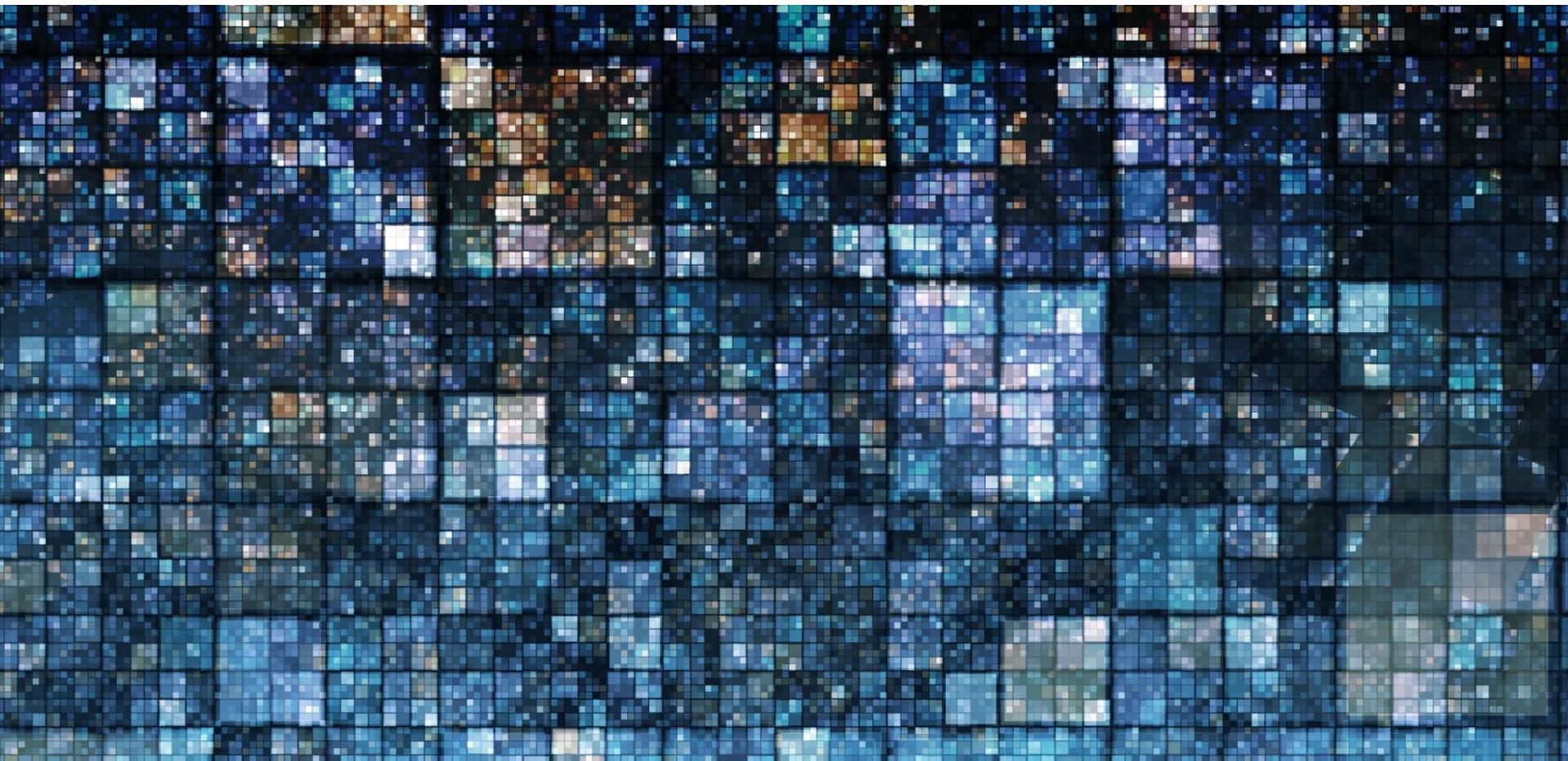


Hadoop et le Big Data

Ecole Centrale Supélec
Mastère SIO - année 2018

Salle VI.336 (3^{ème} étage du bâtiment Eiffel)



Objectifs T1 2018

- Mieux comprendre le **concept Big Data**
- Quelles **technologies** en support ?
- Focus sur **Hadoop**, l'éco-système Big Data open-source par excellence + **Spark**
- Réaliser un **projet** avec Hadoop pour expérimenter cette solution et acquérir des compétences (en plus des **TP**)

3 intervenants extérieurs

contact: sio.bigdata@gmail.com



- **Alzennyr GOMES DA SILVA**
 - Licence + Master en Data Mining à UFPB/UFPE, Brésil
 - Thèse INRIA/Paris Dauphine
 - Actuellement en poste à EDF Lab Paris-Saclay



- **Benoît GROSSIN**
 - Ingénieur UTC + Master Statistiques et Econométrie à l'Université de Toulouse
 - Actuellement en poste à EDF Lab Paris-Saclay



- **Leeley DAIO**
 - Ingénieur INSA Lyon + Executive Master à l'ESCP
 - Actuellement en poste à EDF Commerce La Défense

Agenda du jour

- Ce matin :
 - Concept Big Data et technologies associées, dont le NoSQL
 - Hadoop – MapReduce/HDFS/HIVE + Spark
- Cette après-midi :
 - Vérification de l'environnement technique pour les TPs
 - Présentation du projet qui servira d'évaluation

Agenda des sessions suivantes

- **26/01/18** : 6 heures - Alzennyr & Benoît
 - AM: Introduction générale / Alzennyr
 - PM: Vérification de l'environnement technique et présentation du projet / Benoît
- **02/02/18 AM** : 3 heures / Leeley
Prise en main de HDP + Hive
- **09/02/18 AM** : 3 heures / Benoît
Prise en main de Tableau Software avec HIVE/SparkSQL
- **16/02/18 AM** : 3 heures / Alzennyr
Prise en main de Spark – focus SparkSQL
- **23/02/18 PM** : 3 heures / tous
Séance de questions/réponses sur les séances précédentes et de suivi du Projet
- **16/03/18 PM** : 3 heures / Benoît & Leeley
Soutenance orale des projets (20 minutes par groupe) + Remise du rapport

AM : 9h45 – 13h00

PM : 13h45 – 17h00

DES QUESTIONS ? DES REMARQUES ?

§ 1 - BIG DATA ?



V3

BIG DATA = V³ = VOLUME, VELOCITE, VARIETE- Gartner

Au moins 2 V à ajouter :

- **VALIDITE** : s'appuyer sur des données de qualité
- **VALEUR** : tirer de la valeur du capital de données

Valeur :

- Améliorer/Accélérer les prises de décisions et actions
- Optimiser des processus existants

Ruptures / Coûts :

- Adapter les SI
- Innover dans l'exploitation des données
- Intégrer de nouvelles compétences

Une transformation qui touche tous les secteurs d'activité

- Le Big Data ne se limite pas au secteur des « .com » ... même s'il a un rôle particulier –sur le plan des technologies Big Data notamment



- De nombreux secteurs d'activités, parfois même historiques, sont concerné par cette transformation pas la Donnée
- Des centaines d'exemples possibles en France ou à l'International
- Sélection de 2 exemples d'opportunités et de ruptures liées au Big Data

Big Data et Secteur Médical

- Exemple de l'introduction du monitoring des grands prématurés via des capteurs dont les données sont analysées en continu et au fil de l'eau

[Enterprise Apps](#) / [IBM InfoSphere, Big Data Help Toronto Hospital Monitor Premature Infants](#)

IBM InfoSphere, Big Data Help Toronto Hospital Monitor Premature Infants

By Brian T. Horowitz | Posted 2013-09-26 [Email](#) [Print](#)



The system is capable of processing the 1256 readings a second it currently receives per patient, and has the potential to provide real-time analysis to help clinicians to predict more quickly potential adverse changes in an infant's condition. In addition, the data will be stored to allow doctors to look for common factors that might enable them to make more accurate predictions about a baby's condition.

Source :

http://www.ibm.com/smarterplanet/global/files/ca_en_us_healthcare_smarter_healthcare_data_baby.pdf

Big Data et Secteur de l'Électricité

- Les compteurs communicants (smart meters) et plus généralement l'émergence des réseaux électriques intelligents (smart grids)
- En France, déploiement sur 2015->~2020 de plus de 35 millions de compteurs communicants (Linky, porté par ERDF)

> STRATÉGIE > ENERGIE

Demain, ERDF deviendra gestionnaire de données avec Linky



Philippe Monloubou président du

Romain Charbonnier | 28/0



Avec ces compteurs électriques nouvelle génération, la question de la gestion des données se pose. Comment ERDF va-t-il y faire face ?

Imaginez que les compteurs Linky fournissent toutes les 10 min de nouvelles données. Nous ne pourrons donc plus gérer les données comme nous le faisons aujourd'hui. La sécurité complète des données de nos clients est prioritaire. La question est de savoir si nous construisons un data center en région pour gérer ses milliards de données, ou si l'on externalise cette gestion. Cela fait partie de notre 2e grand projet sur lequel nous travaillons activement. Une exigence forte pour ERDF, qui deviendra, demain, un gestionnaire de données. Un nouveau métier. Une révolution !

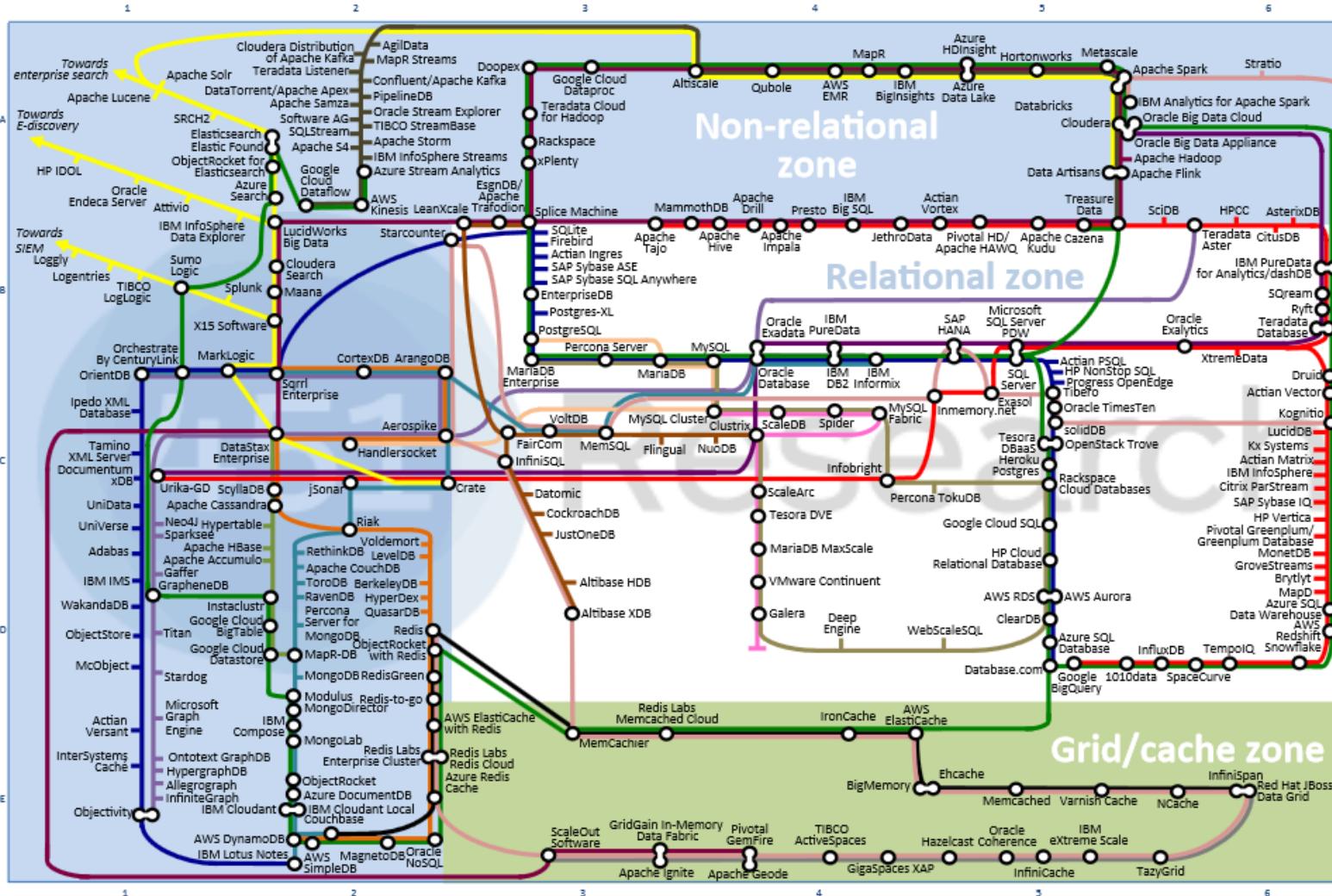
La technologie, support nécessaire au Big Data

- Pour tenir ses promesses, le Big Data s'appuie sur des technologies capables d'adresser les V - V(olume) V(ariété) V(élocité)
 - La technologie Hadoop peut se positionner sur les 3V grâce à un éco-système de solutions
- D'autres solutions se spécialisent sur une dimension en particulier.
 - Par exemple les solution CEP – Complex Events Processing – dédiées à la gestion et au traitement de flux de données temps réel – V(élocité) –
 - CEP utilisés par exemple pour analyser en masse les flux financiers pour le trading haute fréquence
- L'offre technologique data plateforme (de tout type) explose avec le rôle de plus en plus important des data – « data is the new oil » !

Data Platforms Map

January 2016

Key:	
A	General purpose
B	Specialist analytic
C	-as-a-Service
D	BigTables
E	Graph
F	Document
G	Key value stores
H	Key value direct access
I	Hadoop
J	MySQL ecosystem
K	Advanced clustering/sharding
L	New SQL databases
M	Data caching
N	Data grid
O	Search
P	Appliances
Q	In-memory
R	Stream processing



<https://451research.com/state-of-the-database-landscape>

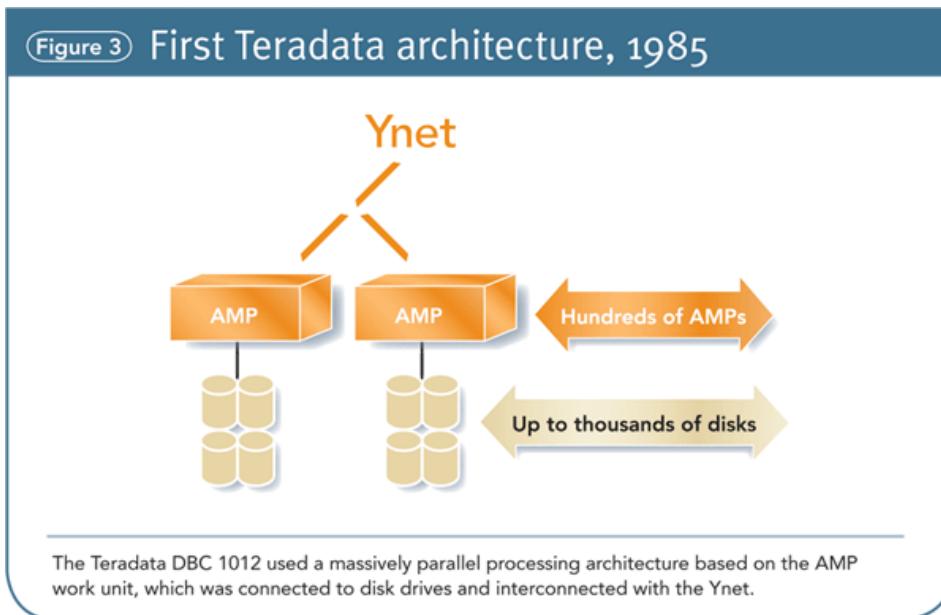
© 2016 by 451 Research LLC.
All rights reserved

Appliance Big Data des éditeurs

- Les éditeurs de solutions de base de données – IBM, Teradata, Oracle ... – ont des solutions Big Data sous forme d'appliance
- Appliance = un produit matériel ET logiciel = solution clef en main
- Le terme data warehouse appliance (DWA) est utilisé dans ce cas
- La plupart des DWA ont une architecture Massively Parallel Processing (MPP) à la fois pour assurer la **performance** et la **scalabilité**
- La scalabilité est la capacité d'un système à monter en charge
- La scalabilité des MPP est faite de manière horizontale « scale out » : on ajoute un nouveau nœud de calcul dans un cluster – pour augmenter sa capacité de traitement et de stockage
- La scalabilité verticale « scale up » consiste à donner plus de ressource à un nœud du système – en augmentant sa mémoire par exemple

Appliance Big Data des éditeurs

- La plupart des DWA MPP offrent un environnement « share-nothing architecture » (SN) : chaque nœud est indépendant et auto-suffisant, idéal pour la scalabilité horizontale !
- La première implémentation commercial MPP SN remonte au début des années 1980 : il s'agit de Teradata.



Exemple d'Appliance Teradata actuelle :
Teradata Active Enterprise
Data Warehouse 6750

Appliance Big Data des éditeurs

- Certains grands noms, comme **Oracle** – ExaData – , ont fait un choix différent : **MPP en « shared-everything »**
- « **Shared-everything** » : les nœuds du cluster partagent des données au niveau disque et/ou mémoire.
- Difficile de déclarer un vainqueur définitif entre « shared-nothing » ou « shared-everything »
- Enfin, bien que le modèle « Appliance » est dominant, il existe des offres logicielles Big Data strictes comme **GreenPlum d'EMC²**

Au-delà du relationnel ...

- Les appliances Big Data des éditeurs – malgré leur excellentes performances et leur degré d'intégration et de sophistication – restent des solutions relationnelles classiques (et aussi onéreuses! – on parle parfois en millions de \$)
- Le « big bang » data – commencé au début des années 2000 – a nécessité des solutions différentes, comme les solutions CEP déjà évoquées mais aussi des solutions NoSQL qui privilégient la haute disponibilité et la simplicité au détriment de la cohérence
- Nous allons maintenant parler du NoSQL, puis d'Hadoop qui essaye – grâce à un écosystème de solutions sans cesse amélioré et grossi – de se positionner sur toutes les facettes du Big Data et de devenir un « Data OS » open-source.
- PAUSE

§ 2 – NOSQL ?

Bases de données NOSQL

- Le terme **NOSQL** (*Not Only SQL*, ~~NoSQL~~) a été popularisé début 2009 par Johan Oskarsson
- Les BD NoSQL ne respectent pas forcément les propriétés ACID (Atomicité, Cohérence, Isolation, Durabilité) des BD relationnelles
- Les BD NoSQL n'ont donc pas vocation à remplacer les BD relationnelles
- Afin d' éviter les jointures, elles poussent à la **dénormalisation**

Pour certains cas d' application, une solution qui ne respect pas toutes les contraintes du SQL (ex: normalisation) peut être envisageable → NOSQL

- Caractéristiques des Bases de Données distribuées
 - Extensibilité
 - Sharding
 - Théorème du CAP

Extensibilité (Scalability)

- *L'extensibilité est la propriété d'un système, d'un réseau ou d'un processus qui témoigne de sa capacité à gérer des charges de travail importantes en toute souplesse ou à être agrandi sans difficultés.*
- Deux types :

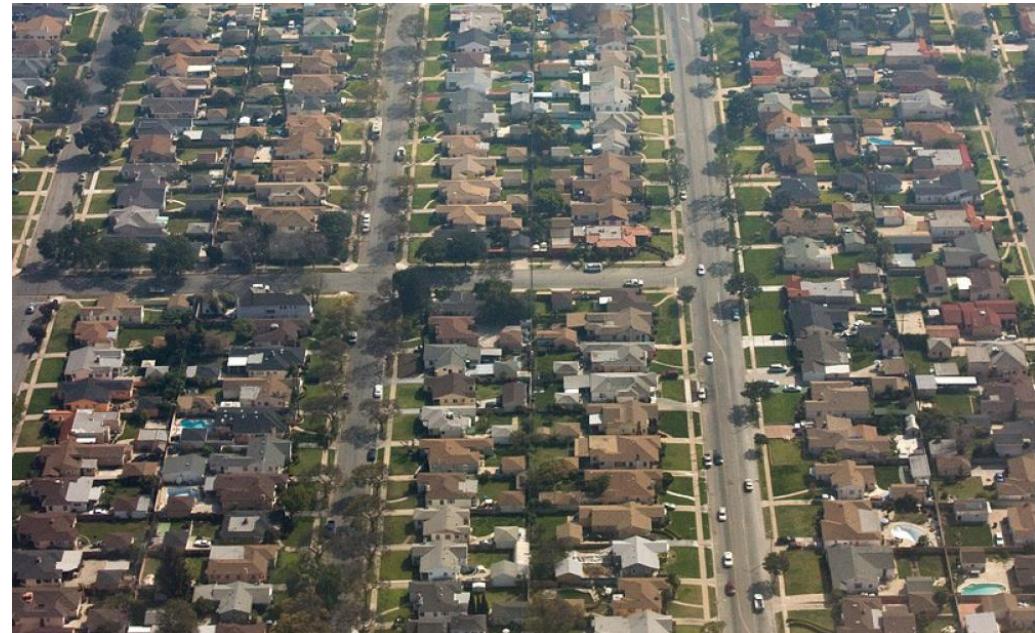
Extensibilité verticale (scale up)

- Ajout de serveurs puissants
- Coût important
- Mise en place facile



Extensibilité horizontale (scale out)

- Ajout de machines ordinaires (commodity hardware)
- Solution moins onéreuse
- Mise en place plus délicate (*load balancing*)

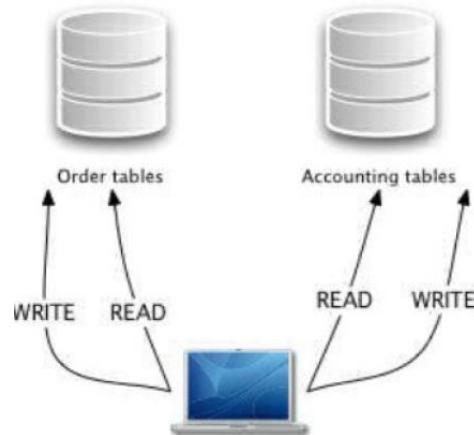


Sharding (partitionnement de données)

- *Un shard (partition) est une division logique d'une base de données en plusieurs parties indépendantes. Cela permet d'obtenir une capacité de stockage supérieure à la taille maximum des disques durs ou d'effectuer des requêtes en parallèle sur plusieurs partitions.*
- Deux types :

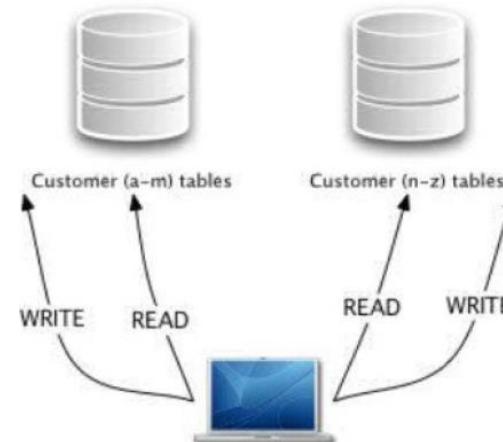
Sharding Vertical

Les serveurs stockent différentes tables d' une base de données



Sharding Horizontal

Chaque serveur stocke un sous ensemble des données (identifié par un intervalle de clés) d' une même table



Le théorème du CAP¹

- Théorème de Eric Brewer évoqué lorsque l' on parle de données massivement distribuées
 - **Consistency** : l' ensemble des clients voient la même donnée, même après des mises à jour
 - **Availability** : tous les clients peuvent trouver la donnée répliquée, même lors d' un crash
 - **Partition tolerance** : garantie de la continuité du fonctionnement en cas d' ajout/suppression de partition (ou nœud) au système distribué

« seulement deux de ces postulats peuvent être appliqués en même temps en environnement distribué »
- Dans le cas du NOSQL
 - privilégier la haute disponibilité grâce à de puissantes capacités de partitionnement, au détriment de la cohérence des données

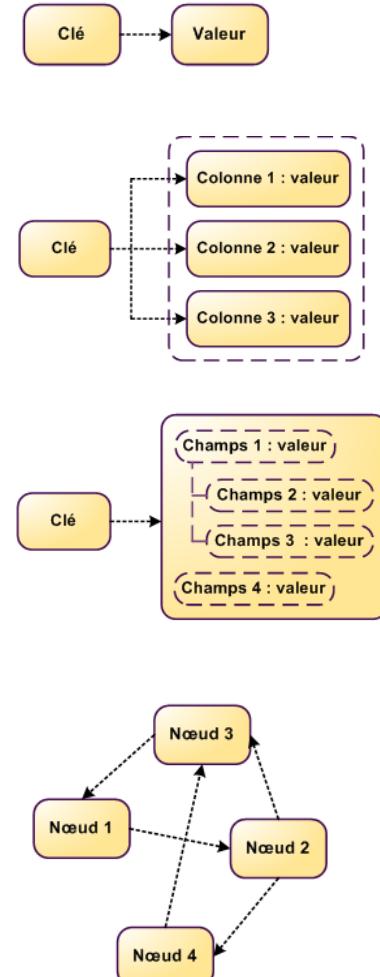
¹ <http://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf>

Les Bases de Données NOSQL

- Caractéristiques
 - Les données sont :
 - partitionnées (gain d' extensibilité - *scalability*)
 - répliquées (gain de disponibilité - *availability*)
 - Modèle de données simple
 - Langage de requêtes adapté
 - avec des contraintes priorisant la performance
- Quid des Jointures ? Inexistantes grâce à la **dénormalisation**
 - Normalisation
 - une seule copie des données
 - réduction de l'espace de stockage
 - Dénormalisation
 - réPLICATION des données sur différentes machines
 - *eventually consistent* != éventuellement consistant
→ les données sont consistantes à la fin

Classification des BD NOSQL

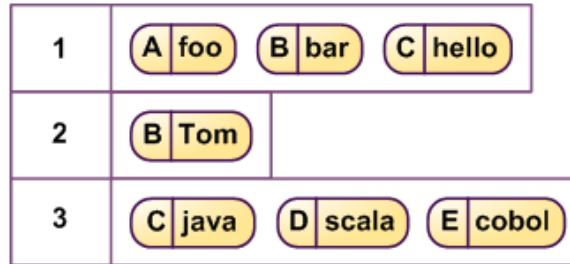
- **Clé/valeur:**
 - Ex: standalone (Redis), distribuées (Dynamo, Voldemort, Riak)
- **Orientées colonne:**
 - Ex: BigTable, HBase, Cassandra, HyperTable
- **Orientées document:** adaptées aux systèmes de gestion documentaire (CMS) où l' unité de stockage est un document.
 - Ex: CouchDB, MongoDB
- **Orientées graphe:** fondées sur la théorie des graphes et basées sur des concepts de nœuds, relations et propriétés.
 - Ex: Neo4J, FlockDB (de Twitter), OrientDB



BD's orientées ligne x BD's orientées colonne

	A	B	C	D	E
1	foo	bar	hello		
2		Tom			
3			java	scala	cobol

Organisation d'une table dans une BDD relationnelle



Organisation d'une table dans une BDD orientée colonnes

- Inconvénient des BDs orientées ligne
 - Redimensionnement onéreux
- Avantages des BDs orientées colonne
 - Compression des données plus facile
 - Lorsque les données d'une même colonne se ressemblent
 - Ajout de nouvelles colonnes plus rapide
 - les lignes n'ont pas besoin d'être redimensionnées
 - Plus efficiente pour le calcul des agrégats
 - SUM, COUNT, AVERAGE, etc.

Visual Guide to NoSQL Systems



NewSQL

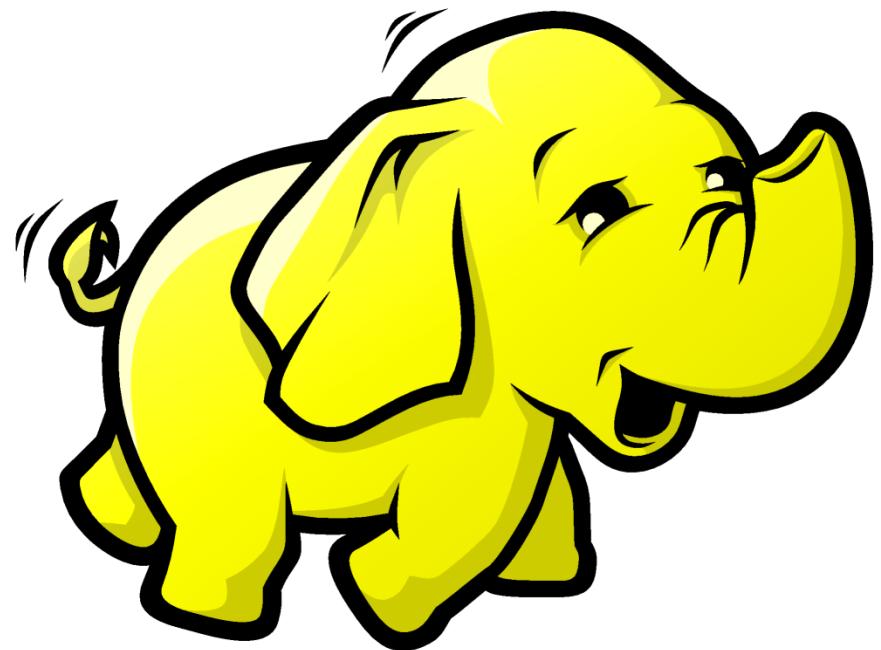
- Terme créé par [Mike Stonebraker](#) en 2011 lors de la conférence 
- NewSQL désigne une catégorie de SGBDs relationnelles qui cherchent à fournir la même puissance que les systèmes NoSQL pour le traitement transactionnel en ligne (lecture-écriture), tout en maintenant les propriétés ACID d'un SGBD traditionnel.

- ❖ *SQL as the primary interface*
- ❖ *ACID support for transactions*
- ❖ *Non-locking concurrency control*
- ❖ *High per-node performance*
- ❖ *Scalable, shared nothing architecture*

Michael Stonebraker

- Exemples de bases de données NewSQL:





§ 3 – HADOOP?

Origine d'Hadoop

- 2002: Doug Cutting développe Nutch, un moteur de recherche Open Source exploitant le calcul distribué : l'implémentation peut tourner seulement sur quelques machines et a de multiples problèmes, notamment en ce qui concerne l'accès et le partage de fichiers.
- 2003/2004: le département de recherche de Google publie deux articles, le premier sur GFS (un système de fichier distribué) et le second sur le paradigme Map/Reduce pour le calcul distribué.
 - [MapReduce: Simplified Data Processing on Large Clusters](#) by Jeffrey Dean and Sanjay Ghemawat from Google
- 2004: Doug Cutting développe framework (encore assez primitif) inspiré des papers de Google et porte le projet Nutch sur ce framework.
- 2006: Doug Cutting (désormais chez Yahoo) est en charge d'améliorer l'indexation du moteur de recherche de Yahoo. Il exploite le framework réalisé précédemment...

Origine d'Hadoop

- ... et créé une nouvelle version améliorée du framework en tant que projet Open Source de la fondation Apache, qu'il nomme Hadoop (le nom d'un éléphant en peluche de son fils).
- A l'époque, Hadoop est encore largement en développement – un *cluster pouvait alors comporter au maximum 5 à 20 machines*, etc.
- 2008 : le développement est maintenant très abouti, et Hadoop est exploité par le moteur de recherche de Yahoo- ainsi que par de nombreuses autres divisions de l'entreprise.



Hadoop aujourd’hui



- Plusieurs distributions d’Hadoop disponibles – dont celle de Cloudera (CDH), celle d’Hortonworks (HDP) et MapR
- Hadoop pénètre de plus en plus les entreprises – tout secteur d’activité (ex banque : HSBC, ING, Crédit Mutuel, ...) – avec l’argument d’un excellent équilibre **coût/performance/stockage/flexibilité**
- Le monde académique utilise aussi Hadoop : MIT, Berkeley, ...
- Dans ce contexte, les éditeurs classiques intègrent aussi Hadoop dans leur offre – comme IBM, Microsoft et Teradata
- Yahoo – toujours le plus gros cluster ?

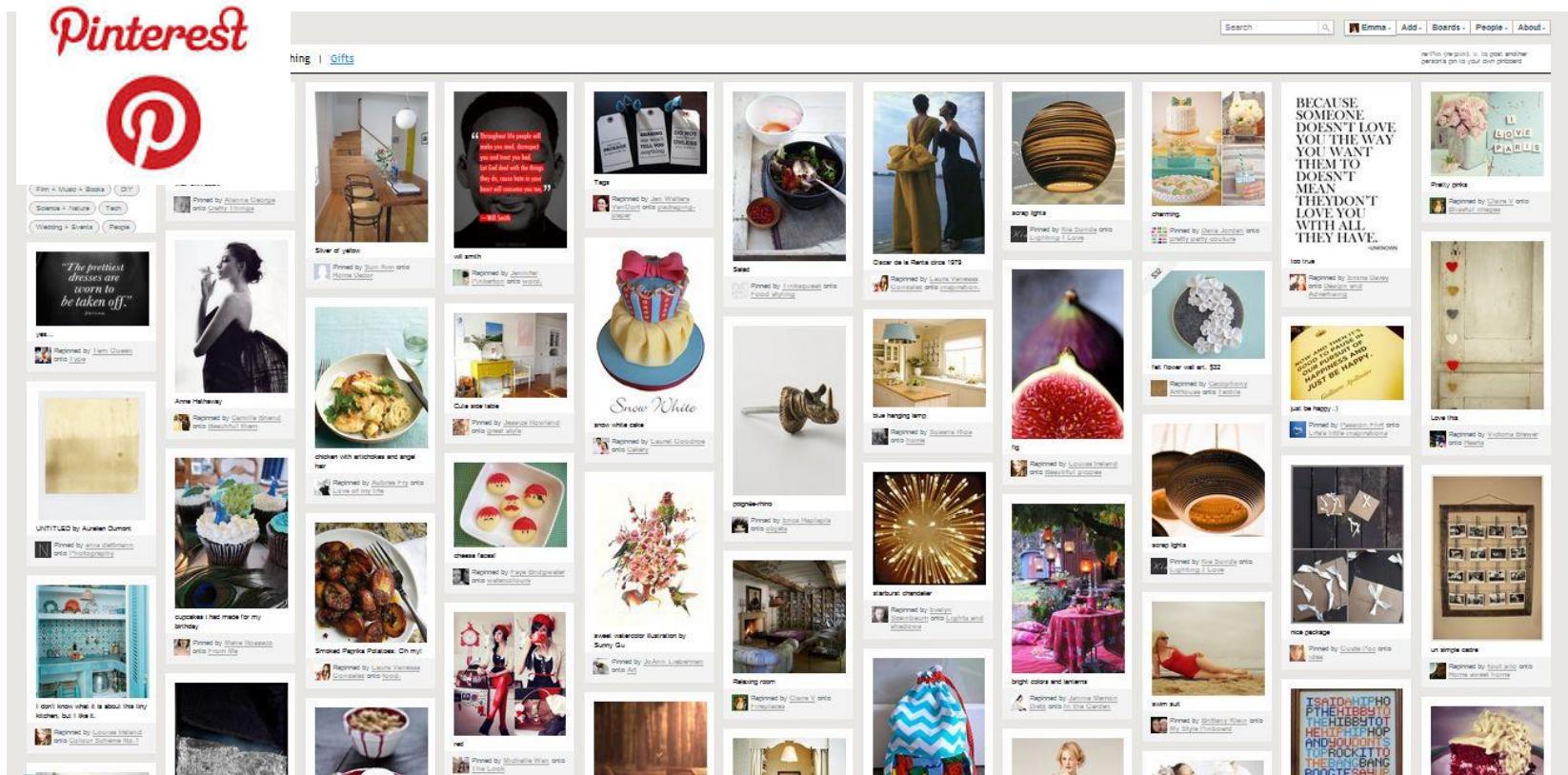
Hadoop at Yahoo (September 2014)

From an interview with Mithun Radhakrishnan, member of the Yahoo Hive team:

"Y!Grid is Yahoo's Grid of Hadoop Clusters that's used for all the "big data" processing that happens in Yahoo today. It currently consists of 16 clusters in multiple datacenters, spanning 32,500 nodes, and accounts for almost a million Hadoop jobs every day. Around 10% of those are Hive jobs."

Source: [ODBMS Industry Watch](#)

Exemple : Hadoop chez Pinterest



- 50+ millions d'utilisateurs, dont 1+ millions en France

Exemple : Hadoop chez Pinterest

- Quelques statistiques :
 - 30 billion Pins in the system
 - 20 terabytes of new data each day
 - over 100 regular Mapreduce users running over 2,000 jobs each day
 - six standing Hadoop clusters comprised of over 3,000 nodes
 - over 20 billion log messages and process nearly a petabyte of data with Hadoop each day



Src: [Pinterest Engineering Blog](#) -

<http://engineering.pinterest.com/post/92742371919/powering-big-data-at-pinterest>

L'équation Hadoop

- Hadoop = excellent équilibre coût/performance/stockage/flexibilité
 - coût : open-source + commodity hardware
 - performance : map-reduce et scalabilité
 - Stockage : hdfs – stockage distribué
 - Flexibilité : écosystème riche

Map Reduce

- Pour pouvoir traiter un grand volume de données sur un cluster de machines, il faut découper le problème en plusieurs problèmes de taille réduite à exécuter sur différentes machines (stratégie « diviser pour régner »).
- De multiples approches existent et ont existé pour cette division d'un problème en plusieurs « sous-tâches ».
- MapReduce est un paradigme (modèle) de programmation distribuée popularisé par Google

MapReduce: Simplified Data Processing on Large Clusters. Jeffrey Dean and Sanjay Ghemawat. OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA, December, 2004.

<http://research.google.com/archive/mapreduce.html>

Map Reduce

- Le modèle MapReduce préconise deux opérations distinctes:
- **Map ()** : transforme les données d'entrée en une série de couples clef/valeur. 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 clefs distinctes produite par l'opération MAP. Chaque machine du cluster recevra une des clefs uniques produites par MAP, accompagnées de la liste des valeurs associées à la clef. Chaque machine effectuera alors l'opération REDUCE pour cette clef et les valeurs associées.

Map Reduce

- Les 4 étapes du traitement MapReduce :
 - **Split**: les données en entrée sont découpées en plusieurs fragments
 - **Mapper**: l'opération Map est appliquée sur chacun de ces fragments afin d'obtenir les couples (*clef;valeur*)
 - **Shuffle**: les couples (*clef;valeur*) sont groupés par clef
 - **Reduce**: l'opération Reduce est appliquée sur chacun de ces groupes indexés par clef afin d'obtenir une valeur finale pour chaque clef

Map Reduce (exemple *Word Count*)

- Objectif: compter le nombre de mots dans un texte
- Première étape: déterminer une manière de découper (split) les données d'entrée pour que chacune des machines puisse travailler sur une partie du texte.
- Notre problème est ici très simple – on peut par exemple décider de découper les données d'entrée ligne par ligne. Chacune des lignes du texte sera un fragment de nos données d'entrée.

Map Reduce (exemple *Word Count*)

- Nos données d'entrée (le texte):

Celui qui croyait au ciel

Celui qui n'y croyait pas

[...]

Fou qui fait le délicat

Fou qui songe à ses querelles

(Louis Aragon,
La rose et le Réséda,
1943, fragment)

- Pour simplifier les choses, avant le découpage, nous allons supprimer toute ponctuation et les caractères accentués. Nous allons également passer l'intégralité du texte en minuscules.

Map Reduce (exemple *Word Count*)

- Ce qui donne, après découpage:

celui qui croyait au ciel

celui qui ny croyait pas

fou qui fait le delicat

fou qui songe a ses querelles

... on obtient 4 fragments depuis nos données d'entrée.

Map Reduce (exemple *Word Count*)

- On doit désormais déterminer la clef à utiliser pour notre opération MAP, et écrire le code de l'opération MAP elle-même.
- Puisqu'on s'intéresse aux occurrences des mots dans le texte, et qu'à terme on aura après l'opération REDUCE un résultat pour chacune des clefs distinctes, la clef qui s'impose logiquement dans notre cas est: ***le mot-lui même***.
- Quand à notre opération MAP, elle sera elle aussi simple: nous allons simplement parcourir le fragment qui nous est fourni et, pour chacun des mots, générer le couple clef/valeur: **(MOT ; 1)**. La valeur indique ici l'occurrence pour cette clef - puisqu'on a croisé le mot une fois, on lui donne la valeur « 1 ».

Map Reduce (exemple *Word Count*)

- Le code de notre opération MAP sera donc (ici en pseudo code):

```
POUR MOT dans LIGNE, FAIRE :  
    GENERER COUPLE (MOT ; 1)
```

- Pour chacun de nos fragments, les couples (clef; valeur) générés seront donc:

celui qui croyait au ciel	→	(celui;1) (qui;1) (croyait;1) (au;1) (ciel;1)
celui qui ny croyait pas	→	(celui;1) (qui;1) (ny;1) (croyait;1) (pas;1)
fou qui fait le delicat	→	(fou;1) (qui;1) (fait;1) (le;1) (delicat;1)
fou qui songe a ses querelles	→	(fou;1) (qui;1) (songe;1) (a;1) (ses;1) (querelles;1)

Map Reduce (exemple *Word Count*)

- Une fois notre opération MAP effectuée (de manière distribuée), Hadoop groupera (shuffle) tous les couples par clef commune.
- Cette opération est effectuée automatiquement par Hadoop. Elle est, là aussi, effectuée de manière distribuée en utilisant un algorithme de tri distribué, de manière récursive. Après son exécution, on obtiendra les 15 groupes suivants:

(celui;1) (celui;1)

(qui;1) (qui;1) (qui;1) (qui;1)

(croyait;1) (croyait;1)

(au;1) (ny;1)

(ciel;1) (pas;1)

(fou;1) (fou;1)

(fait;1) (le;1)

(delicat;1) (songe;1)

(a;1) (ses;1)

(querelles;1)

Map Reduce (exemple *Word Count*)

- Il nous reste à créer notre opération REDUCE, qui sera appelée pour chacun des groupes/clef distincte.
- Dans notre cas, elle va simplement consister à additionner toutes les valeurs liées à la clef spécifiée:

```
TOTAL=0
POUR COUPLE dans GROUPE , FAIRE :
    TOTAL=TOTAL+1
RENVOYER TOTAL
```

Map Reduce (exemple *Word Count*)

- Une fois l'opération REDUCE effectuée, on obtiendra donc une valeur unique pour chaque clef distincte. En l'occurrence, notre résultat sera:

```
qui: 4
celui: 2
croyait: 2
fou: 2
au: 1
ciel: 1
ny: 1
pas: 1
fait: 1
[...]
```

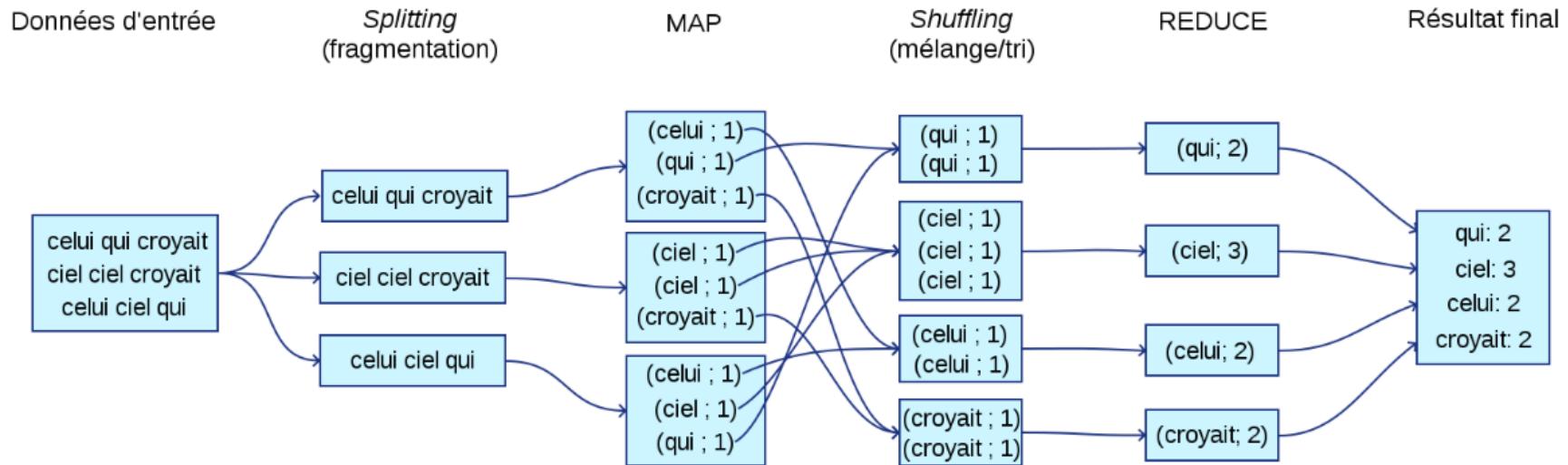
On constate que le mot le plus utilisé dans notre texte est « qui », avec 4 occurrences, suivi de « celui », « croyait » et « fou », avec 2 occurrences chacun.

Map Reduce (exemple *Word Count*)

- Notre exemple est évidemment trivial, et son exécution aurait été instantanée même sur une machine unique, mais il est d'ores et déjà utile: on pourrait tout à fait utiliser les mêmes implémentations de MAP et REDUCE sur l'intégralité des textes d'une bibliothèque Française, et obtenir ainsi un bon échantillon des mots les plus utilisés dans la langue Française.
- L'intérêt du modèle MapReduce est qu'il nous suffit de développer les deux opérations réellement importantes du traitement: MAP et REDUCE, et de bénéficier automatiquement de la possibilité d'effectuer le traitement sur un nombre variable de machines de manière distribuée.

Map Reduce (exemple *Word Count*)

Récapitatif:



HDFS

- HDFS = Hadoop Distributed FileSystem
- HDFS =Système de fichiers (FileSystem) d'Hadoop par défaut
- HDFS est inspiré de GFS – « The Google File System », 2003)
- Principales caractéristiques :
 - Distribué (!!!) : les données sont réparties sur les machines du cluster
 - Répliqué : les données sont répliquées pour palier la défaillance d'un machine
- HDFS est « taillé » pour les fortes volumétries :

HDFS at Twitter (June 2014)

Max int in javascript $2^{53}=8$ PB isn't enough to show some individual users' HDFS usage on our #Hadoop clusters. Overflow is so 20th century!

– Joep R. (@joep) June 27, 2014

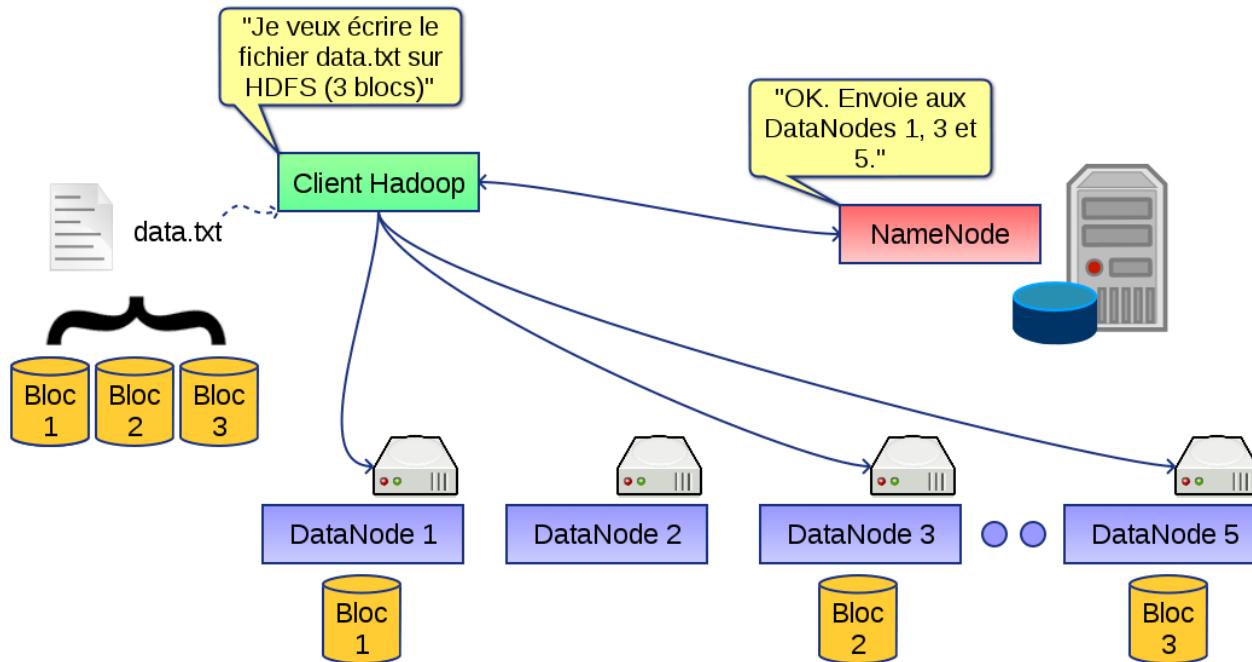
-
- Le mécanisme de gestion des tâches MapReduce (JobTracker/TaskTracker) communique en permanence avec HDFS

HDFS

- Dans HDFS, les données sont découpées en « blocks » - 64Mo par défaut
- HDFS s'appuie sur :
 - Le NameNode : 1 NameNode par Cluster
 - Le DataNode : 1 DataNode par machine du Cluster
- Pour comprendre le rôle de chacun, regardons le mécanisme d'écriture et de lecture d'un fichier dans HDFS
- On illustrera aussi la capacité de robustesse d'HDFS avec l'exemple de lecture

Ecriture d'un fichier

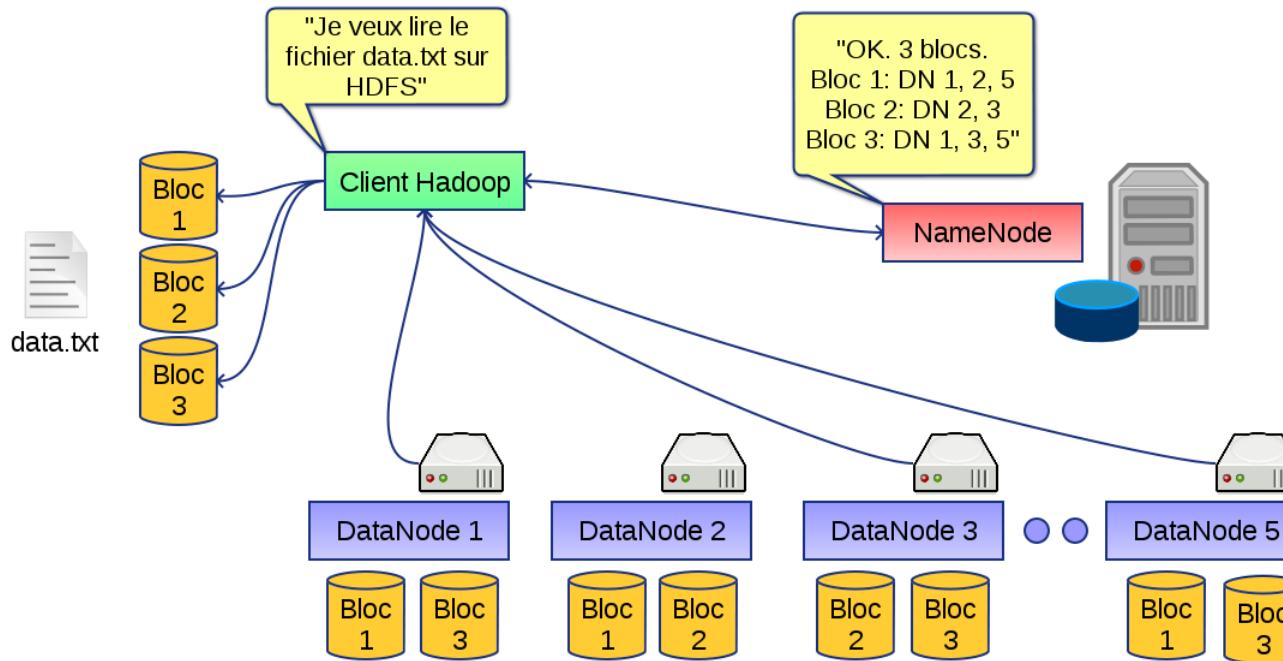
Ecriture HDFS



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

Lecture d'un fichier

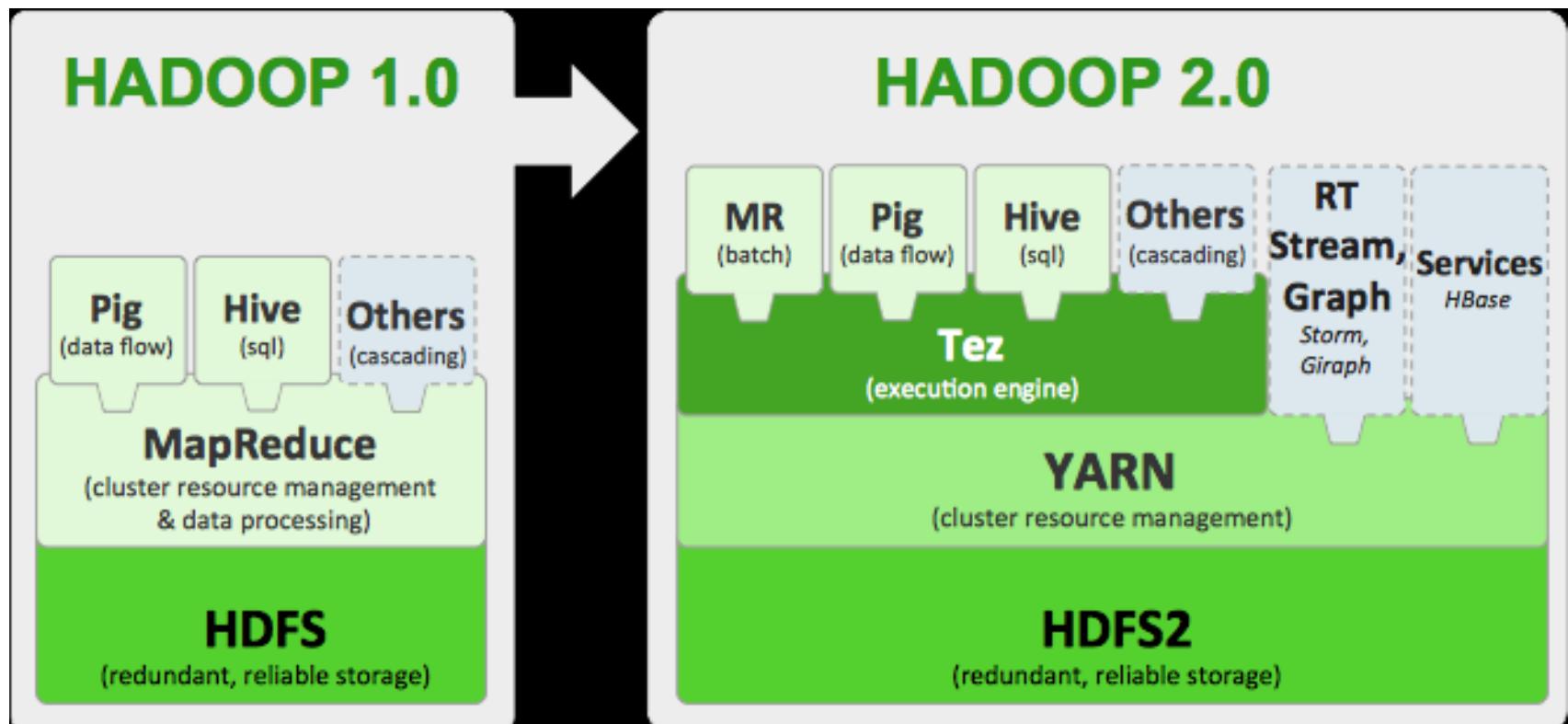
Lecture HDFS



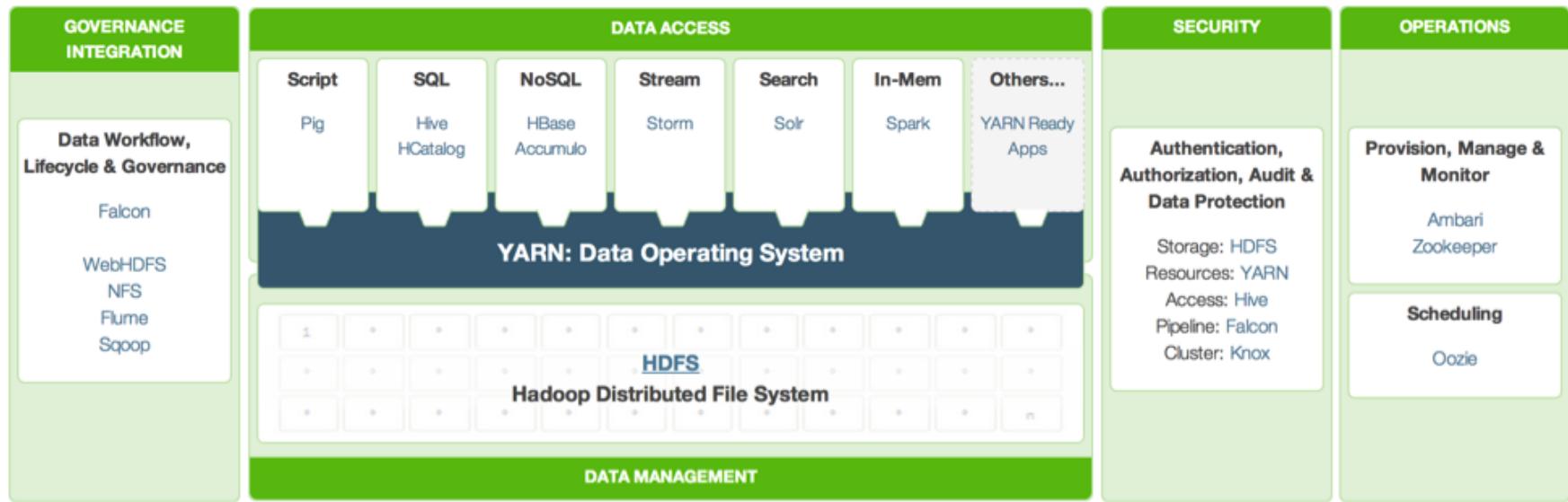
- Le client indique au NameNode qu'il souhaite lire un fichier.
- Celui-ci lui indique sa taille et les différents DataNode contenant les N blocs.
- Le client récupère chacun des blocs à un des DataNodes.
- Si un DataNode est indisponible, le client le demande à un autre.

Attention: cet exemple contient délibérément une erreur, dans Hadoop le nombre de répliques doit le même pour tous les blocs, ors le bloc 2 est répliqué uniquement 2 fois, il manque donc une troisième réplica pour ce bloc.

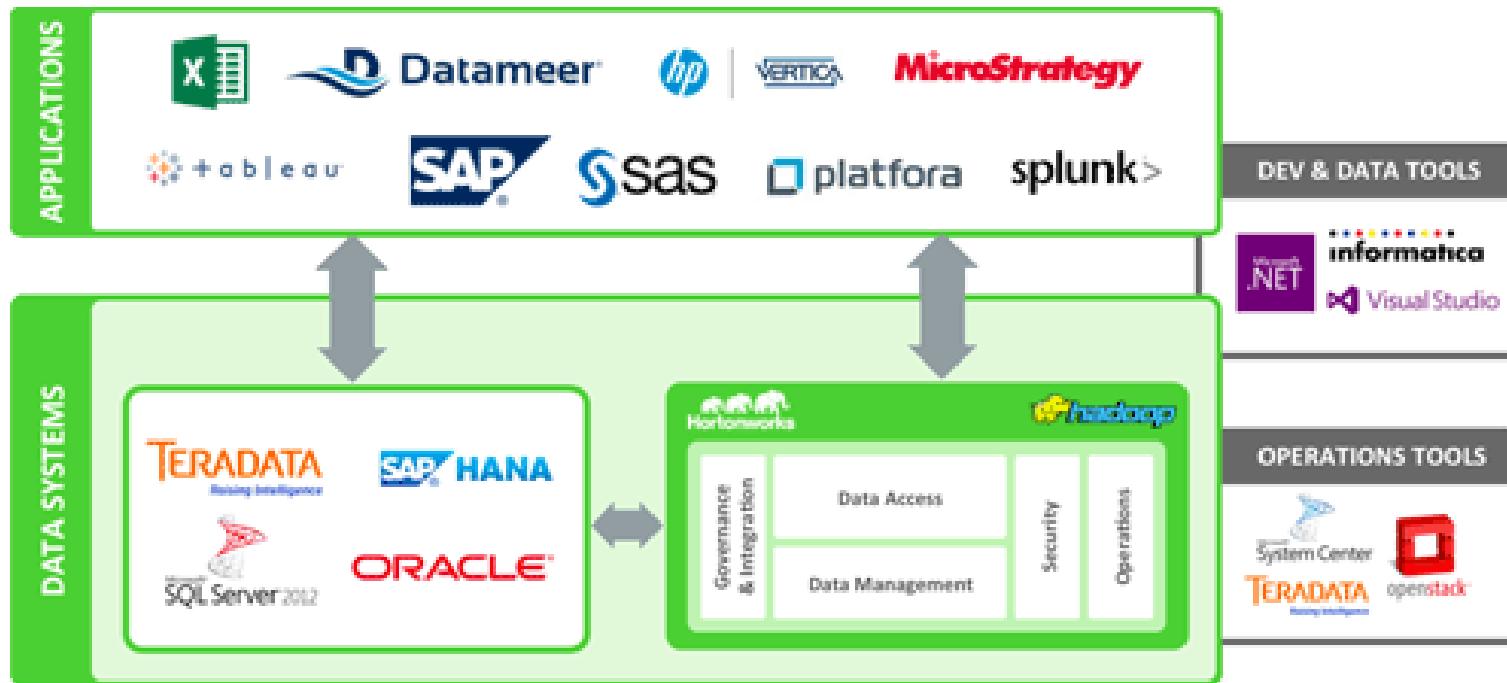
HDFS et MapReduce : le cœur Hadoop 1.0 !



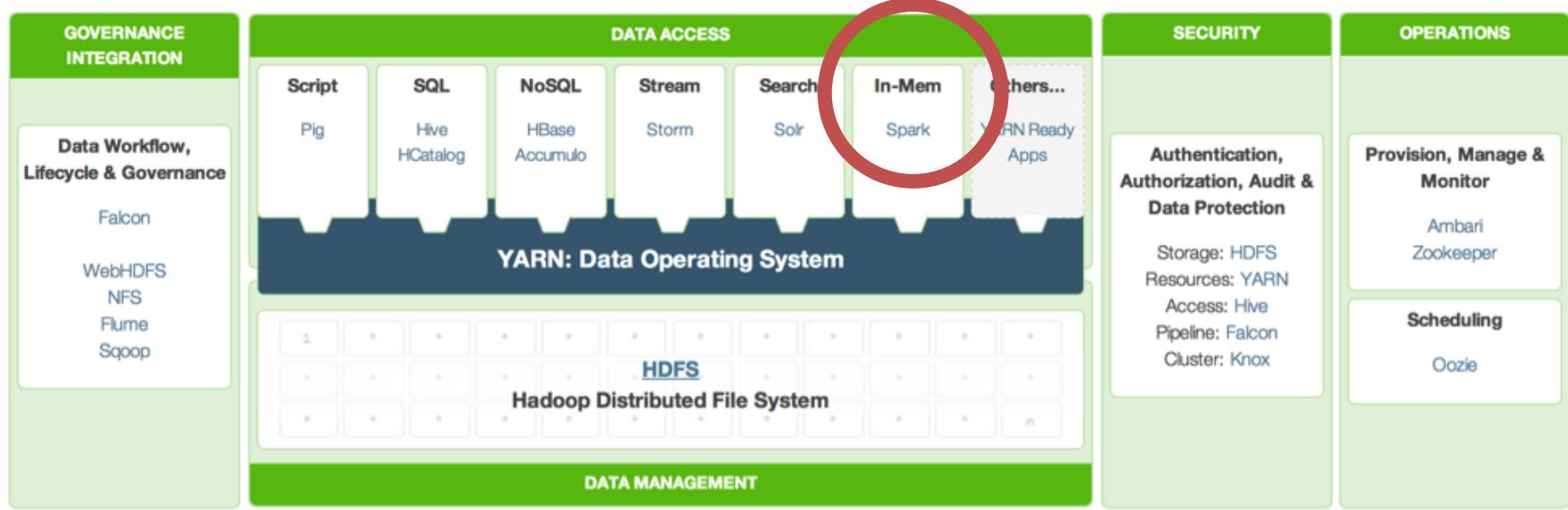
Ecosystème Hadoop : le « data OS »



Au-delà de l'Ecosystème Hadoop



Ecosystème Hadoop : le « data OS »



The background of the image is a dark, moody landscape featuring a range of mountains under a heavy, overcast sky. The lighting is low, creating deep shadows and a sense of mystery.

Spark

IBM Announces Major Commitment to Advance Apache®Spark™, Calling it Potentially the Most Significant Open S Project of the Next Decade

IBM Joins Spark Community, Plans Scientists



NEWS POPULAR VIDEOS FORTUNE 500

TECH BIG DATA

Survey shows huge popularity spike for Apache Spark SEPTEMBER 25, 2015

Trump hints at plan to raise taxes on the wealthy and eliminate it for others SEPTEMBER 25, 2015

VW staff, supplier warned of emissions test cheating years ago SEPTEMBER 27, 2015

Oil prices fall on slowing global economic growth outlook SEPTEMBER 25, 2015

India's Modi grew emotional at Facebook as he recalls his childhood SEPTEMBER 27, 2015

September jobs, Congress' deadline, and Trevor Noah — 5 things to know this week SEPTEMBER 27, 2015

ISIS has a money problem SEPTEMBER 27, 2015

Indian Prime Minister tours Tesla factory, talks batteries & solar with Elon Musk SEPTEMBER 27, 2015

How CST Brands' Kim Lavelle is dealing with her biggest fear SEPTEMBER 27, 2015

Here's how Bono and Mark Zuckerberg aim to bring the Internet to the globe SEPTEMBER 27, 2015

JUN 23, 2016 @ 10:28 AM 30,975 views

Spark Or Hadoop -- Which Is The Best Big Data Framework?



One question I get asked a lot by my clients is: Should we go for Hadoop or Spark as our big data framework? Spark has overtaken Hadoop as the most active open source Big Data project. While they are not directly comparable products, they both have many of the same uses.

an article explaining the essential tried to keep it accessible to

of the most popular tools used

work but recently the newer and te API 2.63% Software Foundation

t mutually exclusive, as they are times faster than Hadoop in system.

Survey shows huge popularity spike for Apache Spark

by Derrick Harris @derrickharris SEPTEMBER 25, 2015, 3:49 PM EDT

[✉](#) [✉](#) [f](#) [in](#)

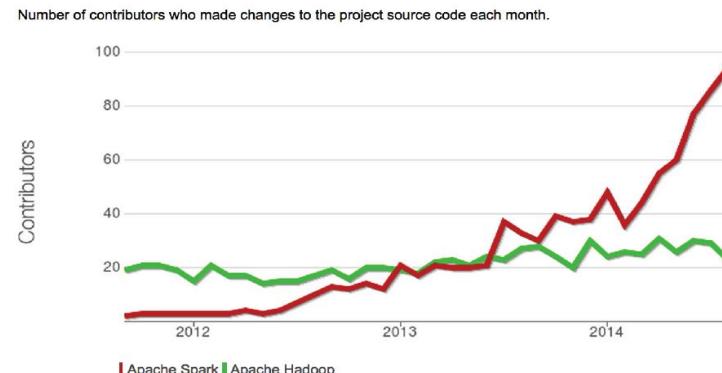
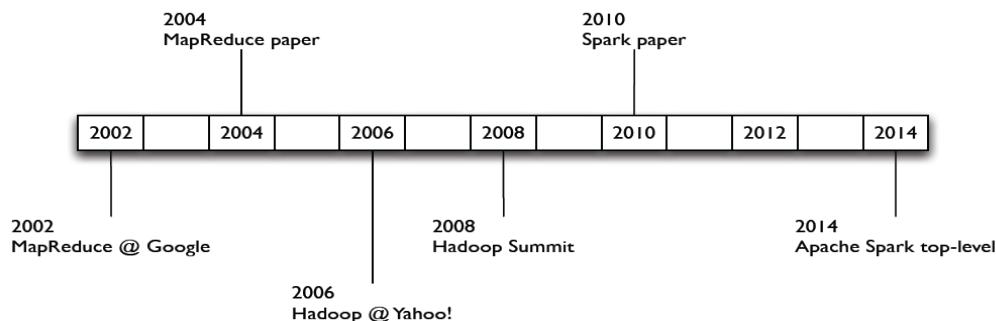


Click to go back, hold to see his

SPARK : UN MOTEUR IN-MEMORY



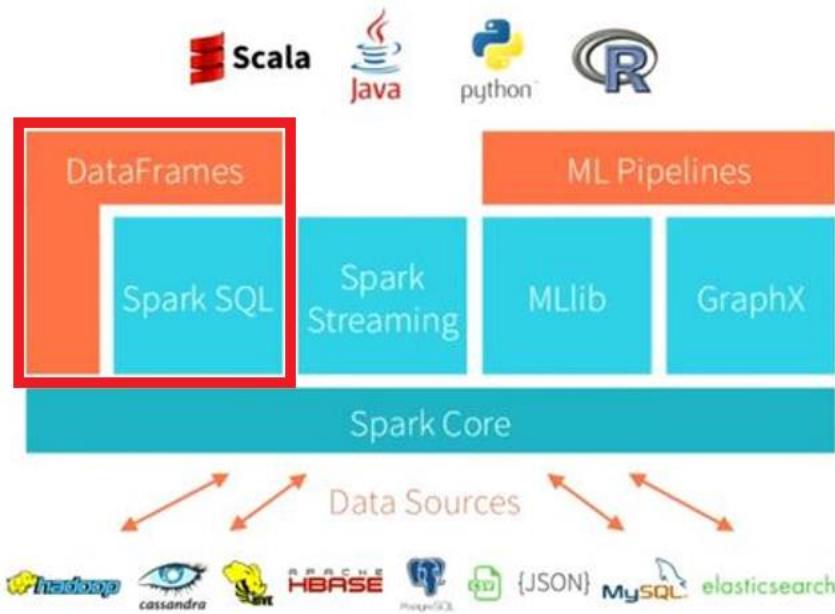
- Développé à l'origine par l'université de Berkeley en **2009** (laboratoire AMPLab)
- Projet adopté par la **fondation Apache** en **juin 2013**
- Croissance très rapide : **180 contributeurs** à ce jour, répartis dans **50 organisations**
- La logique Map-Reduce est conservée, avec une nouvelle approche :
 - Traitements exploitant la **mémoire RAM**
 - **Graphe d'exécution optimisé**
 - Logique de programmation **simplifiée**
 - Java, Scala, **Python**, R



SPARK : UN ECOSYSTEME COMPLET



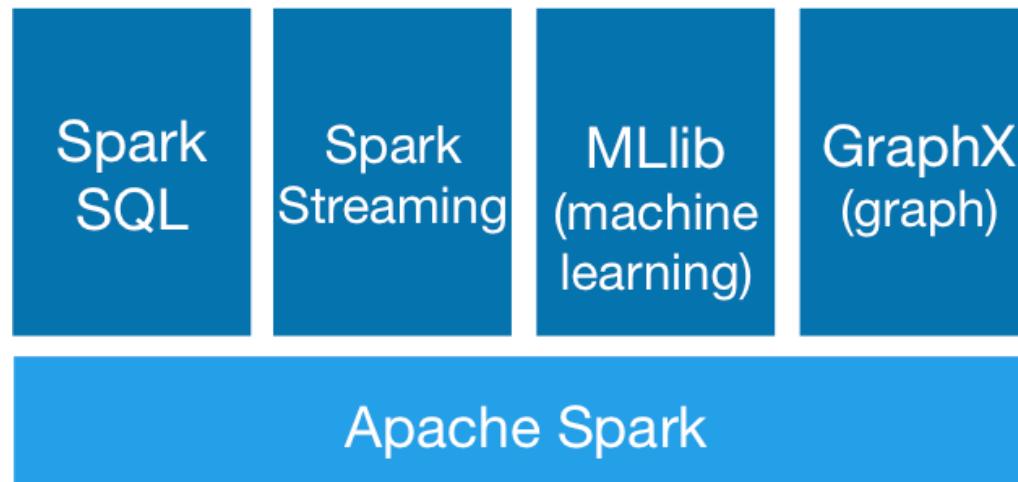
- Plus qu'un simple moteur, Spark est un **véritable framework** :



- Spark propose une **solution unifiée** pour des **traitements batch, interactifs** (faible latence) et **temps réel** (streaming)
- Spark offre une programmation avec un **niveau d'abstraction plus élevé**, réduisant ainsi considérablement la complexité inhérente à la création de jobs Map-Reduce

L'écosystème Spark

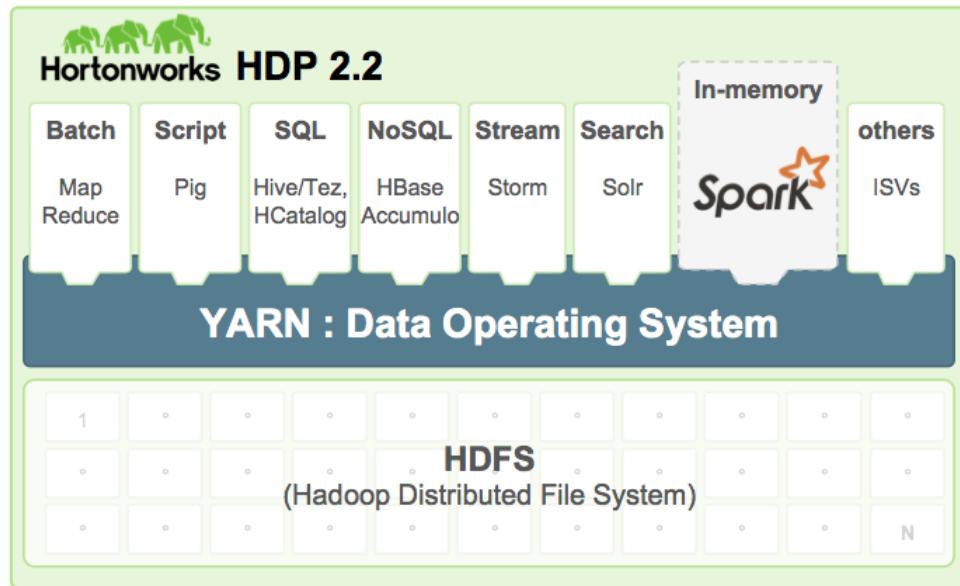
- **Spark Streaming** : utilisé pour traitement temps-réel des données en flux. Il s'appuie sur un mode de traitement en "micro batch" et utilise pour les données temps-réel DStream, c'est-à-dire une série de RDD (Resilient Distributed Dataset).
- **Spark SQL** : permet d'exposer les jeux de données Spark via l'API JDBC et d'exécuter des requêtes de type SQL.
- **Spark MLlib** : librairie de machine learning qui contient des algorithmes d'apprentissage automatique (classification,, clustering, le filtrage collaboratif, régression, réduction de dimensions, etc.)
- **Spark GraphX** : l'API pour le traitement et parallélisation de graphes. GraphX expose un jeu d'opérateurs de base (subgraph, joinVertices, aggregateMessages) ainsi qu'une variante optimisée de l'API Pregel.



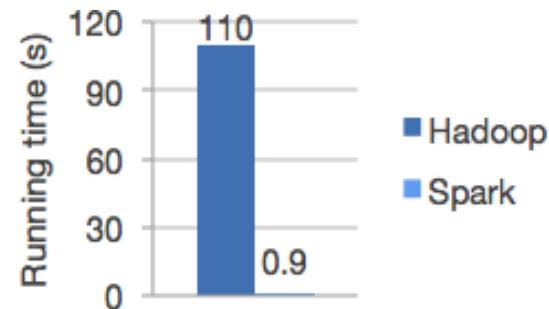
SPARK : INTEGRATION DANS HADOOP



- Intégration de Spark dans Hadoop :



Pour certains traitements, les gains de performances annoncés sont de **X10 sur disque et x100 en full in-memory** :



Conclusion

- Que retenez-vous de cette matinée ?

Références

- **Hadoop: The Definitive Guide.** O'Reilly Media
- **Programming Hive.** Data Warehouse and Query Language for Hadoop. O'Reilly Media
- **Cours Big Data,** Benjamin Renaut, Université de Nice Sophia Antipolis.
- **Tutoriel Yahoo!**
<https://developer.yahoo.com/hadoop/tutorial/>

