Extrait du cours RSX102 / Systèmes distribués

Technologies pour les applications en réseau

RSX102

Document provisoire.
Copie et diffusion non autorisées sans accord écrit.
Documents liés aux cours : https://rsx102.seancetenante.com

2024 *François Lacomme* <*francois.lacomme@2isa.net*>

6 - Systèmes distribués

6.1 - Introduction

- **Définitions**
  - Un **système distribué** (aussi appelé **système réparti**) est un ensemble d'unités de traitement **autonomes** interconnectées entre-elles.
    - Exemples : Internet ; RTC (Réseau téléphonique commuté) ; GPS ; Réseau de capteurs…
    - **Autonomes** => OS et/ou programmes locaux ; mémoires locales ; asynchrones ; pas de temps global
    - **Interconnexion** => Échange d'informations par envoi de messages
  - Un ensemble d'ordinateurs indépendants qui apparaît à un utilisateur comme un système unique et cohérent
    - Les machines sont autonomes
    - Les utilisateurs ont l'impression d'utiliser un seul système.
  - Un **algorithme distribué** (ou réparti) est une suite d'instructions répartie sur plusieurs sites. Chaque site calcule et communique avec d'autres sites.
    - Ex. : Algorithmes de routage ; algorithmes d'équilibrage de charge.

6 - Systèmes distribués

6.2 - Partage de données dans le cloud

- **Problématique**
  - Les systèmes distribués utilisent la puissance de calcul et l'espace de stockage de plusieurs unités de traitement interconnectées par des réseaux.
  - Leur taille peut aller du plus petit système composé de deux entités aux millions de nœuds que forment le réseau internet.
  - La **localisation** des unités de traitement et des données doit être **transparente** pour l'utilisateur.
  - Le système doit gérer la localisation et le transfert des données et ce, de manière transparente.
  - Une **grille informatique** mutualise un ensemble de ressources informatiques géographiquement distribuées dans différents sites.
    - La grille consiste en une infrastructure composée de ressources informatiques hétérogènes interconnectées par un réseau.
    - L'idée est de proposer un super-ordinateur en réunissant la puissance de calcul et l'espace de stockage de sites géographiquement dispersés.
  - **Cohérence des données** : capacité pour un système à refléter sur la copie d'une donnée les modifications intervenues sur d'autres copies de cette donnée

6 - Systèmes distribués

6.2 - Partage de données dans le cloud

- **Définitions**
  - HPC, *high-performance computing* ; Calcul haute performance.
  - MPP, *Massively Parallel Processing* ; Traitement massivement parallèle :
    - Exécution coordonnée d'un programme par plusieurs processeurs.
  - ETL, *Extract, Transform, Load* ; Extraction, Transformation, Chargement.
    - Voir Apache Hive
  - *Grid ;* Grille informatique :
    - Infrastructure virtuelle constituée d'un ensemble de ressources informatiques partagées, distribuées, hétérogènes, délocalisées et autonomes.
  - *Computer cluster* ; grappe de serveurs, ferme de calcul pour réaliser le calcul distribué, *Distributed computing*.

- **Définitions, suite…**
  - *Data store* ; Dépôt de données :
    - Terme générique qui désigne l'ensemble des bases de données, des systèmes de fichiers ou de répertoires.
  - *Data warehouse* ; entrepôt de données ;
    - Type particulier de base de données.
  - *Data lake* ; lac de données :
    - Stockage de données massives où les données sont gardées dans leur formats natifs, dans des base NoSQL par exemple.
  - *Data Mining* ; exploration de données :
    - Consiste à rechercher des relations qui n'ont pas encore été identifiées.
  - *Dataviz* ; visualisation de données.
  - *Open data* ; données ouvertes :
    - Données numériques dont l'accès et l'usage sont laissés libres aux usagers.
  - BI, *Business Intelligence* ; informatique décisionnelle :
    - Processus d'analyse des données qui vise à doper les performances métier en aidant à prendre des décisions plus avisées.

6 - Systèmes distribués

6.2 - Partage de données dans le cloud

- **Big data**
  - Le *Big Data* fait référence à **l'explosion du volume des données** dans l'entreprise et des nouveaux moyens technologiques proposés par les éditeurs pour y répondre.
    - YouTube **reçoit** 500 heures de vidéo toutes les minutes ;
    - Plus d'1 milliard d'heures de vidéos sont visionnées sur YouTube par les internautes à travers le monde tous les jours ;
    - 4 pétaoctets (4×10^{15} octets soit 4000 téraoctets) de données **transitent** chaque jour sur Facebook ;
  - 500 millions de tweets étaient **envoyés** par jour en 2017 ;
  - 8,6 milliards de téléphones mobiles en activité dans le monde ;
  - 1000 articles commandés par minute chez *amazon.fr* pendant le Black Friday ;
  - 90% des données créées dans le monde l'ont été au cours des 2 dernières années ;

6 - Systèmes distribués

6.2 - Partage de données dans le cloud

- **Big data, suite…**
  - *Big data,* données massives :
    - Volume de données numériques créées dans le monde :
      - 2 zettaoctets en 2010
      - 64 zettaoctets en 2020
      - Environ 120 Zo en 2023
      - *180 Zo en 2025 ?*
    - 1 zettaoctet = 1 Zo = 10^{21} octets = 1 000 000 Po.

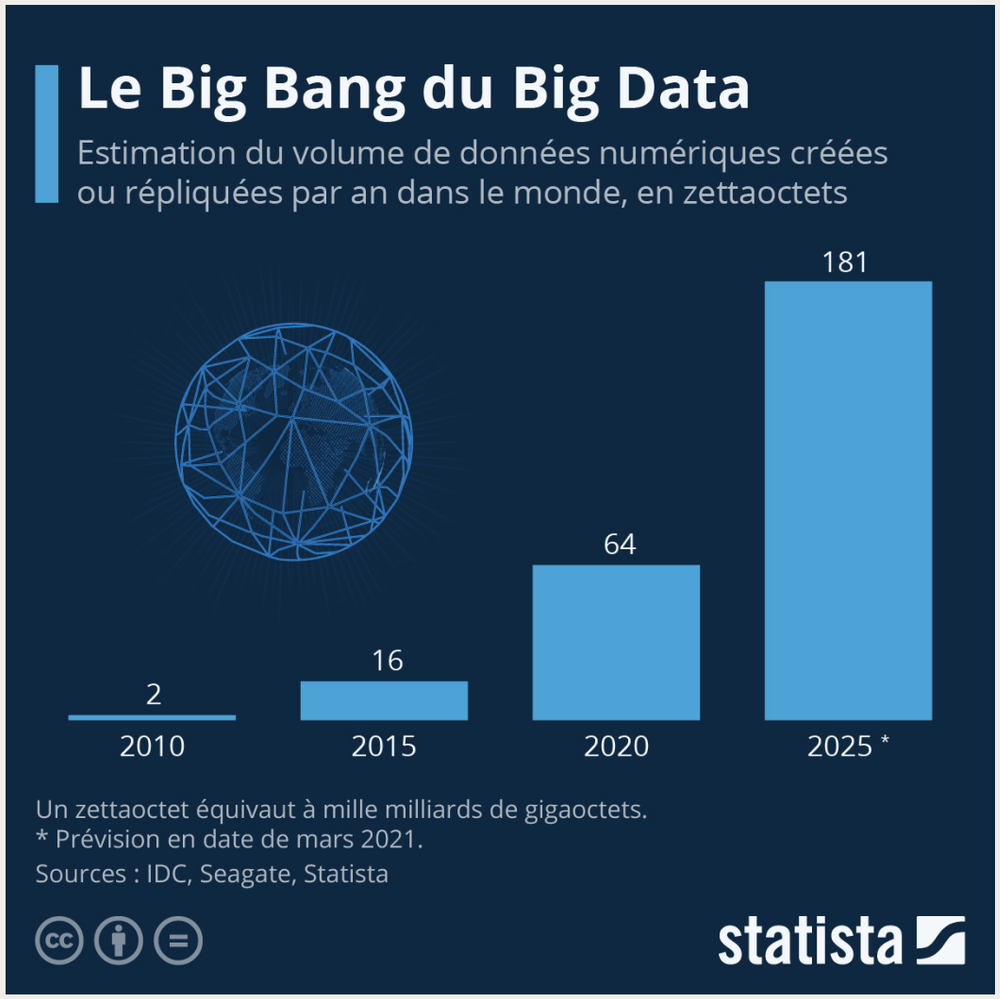


Fig 6.1 - Le Big Bang du Big Data

- Les 4V :
  - **Volume** de données + **Vitesse** de traitement + **Variété** de formats de données + **Véracité** (erreurs, incomplétude, confiance, fraîcheur, etc.)

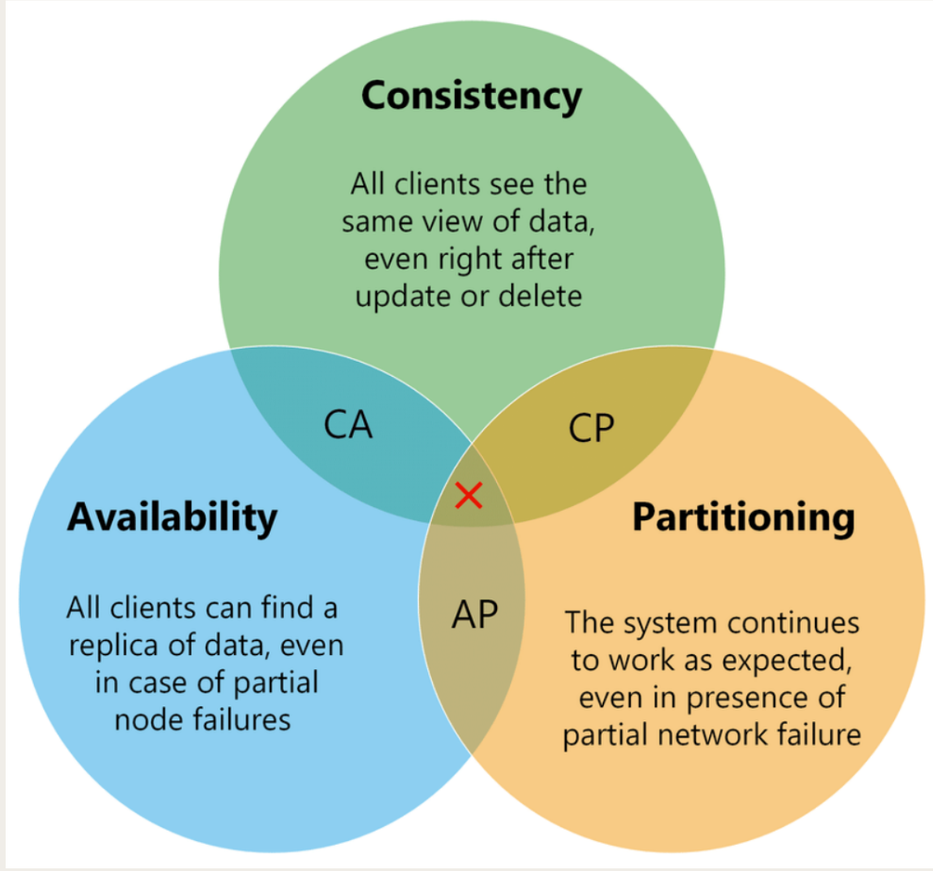
- **La cohérence de données (*data consistency*)**
  - Un système en **cohérence forte** assure que toute lecture d'une copie d'une donnée reflétera toute modification antérieure à la lecture intervenue sur n'importe quelle copie de la donnée.
  - Un système en **cohérence faible** assure que si une copie est modifiée, toutes les copies des données refléteront ces modifications au bout d'un certain temps, mais la modification n'est pas forcément immédiate.
  - **Cohérence stricte** : c'est la forme la plus intuitive de la cohérence. Elle correspond à garantir au programmeur que chaque lecture d'une donnée correspond bien à la version la plus fraîchement modifiée dans tous les caches du système.
  - **Cohérence séquentielle** : Elle consiste à préserver l'ordre global des accès mémoire en lecture. Cet ordre correspond à l'ordre du programme. Le résultat final d'une exécution est similaire au résultat d'une exécution séquentielle du programme.
  - **Cohérence faible** : le programmeur est impliqué dans la gestion de la cohérence en utilisant des opérateurs de synchronisation séquentiellement cohérents.
  - **Cohérence par nœud** : L'ordre des écritures effectuées par un même nœud est préservé tandis que les écritures des différents nœuds peuvent être vues autrement.
  - **Cohérence relâchée** : on considère deux types d'opérateurs de synchronisation : *acquire* (pour acquérir l'accès à une variable partagée), *release* (pour libérer l'accès à une variable partagée). Chaque opérateur est considéré cohérent par nœud.

- **Théorème CAP, alias théorème de Brewer**
  - CAP = ***Consistency, Availability & Partition Tolerance***, soit Cohérence, Disponibilité & Tolérance au partitionnement.
  - **Eric Brewer** de l'université de Californie à Berkeley a énoncé une conjecture, devenu le **théorème de Brewer** ou Théorème CAP, qui prouve que :
  - Il est impossible sur un système informatique de calcul distribué de garantir en même temps (c'est-à-dire de manière synchrone) les trois contraintes suivantes :
    - *Consistency* (Cohérence) : tous les nœuds du système voient exactement les mêmes données au même moment ;
    - *Availability* (Disponibilité) : garantie que toutes les requêtes reçoivent une réponse ;
    - *Partition Tolerance* (Tolérance au partitionnement) : aucune panne moins importante qu'une coupure totale du réseau ne doit empêcher le système de répondre correctement.
  - D'après ce théorème, un système de calcul/stockage distribué ne peut garantir à un instant ***t*** que deux de ces contraintes mais pas les trois.

6 - Systèmes distribués

6.2 - Partage de données dans le cloud

- **Théorème CAP, alias théorème de Brewer, suite…**
  - D'où les trois catégories de systèmes distribués :
    - **Système AP** (disponibilité et tolérance au partitionnement)
      - Exemple : DNS
      - Si une entrée du DNS est modifiée, cela peut prendre plusieurs jours avant que cette modification ne soit transmise à toute la hiérarchie du système et ne puisse être vue par tous les clients
    - **Système CA** (cohérence et disponibilité) ;
      - Exemple : SGBD
      - Où la tolérance au partitionnement joue un rôle mineur
    - **Système CP** (cohérence et tolérance au partitionnement)
      - Exemple : les applications financières et bancaires



- Voir : Théorème CAP : cohérence, disponibilité et tolérance au partitionnement - Digital Guide IONOS

6 - Systèmes distribués

6.2 - Partage de données dans le cloud

- **Big data**
  - Les besoins :
    - Systèmes qui peuvent gérer de gros volumes de données
    - *Scalable* (extensible)
    - Robuste
    - Haute disponibilité
    - Économique
  - Nouveaux modèles de bases de données
    - Les bases de données relationnelles ne sont pas adaptés
    - Des nouveaux modèles de repésentation sont utilisés (des patrons d'architectures), comme BDAF, *Big Data Architecture Framework,* exploités par MapReduce créé par Google.
- Stockage
  - *Data lake*, Lac de données : les données sont gardées dans leur formats originaux dans des base NoSQL par exemple.
  - Le Cloud computing propose des services comme Google BigQuery, AWS Big Data, etc.
  - DFS, *distributed file system,* Systèmes de fichiers distribués : les données sont stockées là où elles peuvent être traitées.

6.3 - Hadoop

- **Apache Hadoop, un framework open source**
  - Pour créer des applications de traitement et de **gestion intensive de données**
  - Doug Cutting est un cofondateur du projet Hadoop (chez Google, puis Yahoo)
  - Framework écrit en Java et inspiré de Google FileSystem et de Google MapReduce
  - Objectifs :
    - Stocker les données dans un espace de stockage massif ;
    - Rechercher et restituer des données ;
    - Avec une énorme puissance de traitement et la possibilité de prendre en charge une quantité de tâches virtuellement illimitée.

- Voir :
  - Hadoop : qu'est-ce que c'est et comment apprendre à l'utiliser ? - DataScientest

6.3 - Hadoop

- **Apache Hadoop, un framework open source**
  - Trois constituants essentiels :
    - **HDFS**, *Hadoop Filesystem*, un système de fichiers virtuel aggrégant le stockage de plusieurs machines d'un cluster ;
    - **Hadoop MapReduce** : un framework logiciel en Java permettant de développer des programmes exécutables de manière distribués grâce à l'utilisation de l'algorithme **MapReduce** développé par Google.
    - Apache **HBase,** une base de données NoSQL, scalable et distribuée conçue pour les analyses Big Data.

    - L'écosystème Hadoop comprend des logiciels et des utilitaires associés, notamment Apache **Hive**, Apache **HBase**, **Spark**, **Kafka**, etc.
- **Acteurs**
  - Fondation Apache
  - **Cloudera**, start-up de la Silicon Valley ; développement de logiciels de Big Data basées sur le framework Hadoop
  - **Hortonworks**, société de logiciels informatique Californienne ; développement et soutien de Hadoop. Elle a fusionné avec Cloudera en 2018.
  - Les GAFAM

6.3 - Hadoop

- **Apache Hadoop, l'écosystème**

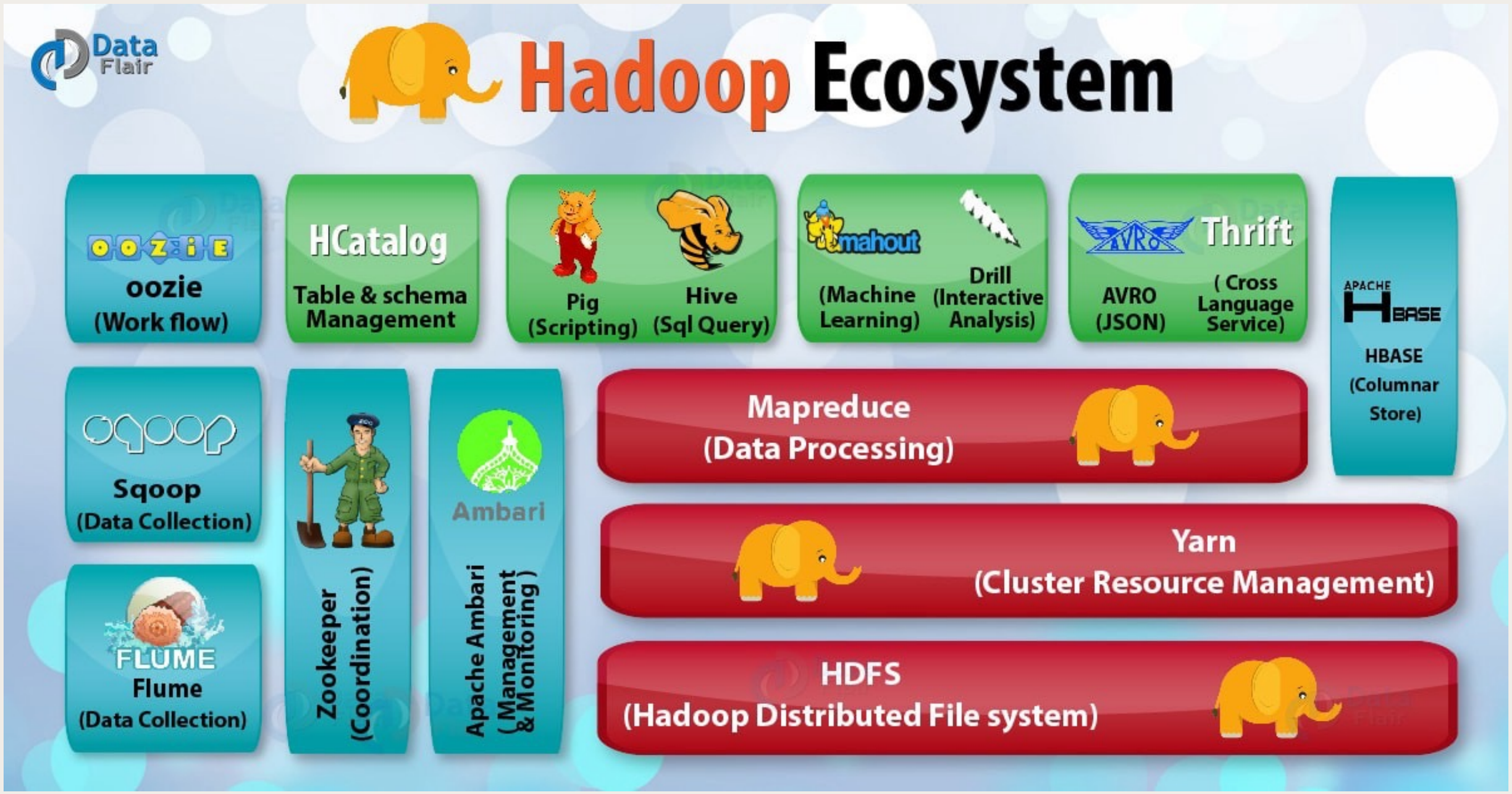


Fig 6.1 - L'écosystème Hadoop

- **MapReduce et Spark**
  - **MapReduce** est un modèle de programmation (un patron d'architecture) qui fournit un cadre pour automatiser le calcul parallèle sur des données massives.
    - Deux étapes :
      - **map** : consiste à appliquer une même fonction à tous les éléments de la liste ;
      - **reduce** : applique une fonction récursivement à une liste et retourne un seul résultat.
    - **map** et **reduce** sont des opérateurs génériques et leur combinaison permet donc de modéliser énormément de problèmes
    - Dans un calcul MapReduce :
      - on commence par découper le problème en de nombreux sous-problèmes indépendants (étape *Map*)
      - que l'on confie à des ordinateurs distincts.
      - Ces machines résolvent les sous-problèmes et envoient leurs résultats à d'autres machines qui ont pour tâche de combiner les résultats (étape *Reduce*).
      - L'objectif est de pouvoir travailler sur d'énormes volumes de données en parallélisant les calculs sur des grappes d'ordinateurs.
      - MapReduce ne permet de traiter que des problèmes qui peuvent se décomposer en de multiples tâches parallèles.
  - Voir : https://www.youtube.com/watch?v=mhnxcYbdE6M&t=476s

MapReduce et Spark

- **Apache Spark** est également une solution pour écrire simplement des applications distribuées
  - **Spark** propose des bibliothèques de traitement classique.
  - Sa performance est remarquable ; il peut travailler sur des données sur disque ou des données chargées en RAM.
  - Il est plus jeune que MapReduce mais il dispose d'une communauté énorme.
  - c'est donc une solution qui s'avère être le successeur de MapReduce.
  - Voir : https://www.youtube.com/watch?v=ymtq8yjmD9I

- **Pour en savoir plus**
  - Un déluge de données - Interstices
  - Définition : Qu'est-ce que le Big Data ? - Le Big Data
  - Hadoop – Tout savoir sur la principale plateforme Big Data - Le Big Data
  - Apache Hadoop
  - Hadoop : qu'est-ce que c'est et comment apprendre à l'utiliser ? - DataScientest
  - Apache Spark (ne pas confondre avec 🎸 Sparks)
  - Qu'est-ce qu'Apache Hadoop dans Azure HDInsight ?