Introduction au Big Data & Apprentissage machine à grande échelle

Carine TOURE

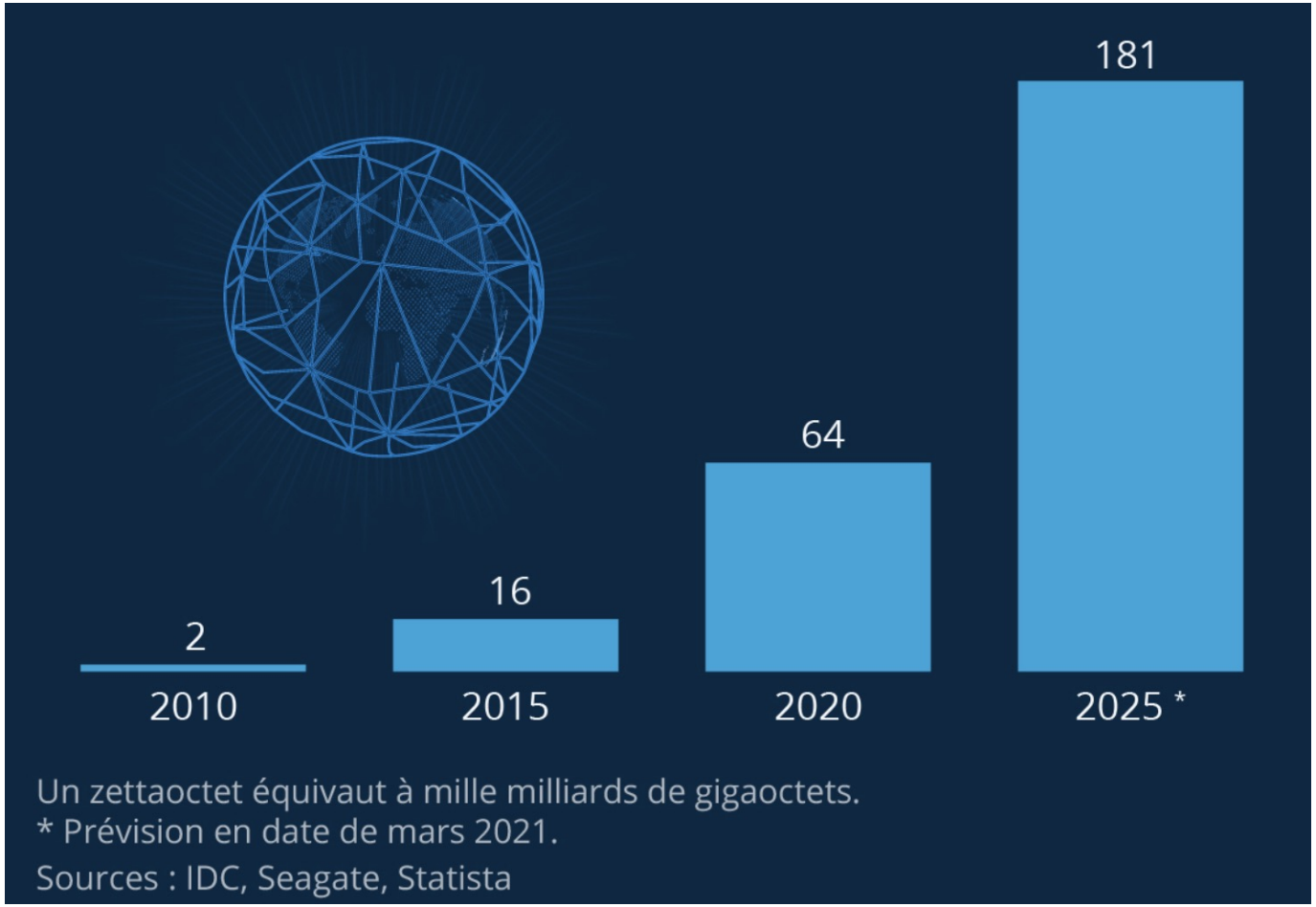
LMA36-22

Université des Lagunes

Outline

La place des données aujourd'hui

- Tous les logiciels, applications, sites web, objets connectés, capteurs... produisent leurs propres données aujourd'hui



- Depuis la fin des années 80,
  - boom internet
  - intensification des usages
  - augmentation de la production de données
  - vers la fin des années 2000, commercialisation des smartphones
  - on communique, on produit de la données, on partage via les réseaux sociaux
- Les données produites sont énormes (voir figure)
- 1 caractère $\approx$ 4 octets :
  - en français, un mot fait en moyenne 4.8 caractères
  - environ 20 octets pour stocker un mot
  - environ 1963 mots/personne/jour $\Leftrightarrow$ 40k Octets/personne/jour (plus de 292.5 To pour la population mondiale)

Figure. Estimation du volume de données numériques crées oàu repliquées dans le monde par an, en zettaoctet.
Source : https://fr.statista.com/infographie/17800/big-data-evolution-volume-donnees-numeriques-genere-dans-le-monde/

La place des données aujourd'hui

- Facebook $\approx$ 4 pétaoctets/jour soit 13.5 fois la quantité de données prononcée à l'oral dans le monde
- Sur le web en général, on prédit 181 zétaoctets de données produites en 2025 (voir fig précédente)
- Résumons :

Unité	Abbréviation	Taille (en octets)	Exemple
1 octet		1	4 octets ~ un caractère au format utf-8
1 kilooctet	ko	10^3	40 ko ~ mots prononcés par 1 personne/jr
1 mégaoctet	Mo	10^6	3 Mo ~ texte d'1 édition quotidienne du NY Times
1 gigaoctet	Go	10^9	1 Go ~ mots prononcés par 1 personne en 1 vie
1 téraoctet	To	10^{12}	>292.5 To ~ mots prononcés dans le monde/jr
1 pétaoctet	Po	10^{15}	4 Po ~ données générées par Facebook/jr
1 exaoctet	Eo	10^{18}	605Eo ~ données générées par le téléscope SKA/jr
1 zétaoctet	Zo	10^{21}	40 Zo ~ données générées par le web en 2020

La place des données aujourd'hui

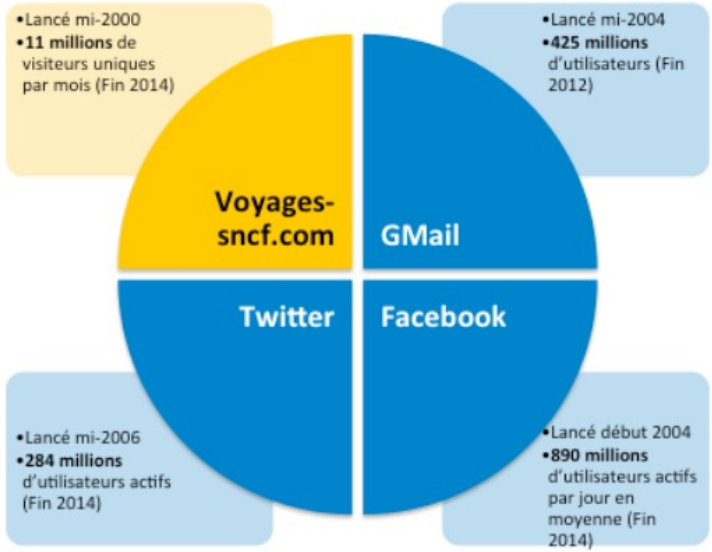
- Toutes ces données sont sources de valeurs à condition de les exploiter correctement et soulèvent des défis : comment les collecter, les stocker, les filtrer, les analyser ?
- Mais avant d'y arriver, définissons le terme big data :
  - origine controversée – fin des années 90/début 2000
  - se traduit littéralement par 'grosses données'
  - sa traduction française officielle recommandée est "mégadonnées", parfois on parle de "données massives".
- A partir de quel moment parle-t-on de big data ?
  - si la quantité de données excède la faculté de la machine à stocker et à analyser en temps acceptable (pour l'infrastructure et le business)
  - si les données sont trop grosses pour être stockées en RAM
  - il faut donc passer à l'échelle
- Le passage à l'échelle ne signifie pas seulement augmenter la RAM, et disque dur, l'infrastructure informatique, il s'accompagne presque toujours d'une transformation des usages du fait des caractéristiques des big data
- On parle des 3V (+2V supplémentaires) : Volume, Variété, Vélocité (+ Valeur, Validité/Véracité)

Volume

- Caractérisation des big data : Volume, Variété, Vélocité (+ Valeur, Validité/Véracité)

- Le volume :
  - Fait référence à la quantité d'informations, trop volumineuses pour être acquises, stockées, traitées, analysées et diffusées par des outils traditionnels
  - Peut s'interpréter comme le traitement d'objets informationnels de grande taille ou de grandes collections d'objets

- Exemple :
  - SNCF, Facebook, Gmail, Twitter
  - en 15 ans, ces entreprises ont du faire face à un changement d'échelle passant de BDD relationnelles classiques à des outils Big Data



Variété

- La variété :
  - Fait référence à l'hétérogénéité des formats, des types, et de la qualité des informations
  - Est lié au fait que ces données peuvent présenter des formes complexes du fait qu'elles trouvent leurs origines dans :
    - des capteurs divers et variés (température, vitesse du vent, hygrométrie, tours/mn, luminosité...),
    - des messages échangés (e-mails, médias sociaux, échanges d'images, de vidéos, musique),
    - des textes, des publications en ligne (bibliothèques numériques, sites web, blogs, ...),
    - enregistrements de transactions d'achats, des plans numérisés, des annuaires, des informations issues des téléphones mobiles, etc.

- Ces nouveaux types de données nécessitent l'usage de technologies nouvelles pour analyser et recouper les données non structurées (mails, photos, conversations...) représentant une grande par des informations collectées.

Variété

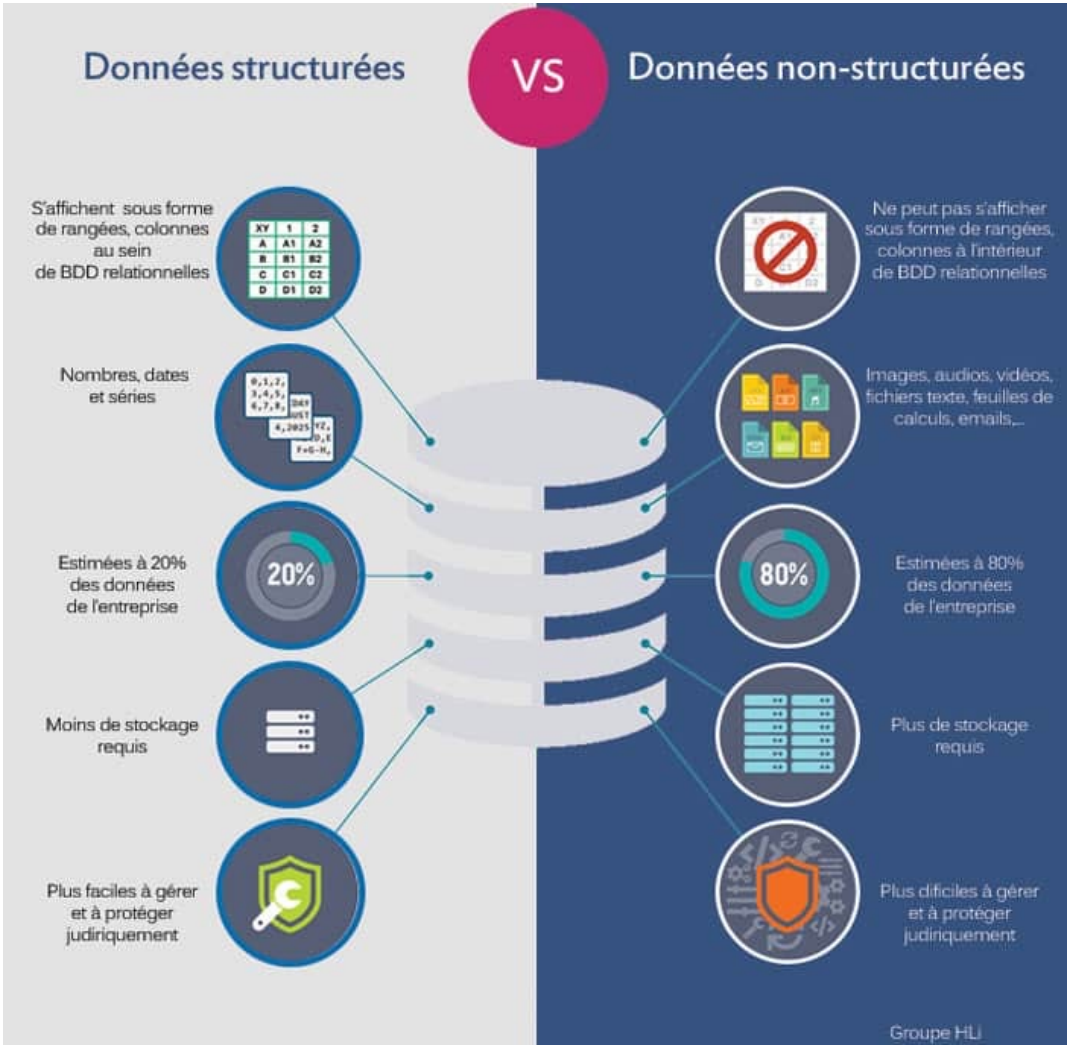
- Types de données stockées et générées par secteurs d'activités (cabinet McKinsey)

The type of data generated and stored varies by sector[1]

	Video	Image	Audio	Text/ numbers
Banking	Medium	Medium	Medium	High
Insurance	Low	Low	Low	High
Securities and investment services	Low	Low	Low	High
Discrete manufacturing	Medium	Medium	Low	High
Process manufacturing	Medium	Medium	Low	High
Retail	Medium	Low	Low	High
Wholesale	Low	Low	Low	High
Professional services	Medium	Medium	Medium	High
Consumer and recreational services	Medium	Low	Medium	Low
Health care	Low	High	Low	High
Transportation	Low	Low	Low	High
Communications and media[2]	High	Medium	High	High
Utilities	Low	Low	Low	High
Construction	Low	High	Low	Low
Resource industries	Medium	Medium	Low	High
Government	High	Medium	High	High
Education	High	Medium	High	Low

Penetration: High, Medium, Low

1 We compiled this heat map using units of data (in files or minutes of video) rather than bytes.
2 Video and audio are high in some subsectors.
SOURCE: McKinsey Global Institute analysis



Vélocité

- La vélocité :
  - Fait référence à l'aspect dynamique et/ou temporel des données, à leur délai d'actualisation et d'analyse
  - les données ne sont plus traitées, analysées, en différé, mais en temps réel ou quasi réel,
  - elles sont produites en flots continus, sur lesquels des décisions en temps réel peuvent être prises

- Cela concerne notamment les données issues de capteurs, nécessitant un traitement rapide pour une réaction en temps réel

- L'algorithme de Tiktok par exemple nécessite un fonctionnement en temps réel afin de faire ses recommendations aux utilisateurs

- Dans le cas de données de grande vélocité engendrant des volumes très importants, il n'est plus possible de les stocker en l'état, mais seulement de les analyser en flux (streaming), voire de les résumer.

Valeur

- La valeur :
  - fait référene à l'usage qui peut être fait de ces mégadonnées, de leur analyse, notamment d'un point de vue économique.

- L'analyse de ces mégadonnées demande des expertises en statistiques, en analyse de données, ainsi que des compétences pour l'interprétation de ces analyses.

- Selon une étude du McKinsey Global Institute :
  - en 2013, il manquait environ 150 000 personnes avec une expertise en analyse de Big Data
  - les termes de Data Scientist et de Data Science sont des métiers qui ont émergés liés à ce besoin
  - le système de santé américain pourrait créer 300 milliards de dollars de valeur par an dont deux tiers correspondrait à des réductions de coûts d'environ 8%.

Validité/Véracité

- La validité/véracité :
  - Fait référence à la qualité des données et/ou aux problèmes éthiques liés à leur utilisation
  - comprend les problèmes de valeurs aberrantes ou manquantes (ces problèmes pouvant être résolus par le volume de données),
  - fait référence aussi au niveau de la confiance que l'on peut avoir dans les données.

  $\Longrightarrow$ problème d'autant plus présent à l'heure des IA conversationnelles qui génèrent automatiquement leur propres contenus

- On parle d'hallucinations (ou délires) des robots conversationnels : ce sont des réponses qui semblent correctes mais qui ne le sont pas en réalité, qui proviennent de sources non fiables, ou qui sont déformées par des mensonges et des informations trompeuses. Les machines, aussi avancées soient-ells aujourd'hui, ne sont pas en mesure de reconnaître ou de comprendre ces réponses erronées. Ils rendent ce qu'ils ont appris d'internet par exemple

- Les résultats générés peuvent donc ne pas être objectifs, ni équitables, il faut donc demeurer attentifs et toujours verifier la qualité des données utilisées.

- On a des critères permettant de qualifier la qualité des données, mais dans le cas du Big Data, la vérification de la qualité est rendue difficile voire impossible du fait du volume, de la variété et de la vélocité spécifiques au Big Data.

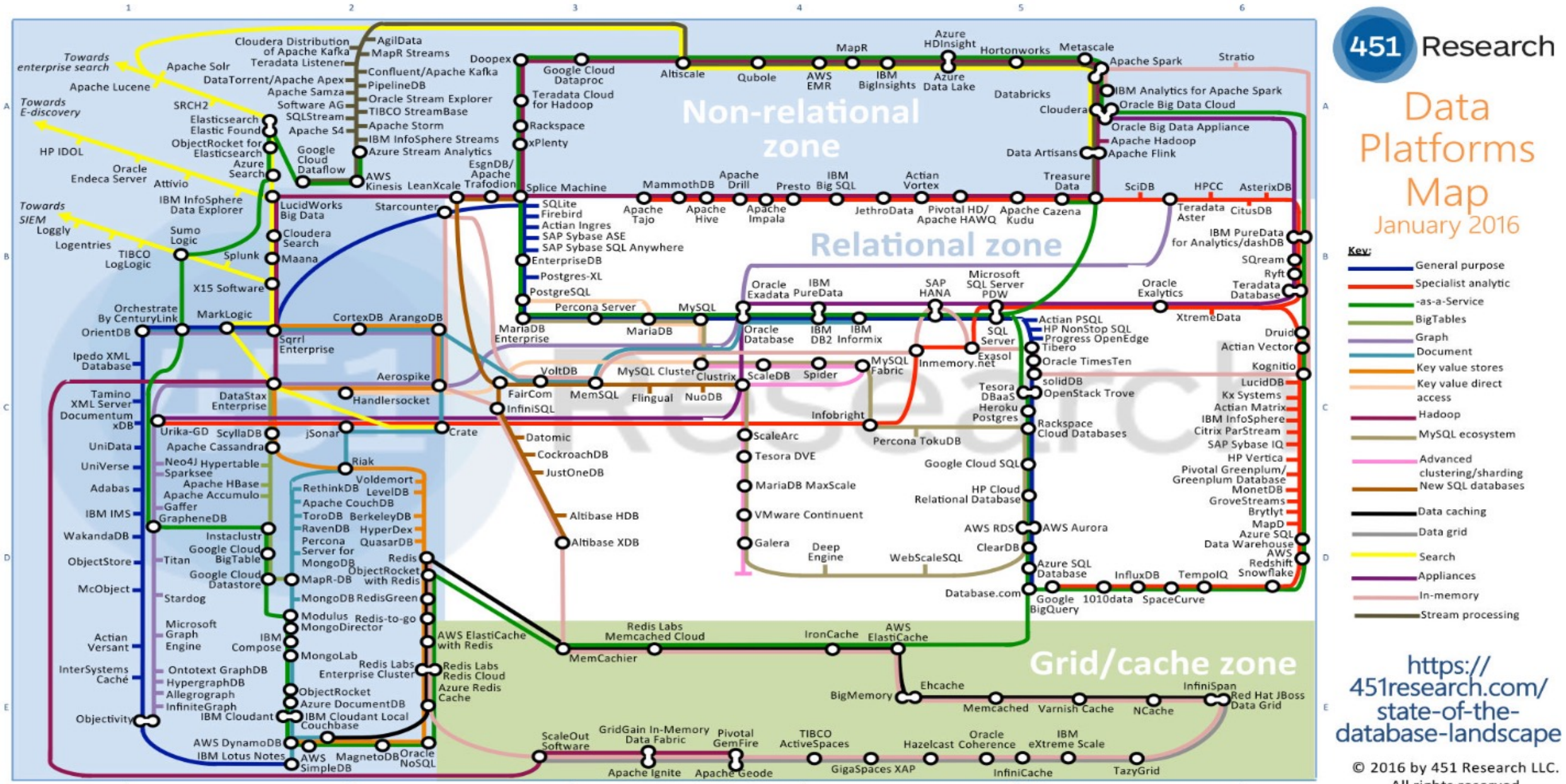
Définition

Mégadonnées (Big Data) =

- des données qui :
  - sont trop volumineuses
  - ou ayant une arrivée trop rapide
  - ou une variété trop grande
- pour :
  - permettre de les ranger directement dans des bases de données traditionnelles (Relationnelles)
  - ou de les traiter par les algorithmes actuels [1].
- et qui nécessitent l'utilisation d'outils spécifiques
  - La figure suivante présente différents outils qui ont été développés pour traiter du big data. Il en existe un grand nombre.

[1] MOTHE J., PITARCH Y., GAUSSIER E. Big Data: Le cas des systèmes d'information, Revue Ingénierie des Systèmes d'Information, Hermès Editeur, Vol. 19/3, (2014).

Outils


451 Research
Data Platforms Map
January 2016
Key:
General purpose
Specialist analytic
-as-a-Service
BigTables
Graph
Document
Key value stores
Key value direct access
Hadoop
MySQL ecosystem
Advanced clustering/sharding
New SQL databases
Data caching
Data grid
Search
Appliances
In-memory
Stream processing
Non-relational zone
Relational zone
Grid/cache zone
Towards enterprise search
Towards E-discovery
Towards SIEM
Apache Solr
Apache Lucene
HP IDOL
Oracle Endeca Server
Attivio
Loggly
Logentries
TIBCO LogLogic
Sumo Logic
Splunk
X15 Software
IBM InfoSphere Data Explorer
Cloudera Distribution of Apache Kafka
Teradata Listener
DataTorrent/Apache Apex
Apache Samza
Software AG
SRCH2
Elasticsearch
Elastic Found
ObjectRocket for Elasticsearch
Azure Search
SQLStream
Apache S4
Google Cloud Dataflow
AWS Kinesis
AgilData
MapR Streams
Confluent/Apache Kafka
PipelineDB
Oracle Stream Explorer
TIBCO StreamBase
Apache Storm
IBM InfoSphere Streams
Azure Stream Analytics
LucidWorks Big Data
Cloudera Search
Maana
Starcounter
LeanXcale
EsgnDB/ Apache Trafodion
Doopex
Google Cloud Dataproc
Teradata Cloud for Hadoop
Rackspace
xPlenty
Altiscale
Qubole
AWS EMR
MapR
IBM BigInsights
Azure HDInsight
Azure Data Lake
Hortonworks
Databricks
Metascale
Cloudera
Data Artisans
Apache Spark
Stratio
IBM Analytics for Apache Spark
Oracle Big Data Cloud
Oracle Big Data Appliance
Apache Hadoop
Apache Flink
Splice Machine
SQLite
Firebird
Actian Ingres
SAP Sybase ASE
SAP Sybase SQL Anywhere
EnterpriseDB
Postgres-XL
PostgreSQL
Percona Server
MariaDB Enterprise
MariaDB
MySQL
MammothDB
Apache Tajo
Apache Hive
Apache Drill
Apache Impala
Presto
IBM Big SQL
JethroData
Actian Vortex
Pivotal HD/ Apache HAWQ
Apache Kudu
Apache Cazena
Treasure Data
SciDB
HPCC
AsterixDB
Teradata Aster
CitusDB
IBM PureData for Analytics/dashDB
SQream
Ryft
Teradata Database
Oracle Exadata
IBM PureData
SAP HANA
Microsoft SQL Server PDW
Oracle Exalytics
XtremeData
Oracle Database
IBM DB2
IBM Informix
SQL Server
Exasol
Inmemory.net
Actian PSQL
HP NonStop SQL
Progress OpenEdge
Tibero
Oracle TimesTen
solidDB
OpenStack Trove
Tesora DBaaS
Heroku Postgres
Rackspace Cloud Databases
Google Cloud SQL
HP Cloud Relational Database
AWS RDS
AWS Aurora
ClearDB
Azure SQL Database
Database.com
Google BigQuery
1010data
InfluxDB
TempoIQ
SpaceCurve
Druid
Actian Vector
Kognitio
LucidDB
Kx Systems
Actian Matrix
IBM InfoSphere
Citrix ParStream
SAP Sybase IQ
HP Vertica
Pivotal Greenplum/ Greenplum Database
MonetDB
GroveStreams
Brytlyt
MapD
Azure SQL Data Warehouse
AWS Redshift
Snowflake
VoltDB
MySQL Cluster
Clustrix
ScaleDB
Spider
MySQL Fabric
FairCom
MemSQL
Flingual
NuoDB
InfiniSQL
Aerospike
Handlersocket
DataStax Enterprise
Crate
Infobright
Percona TokuDB
ScaleArc
Tesora DVE
MariaDB MaxScale
VMware Continuent
Galera
Deep Engine
WebScaleSQL
Altibase HDB
Altibase XDB
Datomic
CockroachDB
JustOneDB
Orchestrate By CenturyLink
OrientDB
MarkLogic
Sqrrl Enterprise
CortexDB
ArangoDB
Ipedo XML Database
Tamino XML Server
Documentum xDB
UniData
UniVerse
Adabas
IBM IMS
WakandaDB
ObjectStore
McObject
Actian Versant
InterSystems Caché
Objectivity
Urika-GD
ScyllaDB
Apache Cassandra
Neo4J
Hypertable
Sparksee
Apache HBase
Apache Accumulo
Gaffer
GrapheneDB
Instaclustr
Google Cloud BigTable
Titan
Google Cloud Datastore
Stardog
Microsoft Graph Engine
IBM Compose
Ontotext GraphDB
HypergraphDB
Allegrograph
InfiniteGraph
IBM Cloudant
AWS DynamoDB
IBM Lotus Notes
jSonar
Riak
RethinkDB
Voldemort
LevelDB
Apache CouchDB
ToroDB
BerkeleyDB
RavenDB
HyperDex
Percona Server for MongoDB
QuasarDB
Redis
ObjectRocket with Redis
MapR-DB
MongoDB
RedisGreen
Modulus
MongoDirector
Redis-to-go
MongoLab
Redis Labs Enterprise Cluster
ObjectRocket
Azure DocumentDB
IBM Cloudant Local
Couchbase
AWS SimpleDB
MagnetoDB
Oracle NoSQL
AWS ElastiCache with Redis
Redis Labs Redis Cloud
Azure Redis Cache
Redis Labs Memcached Cloud
MemCachier
IronCache
AWS ElastiCache
BigMemory
Ehcache
Memcached
Varnish Cache
NCache
InfiniSpan
Red Hat JBoss Data Grid
ScaleOut Software
GridGain In-Memory Data Fabric
Apache Ignite
Pivotal GemFire
Apache Geode
TIBCO ActiveSpaces
GigaSpaces XAP
Oracle Hazelcast Coherence
InfiniCache
IBM eXtreme Scale
TazyGrid
https://451research.com/state-of-the-database-landscape
© 2016 by 451 Research LLC.

Déroulé du cours

- 12h CM / 12h TP
- Rapporteurs
  - Les notes de cours sont mises à disposition par les étudiants (viennent à l'appui du support de cours)
- Objectifs
  - CM / TP :
    - Définir le Big Data
    - Introduction à la collecter, l'exploration, l'expérimentation, la mise en production, l'hébergement
    - Décrire les différentes étapes d'implémentation d'un système grande échelle
    - Designer un système grande échelle
- Évaluation
  - Note de participation (potentiellement sous forme de bonus)
  - Contrôle continu
  - Examen final
  - Rattrapage

Quiz

- Combien y a-t-il de téraoctets dans un exaoctet ?
  - 0.001
  - 10
  - 1000
  - 1000000
  - Aucun

Unité	Abbréviation	Taille (en octets)
1 octet		1
1 kilooctet	ko	10^3
1 mégaoctet	Mo	10^6
1 gigaoctet	Go	10^9
1 téraoctet	To	10^{12}
1 pétaoctet	Po	10^{15}
1 exaoctet	Eo	10^{18}
1 zétaoctet	Zo	10^{21}

- La BNF (Bibliothèque Nationale de France) contient (à la louche) 30 millions de livres. En admettant que, en moyenne, chaque livre contienne 300 pages et que chaque page contienne 2000 caractères, combien de données non compressées cela représente-t-il ?
  - Moins de 1 Go
  - Entre 1 Go et 1 To
  - Entre 1 To et 1 Po
  - Entre 1 Po et 1 Eo

Quiz

- Trouvez l'intrus :
  - Volume
  - Vélocité
  - Vendange
  - Variété

- Quand on ajoute des disques durs à un serveur pour augmenter sa capacité, il s'agit :
  - d'un passage à tabac
  - d'un passage à carte
  - d'un passage à l'échelle

- Qu'est ce qu'une grille de données ?

- Donnez 3 exemples de donnnée non-relationnelle

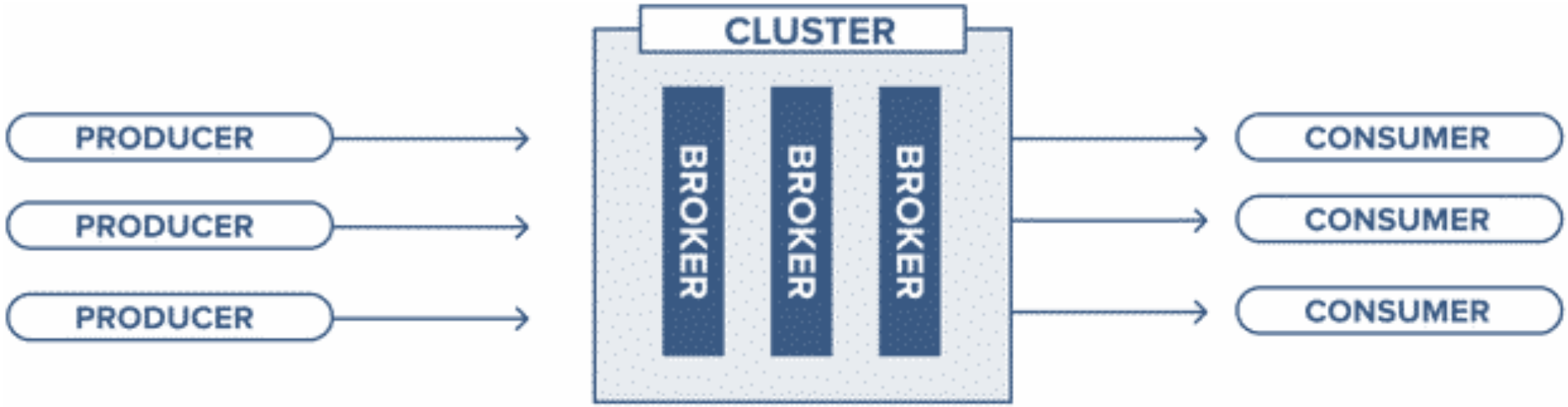
- Quels sont les 3V du big data ?

Outline

- Introduction
- Collecte des données
  - Vue d'ensemble
  - Ingestion
  - Stockage
  - Traitement
  - Orchestration
- Exploration
- Expérimentation
- Apprentissage grande échelle
- Mise en production et hébergement
- Q&A

Vue d'ensemble

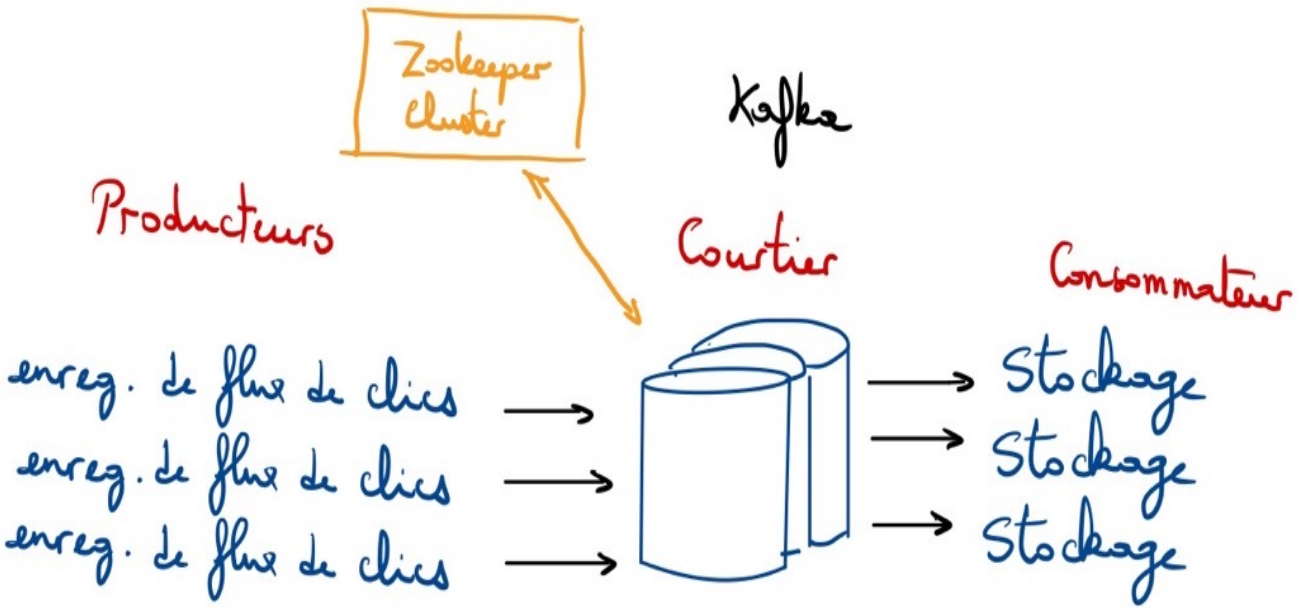
- La première étape pour l'exploitation de big data est la mise en place d'un mécanisme d'ingestion et de stockage des données.
- On utilise pour cela le système PRODUIRE/CONSOMMER encore connu sous le nom de PUBLIER/SOUSCRIRE



- Permet de collecter de grandes quantités de données en flux continu
  - flux de clics
  - changements en base de données
  - flux video et/ou audio (streaming, live...)
- Apache Kafka : outil open-source de type Publier/souscrire permettant la gestion de données en flux continu

Flux de clics

- Prenons le cas de l'ingestion de flux de clics à partir d'une application web



- 1 enregistrement :
  - date et heure
  - agent utilisateur
  - identifiant utilisateur
  - détail de l'évènement
  - information de cookies
  - adresse IP
- Chaque enregistrement est envoyé au courtier :
  - cluster de courtier
  - le leader du cluster s'occupe du bon routage des enregistrements
  - chaque enregistrement est taggé par un sujet (topic)
  - chaque partition est connecté à un consommateur (dans notre exemple, le consommateur est un espace de stockage)
- AWS Kinesis utilise un mécanisme similaire à la différence qu'au lieu de partitions, on parle de fragments (shards)

Changements en base de données

- Dans le schéma précédent, on a parlé d'enregistrements de clics mais il existe d'autres types de données collectables en flux continu avec des producteurs de nature diverse :
  - robots crawler qui parcourent internet et envoient automatiquement et en continu des informations aux courtiers
  - flux RSS
  - base de données

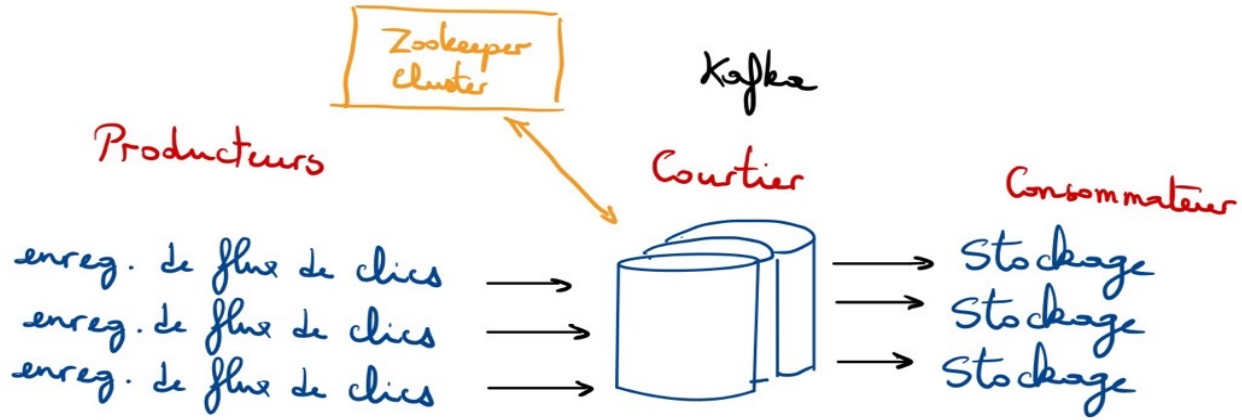
- Cas de l'ingestion de changements en base de données
  - utilisé lorsqu'on a besoin de tenir des enregistrment de changements dans des bases de données
  - Exple BD Netflix :
    - Information sur l'identité de l'utilisateur (identifiant, abonné actif ou pas, date de renouvellement...)
    - Besoin de prédire si une personne donnée va renouveller son abonnement en fonction de son activité
    - Besoin de l'historique du statut d'abonné de la personne et de son activité
    - est ce que le désabonnement fait suite à une baisse de l'activité sur la plateforme ?
  - Outil de CDC (change data capture) pour capturer les changements de la BD
  - La plupart des systèmes de gestion de base de données (SGBD) actuels proposent des fonctionnalités de CDC

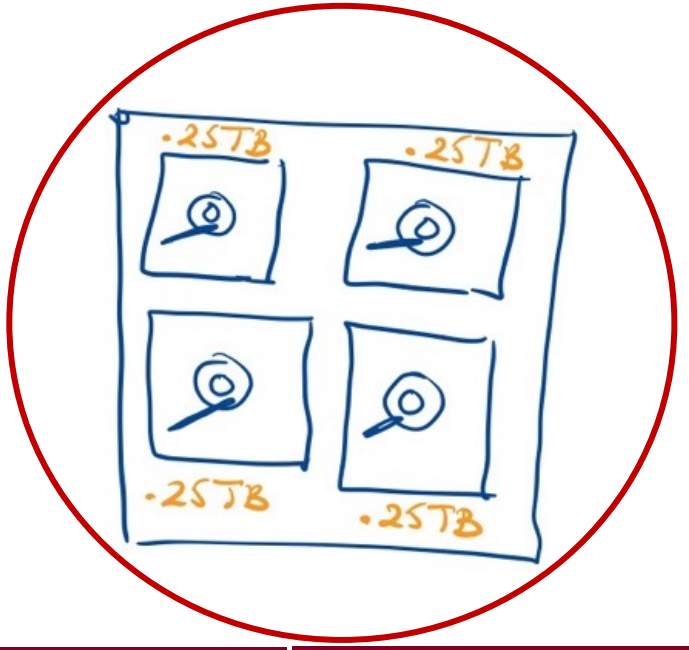
Flux vidéo

- Les flux vidéos comprennent les contenus provenant de :
  - caméras de circulation
  - caméras de sécurité
  - services de diffusion de live vidéo
- HLS (HTTP live stream) : protocole de streaming basé sur HTTP
  - prend le format MP4
  - le découpe en segments
  - et envoie les segments sur http
- Dans le modèle PRODUCTEUR/CONSOMMATEUR :
  - le producteur est la caméra qui filme
  - le film est envoyé à un collecteur (étape intermédiaire au courtier)
  - le collecteur désassemble la vidéo image par image et chaque image est compressée
  - chaque image est ensuite envoyée au courtier pour être ensuite transmise à un consommateur
- Ingestion en continu vs ingestion par lots
  - au lieu d'ingérer des fragments de données unité par unité, on fait un envoi plus global (copie de la base entière plutôt que des changements de lignes)
- Le choix du type d'ingestion dépend de la taille des données à traiter, du rythme d'arrivée des données au courtier, du format de données, de la disponibilité et de la tolérance aux pannes du courtier

Stockage des données

- On va maintenant voir la partie stockage dans notre schéma précédent



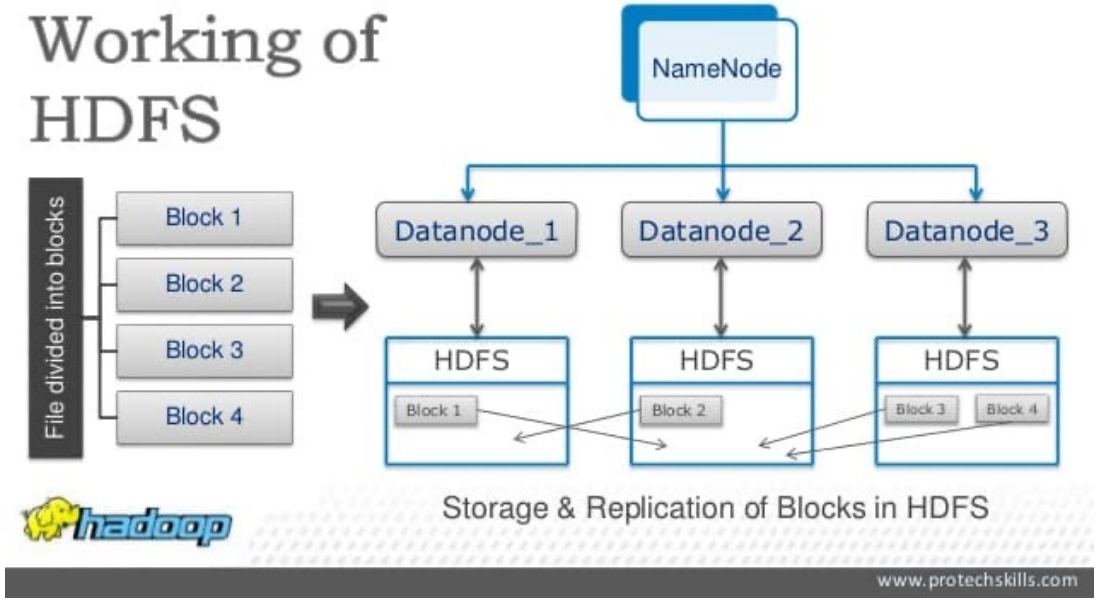


- Considérons 1 serveur :
  - 4 disques durs
  - 100Mo/s de capacité de lecture écriture
  - 40mn pour lire/écrire 1To de data
- Avec un cluster de 10 serveurs aux mêmes capacités :
  - 40 disques durs
  - 15mn pour lire/écrire 1To de data
- Le passage à l'échelle permet de gagner en temps et en sécurité

Framework HADOOP

- Pour stocker de façon distribuée, on peut avoir recours à un système de gestion de fichiers distribués.
- Framework utilisé : Hadoop basé sur HDFS (Hadoop Distributed File System)

- Chaque machine comprend un ensemble de noeud de données, chaque noeud est relié au HDFS.
- Le HDFS permet de savoir quelle donnée est stockée par quel noeud.



- Le namenode :
  - détermine les datanode sur lesquels les blocs de fichiers peuvent être stockés
  - garde une trace du fichier original, des blocs, et des nœuds où ils sont stockés
  - stocke une carte clé valeur où la clé est le bolc et les valeurs, tous les nœuds sur lesquels sont stockés le bloc
- en cas de besoin, HDFS reconstitue le fichier original à partir des blocs et le renvoie à l'utilisateur

Traitement des données

- Après l'ingestion et le stockage, on passe au traitement des flux de données
  - consiste notamment en l'aggrégation des données et la construction des attributs du jeu de données
- Exemple : Prédire si un utilisateur Netflix va annuler ou non son abonnement

- On considère 2 pages d'ingestion de données :
  - la page d'accueil avec recommandations
  - la page de recherche
- L'agrégation des enregistrement de flux de clics :
  - compter le nombre de recherche par utilisateur
  - compter le nombre de sélection de recommandations par utilisateur
  - transformer les dates des enregistrements de flux de clics (passer de la date complète au mois de l'année par exemple)
- 3 types de traitement des données
  - agrégation
  - jointure
  - transformation

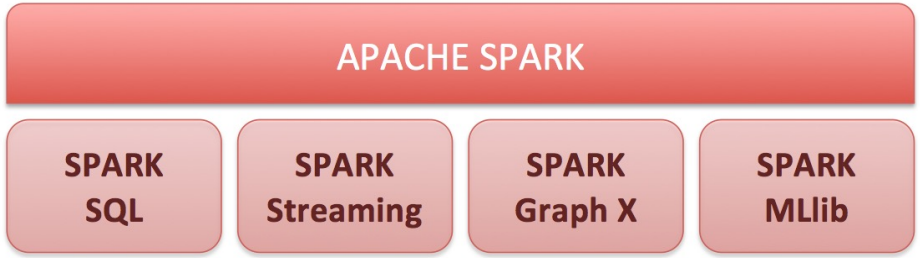
- Lorsque les flux de données arrivent au consommateur (client HDFS par exemple), comment celui-ci opère t-il les traitements ?

Mise à l'échelle des traitements

- Les requêtes arrivant au consommateur peuvent être très nombreuses dans le cas du big data
- Exemple :
  - 50000 enregistrements /seconde de flux Netflix arrivent au serveur HDFS
  - pour 1 processeur à 2 coeurs compter plus de 10h de traitement pour 3 mois de données
- Quelle solution ?
  - mise à l'échelle des processeurs
  - chaque de noeud de données distribué envoie ses calculs à un cluster de processeurs distants
- Comment savoir quel cluster a déjà reçu des traitements à effectuer
- Comment distinguer les étapes d'un traitement particulier ?
  - par exemple : effectuer une transformation puis une jointure
- Besoin d'utiliser un outil d'orchestration

Orchestration des traitements

- Apache Spark : framework populaire de calcul distribué
  - ensemble d'outils permettant de réaliser une lecture des données au niveau d'un cluster (grappe de serveurs sur un réseau), et d'effectue toutes les opérations d'analyse nécessaires, puis de réécrire les résultats à ce même niveau.



- Spark SQL
  - exécuter des requêtes en langage SQL pour charger et transformer des données. Le langage SQL est issu des bases de données relationnelles, mais dans Spark, il peut être utilisé pour traiter n'importe quelles données, quel que soit leur format d'origine.
- Spark Streaming
  - traitement des données de flux comme Kafka ou Kinesis
- Spark Graph X
  - traiter les informations issues de graphes.
- Spark Mllib
  - bibliothèque d'apprentissage automatique, qui contient tous les algorithmes et utilitaires d'apprentissage classiques, comme la classification, la régression, le clustering...

Source : https://fr.wikipedia.org/wiki/Apache_Spark

Quiz

- Indiquez l'affirmation correcte.
  - Hadoop est un environnement idéal pour extraire et transformer de petits volumes de données.
  - Hadoop stocke les données dans HDFS et prend en charge la compression/décompression des données.
  - Aucune de ces réponses

- Sous quelle licence Hadoop est-il distribué ?
  - Licence Apache 2.0
  - Mozilla Public License
  - Shareware
  - Commerciale

- Lequel des éléments suivants est produit par Hadoop ?
  - Système de fichiers distribués
  - Service de messages Java
  - JAX-RS
  - Système de gestion de bases de données relationnelles

Quiz

- L'analyse des données volumineuses porte sur les données non structurées, pour lesquelles aucun modèle spécifique n'est défini.
  - Vrai
  - Faux

- L'interprétation des données se réfère à _____.
  - Processus consistant à donner un sens aux données
  - Convertir le texte en informations pertinentes
  - Conclusion efficace
  - Tous les éléments mentionnés ci-dessus

- Le big data concerne des informations de grand volume, de grande vitesse et de grande variété:
  - Vrai
  - Faux

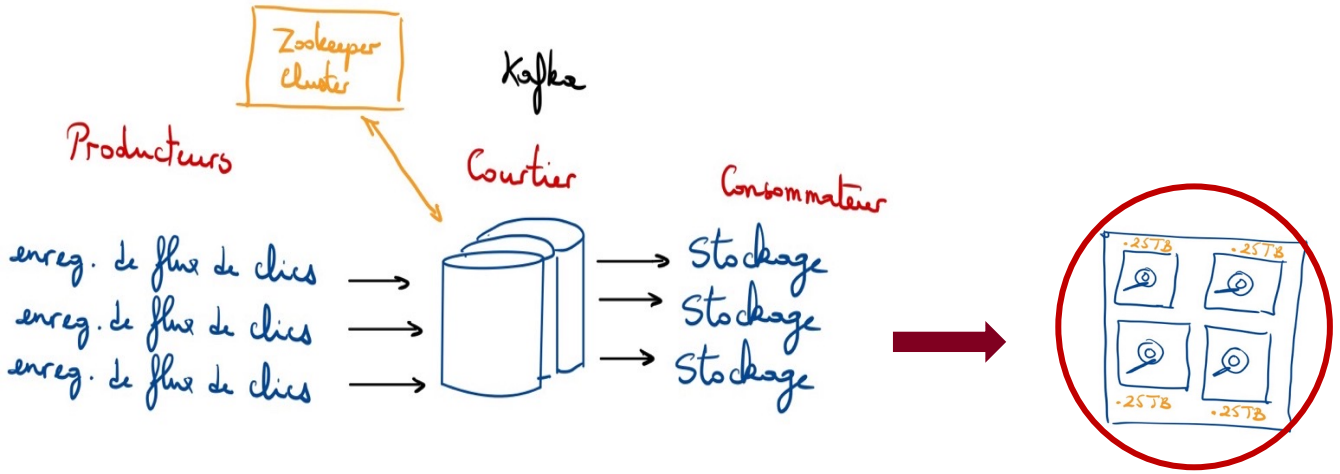
- Quelles sont les 3 composantes du modèle d'ingestion de données utilisé par le logiciel Apache Kafka ?

Outline

- Introduction
- Collecte des données
- Exploration
  - Le travail d'exploration
  - Les espaces de travail
- Expérimentation
- Apprentissage grande échelle
- Mise en production et hébergement
- Q&A

Le travail d'exploration des données

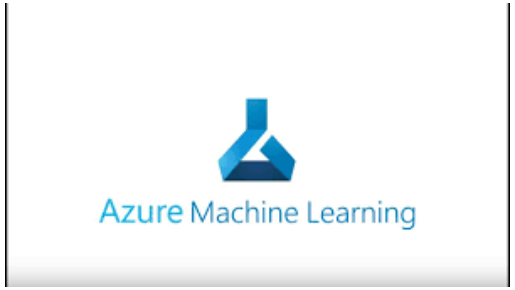
- Cas d'usage : Prédire le renouvellement ou la résiliation d'un abonnement Netflix en fonction des traces utilisateur



- Objectif des étapes précédentes : prérequis afin de pouvoir effectuer le 'travail' de data science à proprement parlé, ie explorer les attributs et les colonnes cibles de nos jeux de données, ainsi que choisir les modèles appropriés
- En quoi consiste précisément ce 'travail' ?
  - observer les attributs et les colonnes cibles
  - identifier le ou les modèles que l'on souhaite expérimenter – il ne faut pas se priver d'explorer
  - vérifier l'exhaustivité des données
  - vérifier la stabilité des données
  - vérifié la validité des données

Espaces de travail

- Tout cela se fait dans un espace de travail dédié
- L'un des plus populaires est le projet Jupyter[1]
- Les notebooks existent en version cloud :
  - Amazon propose SageMaker Studio
  - Google propose Google Colab
  - Microsoft propose AzureML



- Si vous ne souhaitez pas utiliser les espaces de travail proposer dans le cloud, il y a aussi la possibilité d'utiliser votre environnement de développement habituel (en anglais, IDE – Integrated Development Environment)
  - VSCode
  - Netbeans
  - Notepad++
  - Sublime Text ...

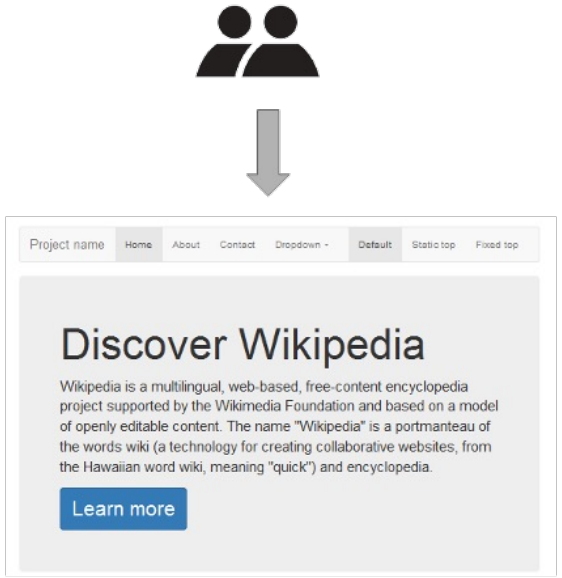
Source : https://en.wikipedia.org/wiki/Project_Jupyter

Outline

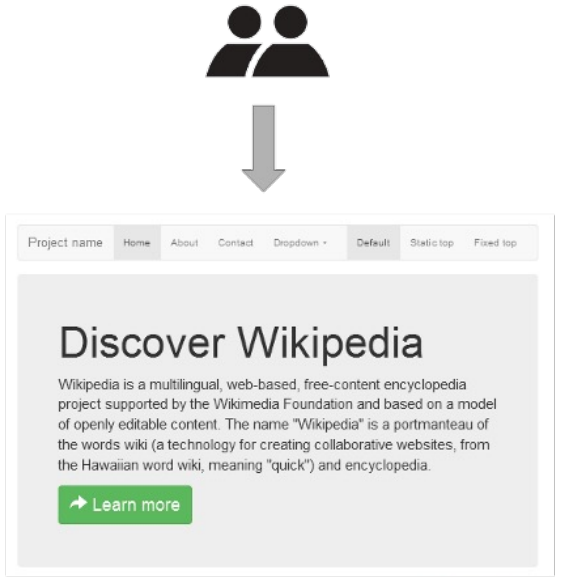
- Introduction
- Collecte des données
- Exploration
- Expérimentation
  - A/B testing
  - Modèle d'hypothèse de test
  - Approche fréquentiste
- Apprentissage grande échelle
- Mise en production et hébergement
- Q&A

Définition

- Dans cette section, on va parler de méthodes pour tester le modèle auprès d'utilisateurs réels
- Test A/B (en anglais, A/B testing) : technique d'inférence probabiliste utilisée notamment en marketing afin de tester différentes versions d'un produit en ligne auprès d'utilisateurs réels afin d'en mesurer la plus effficace.
- On pose une hypothèse pour décrire comment remplacer une expérience utilisateur par une autre et comment l'utilisateur va réagir face à ces changements



Clic rate : 52 %

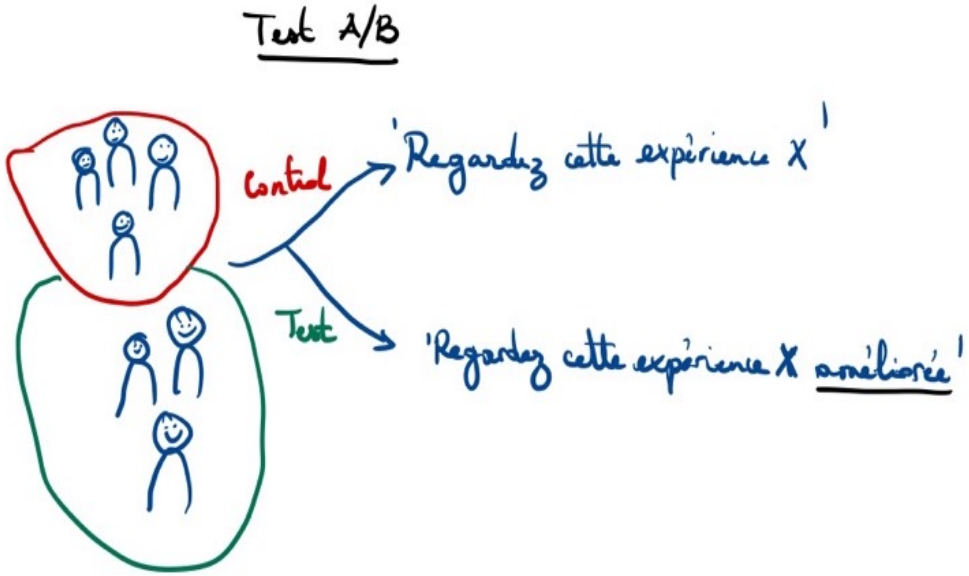


72 %

- On décrit précisément :
  - l'expérience utilisateur à remplacer
  - les variables dépendantes
  - la direction du changement de chaque variable dépendante : est ce qu'elle va augmenter, baisser ou rester stable
  - les participants à l'expérience : caractéristiques des groupes utilisateurs choisis
  - les interfaces utilisateurs ou versions du logiciel concernées par le test

Modèle d'hypothèse de test

- Lorsqu'on pose l'hypothèse, toujours respecter la forme :
  - Si ... *(on effectue tel changement)*, alors ... *(on aura tel effet)*
- Exemple : Si on remplace X par Y pour cet ensemble d'utilisateurs, alors [a, b, c...] [augmentera/baissera] et [les invariants] ne changeront pas.
- On vérifie l'hypothèse en procédant au test AB qui consiste à comparer un groupe test à un groupe de contrôle



- On sépare l'ensemble des utilisateurs en 2 groupes :
  - un groupe de test/traitement et un groupe de contrôle
  - le groupe de traitement reçoit le changement et le groupe de contrôle ne reçoit aucun changement (teste la fonctionnalité précédente)
  - comparer les réactions des 2 groupes
- On détermine si l'expérience de traitement remplace celle de contrôle si on arrive à mesurer des différences significatives entre les 2 groupes allant dans le sens de notre hypothèse

Approche fréquentiste

- Pour appliquer l'approche fréquentiste, on a besoin des valeurs suivantes :
  - d'une référence noté p_1, ce sont les fréquences observées sur la plateforme avant changement
  - d'un seuil de signifiance pratique, c'est le changement minimal souhaité
  - de la puissance noté β, c'est le nombre de fois que le seuil est trouvé dans l'hypothèse qu'il existe
  - de la signifiance noté α, c'est le nombre de fois que le seuil est trouvé dans l'hypothèse qu'il n'existe pas
  - de la taille n des groupes

- La taille n de chaque groupe est donnée par la formule

$$n = \frac{\left(z_\alpha \sqrt{2p_{1(1-p_2)}} + z_\beta \sqrt{p_1(1-p_1)+p_2(1-p_2)}\right)^2}{(p_2-p_1)^2}$$

- On peut exprimer le test autrement en terme d'hypothèse nulle
  - c'est le cas où l'hypothèse initiale est incorrecte
  - accepter l'hypothèse nulle revient à dire qu'il n'y a pas de différence significative entre les 2 groupes
  - rejeter l'hypothèse nulle revient à dire qu'il y a une différence significative entre les 2 groupes

Hypothèse nulle et p-value

- Comment savoir que la différence entre le groupe de test et de contrôle est significative ?
  - on calcule la p-value à laquelle on compare la valeur seuil de 0.05 en général
- Exemple :

	Probabilité de clics	Nombre d'utilisateurs
Groupe de contrôle	7%	1062
Groupe de test	8%	982

- Calcul de la p-value :
- r-score $= \frac{p_1 n_1 + p_2 n_2}{n_1 + n_2}$; r-score $= 7.48$
- z-score $= \frac{\bar{p}_2 - \bar{p}_1}{\sqrt{r(1-r)} \cdot (\frac{1}{n_1} + \frac{1}{n_2})}$; z-score $= 0.858$
- L'observation du tableau des z-scores unilatéraux donne une p-value $= 0.1949$ ce qui est supérieure à notre seuil de 0.05 => l'hypothèse nulle ne peut pas être rejetée.
- Cependant, vous verrez qu'en multipliant le nombre de participants dans les 2 groupes par 10, on obtient une p-value $= 0.004$ qui est inférieure à 0.05 => l'hypothèse nulle peut être rejetée dans ce cas.

Intervalle de confiance

- L'intervalle de confiance IC pour la limite inférieure de chaque probabilité à 95% est donné par la formule :
  - $IC_i = p_i - z_{95\%}\sqrt{\frac{p_i(1-pi)}{n_i}}$
- $IC_1 = p_1 - z_{95\%}\sqrt{\frac{p_1(1-p_1)}{n_1}} = 0.07 - 1.645\sqrt{\frac{0.0651}{1062}} = -\ 1.28\%$
- $IC2 = 0.08 - 1.645\sqrt{\frac{0.0736}{982}} = -\ 1.42\%$
- Exemple :

	Probabilité de clics	Nombre d'utilisateurs	Avec IC
Groupe de contrôle	7%	1062	5.72 (-1.28)%
Groupe de test	8%	982	6.58 (-1.42)%

- => Ces intervalles de confiance signifient que si nous devions réaliser la même expérience 100 fois, environ 95 de ces cas, soit 95 %, donneraient lieu à une probabilité de clic au moins égale à ces valeurs.

Outline

- Introduction
- Collecte des données
- Exploration
- Expérimentation
- Apprentissage grande échelle
  - Focus sur MapReduce et Hadoop
  - Limites de Hadoop
  - Spark et RDD
  - Adaptation de quelques modèles de ML pour le big data
  - Quiz
- Mise en production et hébergement
- Q&A

Rappel sur l'analyse distribuée

- L'analyse distribuée consiste à assigner à chaque noeud d'un cluster une tâche correspondant à l'analyse d'une portion temporelle et spatiale des données.

- Calcul parallèle : exécution simultanée de différents *threads* partageant une mémoire commune qui leur permet de se synchroniser entre eux
- Calcul distribué : exécution des calculs sur des noeuds distants, autonomes et ne partageant pas les mêmes ressources

  - Passage à l'échelle vertical vs passage à l'échelle horizontal

  - Augmenter la puissance du processeur vs augmenter le nombre de noeuds

- Le calcul distribué vient résoudre certaines limites du calcul parallèle :
  - résistance aux pannes : lorsqu'un nœud du cluster subit une panne, il suffit d'affecter la tâche qu'il était en train de traiter à un autre nœud, alors que dans le modèle parallèle la machine sur laquelle le calcul est exécuté constitue un point unique de défaillance.
  - ralentissement de la loi de Moore

MapReduce

- MapReduce offre un cadre générique au calcul distribué
- Le principe consiste à diviser pour distribuer pour règner :
  - découper les données en sous-ensembles de petite taille (lots ou fragments), affecter chaque lot à une machine du cluster permettant ainsi leur traitement parallèle, agréger les résultats intermédiaires obtenus pour chaque lot pour construire le résultat final
- Basé sur deux opérateurs génériques utilisé en programmation fonctionnelle : map et reduce
  - map : appliquer une même fonction à tous les éléments d'une liste

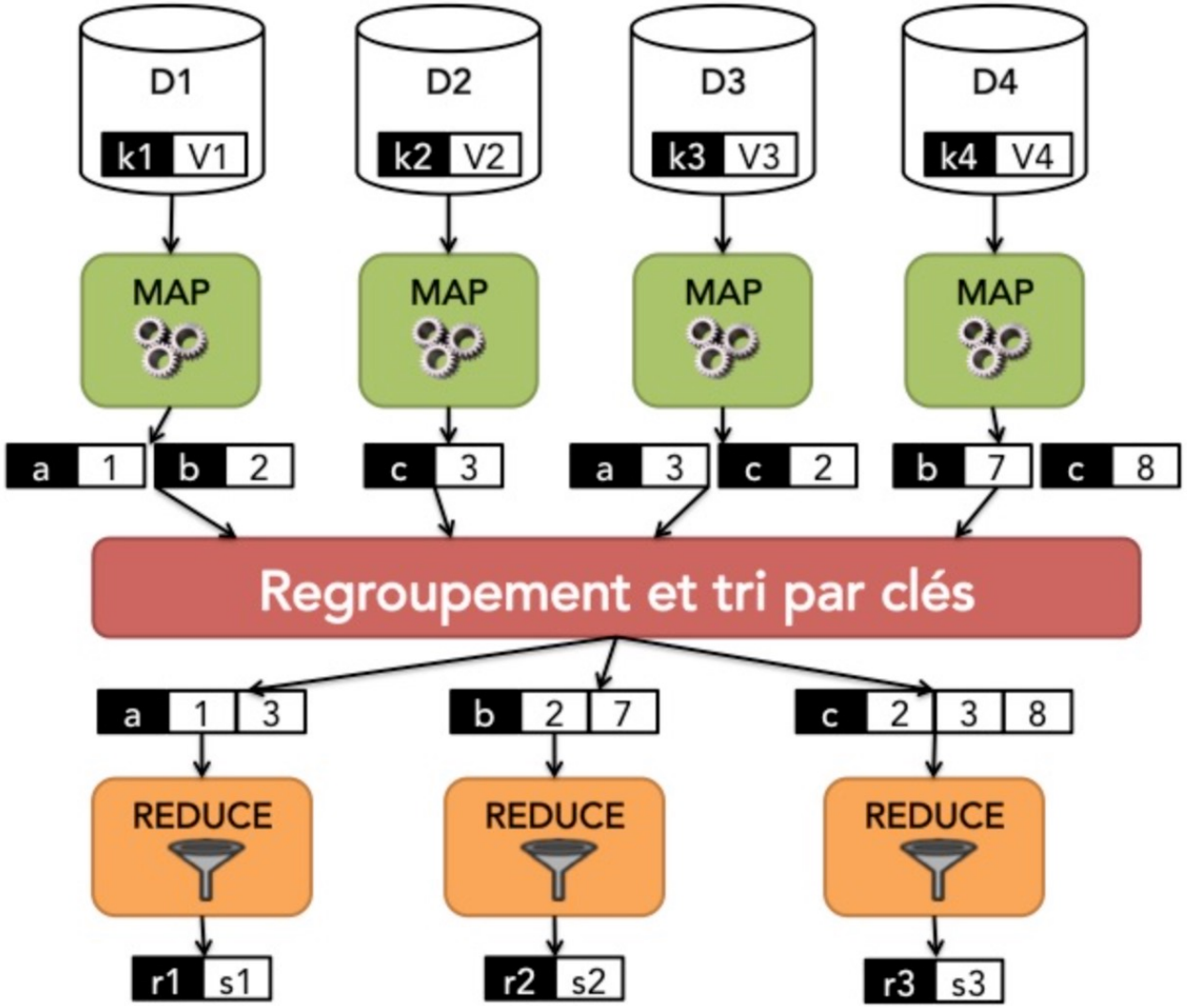
```
1 map(f)[x0, ..., xn] = [f(x0, ...,f(xn)]
2 map(*4)[2, 3, 6] = [8, 12, 24]
```

  - reduce : appliquer une fonction récursivement à une liste et retourne un seul résultat

```
1 reduce(f)[x0, ..., xn] = f(x0, f(x1, f(x2, ...)))
2 reduce(+)[2, 3, 6] = (2 + (3 + 6)) = 11
```

- La combinaison de ces 2 opérateurs permet de modéliser de nombreux problèmes
- MapReduce opère sur des paires. C'est à dire que l'ensemble des données est représenté sous forme de paires (clé, valeur) comme dans les tables d'association

MapReduce : Fonctionnement général



- Les données sont découpées en plusieurs lots ou sous-ensembles.
- Dans l'étape MAP, l'opération map est appliquée à chaque lot et transforme la paire(clé, valeur) représentant le lot en une liste de nouvelles paires (clé, valeur)
- Les résultats intermédiaires sont regroupés et triés par clé. C'est l'étape de SHUFFLE and SORT.
- Dans l'étape REDUCE, l'opération reduce est appliquée à chaque clé et renvoie donc pour chaque clé une valeur unique.

=> Le rôle des développeurs d'application distribuées, est de penser en 'mode MapReduce'

Hadoop

- Pour le traitement de données big data, nécessité d'associer MapReduce à une infrastructure logicielle dédiée et prenant en charge les enjeux du calcul distribué :
  - scalabilité : adaptation de la puissance au besoin
  - localité des données : optimisation des transferts de disques
  - résistance aux pannes

- Hadoop, implémentation de référence dans le monde open-spource d'une telle architecture
- Le socle technique d'Hadoop est composé :
  - De toute l'architecture support nécessaire pour l'orchestration de MapReduce, c'est-à-dire :
    - l'ordonnancement des traitements,
    - la localisation des fichiers,
    - la distribution de l'exécution.
  - D'un système de fichiers HDFS qui est :
    - Distribué : les données sont réparties sur les machines du cluster.
    - Répliqué : en cas de panne, aucune donnée n'est perdue.
    - Optimisé pour la colocalisation des données et des traitements.

Les limites de Hadoop

- A l'usage, Hadoop MapReduce présente deux inconvénients majeurs :
  - l'écriture sur disque :
    - les écritures et lectures sur le disque sont coûteuses en temps.
  - le nombre d'opérateurs :
    - il est difficile d'exprimer des opérations complexes en n'utilisant que cet ensemble de deux opérations.

- Apache Spark est une alternative à Hadoop MapReduce pour le calcul distribué qui vise à résoudre ces deux problèmes.
  - il écrit en RAM et non sur disque => plus rapide
  - il propose plus opérateurs regroupés en 2 groups : les transformations et les actions

Technologie	Latence (s)	Taux de transfert (Go/s)
Disque dur	10^{-2}	0.15
SSD	10^{-4}	0.5
DDR3 SDRAM	10^{-8}	15

Spark et RDD

- A l'instar d'Hadoop qui est basé sur le système de fichier HDFS, Spark est basé sur les RDD (resilient distributed dataset)
- RDD est la représentation par Spark d'un ensemble de données distribué dans la RAM (la mémoire) d'un cluster de plusieurs machines.
- Equivalent au Dataframe de pandas, RDD est un objet constitué d'une collection d'éléments :
  - listes de tuples,
  - des dictionnaires,
  - des listes, etc.

- Nous pouvons y charger un ensemble de données, puis exécuter n'importe laquelle des méthodes accessibles à cet objet.

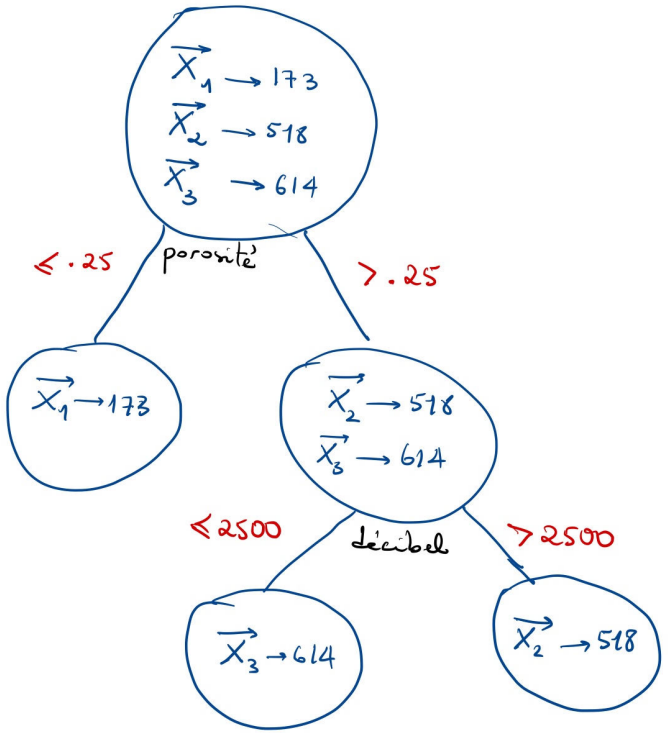
- Spark est écrit en Scala et prévu pour être compile en bytecode Java executable sur la JVM.
- PySpark est une boite à outils qui permet d'interfacer les RDD avec Python. Grâce à une bibliothèque appelée Py4J, Python peut s'interfacer avec des objets Java (dans notre cas, des RDD). Py4J est également l'un des outils qui permet à PySpark de fonctionner.

Spark et RDD

- Ainsi, vous allez utilisé Spark au lieu de Python classique lorsque vous serez amenés à travailler sur des données massives
- Spark offre de nombreux avantages par rapport à Python :
  - l'abstraction RDD permet d'exécuter localement les calculs, Spark simule la distribution en découpant automatiquement la RAM en partitions
  - l'évaluation paresseuse : attente qu'une action soit demandée avant d'exécuter effectivement la logique du code.
- L'avantage de l'évaluation paresseuse est que nous pouvons créer une file d'attente et laisser Spark optimiser le flux de travail en arrière plan
- De quoi parle t-on lorsqu'on dit que Spark attend une action ?
- En effet, il y a 2 types de méthodes dans Spark :
  - Les transformations — map(), reduceByKey()
  - Les actions — take(), reduce(), saveAsTextFile(), collect()
- Les transformations sont des opérations paresseuses et renvoient des pointeurs
- Les actions renvoient des valeurs

Arbres de décision

Comment adapter les modèles d'apprentissage machine en contexte de données massives ?



- Considérons un arbre de décision.
  - 100 millions d'exemples
  - 200 attributs
  - 200 points de séparation possibles
- Algorithme AAD
  - évaluer chaque attribut pour savoir s'il constitue un point de séparation pertinent
  - stocker cet attribut et le seuil de séparation
  - traiter de façon récursive la branche gauche puis la branche droite de l'arbre jusqu'à convergence
- Problème : Ne fait pas usage de l'analyse parallèle
- Solution :
  - paralléliser l'analyse de chaque attribut à chaque point de séparation
  - paralléliser l'analyse de tous les nœuds d'un niveau particulier
- Pour les forêts aléatoires, il est possible de paralléliser la construction de chaque arbre de décision les constituant car ceux-ci sont indépendants les uns des autres.

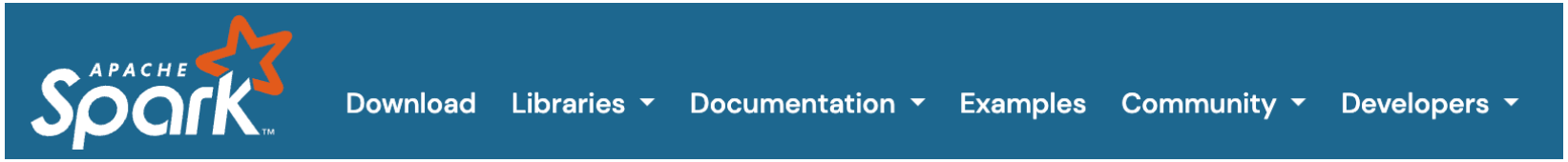
Régression logistique

- Rappel de la formule de la régression logistique

$$h_\beta(x) = = \frac{1}{1+ e^{-(\beta_1 x+\beta_0)}}$$

- Le calcul du gradient tel que présenté précédemment implique que les calculs sont réalisés sur une seule machine
- Si nous disposons de plusieurs machines :
  - séparer les données en mini-lots
  - envoyer différents lots à chaque machine
  - chaque machine calcule un gradient partiel
  - les gradients partiels sont transmis à une machine du cluster qui est chargée de les agréger
- Parralléliser permet de converger plus rapidement vers des valeurs optimales de paramètres
  - sans cela, l'intérêt même d'utiliser de l'apprentissage machine serait nul
  - Exemple : modèle de prédiction boursier
    - s'il faut 3 jours pour entrainer un modèle dont on a besoin pour le lendemain, le modèle ne sera jamais utilisé

Spark et la bibliothèque MLlib



MLlib is Apache Spark's scalable machine learning library.

- Quelques modèles pris en charge[1] :
  - régression linéaire et logistique
  - SVM
  - arbres de décision et forêts aléatoires
  - modèles naïfs de bayes
  - filtrage collaboratifs
  - k-means
  - décomposition en valeurs singulières
  - analyse en composantes principales ...
- Fonctionnalités :
  - d'analyse de performance des modèles
  - techniques de normalisation et de transformation des données ...

1 https://spark.apache.org/mllib/

Quiz

- Spark est développé dans quel langage?
  - R
  - Python
  - Scala
  - Java
  - Toutes les réponses sont varies

- Considérons une ingestion de données en flux, de quelles sources peuvent provenir les données?
  - Google Dataflow
  - Kafka
  - Kinesis
  - Azure Stream Analytics
  - Toutes les réponses sont vraies

- Apache Spark dispose d'API en _____
  - Python
  - Scala
  - Java
  - Toutes les réponses sont vraies

Quiz

- Indiquez l'affirmation correcte.
  - MapReduce essaie de placer les données et les calculs de façon aussi proche que possible.
  - La tâche Map dans MapReduce est exécutée à l'aide de la fonction Mapper().
  - La tâche Reduce de MapReduce est exécutée à l'aide de la fonction Map().
  - Aucune de ces réponses

- Bien que le framework Hadoop soit implémenté en Java, les applications MapReduce n'ont pas besoin d'être écrites en _____.
  - Java
  - C
  - C#
  - Aucune de ces réponses

- Dans les environnements Big Data, la variété des données comprend _____
  - multiples formats et types de données
  - Comprend des données structurées sous la forme de transactions financières.
  - des données semi-structurées sous forme d'e-mails et des données non structurées sous forme d'images.
  - Toutes les réponses ci-dessus

Quiz

- Indiquez l'affirmation incorrecte.
  - Spark est destiné à remplacer, la pile Hadoop
  - Spark a été conçu pour lire et écrire des données depuis et vers HDFS, ainsi que d'autres systèmes de stockage.
  - Les utilisateurs d'Hadoop qui ont déjà déployé ou qui prévoient de déployer Hadoop Yarn peuvent simplement exécuter Spark sur Yarn.
  - Aucune de ces réponses

- L'abstraction de base de Spark est _____.
  - Dstream
  - RDD
  - Variable partagée
  - Aucun des éléments ci-dessus

- Lequel des éléments suivants n'est pas une caractéristique de Spark ?
  - Prise en charge du calcul en mémoire
  - Tolérance aux pannes
  - Il est rentable
  - Compatible avec d'autres systèmes de stockage de fichiers

Outline

- Introduction
- Collecte des données
- Exploration
- Expérimentation
- Apprentissage grande échelle
- Mise en production et hébergement
- Q&A

Mise en production

- L'étape de mise en production formalise et implémente le déploiement des changements approuvés et validés sur un ou plusieurs éléments de notre architecture (ie le modèle, les données attendues par ce modèle ou encore l'environnement logiciel)

- Les étapes pour déployer un modèle final sont nombreuses :
  - ingestion
  - stockage
  - traitement
  - orchestration
  - exploration
  - experimentation ...

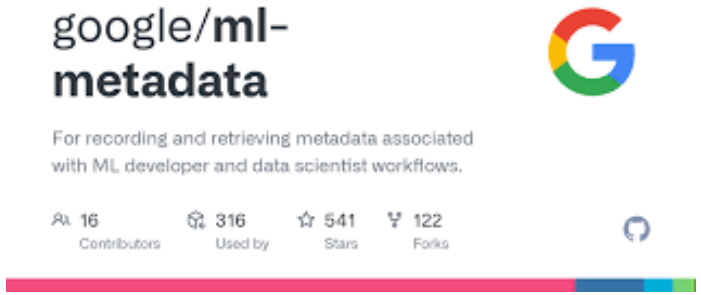
- Des erreurs peuvent survenir à chacune de ces étapes
  - dans les grandes organisations, le risque de survenue d'erreurs est quasiment de 100% au cours d'une année

  => nécessité de réduire autant que possible les risques d'erreurs humaine

- On pose certaines actions afin de minimiser le risque sur chacune des étapes de déploiement

Mise en production

- Mettre en place (au moins) 2 environnements de travail :
  - un environnement de développement
  - un environnement de production
- Pendant l'ingestion des données de flux :
  - mise en place de tests automatisés à l'aide de navigateurs 'headless'
- Pendant le traitement des données :
  - tests unitaires avec le framework Pytest par exemple
  - contrôle de version
  - revue de code
- Pendant le stockage des données :
  - vérifier les schémas de données
  - vérifier la coherence des partitions
  - s'assurer de la preservation des données
- Pendant l'étape d'orchestration :
  - tests unitaires
  - contrôle de versions des flux de travail
- Pendant l'étape d'exploration des données et des modèles :
  - s'assurer de la disponibilité du même nombre de données en dév qu'en prod
  - s'assurer du versioning des données



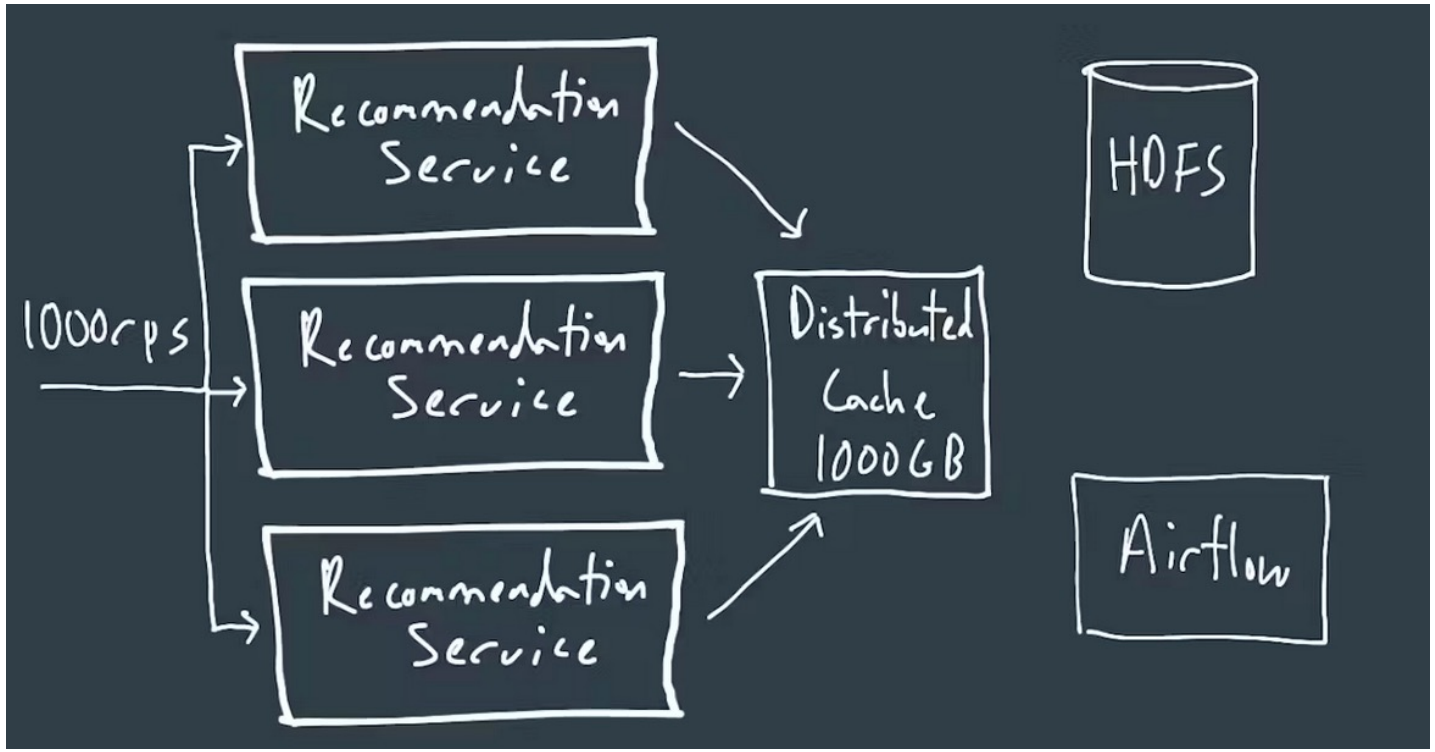
Mise en production

- Pendant l'étape d'expérimentation
  - suivi de l'expérimentation :
    - qui sont les utilisateurs affectés
    - quelles sont les fonctionnalités affectées
    - sur quelles interfaces les changements sont-ils effectués
    - pendant combien de temps dure l'expérimentation ?
  - permet d'éviter les conflits,
  - garder une trace de ce qui a marché et moins marché,
  - assurer la reproductibilité de l'expérimentation

- L'un des objectifs de la mise en production est de 'tout' synchroniser lors du déploiement, ie
  - le modèle,
  - les données attendues par ce modèle,
  - et l'environnement en termes de progiciels

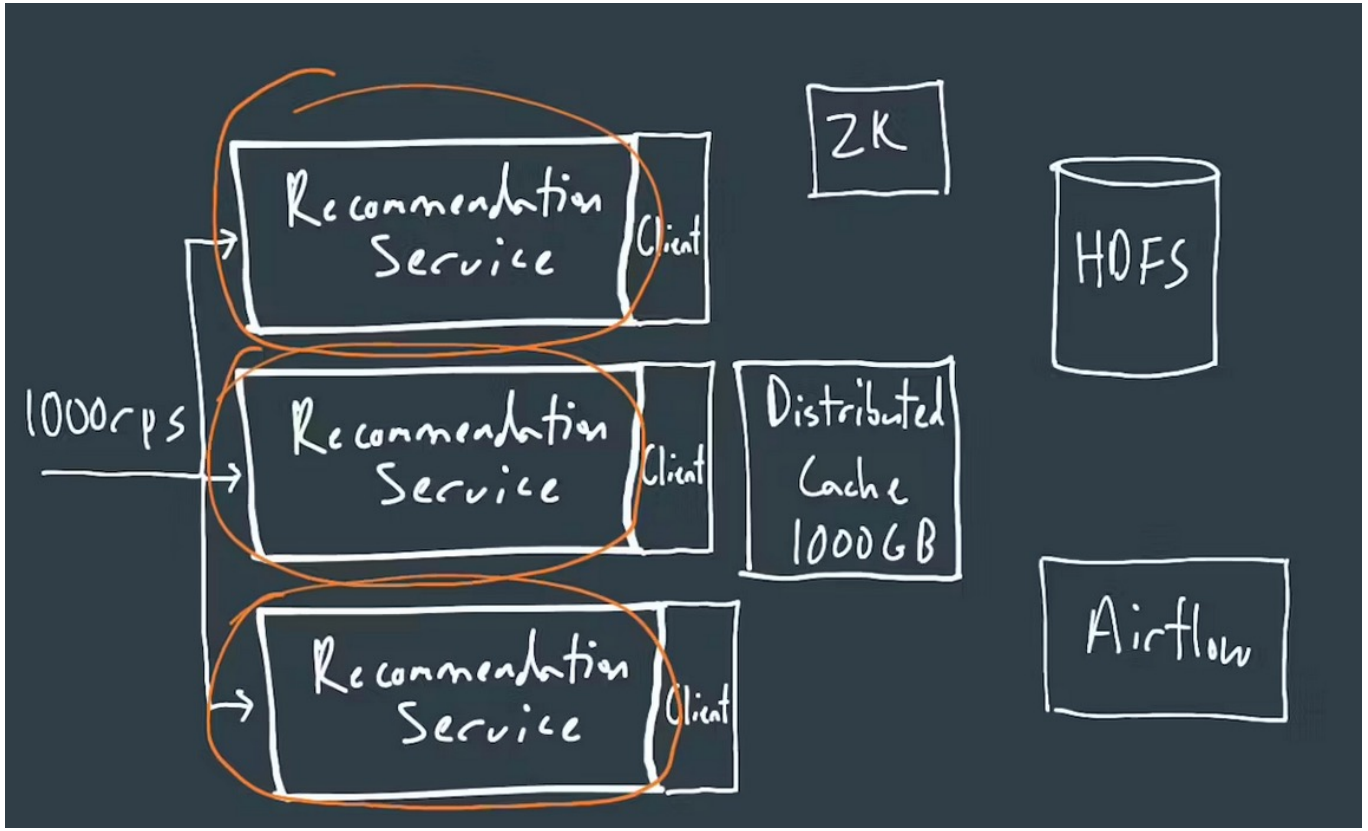
Hébergement des données

- Prenons l'exemple d'un service de recommendation que l'on aurais mis en place sur la page d'accueil de Netflix
  - utilisateur demande une page
  - on lui présente une page avec une recommendation provenant du service de recommendation
  - le service de recommendation récupère les données de HDFS à partir de Airflow



Hébergement des modèles

- Le service de recommendation assure :
  - la récupération des caractéristiques
  - leur modification
  - la gestion des inférences du modèle et la qualité de service (temps de latence notamment)
- Pour gérer la latence :
  - inférence locale
  - inférence distante
  - optimisation du langage
  - optimisation des ressources

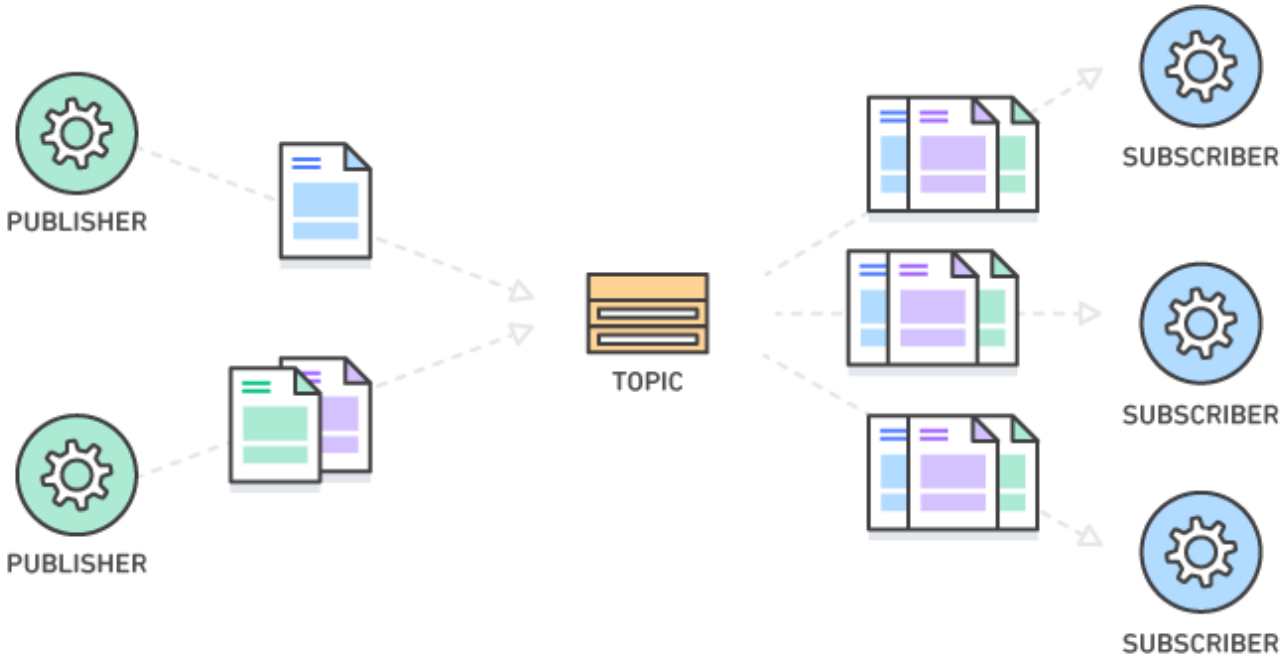


Outline

- Introduction
- Collecte des données
- Exploration
- Expérimentation
- Apprentissage grande échelle
- Mise en production et hébergement
- Q&A

Q&A

- Qu'est ce que le concept Publier/souscrire[1] ?
  - La messagerie Publier/S'abonner, ou messagerie Pub/Sub, est un modèle de communication asynchrone qui permet aux développeurs de créer facilement des applications hautement fonctionnelles et architecturalement complexes dans le cloud.
  - Dans le cloud, les applications sont découplées en des composants indépendants appelés services. La messagerie Pub/Sub fournit des notifications d'événements instantanées pour ces systèmes distribués. Elle prend en charge une communication évolutive et fiable entre des modules logiciels indépendants.



Source : https://aws.amazon.com/fr/what-is/pub-sub-messaging/#:~:text=Pub%2Fsub%20messaging%20instantly%20pushes,when%20a%20message%20is%20available.

Q&A

- Qu'est ce que le concept Publier/souscrire ?
  - 4 composantes clés :
    - Messages : données de communication envoyées par un expéditeur à un destinataire. Les types de données de messages vont des chaînes aux objets complexes représentant du texte, du contenu vidéo, audio, des données de capteur ou autre contenu numérique.
    - Rubriques : Chaque message est associé à une rubrique. La rubrique joue le rôle de canal intermédiaire entre les expéditeurs et les destinataires. Elle maintient une liste de destinataires intéressés par les messages de cette rubrique.
    - Abonnés (consommateurs) : destinataire du message. Les abonnés doivent s'enregistrer (ou s'abonner) auprès des rubriques qui les intéressent. Ils peuvent réaliser différentes fonctions, ou faire quelque chose de différent avec le message en parallèle.
    - Éditeurs (producteurs) : le composant qui envoie les messages. Il crée des messages relatifs à une rubrique et les envoie une fois seulement à tous les abonnés de cette rubrique. Cette interaction entre l'éditeur et les abonnés est une relation de type « un à plusieurs ». L'éditeur n'a pas besoin de savoir qui utilise l'information qu'il diffuse et les abonnés n'ont pas besoin de savoir d'où provient le message.
  - Dans le contexte du big data, on ajoute un cluster de 'brokers' à cette architecture. Il vient s'insérer entre les producteurs et les consommateurs pour gérer plus efficacement le flux massif des messages entrants.

Q&A

- Qu'est ce qu'une datagrid (grille de données) ?
  - Une grille de données est un ensemble d'ordinateurs qui interagissent directement les uns avec les autres pour coordonner le traitement de tâches volumineuses. Les ordinateurs participants sont généralement répartis sur plusieurs sites géographiquement éloignés. Chaque site peut posséder un ou plusieurs des ordinateurs de la grille, et chaque site partage des données et des ressources avec les autres sites.
  - L'objectif principal d'une grille de données est de tirer parti de la puissance collective de tous les ordinateurs pour accomplir une tâche donnée, dans le cadre d'une pratique connue sous le nom d'informatique en grille. Un logiciel fonctionnant sur tous les ordinateurs d'une grille gère la coordination des tâches et l'accès des utilisateurs aux données sur l'ensemble de la grille.
  - Un exemple est l'utilisation de la grille de données est le magasin de données. Chaque site d'une grille de données peut stocker des données qui lui appartiennent, mais le partage coordonné de toutes les données entre tous les utilisateurs de la grille favorise la collaboration et le transfert de connaissances.
  - Les grilles de données peuvent également servir de base à des nuages privés, dans lesquels les ordinateurs sont mis en commun, puis un sous-ensemble des ressources de ce pool est dédié à divers utilisateurs par le biais de machines virtuelles. Chaque machine virtuelle ressemble à un véritable ordinateur, mais n'utilise généralement qu'une partie des ressources d'un ordinateur physique. Cette configuration est similaire à celle d'un fournisseur de nuages publics, sauf que les ordinateurs appartiennent aux organisations participantes de la grille de données. Cette configuration est particulièrement efficace lorsque les utilisateurs du nuage privé ont des besoins informatiques à court terme qui sont rapidement libérés afin qu'une autre tâche puisse exploiter ces ressources libérées.
  - Un type spécifique de grille de données est la grille de données en mémoire (IMDG). Il s'agit d'un ensemble d'ordinateurs en réseau/en grappe qui mettent en commun leur mémoire vive (RAM) pour permettre aux applications de partager des données avec d'autres applications fonctionnant dans la grappe. Les IMDG sont conçus pour le traitement de données à des vitesses extrêmement élevées. Ils sont conçus pour la création et l'exécution d'applications à grande échelle qui nécessitent plus de mémoire vive que ce qui est généralement disponible dans un seul serveur informatique. Cela permet d'obtenir les meilleures performances d'application en utilisant la mémoire vive avec la puissance de traitement de plusieurs ordinateurs qui exécutent des tâches en parallèle. Les IMDG sont particulièrement utiles pour les applications qui effectuent des traitements parallèles importants sur de grands ensembles de données.