dépendant du noyau du système d'exploitation, Il assure une portabilité et peut être déployé sur différents systèmes d'exploitation.

HDFS est un système distribué sur un système classique. Dans HDFS, chaque nœud d'un cluster correspond à un sous-ensemble du volume global des données du cluster.

HDFS utilise des tailles de blocs largement supérieures à ceux des systèmes classiques. Par défaut, la taille est fixée à 64 Mo. Il est toutefois possible de monter à 128 Mo, 256 Mo, 512 Mo voire 1 Go. Alors que sur des systèmes classiques, la taille est généralement de 4 Ko

Le système de fichier distribué d'Hadoop HDFS

HDFS fournit un système de réplication des blocs dont le nombre de réplications est configurable. Pendant la phase d'écriture, chaque bloc correspondant au fichier est répliqué sur plusieurs nœuds. Pour la phase de lecture, si un bloc est indisponible sur un nœud, des copies de ce bloc seront disponibles sur d'autres nœuds.



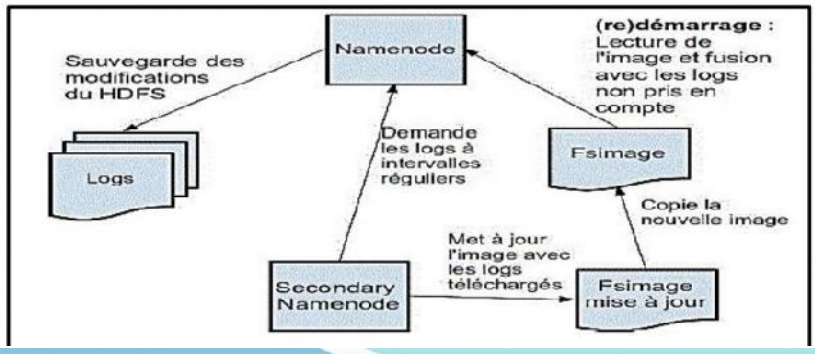
Les composantes d'HDFS

HDFS définit deux types de nœuds :

- Le nœud principal ou NameNode
- Le nœud de données ou DataNode

Le NameNode dans l'architecture Hadoop est un point unique de défaillance. Si ce service est arrêté, il n'y a pas moyen de pouvoir extraire les blocs d'un fichier donné. Pour résoudre ce problème, un NameNode secondaire appelé Secondary NameNode a été mis en place dans l'architecture Hadoop. Son rôle consiste à :

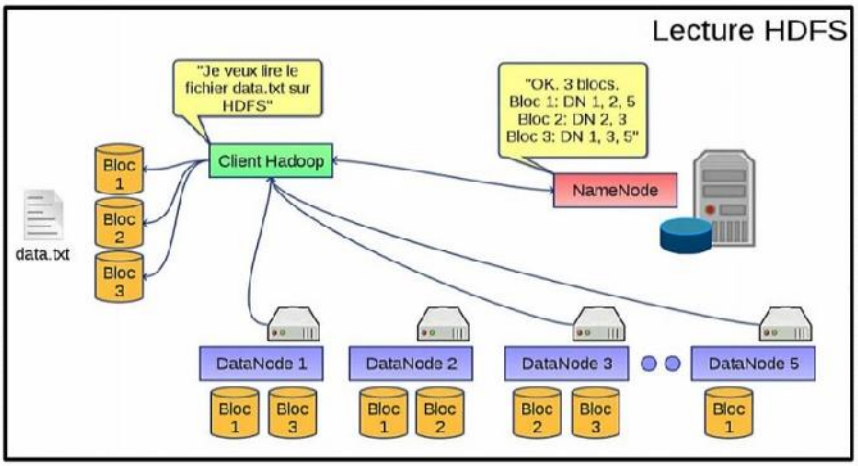
Télécharger régulièrement les logs sur le NameNode ;
Crée une nouvelle image en fusionnant les logs avec l'image HDFS ;
Renvoie la nouvelle image au NameNode.



61

Lecture d'un fichier HDFS

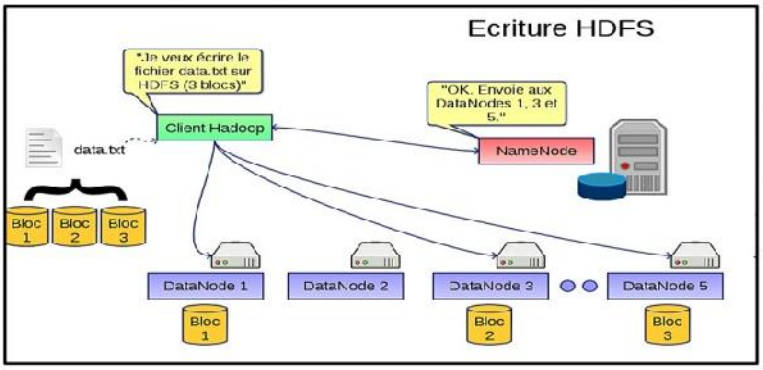
Pour lire un fichier au sein de HDFS, il faut suivre les étapes suivantes :
Etape 1 : Le client indique au NameNode qu'il souhaite lire le fichier data.txt
Etape 2 : Le NameNode lui indiquera la taille de fichier (nombre de blocs) ainsi que les différents DataNode hébergeant les n blocs.
Etape 3 : Le client récupère chacun des blocs à un des DataNodes.
Etape 4 : En cas d'erreur/non réponse d'un des DataNode, il passe au suivant dans la liste fournie par le NameNode



62

Ecriture dans un fichier ou volume HDFS

Etape 1 : On va utiliser la commande principale de gestion de Hadoop: Hadoop, avec l'option fs. Admettons qu'on souhaite stocker le fichier data.txt sur HDFS.
Etape 2 : Le programme va diviser le fichier en blocs de 64KB (ou autre, selon la configuration) – supposons qu'on ait ici 3 blocs.
Etape 3 : Le NameNode lui indique les DataNodes à contacter.
Etape 4 : Le client contacte directement le DataNode concerné et lui demande de stocker le bloc.
Etape 5 : les DataNodes s'occuperont – en informant le NameNode – de répliquer les données entre eux pour éviter toute perte de données.
Etape 6 : Le cycle se répète pour le bloc suivant.



MapReduce

MapReduce est un paradigme (modèle) de programmation parallèle proposé par Google. Il est principalement utilisé pour le traitement distribué sur de gros volumes de données aux seins d'un cluster de nœuds. Il est conçu pour la scalabilité et la tolérance aux pannes.
Le modèle de programmation fournit un cadre à un développeur afin d'écrire une fonction Map et une fonction Reduce. Tout l'intérêt de ce modèle de programmation est de simplifier la vie du développeur. Ainsi, ce développeur n'a pas à se soucier du travail de parallélisation et de distribution du travail. MapReduce permet au développeur de ne s'intéresser qu'à la partie algorithmique.

MapReduce

Un programme MapReduce peut se résumer à deux fonctions Map () et Reduce ()

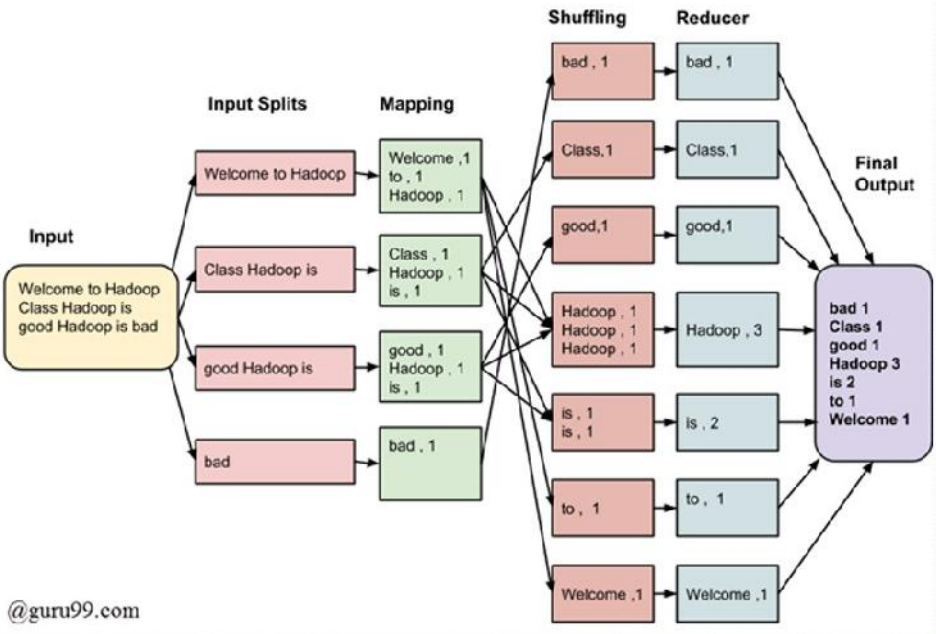
La première, MAP, va transformer les données d'entrée en une série de couples clef /valeur. Elle va regrouper les données en les associant à des clefs, choisies de telle sorte que les couples clef/valeur aient un sens par rapport au problème à résoudre. Par ailleurs, cette opération doit être parallélisable: 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. La fonction Map s'écrit de la manière suivante :

Map (clé1, valeur1) → List (clé2, valeur2).

La seconde, REDUCE, va appliquer un traitement à toutes les valeurs de chacune des clefs distinctes produite par l'opération MAP. Au terme de l'opération REDUCE, on aura un résultat pour chacune des clefs distinctes. Ici, on attribuera à chacune des machines du cluster une des clefs uniques produites par MAP, en lui donnant la liste des valeurs associées à la clef. Chacune des machines effectuera alors l'opération REDUCE pour cette clef. La fonction Reduce s'écrit de la manière suivante :

Reduce (clé2, List (valeur2)) → List (valeur2)

MapReduce



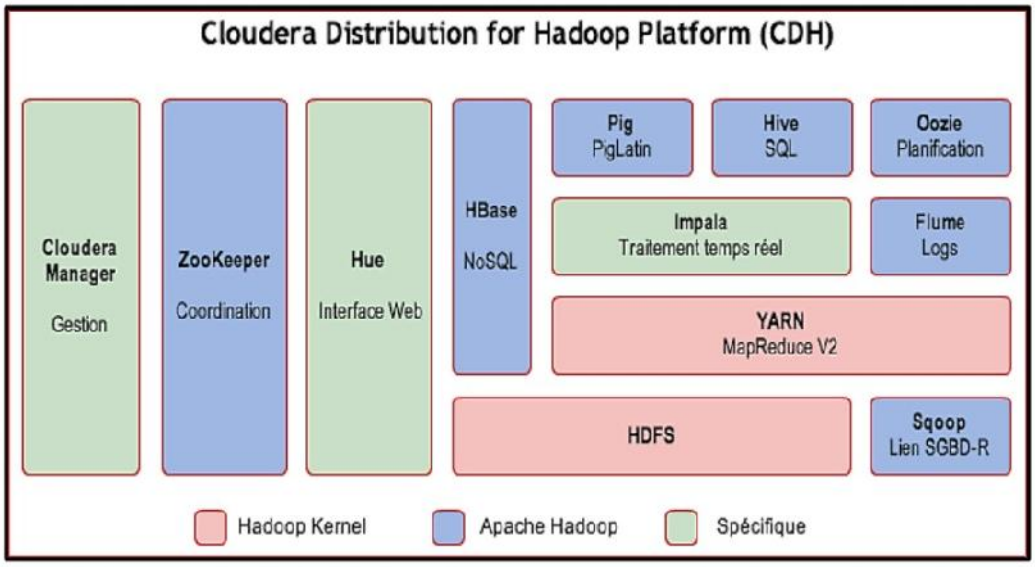
Implémentation JAVA de la fonction map() :

```
public class WordCountMapper extends Mapper<LongWritable, Text,
Text, IntWritable> {
  public void map(LongWritable key, Text value, Context context)
throws IOException, InterruptedException {
StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
         String string = itr.nextToken();
         if (!string.equals("se")) {
            Text word = new Text();
word.set(string);
context.write(word, new IntWritable(1));}}}}
```

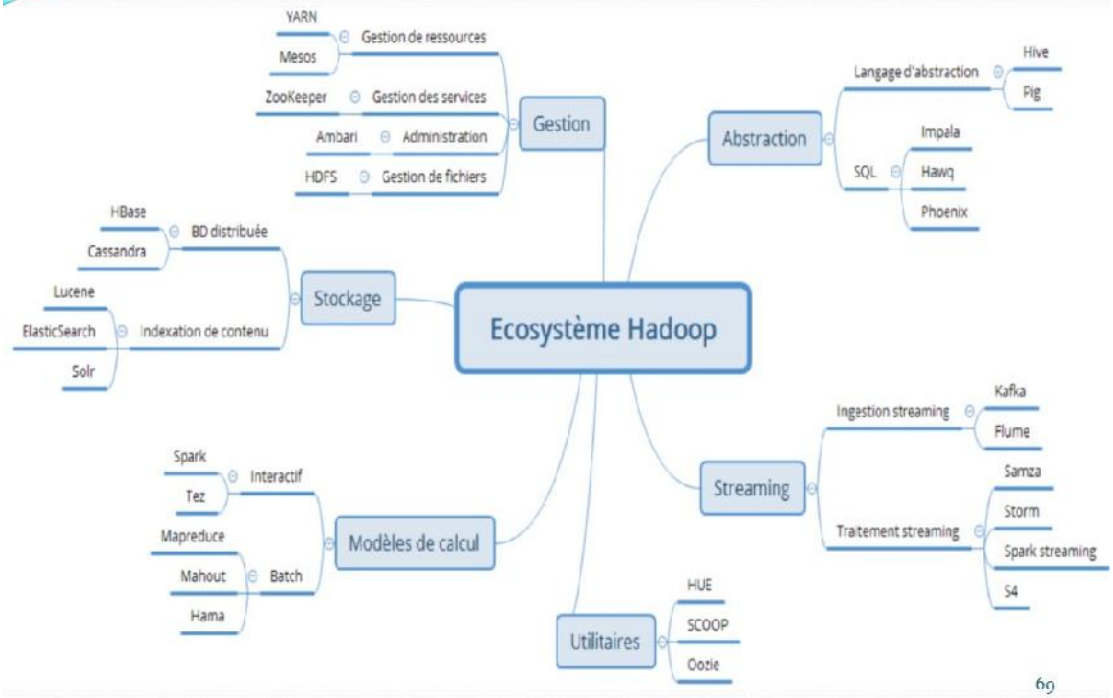
Implémentation JAVA de la fonction reduce():

```
Public class WordCountReducer extends Reducer<Text, IntWritable,
Text, IntWritable>{
Public void reduce(TextKey, Iterable<IntWritable>values, Context
context)
Int sum=0;
For(IntWritablecurrent:values){
Sum+=current.get();
}
Context.write(Key, new IntWritable(sum));}}
```


Ecosystème d'Hadoop



Ecosystème d'Hadoop



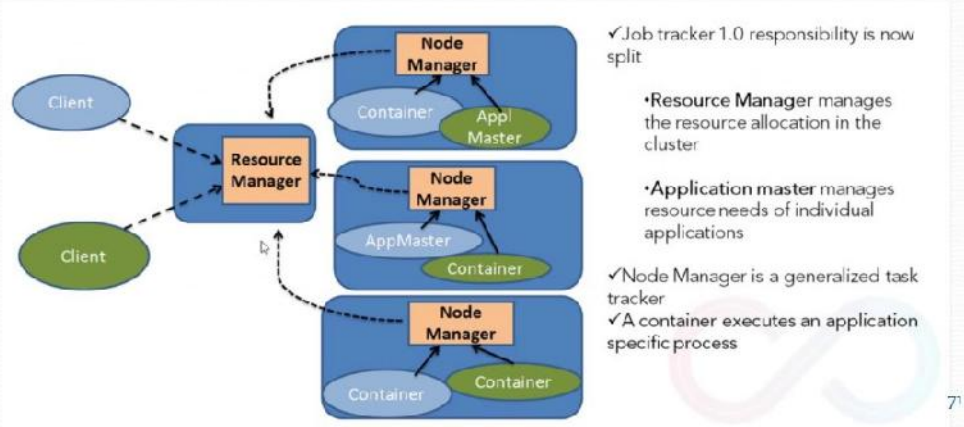
RECAPITULATIF

1. C'est quoi Hadoop ?
2. Quelles sont les caractéristiques de Hadoop ?
3. Donnez qq exemples de distribution Hadoop ?
4. C'est quoi HDFS ?
5. Avec quoi se démarque HDFS sur un système de fichiers classique ?
6. Comment HDFS gère les blocs de données ?
7. Combien de nœuds sont définis sur HDFS ?
8. Quel est le rôle du Secondary NameNode ?
9. Quelles sont les étapes à suivre pour lire un fichier au sein de HDFS ?
10. C'est quoi MapReduce ?
11. Que fait la fonction MAP ?
12. Que fait la fonction Reduce ?
13. Donnez huit outils de l'écosystème Hadoop ?

Ecosystème d'Hadoop

Yarn

Yarn « Yet Another Resource Negotiator » est une technologie qui gère l'utilisation des ressources dans un « Cluster ». Depuis la version 2.0 de Hadoop, Yarn est plus générale que MapReduce et propose une nouvelle architecture de la fonction « JobTracker » qui consiste à séparer ses deux tâches de gestion de ressources et celle de planification et surveillance des travaux. Cela permet d'exécuter des tâches qui ne se basent pas sur MapReduce comme Spark ainsi que des tâches MapReduce sur le même « Cluster ».



Sqoop

- Apache Sqoop est un outil conçu pour transférer efficacement et rapidement des données en mode chargement rapide « Bulk » depuis des bases de données relationnelles vers HDFS, Hive ou HBase. Inversement, Sqoop peut aussi être utilisé pour extraire des données du HDFS et de les exporter vers des bases de données relationnelles ou des entrepôts de données.
- Sqoop fournit un mécanisme de connecteur pour la connexion optimale aux systèmes externes.
- L'API Sqoop fournit un Framework pratique pour la construction de nouveaux connecteurs qui peuvent être déposés dans le répertoire d'installation de Sqoop pour fournir une connectivité à différents systèmes. Sqoop est livré avec les connecteurs les plus populaires comme MySQL, Oracle, PostgreSQL, SQL Server

Sqoop

Exemples d'utilisation de Sqoop :

Importation d'une table depuis Microsoft SQL Server (avec 6 processus Map) :

```
Sqoop import --driver com.microsoft.sqlserver.jdbc.SQLServerDriver
--table MYTABLE --connect
'jdbc:sqlserver://MSSQLSeverHost;database=DBNAME;username=USER;
password=PASSWORD' -m 6
```

Importation d'une table depuis DB2 (avec 1 seul processus Map) :

```
sqoop import --driver com.ibm.db2.jcc.DB2Driver --connect
jdbc:db2://db2host:50000/DBNAME --username USER
--password PASSWORD --table MYTABLE -m 1
```


Zookeeper

- Apache Zookeeper est un service centralisé libre qui a pour rôle de conserver les informations de configuration et de nommage, assurer la synchronisation distribuée des données.
- ZooKeeper est un projet important de l'écosystème de Hadoop avec une architecture qui supporte la haute disponibilité. Un client peut communiquer avec un autre « Zookeeper master » si le premier échoue.
- Zookeeper garantie qu'une écriture de données d'un même client est procédée dans l'ordre envoyé par ce client, et permet aux processus distribués de se coordonner autour d'un espace de nom sous forme d'arborescence partagée ou les données sont stockées dans les nœuds de l'arbre. On appelle ces noeuds des « Znodes ». Chaque Znode est associé à des données spécifiques et est identifié par un chemin et un parent à l'exception de la racine '/' qui n'a aucun parent. La mise à jour des nœuds se fait d'une manière atomique et les clients peuvent être notifiés de cette mise à jour. Zookeeper peut fonctionner sur les mêmes machines qui hébergent d'autres services Hadoop

Oozie

Apache Oozie est une application écrite en Java qui fait partie de l'écosystème de Hadoop et qui est utilisée pour planifier les « Jobs » de Hadoop. Oozie combine plusieurs travaux dans une seule unité logique de travail, il peut être utilisé avec MapReduce ou Yarn et supporte les outils Pig, Hive et Sqoop. Il peut également être utilisé pour planifier des travaux d'un système basé sur des programmes écrit en Java.

Il existe deux types de travaux Oozie :

- Oozie Workflow : les travaux représentent des séquences d'actions à exécuter qui sont regroupées dans ce qu'on appelle « DAGs »;
- Oozie Coordinator : les travaux sont des « Oozie Workflow » récurrents qui sont déclenchés par le temps et par la disponibilité des données.

Pour un travail de MapReduce contrôlé par Oozie, ce dernier planifie et déclenche les actions, mais c'est le MapReduce qui les exécute. Cela permet à Oozie de balancer la charge et de gérer les erreurs

Hive / Pig / HCatalog

Les données sur un système HDFS, peuvent être interrogées de plusieurs façons. Programmer directement en MapReduce pour extraire des données de HDFS est une tâche compliquée qui nécessite beaucoup d'efforts. Dans ce contexte, des applications libres avec un niveau d'abstraction plus élevé que la programmation MapReduce sont disponibles pour répondre à ce besoin, surtout quand il s'agit des tâches avec des jointures.

- Pig : développé par Yahoo, utilise un langage de scripts appelé Pig Latin et transforme le programme en tâches MapReduce .
- Hive : similaire à Pig, mais utilise un langage HiveQL semblable à SQL. Hive était créé principalement pour les analystes de Facebook pour leur permettre d'analyser des données dans HDFS sans le passage à la programmation MapReduce.
- HCatalog : devenu une partie du projet Hive depuis mars 2013 , a pour rôle principal de centraliser les méta-données qui définissent l'endroit où les données sont stockées.

HCatalog expose son service à d'autres applications de l'écosystème de Hadoop pour partager ces méta-données et leur permettre un accès rapide aux données correspondantes sans aucun besoin par l'application de connaitre l'endroit physique ou les données sont stockées sur HDFS.

Exemple (hortonworks.com) :
Une table qui représente la liste des clients est stockée sur HDFS en format délimité et créé avec Apache HCatalog, peut être accessible avec Hive et Pig. Pour extraire de cette table les informations du client « IBM » en utilisant Hive et Pig on procède comme suit :

Hive

```
SELECT * FROM clients WHERE name = 'IBM';
```

Pig

```
a – LOAD 'default.clients' USING
org.apache.hcatalog.pig.HCatLoader();
b = filter a by name == 'IBM';
c = group b all;
dump c;
```


Spark SQL

Spark SQL (anciennement Shark) est un projet Apache basé sur Apache Spark qui est un engin qui exécute à travers Yarn des programmes en parallèle 100x plus rapide que Hive avec l'utilisation maximale de mémoire contrairement à Hive qui utilise un maximum d'E/S sur les disques .
Les caractéristiques de Spark SQL sont :

- Licence Apache et 100 % libre;
- Codage facile d'application en utilisant l'API Spark avec des langages haut niveau comme Java, Scala, Python et R;
- Chargement et interrogation des données depuis plusieurs sources : HDFS, Cassandra, Hive, HBase, Amazon S3;
- Connexion ODBC, JDBC

HBase

HBase est un projet Apache libre conçu pour gérer en temps réel des accès lecture/écriture aléatoires basé sur HDFS. Le but principal de ce projet est de pouvoir stocker de grandes tables à l'échelle de millions de colonnes groupées dans ce qu'on appelle « Familiy group » et billions de lignes de données. HBase est basé sur le concept clé-valeur.

```
put 'hbase_table_name', 'unique_row_id', 'family_group :field1', 'value1'
put 'hbase_table_name', 'unique_row_id', 'family_group :field2', 'value2'
get 'hbase_table_name', 'unique_row_id'
```

HBase a le même concept que HDFS, mais utilise HRegion au lieu des DataNode et HRegionServer au lieu du NameNode. Il est tout à fait possible d'utiliser les mêmes serveurs HDFS pour ajouter la couche HBase.

RECAPUTILATIF

1. Que signifie l'abréviation YARN ? Que gère la technologie YARN ?
2. Quel est le fonctionnement de Sqoop ? que fournie l'API YARN ?
3. Que permet le service Zookeeper ? qu'est ce que un Znode ?
4. Que permet Oozie ? quels sont les deux types de travaux Oozie ?
5. Les données HDFS peuvent être interroger de plusieurs façons; citez trois exemples de ces façons ?
6. Qu'est ce que Spark SQL?
7. Quelles sont les caractéristiques de Spark ?
8. Qu'est ce que HBASE ?
9. Quelle la correspondance entre les Nœuds Hbase et HDFS ?

NoSQL

Nouvelle contrainte des bases de données

La force du modèle relationnel classique réside avant tout dans la flexibilité de l'usage de données qu'il permet, et dans les garanties offertes par le caractère ACID des transactions.
Dans le contexte des applications web à grande échelle, ce modèle universel est désormais remis en question. Les exigences de performance et de disponibilité demandent de l'abandonner au profit des systèmes dont les modèles de données sont spécifiquement adaptés aux applications qui les exploitent tout en étant simultanément moins flexible.
Les applications métiers devront par ailleurs prendre en charge tout ou partie de la gestion de la cohérence des données. Les bases de données NoSQL coexisteront encore pendant de nombreuses années avec les SGBDR. Le recours à l'option NoSQL se fera avant tout sur la base d'un examen lucide et chiffré des compromis acceptables entre consistance des données et disponibilité des systèmes. Il s'agit donc au moins autant d'un choix commercial que d'un choix technologique.

NB : Le caractère ACID des SGBDR

Les exigences qui caractérisent une transaction SGBDR sont définies par le célèbre acronyme **ACID. Rappelons en brièvement la signification :**

• Atomicité : désigne le caractère indivisible d'une transaction évoqué plus haut.

• Cohérence : désigne l'objectif même des transactions, à savoir laisser les données dans un état cohérent.

• Isolation : deux transactions simultanées A et B n'interfèrent jamais. La transaction B ne peut ni voir ni modifier les données sur lesquelles A opère tant
que A n'a pas été validé.

• Durabilité : une fois une transaction validée, les données doivent être permanentes.

Qu'est-ce que le NoSQL ? Not Only SQL

Catégorie de SGBD s'affranchissant du modèle relationnel des SGBDR. Mouvance apparue par le biais des "grands du Web", popularisée en 2010

Il existe quatre types de SGBD NoSQL :

1. Orienté document (MongoDB, ...)
2. 2. Clé / valeur (Redis, ...)
3. 3. Orienté colonne (Cassandra, ...)
4. 4. Orienté graphe (Neo4J, ...)

Pourquoi NoSQL ?

- Licence des SGBDR très chère (Oracle, ...).
- Le SQL a un schéma fermé.
- Performances faibles, sur de gros volumes de données, comparées au NoSQL.

Le NoSQL vise :

1. Gestion d'énormes quantités de données
2. Structuration faible du modèle
3. Montée en charge

Caractéristiques NoSQL

- Principalement utilisé sur des clusters de serveurs
- Permet un modèle qui peut s'étendre plus facilement (scalability)
- Assouplit les contraintes habituellement présentes sur les bases de données relationnelles
- Permet de gérer rapidement des tonnes de données

NoSQL et réseaux sociaux

Les entreprises du WEB 2.0 avaient besoin de solutions technologiques plus adaptées à leurs besoins

Développement des systèmes NoSQL propriétaires

- Facebook : Cassandra, Hbase
- Google : BigTable
- LinkedIn : Projet Voldemort
- Amazon : DynamoDB, SimpleDB
- Twitter : Cassandra

Théorème CAP

Énoncé par Eric Brewer en 1999

indique qu'il est impossible, pour un système distribué, de garantir en même temps les trois contraintes suivantes :

- Cohérence : Tous les nœuds du système voient les mêmes données au même moment
- Disponibilité : Toutes les requêtes reçoivent une réponse
- Tolérance au partitionnement : Aucune panne ne doit empêcher le système de répondre correctement (sauf une coupure complète du réseau)
- Il est possible de garantir 2, mais pas 3 contraintes

- Les bases de données NoSQL tendent à privilégier la disponibilité et la tolérance au partitionnement
- Il peut être préférable que deux personnes faisant la même recherche sur Google obtiennent des résultats différents que pas de réponses du tout
- Facebook, Twitter, etc. utilisent le même principe

NoSQL - Types

Il existe différents types de bases de données NoSQL

- Colonne
- Clé/Valeur
- Document
- Graphe
- Etc.

Les types Document et Graphe sont basés sur le type Clé/Valeur

NoSQL – Comparaison

Modèle	Performance	Évolutivité	Flexibilité	Complexité	Fonctionnalité
Clé/Valeur	Élevée	Élevée	Élevée	Aucune	Variable (Aucune)
Colonne	Élevée	Élevée	Modérée	Faible	Minimale
Document	Élevée	Variable	Élevée	Faible	Variable (Faible)
Graphe	Variable	Variable	Élevée	Grande	Théorie des graphes

Présentation de MongoDB

MongoDB est une base de données NoSQL orientée document qui stocke les données sous forme de documents (paires clé/valeur). Il s'agit d'une base de données Open Source offrant des performances et une évolutivité élevées, ainsi que la modélisation et la gestion de données dans une application d'entreprise.

MongoDB fournit également la fonctionnalité d'Auto-Scaling. En plus, MongoDB est une base de données multiplateforme et peut être installé sur différentes plates-formes telles que Windows, Linux, etc.

Récapitulatif

1. Quelle est la signification de l'acronyme ACID en base de données
2. Fournir une définition du NoSQL
3. Que vise le traitement NoSQL
4. Quelles sont les principales caractéristique du NoSQL
5. Donnez quelques exemples de solutions Web NoSQL
6. Citez le théorème CAP
7. Donnez qq types de base données NoSQL

Présentation de MongoDB

MongoDB est un SGBD :

- orienté documents
- libre
- scalable : réplication, auto-sharding
- flexible : pas de schéma de données, full-text index
- écrit en C++.
- Il fait partie de la mouvance NoSQL et vise a fournir des fonctionnalités avancées. Utilisée par Foursquare, bit.ly, Doodle, Disqus

Présentation de MongoDB

MongoDB est orienté document. Qu'est ce qu'un document ? Un document est la représentation d'une donnée en BSON. BSON = Binary JSON. Extension du JSON (support officiel du type Date, ...).

Exemple :

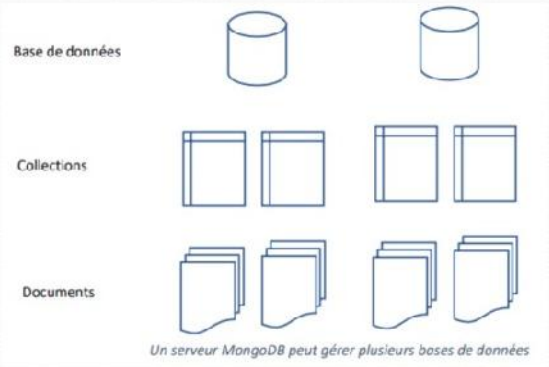
```
{
        "name" : "MongoDB",
        "type" : "database",
        "count" : 1,
        "info" : {
                x : 203,
                y : 102
                }
}
```

Présentation de MongoDB

Le modèle de données

Un serveur MongoDB peut gérer plusieurs bases de données. Chacune de de ces bd peut contenir 0 ou plusieurs collections.

Une collection regroupe des documents. Si on voulait faire une analogie avec le modèle relationnel, on dirait que le document est l'équivalent d'un enregistrement et que la collection est l'équivalent d'une table.



Un serveur MongoDB peut gérer plusieurs boses de données

Les documents

- Du point de vu de l'usager, un document est défini avec le langage JSON;
- Un document JSON est un ensemble d'associations { clé : valeur}. Le nom de clé _id est réservé au système;

```
{   "num_ad": "20183383518",
    "nom": "Patoche",
    "prénom": "Alain"   }
```


DBObject

Le document :

```
{ "name" : "MongoDB", "type" : "database", "count" : 1,
"info" : { x : 203, y : 102 } }
```

Sera représenté ainsi en Java :

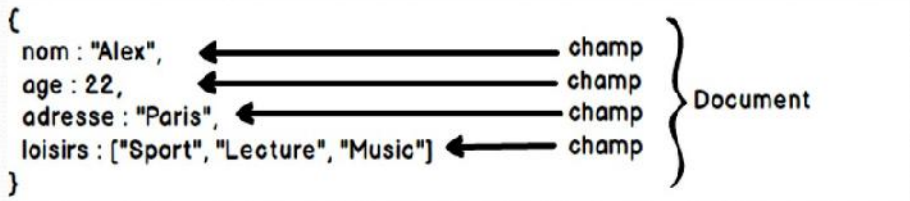
```
BasicDBObject doc = new BasicDBObject("name",
"MongoDB").          append("type", "database").
        append("count", 1).
        append("info", new BasicDBObject("x", 203).
        append("y", 102));
```

- Organisation

- Un serveur MongoDB est composé de bases de données.
- Une base de données contient des collections(les tables en SQL).
- Les collections possèdent les documents.
- Chaque document possède un identifiant unique généré par MongoDB, le champ _id.

Base de données, collection et document

MongoDB fonctionne sur le concept de « base de données, collection et document ».
Base de données est un conteneur physique pour les collections.
Collection est un groupe de documents MongoDB.
Un document est un ensemble de paires clé-valeur.



Les collections

• Une collection est un regroupement de documents;
• Une collection n'a pas de structure prédéfinie comme une table dans une base de données
relationnelle;
• Des documents de forme différente peuvent être ajouté dans une même collection;
Ex. Les deux documents suivants pourraient être contenus dans la même collection;

```
{
 "nom": "Informatique",
 "code": "420"
}
{
 "num_ad": "20183383518",
 "nom": "Patoche",
 "prénom": "Alain"
}
```

• Typiquement on veut conserver des documents d'un même type dans une collection;
Les base d

Le tableau suivant montre la relation entre la terminologie de SGBDR et MongoDB.

SGBDR	MongoDB
Base de données	Base de données
Table	Collection
Tuple	Document
Colonne	Champ
Jointure	Document incorporé
Clé primaire	Clé primaire (Clé par défaut _id fournie par mongodb elle-même)

Les base de données

- Les collections sont regroupées dans base de données;
- Un serveur MongoDB peut gérer plusieurs bases de données;
- Les bases de données suivantes sont utilisées par le gestionnaire MongoDB;
  - o **admin**
    Utilisé pour la gestion des usagers et des autorisations;
  - o **local**
    Utilisé pour conserver les collections que l'on ne veut pas répliquer à d'autre serveur MongoDB;
  - o **config**
    Utilisé pour la gestion du mode « shared » (partitionnement des donnés entre plusieurs serveur MongoDB);

Installer MongoDB

Installation :

• Téléchargez la version la plus récente du logiciel (choisissez la version en format .zip);

https://www.mongodb.com/download-center/community

• Décompressez le fichier sur le bureau;

• Démarrez un interpréteur de commandes (cmd.exe) et déplacez-vous dans le sous-répertoire 'bin' du logiciel MongoDB que vous avez décompressé;

• Vous devez créer le répertoire où seront conservés les fichiers des bases de données (par défaut MongoDB utilise le répertoire 'C:\data\db' pour conserver ses fichiers);

o Créez le répertoire 'mongo_data' sur le bureau;

Démarrage du serveur MongoDB

• Dans l'interpréteur de commande que vous avez démarré, taper la commande suivante en spécifiant le chemin vers le répertoire que vous avez créé.

mongod.exe -dbpath C:\Users\beaul\OneDrive\Desktop\mongo_data

• Pour arrêter MongoDB simplement, tapez Ctrl-C dans la fenêtre de commandes;

Langage MongoDB

Mise-à-jour

- db.collection.insert()
- db.collection.update()
- db.collection.save()
- db.collection.findAndModify()
- db.collection.remove()

Interrogation

- db.collection.find()
- db.collection.findOne()

Mongo DB : insert

```
db.collection.insert( [, ]*)
> collection = db.contactinfo
> doc = { "nom" : "toto",
        "info" : {
                "twitter" : "@toto5646",
                "email" : "toto@upmc.fr" },
        "amis" : 87687,
        "photo" : BinData(...)
        "date_login" : new Date()}
> db.contactinfo.insert(doc)
```

MongoDB : update

db.collection.update(query, update, ,)

- query : requête
- update : modifications
- upsert (booléen) :
  - false (défaut) : update
  - true : update or insert if not exists
- multi (booléen) :
  - false (défaut) : maj de la 1ere occurrence trouvée
  - true : maj toutes les occurrence

MongoDB: update

- db.products.update({ item: "book", qty: { $gt: 5 } }, { $set: { x: 6 }, $inc: { y: 5} })

<>

- db.products.update({ item: "book", qty: { $gt: 5 } }, { x: 6, y: 15 })

- db.products.update({ item: "book", qty: { $gt: 5 } }, { $set: { x: 6, y: 15 } }, { multi: true })

=

- db.products.update({ item: "book", qty: { $gt: 5 } }, { $set: { x: 6, y: 15 } }, false, true)

- db.products.update({ item: "magazine", qty: { $gt: 5 } }, { $set: { x: 25, y: 50 } }, { upsert: true })

Mongo DB : requêtes

db.collection.find(<query>, <projection>)

db.collection.findOne(<query>, <projection>)

```
> db.inventory.find( {} )
> db.contactinfo.findOne({"nom" : "toto"});
> db.contactinfo.findOne({"contact.twitter" : "@toto5646"});
> db.contactinfo.find({"date_login" : {
        "$gt" : ISODate("2015-09-17T23:25:56.314Z"),
        "$lt" : ISODate("2014-09-17")}}).sort({nom :
1}).limit(10).skip(100)
```


Les outils fournis avec MongoDB

Vous trouverez parmi les fichiers du dossier bin :

- mongod : Le serveur MongoDB
- mongo : Le shell MongoDB (avec le langage Javascript)
- mongodump : Pour sauvegarder (binaire) vos bases de données
- mongorestore : Pour restaurer (binaire) vos bases de données
- mongoimport : Pour importer un fichier JSON ou CSV
- mongoexport : Pour exporter un fichier JSON ou CSV
- bsondump : Conversion du BSON en JSON

Récapitulatif

1. Donnez qq caractéristiques de MongoDB
2. Qu'est ce que un document en MongoDB
3. Fournir un tableau de corespondance entre un SGBDR et MongoDB
4. Listez qq outils MangoDB
5. Qu'est ce que un document et une collection on MongoDB
6. Donnez les bases de données utilisées par le gestionnaire MongoDB

MongoDB : requêtes

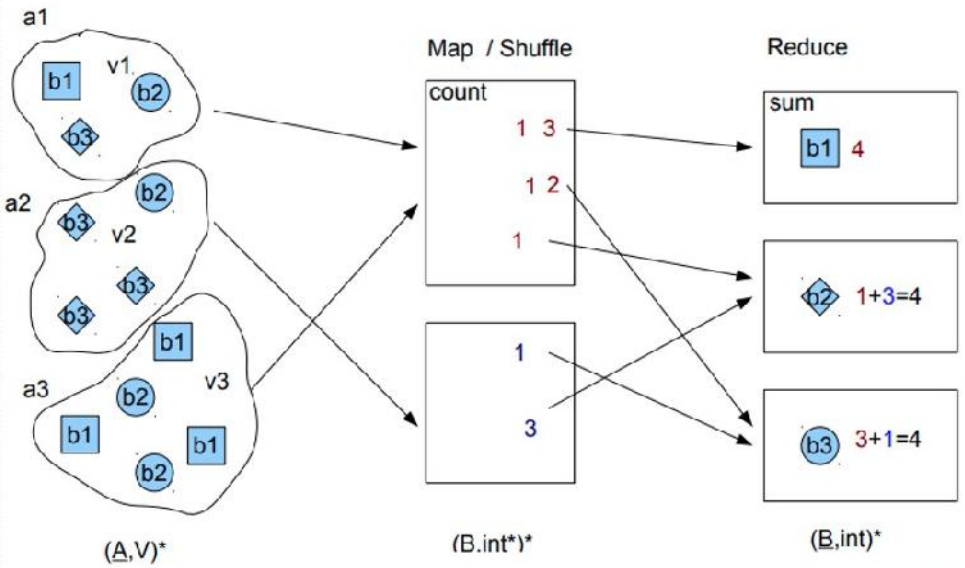
```
> db.employee.insert(
    {
        " name ": " John Smith ",
        " address ": {
        " street ": " Lily Road ",
        " number ": 32,
        " city ": " Owatonna ",
        "zip ": 55060
    },
    " hobbies ": [ " yodeling ", "ice skating " ]
})
>db. employee . FindOne (
    {" name ": " John Smith "})

> db. employee . find (
    {" address . city ": " Owatonna "},
    {" name ": 1})

> db. employee . find (
    {" hobbies ": {" $ne ": " yodeling "}})
```

```
> db. employee . find (
    {" address . city ": " Owatonna "},
    {" address.city ": 0, age : 1, name : 1})
```


MapReduce (exemple)



MongoDB : MapReduce

```
db.runCommand(
 { mapreduce : <collection>,
   map : <mapfunction>,
   reduce : <reducefunction>
   [, query : <query filter object>]
   [, sort : <sorts the input objects using this key. Useful for optimization, like
            sorting by the emit key for fewer reduces>]
   [, limit : <number of objects to return from collection>]
   [, out : <see output options below>]
   [, keeptemp: <true|false>]
   [, finalize : <finalizefunction>]
   [, scope : <object where fields go into javascript global scope >]
   [, jsMode : true]
   [, verbose : true]
 }
);
```


MapReduce : exemple

Collection de commentaires :

```
{ username: "jones", likes: 20,
   text: "J'aime cette photo!" }
```

Fonction Map :

```
function() {
   emit( this.username,
         {count: 1, likes: this.likes} );
}
```

Fonction Reduce :

```
function(key, values) {
   var result = {count: 0, likes: 0};
   values.forEach(function(value) {
     result.count += value.count;
     result.likes += value.likes;
   });
   return result;
}
```

- Map / shuffle : [(C,V)] → [(C',[V'])]
- Reduce : [(C',[V'])] → [(C',V')]

} MapReduce incrémental

EXPOSES

1- SPARK Zobir elkourdachi
2-CASSANDRA -
3- HBASE Gaoussou COULIBALY
4-HIVE -
5-YARN
6-MongoDB Elom PEDASSOU
7-ZooKeeper
8-Oozie
9-Impala