SYSTÈMES D'INFORMATION DÉCISIONNELS

BIG DATA

Département d'informatique

Module: Systèmes d'information décisionnels

Chargé du module: Mokeddem, S

Email: sidahmed.mokadem@univ-mosta.dz

Année universitaire: 2018/2019

POURQUOI CE COURS

- Selon LinkedIn, les compétences les plus recherchées en 2017 dans le monde sont :
  - Cloud and Distributed Computing (Hadoop, Big Data)
  - Statistical Analysis and Data Mining (R, Data Analysis)
  - Storage Systems and Management (SQL)

AVANT BIG DATA

- Calcul répartie
  - Effort concentré sur les problèmes « calcul intensive »
  - Le réseau n’étant pas si performant, 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
- Data mining
  - Méthodes statistiques pour l’extraction des connaissances
  - D’abord un modèle, qui sera ensuite validé
  - Echantillonnage des données pour tenir dans la mémoire

PRÉFIXES MULTIPLICATIFS

signe	préfixe	facteur	exemple représentatif
k	kilo	10^3	
M	méga	10^6	vitesse de transfert par seconde
G	giga	10^9	DVD, clé USB
T	téra	10^{12}	disque dur
P	**péta**	10^{15}	
E	**exa**	10^{18}	FaceBook, Amazon
Z	**zetta**	10^{21}	internet tout entier depuis 2010

MÉGADONNÉES

- Les mégadonnées ou Big Data sont des collections d'informations qui auraient été considérées comme gigantesques, impossible à stocker et à traiter, il y a une dizaine d'années.
  - Internet :
    - Google en 2015 : 10 Eo (10 milliards de Go),
    - Facebook en 2014 : 300 Po de données (300 millions de Go), 4 Po de nouvelles données par jour,
    - Amazon : 1 Eo.
  - BigScience : télescopes (1 Po/jour), CERN (500 To/jour, 140 Po de stockage), génome, environnement. . .
    - Les informations sont très difficiles à trouver. La raison est que tout est enregistré sans discernement, dans l'idée que ça pourra être exploité. Certains prêchent pour que les données collectées soient pertinentes (smart data) plutôt que volumineuses.

MÉGADONNÉES

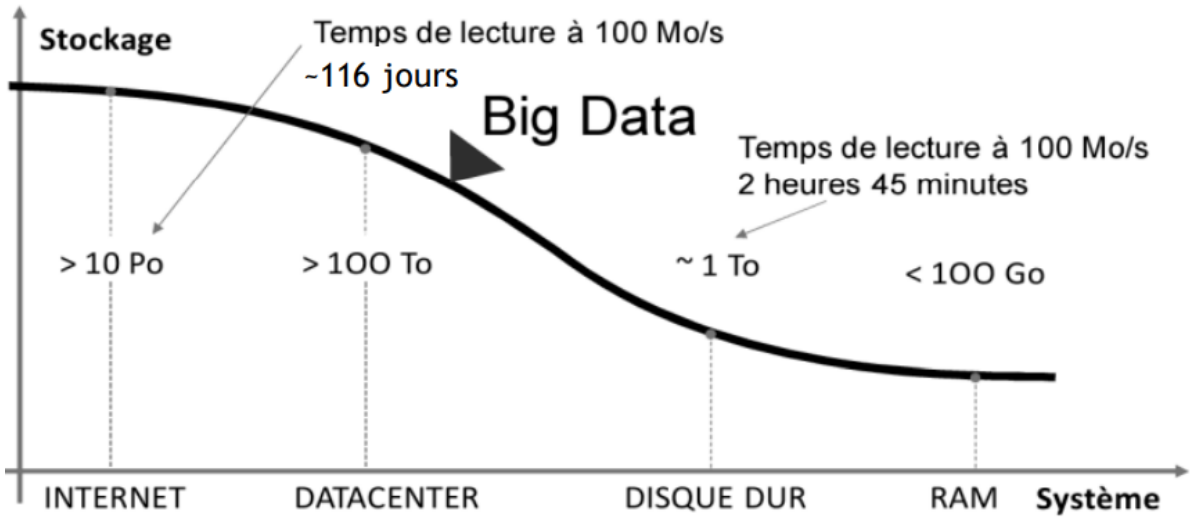
- Les mégadonnées ou Big Data sont des collections d'informations qui auraient été considérées comme gigantesques, impossible à stocker et à traiter, il y a une dizaine d'années.
  - Internet :
    - Google en 2015 : 10 Eo (10 milliards de Go),
    - Facebook en 2014 : 300 Po de données (300 millions de Go), 4 Po de nouvelles données par jour,
    - Amazon : 1 Eo.
  - BigScience : télescopes (1 Po/jour), CERN (500 To/jour, 140 Po de stockage), génome, environnement. . .
    - Les informations sont très difficiles à trouver. La raison est que tout est enregistré sans discernement, dans l'idée que ça pourra être exploité. Certains prêchent pour que les données collectées soient pertinentes (smart data) plutôt que volumineuses.

UN AUTRE REGARD SUR L'INFORMATIQUE

- L'accumulation et la capacité de traiter les données a crée une révolution dans notre vie courante
  - Services de recommandation
  - Réseaux sociaux
- Les entreprises ont rapidement reconnu l'intérêt
  - Des recherches plus pertinentes
  - Meilleur ciblage des publicités
  - Analyse et prédiction des tendances du marché
  - Une relation plus personnelle avec les clients

LA FRONTIÈRE DU BIGDATA

- Règle générale, on considère du BigData quand le traitement devient trop long pour une seule machine



LES TROIS « V » DU BIGDATA

- Volume (Volume)
- Variété (Variety)
- Vitesse (Velocity)

VOLUME

- Le prix de stockage a beaucoup diminue
- Des solutions de stockage fiables sont nombreuses
  - SAN (Storage Area Networks)
  - Stockage sur le cloud (Amazon S3)
- 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)

VARIÉTÉ

- Les bases de données ou entrepôts de données imposent souvent un format prédéfini
- La plupart des données existantes sont non-structurées
  - Données sous plusieurs formats et types
  - On veut tout stocker
- Certaines données peuvent paraître obsolètes mais peuvent être utiles pour certaines décisions
  - Ex : Transport de marchandises – quel camion choisir ?
  - Données GPS, plan de livraison du camion, circulation, chargement du camion, niveau de combustible, horaires de travail du conducteur

VITESSE

- Rapidité d’arrivée des données
  - Vitesse de traitement
  - Les données doivent être stockées à l’arrivée, parfois même des teraoctets par jour
- Exemple
  - Il ne suffit pas de savoir quel article un client a acheté ou réservé
  - Si si on sait que vous avez passé plus de 5mn à consulter un article dans une boutique d’achat en ligne, il est possible de vous envoyer un email dès que cet article est soldé

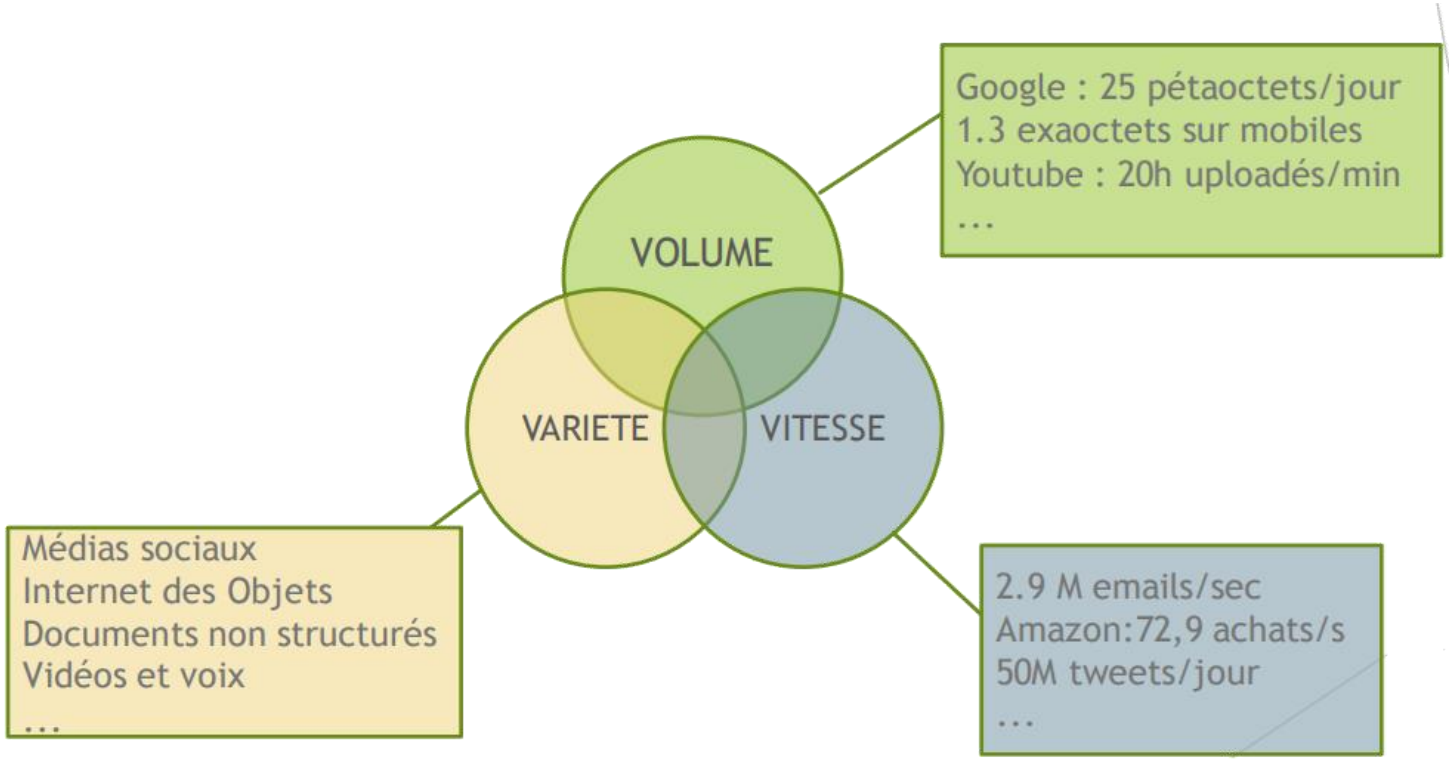
VITESSE

- Rapidité d’arrivée des données
  - Vitesse de traitement
  - Les données doivent être stockées à l’arrivée, parfois même des teraoctets par jour
- Exemple
  - Il ne suffit pas de savoir quel article un client a acheté ou réservé
  - Si si on sait que vous avez passé plus de 5mn à consulter un article dans une boutique d’achat en ligne, il est possible de vous envoyer un email dès que cet article est soldé

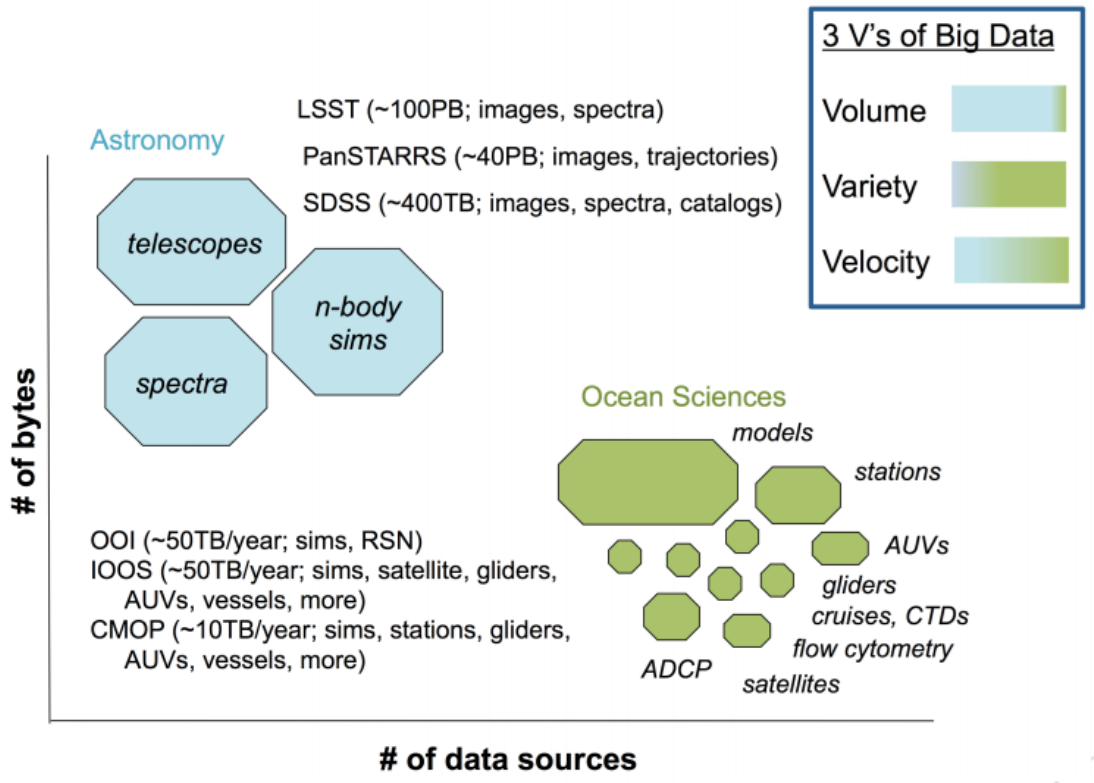
VITESSE

- Rapidité d’arrivée des données
  - Vitesse de traitement
  - Les données doivent être stockées à l’arrivée, parfois même des teraoctets par jour
- Exemple
  - Il ne suffit pas de savoir quel article un client a acheté ou réservé
  - Si si on sait que vous avez passé plus de 5mn à consulter un article dans une boutique d’achat en ligne, il est possible de vous envoyer un email dès que cet article est soldé

LE BIGDATA À L'INTERSECTION DES 3V



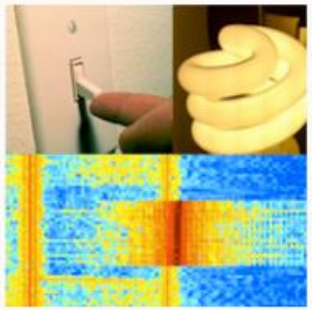
SOURCES DE DONNÉES



DES CAPTEURS PARTOUT ?

- Trackeur d'activité
- Positionnement GPS avec le Téléphone (+ pression, température, luminosité)
- Domotique

CONNECT'



ElectriSense

Electrical Device Energy Usage with a Single Sensor

ElectriSense is a single plug-in sensor that provides whole home device level usage data. That is, using a single sensor plugged in anywhere in the home, ElectriSense can infer which electrical appliances are on and which off. This data could be used for numerous applications, for example, for providing home owners with itemized electrical bill that not only shows the total energy consumption but breaks the total on a per appliance basis (TV consumed 20 KWh, Lighting consumes 18 KWh and so on).

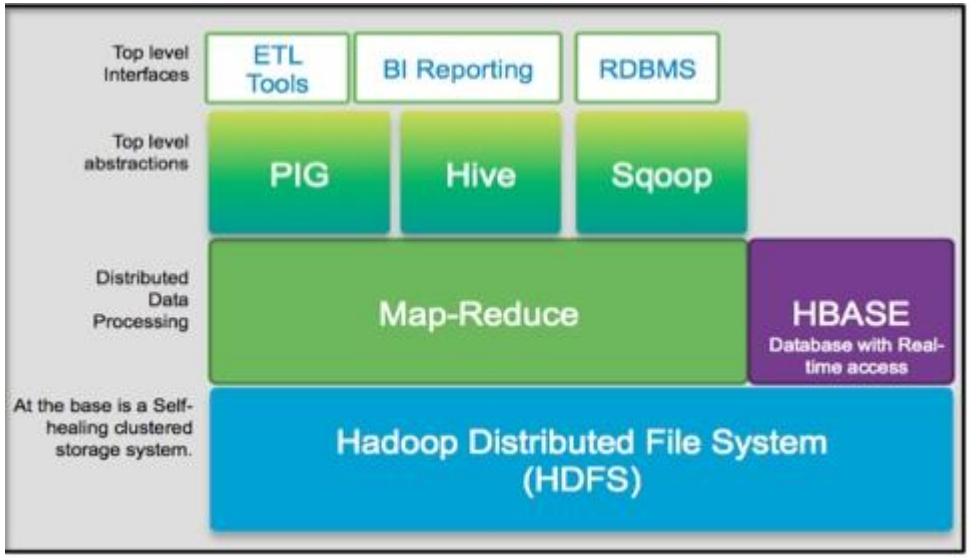
Lead Researchers: Sidhant Gupta, Shwetak Patel

LES DEUX « V » SUPPLÉMENTAIRES

- Véracité (Veracity)
  - La qualité et la précision des données sont aussi importantes
  - Comment se trouver dans un déluge de hashtags ?
  - Comment gérer les données partielles ou incomplètes ?
- Valeur (Value)
  - La valeur ajoutée des données ou des informations extraites
  - Sans une réelle valeur, ce n’est qu’un gaspillage de ressources

UN ÉCOSYSTÈME PARTICULIER

- Hadoop est un système de gestion de données et de traitements distribués.
  - Il contient de beaucoup de composants, dont :
    - HDFS un système de fichier qui répartit les données sur de nombreuses machines, YARN un mécanisme d'ordonnancement de programmes de type MapReduce.



HADOOP FILE SYSTEM (HDFS)

- HDFS est un système de fichiers distribué. C’est à dire :
  - un dispositif de stockage et d’accès à des fichiers
  - ces fichiers sont stockés sur un grand nombre de machines de manière à rendre invisible la position exacte d’un fichier.
  - L’accès est transparent, quelle que soient les machines qui contiennent les fichiers.
- HDFS permet de voir tous les dossiers et fichiers de ces milliers de machines comme un seul arbre, contenant des Po de données, comme s’ils étaient sur le disque dur local.

HADOOP FILE SYSTEM (HDFS)

- Vu de l'utilisateur, HDFS ressemble à un système de fichiers Unix : il y a une racine, des répertoires et des fichiers. Les fichiers ont un propriétaire, un groupe et des droits d'accès comme avec ext4.
- Sous la racine /, il y a :
  - des répertoires pour les services Hadoop : /hbase, /tmp, /var
  - un répertoire pour les fichiers personnels des utilisateurs : /user, Dans ce répertoire, il y a aussi trois dossiers système : /user/hive, /user/history et /user/spark
  - un répertoire pour déposer des fichiers à partager avec tous les utilisateurs : /share

ÉCHANGES ENTRE HDFS ET LE MONDE

- Pour placer un fichier dans HDFS, deux commandes équivalentes :
  - hdfs dfs -copyFromLocal fichiersrc fichierdst
  - hdfs dfs -put fichiersrc [fichierdst]
- Pour extraire un fichier de HDFS, deux commandes possibles :
  - hdfs dfs -copyToLocal fichiersrc dst
  - hdfs dfs -get fichiersrc [fichierdst]

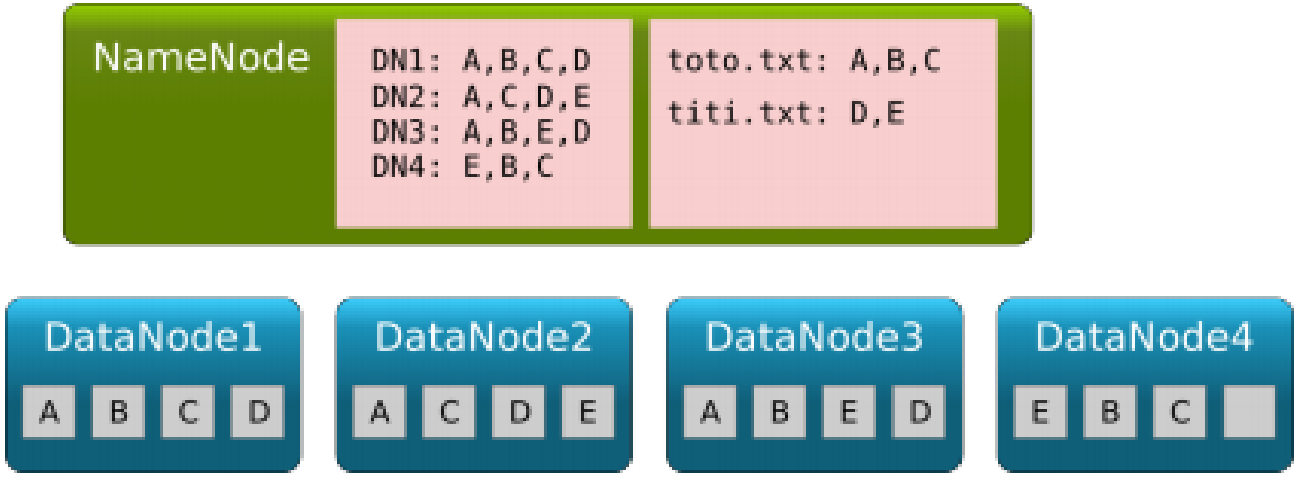
```
hdfs dfs -mkdir -p livres
wget http://www.textfiles.com/etext/FICTION/dracula
hdfs dfs -put dracula livres
hdfs dfs -ls livres
hdfs dfs -get livres/center_earth
```

ÉCHANGES ENTRE HDFS ET LE MONDE

- Pour placer un fichier dans HDFS, deux commandes équivalentes :
  - hdfs dfs -copyFromLocal fichiersrc fichierdst
  - hdfs dfs -put fichiersrc [fichierdst]
- Pour extraire un fichier de HDFS, deux commandes possibles :
  - hdfs dfs -copyToLocal fichiersrc dst
  - hdfs dfs -get fichiersrc [fichierdst]

```
hdfs dfs -mkdir -p livres
wget http://www.textfiles.com/etext/FICTION/dracula
hdfs dfs -put dracula livres
hdfs dfs -ls livres
hdfs dfs -get livres/center_earth
```

UN SCHÉMA DES NODES HDFS



ALGORITHMES « MAP-REDUCE »

- Le but est de recueillir une information synthétique à partir d'un jeu de données.
- Exemples sur une liste d'articles définis par leur prix :
  - calculer le montant total des ventes d'un article,
  - calculer l'article le plus cher,
  - calculer le prix moyen des articles. Pour chacun de ces exemples, le problème peut s'écrire sous la forme de la composition de deux fonctions :
    - map : extraction/calcul d'une information sur chaque n-uplet,
    - reduce : regroupement de ces informations.

ALGORITHMES « MAP-REDUCE »: EXEMPLE

Id	Marque	Modèle	Prix
1	Renault	Clio	4200
2	Fiat	500	8840
3	Peugeot	206	4300
4	Peugeot	306	6140

- pour chaque n-uplet, faire :
  - valeur = FonctionM(n-uplet courant)
- retourner FonctionR(valeurs rencontrées)
- FonctionM est une fonction de correspondance : elle calcule une valeur qui nous intéresse à partir d'un n-uplet,
- FonctionR est une fonction de regroupement (agrégation) : maximum, somme, nombre, moyenne, distincts. . .
  - Par exemple, FonctionM extrait le prix d'une voiture, FonctionR calcule le max d'un ensemble de

ALGORITHMES « MAP-REDUCE »: EXEMPLE

```
data = [
    {'id':1, 'marque':'Renault', 'modele':'Clio', 'prix':4200},
    {'id':2, 'marque':'Fiat',    'modele':'500',  'prix':8840},
    {'id':3, 'marque':'Peugeot', 'modele':'206',  'prix':4300},
    {'id':4, 'marque':'Peugeot', 'modele':'306',  'prix':6140} ]

## retourne le prix de la voiture passée en paramètre
def getPrix(voiture): return voiture['prix']

## affiche la liste des prix des voitures
print map(getPrix, data)

## affiche le plus grand prix
print reduce(max, map(getPrix, data) )
```

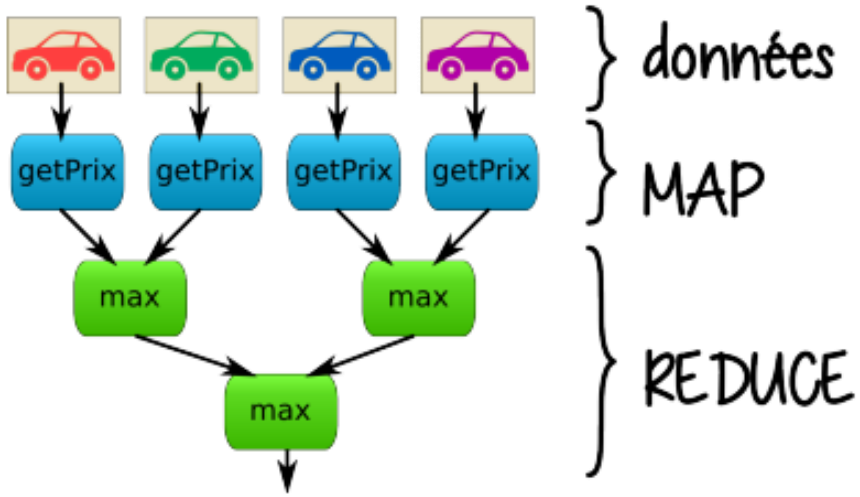
ALGORITHMES « MAP-REDUCE »: EXEMPLE

```
data = [
    {'id':1, 'marque':'Renault', 'modele':'Clio', 'prix':4200},
    {'id':2, 'marque':'Fiat',    'modele':'500',  'prix':8840},
    {'id':3, 'marque':'Peugeot', 'modele':'206',  'prix':4300},
    {'id':4, 'marque':'Peugeot', 'modele':'306',  'prix':6140} ]

## retourne le prix de la voiture passée en paramètre
def getPrix(voiture): return voiture['prix']

## affiche la liste des prix des voitures
print map(getPrix, data)

## affiche le plus grand prix
print reduce(max, map(getPrix, data) )
```

ALGORITHMES « MAP-REDUCE »: EXEMPLE

- L’écriture map(fonction, liste) applique la fonction à chaque élément de la liste. Elle effectue la boucle « pour » de l’algorithme précédent et retourne la liste des prix des voitures. Ce résultat contient autant de valeurs que dans la liste d’entrée.
- La fonction reduce(fonction, liste) agglomère les valeurs de la liste par la fonction et retourne le résultat final1 . Ces deux fonctions constituent un couple « map-reduce » et le but de ce cours est d’apprendre à les comprendre et les programmer.
- Le point clé est la possibilité de paralléliser ces fonctions afin de calculer beaucoup plus vite sur une machine ayant plusieurs cœurs ou sur un ensemble de machines reliées entre elles.

ALGORITHMES « MAP-REDUCE »: EXEMPLE



YARN ET MAPREDUCE

- YARN (Yet Another Resource Negociator) est un mécanisme permettant de gérer des travaux (jobs) sur un cluster de machines.
- YARN permet aux utilisateurs de lancer des jobs Map-Reduce sur des données présentes dans HDFS, et de suivre (monitor) leur avancement, récupérer les messages (logs) affichés par les programmes.
- Éventuellement YARN peut déplacer un processus d'une machine à l'autre en cas de défaillance ou d'avancement jugé trop lent.
- En fait, YARN est transparent pour l'utilisateur. On lance l'exécution d'un programme MapReduce et YARN fait en sorte qu'il soit exécuté le plus rapidement possible.

YARN ET MAPREDUCE

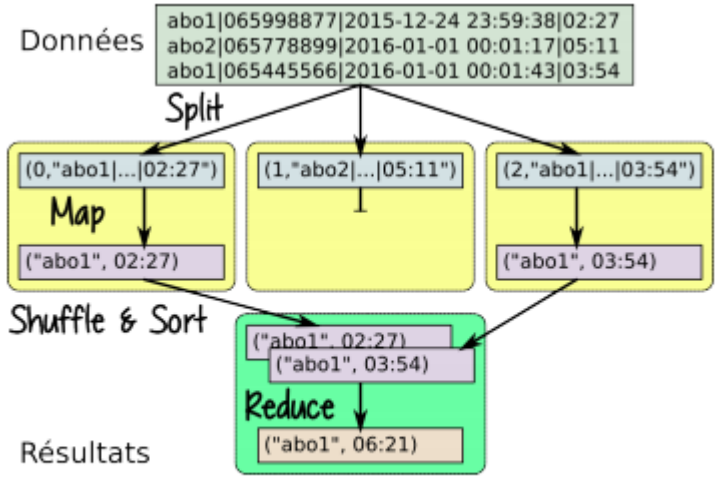


Figure 9: Étapes MapReduce

YARN ET MAPREDUCE

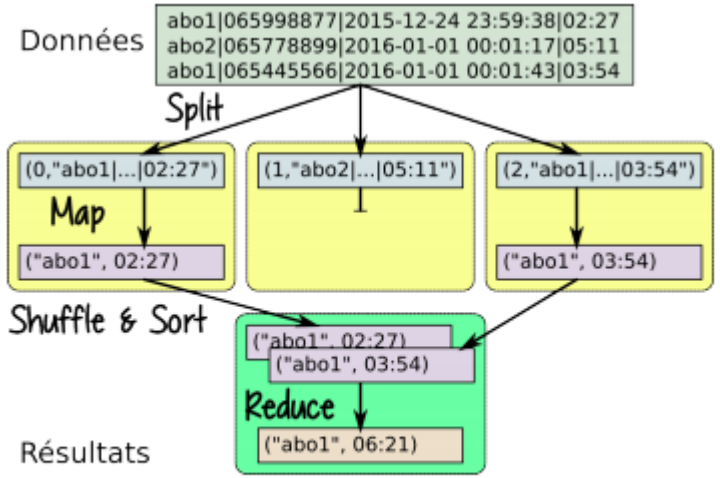


Figure 9: Étapes MapReduce

PIG

- un langage de programmation de requêtes sur des fichiers HDFS qui se veut plus simple que Java pour écrire des jobs MapReduce.
  - Pig sert à lancer les programmes Pig Latin dans l'environnement Hadoop.

```
personnes = LOAD 'personnes.csv' USING PigStorage(';')
        AS (userid:int, nom:chararray, age:int);
jeunesadultes = FILTER personnes BY age >= 18 AND age < 24;
classement = ORDER jeunesadultes BY age;
resultat = LIMIT classement 10;
DUMP resultat;
```

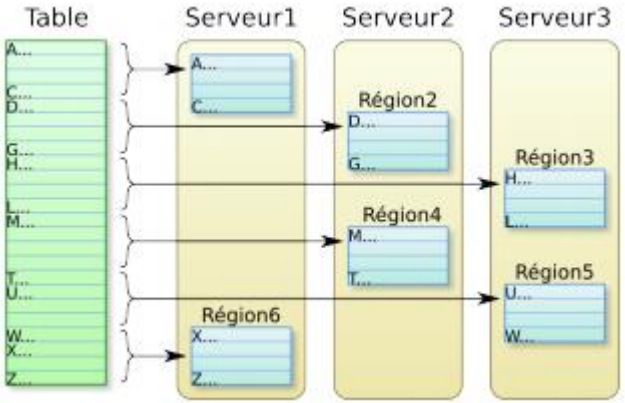
HBASE

- HBase est un système de stockage efficace pour des données très volumineuses. Il permet d'accéder aux données très rapidement même quand elles sont gigantesques.
- Une variante de HBase est notamment utilisée par FaceBook pour stocker tous les messages SMS, email et chat
- HBase mémorise des n-uplets constitués de colonnes (champs). Les n-uplets sont identifiés par une clé. À l'affichage, les colonnes d'un même n-uplet sont affichées successivement :

```
Clés       Colonnes et Valeurs
isbn7615   colonne=auteur valeur="Jules Verne"
isbn7615   colonne=titre  valeur="De la Terre à la Lune"
isbn7892   colonne=auteur valeur="Jules Verne"
isbn7892   colonne=titre  valeur="Autour de la Lune"
```

HBASE

- Pour obtenir une grande efficacité, les données des tables HBase sont séparées en régions. Une région contient un certain nombre de n-uplets contigus (un intervalle de clés successives).



DIFFÉRENCES ENTRE HBASE ET SQL

- Voici quelques caractéristiques de HBase :
  - Les n-uplets sont classés selon leur clé, dans l'ordre alphabétique. Cette particularité est extrêmement importante pour la recherche d'informations. On est amené à définir les clés de façon à rapprocher les données connexes.
  - Les n-uplets de HBase peuvent être incomplets. Les colonnes ne sont pas forcément remplies pour chaque n-uplet, au point qu'on peut même avoir des colonnes différentes pour les n-uplets. Ce ne sont pas des valeurs null, mais des colonnes carrément absentes. On qualifie ça de « matrice creuse » (sparse data).
  - Les colonnes appelées qualifiers sont groupées en familles.
  - Les valeurs, appelées cellules sont enregistrées en un certain nombre de versions, avec une date appelée timestamp.

HBASE: STRUCTURE DES DONNÉES

- Au plus haut niveau, une table HBase est un dictionnaire trié sur les clés,
- Chaque n-uplet est une liste de familles,
- Une famille est un dictionnaire trié sur les noms de colonnes (aussi appelées qualifier),
- Une cellule est une liste de (quelques) paires . La date, un timestamp permet d'identifier la version de la valeur.
- Donc finalement, pour obtenir une valeur isolée, il faut fournir un quadruplet :
  - (clé, nomfamille, nomcolonne, date)

HBASE: SHELL

- HBase offre plusieurs mécanismes pour travailler, dont :
  - Un shell où on tape des commandes,
  - Une API à utiliser dans des programmes Java, voir plus loin,
  - Une API Python appelée HappyBase.
- hbase shell
- Il faut savoir que c’est le langage Ruby qui sert de shell. Les commandes sont écrites dans la syntaxe de ce langage.

HBASE: SHELL

- Creation, Il y a deux syntaxes :
  - create 'NOMTABLE', 'FAMILLE1', 'FAMILLE2'...
  - create 'NOMTABLE', {NAME=>'FAMILLE1'}, {NAME=>'FAMILLE2'}...
- Supression
  - disable 'NOMTABLE'
  - drop 'NOMTABLE'
- Affichage
  - get 'NOMTABLE', 'CLE' get 'NOMTABLE', 'CLE', 'FAM:COLONNE'
  - get 'NOMTABLE', 'CLE', 'FAM:COLONNE', TIMESTAMP

HBASE: HIVE

- Hive simplifie le travail avec une base de données comme HBase ou des fichiers CSV.
- Hive permet d’écrire des requêtes dans un langage inspiré de SQL et appelé HiveQL. Ces requêtes sont transformées en jobs MapReduce.

```
CREATE TABLE releves (
        idreleve STRING,
        annee INT, ...
        temperature FLOAT, quality BYTE,
        ...)
    ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';
```