Université Batna 2
Faculté de Mathématiques et Informatique
Département de Mathématique
Année universitaire 2019-2020

Cours Big data et deep learning

Master 1 SAD

Dr Saadna yassmina

Chapitre 1: Big data

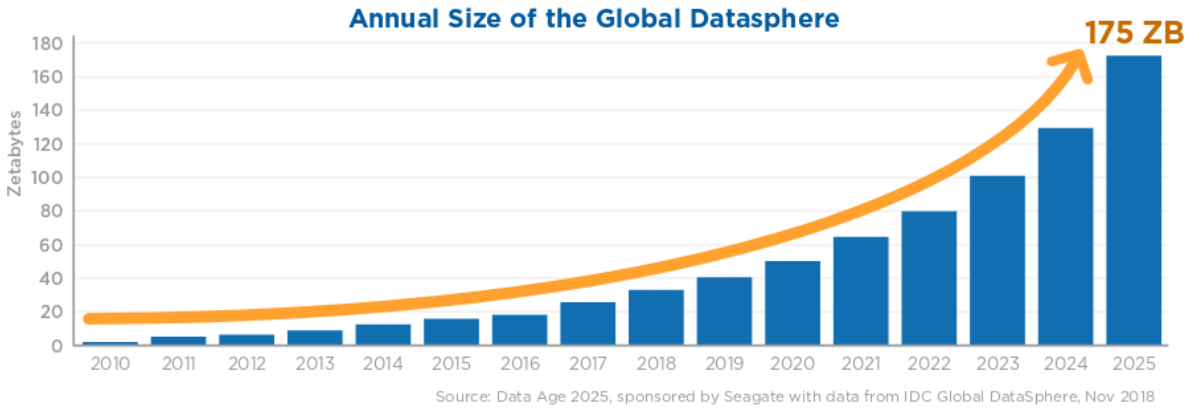
Introduction

En moins de dix ans, le volume total de données à analyser devrait être multiplié par plus de huit selon le cabinet d'analystes IDC, pour atteindre 163 Zettaoctets (163 milliards de Teraoctets). « Ce qui est étonnant, ce n'est pas que la production de données à stocker augmente, mais le rythme effréné de cette augmentation », explique Jeff Fochtman, responsable marketing chez Seagate. « Nous -mêmes sommes surpris. Et la vague de l'Internet des objets ne fait que commencer ».

Il ne s'agit pas que des bracelets connectés. Voitures autonomes, implants médicaux, caméras connectées, compteurs électriques intelligents ou jouets connectés sont autant de machines qui génèrent de plus en plus de données. Corollaire de ce changement, la nature de données produites change. C'est d'ailleurs le principal enseignement de l'étude, au-delà de la rapide montée en puissance de la production.

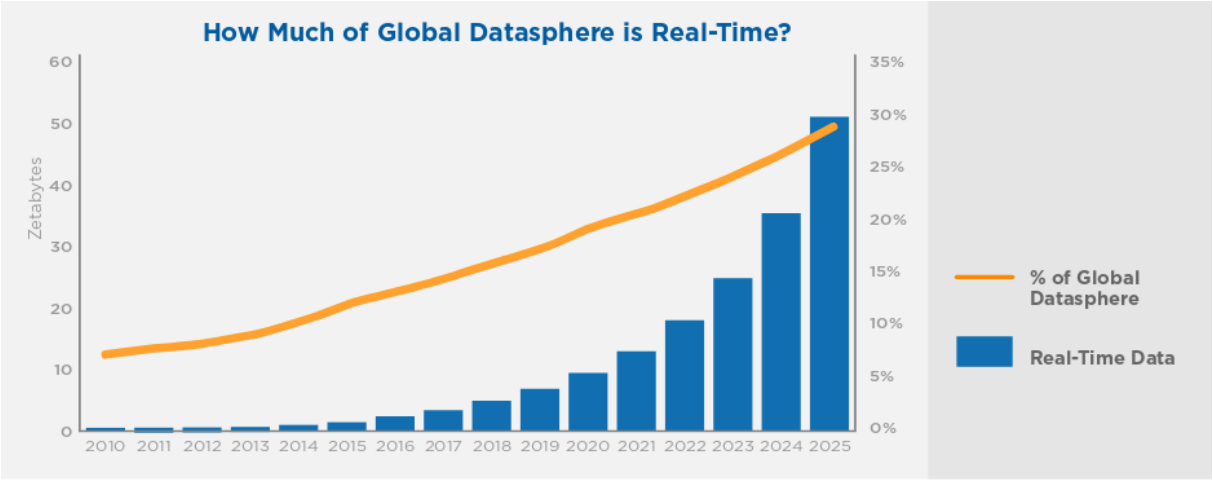
Alors qu'une majorité des besoins de stockage aujourd'hui concerne le divertissement (films, photos...), les données collectées à l'avenir seront de plus en plus critiques, au sens où elles sont le support d'activités humaines considérées comme vitales. Il peut s'agir des données de santé, mais aussi de celles qui serviront à diriger les voitures autonomes par exemple.

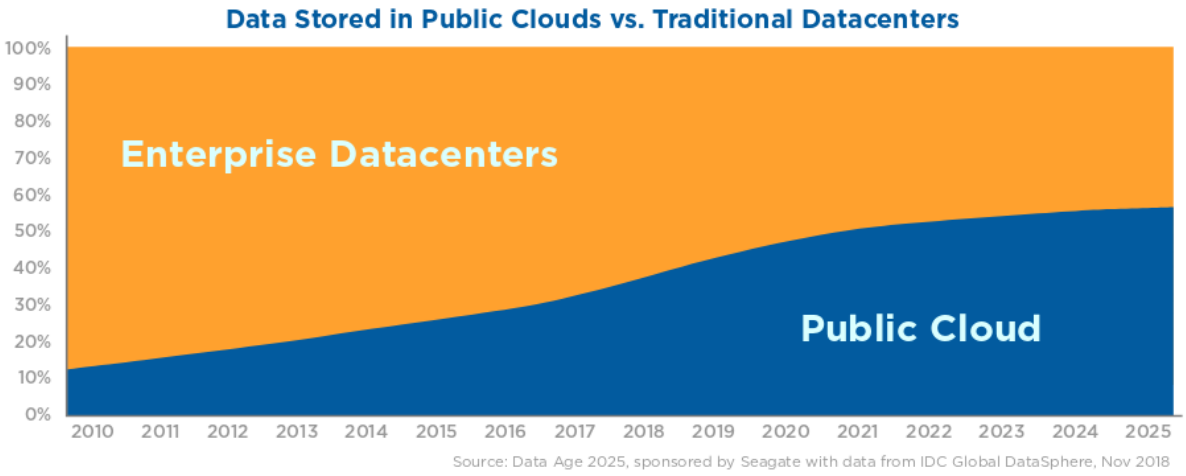
L'industrie se frotte les mains. Certes, IDC explique qu'il est hors de question que le stockage suive le rythme de la production. Il faudrait si c'était le cas 16 milliards des plus gros disques durs d'entreprises de 12 Teraoctets pour conserver les 163 Zettaoctets que nous sommes censés produire dès 2025, soit deux fois plus de disques durs que l'industrie n'en a produit au cours des 20 dernières années.



Le volume total de données pourrait atteindre 175 zettaoctets (175 milliards de teraoctets) en 2025, contre 33 zettaoctets en 2018.

Ces données circulent entre cloud, terminaux/objets connectés (IoT) et périphérie du réseau (edge). Justement, près de 30% des données mondiales nécessiteront un traitement en temps réel, au plus près du lieu où elles sont générées, contre 15% en 2017.





IDC prévoit ainsi que 49% des données mondiales stockées le seront dans le cloud public à horizon 2025. Un mouvement qui devrait s'exercer au détriment des datacenters traditionnels d'entreprise et du stockage dans les terminaux (endpoints).

En revanche, le stockage à la périphérie du réseau (edge) va lui aussi progresser. Pour une analyse des données là où elles sont générées, plutôt que vers un datacenter tiers.

Les notions de base de Big Data

Chaque jour, nous générons des trillions d'octets de données (Big Data). Ces données proviennent de partout : de capteurs utilisés pour collecter les informations climatiques, de messages sur les sites de médias sociaux, d'images numériques et de vidéos publiées en ligne, d'enregistrements transactionnels d'achats en ligne et de signaux GPS de téléphones mobiles, pour ne citer que quelques sources.

Les Big Data se caractérisent par leur volumétrie (données massives); ils sont connus aussi par leur variété en termes de formats et de nouvelles structures, ainsi, qu'une exi- gence en termes de rapidité dans le traitement. Mais jusqu'à maintenant d'après nos recherches, aucun logiciel est encore capable de gérer toutes ces données qui ont plu- sieurs types et formes et qui augmentent très rapidement. Alors les problématiques du Big Data font partie de notre quotidien, et il faudrait des solutions plus avancées pour gérer cette masse de données dans un petit temps.

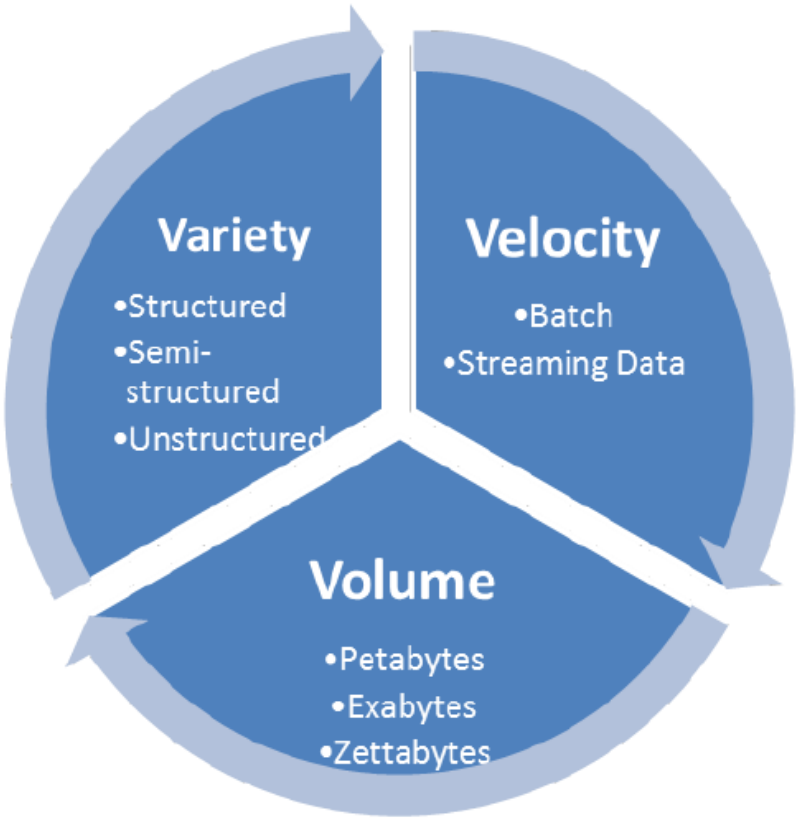
Le calcul distribué concerne le traitement de grandes quantités de données. Ce traitement ne peut être réalisé avec les paradigmes classiques de traitement de don- nées, il nécessite l'utilisation de plateformes distribuées. Dans la littérature, il existe plusieurs solutions, pour l'implémentation de ce paradigmes. Parmi ces solutions on trouve l'exemple Google qui a développé un modèle de programmation très fiable pour le traitement de Big Data : c'est le modèle MapReduce. Ce modèle est implémenté sur plusieurs plateformes comme la plateforme Hadoop. Malgré tous ces avantages, Hadoop souffre de problèmes de la latence qui est la cause principale de développement d'une nouvelle alternative pour améliorer les performances du traitement, c'est la plateforme Spark qui est plus puissante, plus souple et rapide que Hadoop MapReduce.

Définitions

« Le Big Data désigne un très grand volume de données souvent hétérogènes qui ont plusieurs formes et formats (texte, données de capteurs, son, vidéo, données sur le parcours, fichiers journaux, etc.), et comprenant des formats hétérogènes : données structurées, non structurées et semi-structurées. Le Big Data a une nature complexe qui nécessite des technologies puissantes et des algorithmes avancés pour son traitement et stockage. Ainsi, il ne peut être traité en utilisant des outils tels que les SGBD traditionnels. La plupart des scientifiques et experts des données définissent le Big Data avec le concept des 3V comme suit» :

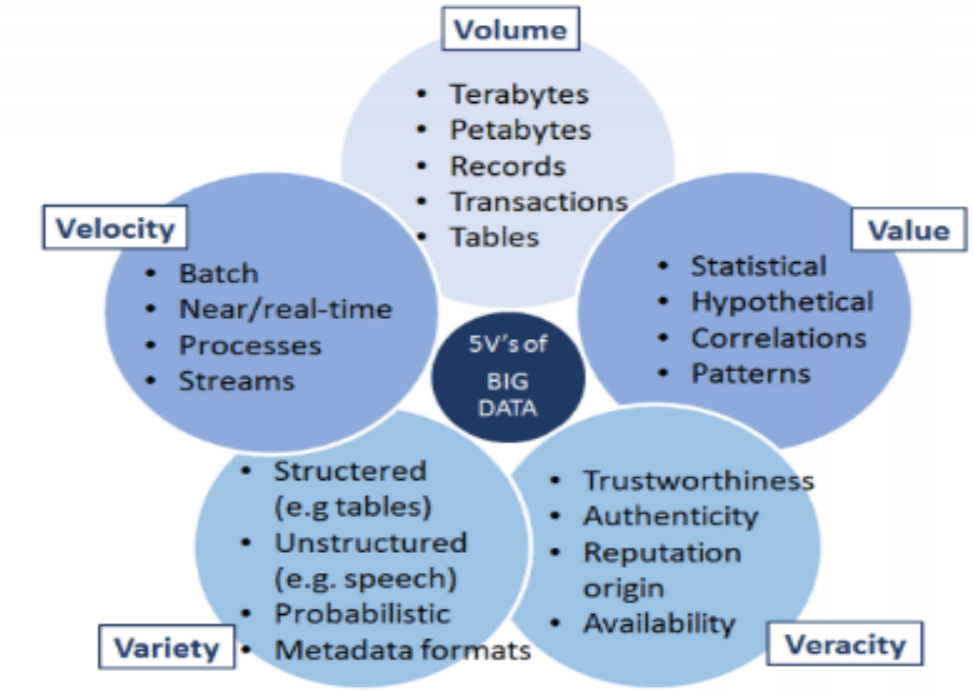
- **Vélocité** : Les données sont générées rapidement et doivent être traitées rapidement pour extraire des informations utiles et des informations pertinentes. Par exemple, Wallmart (une chaîne internationale de détaillants à prix réduits) génère plus de 2,5 petabyte(PB) de données toutes les heures à partir des transactions de ses clients. YouTube est un autre bon exemple qui illustre la vitesse rapide du Big Data.
- **Variété** : Les données volumineuses sont générées à partir de diverses sources distribuées dans plusieurs formats (vidéos, documents, commentaires, journaux, par exemple). Les grands ensembles de données comprennent des données structurées et non structurées, publiques ou privées, locales ou distantes, partagées ou confidentielles, complètes ou incomplètes, etc.

- **Volume** : il représente la quantité de données générées, stockées et exploitées. Le volume des données stockées aujourd'hui est en pleine explosion il est presque de 800.000 Péta-octets, Twitter générer plus de 7 téraoctets chaque jour de données, Facebook générer plus de 10 téraoctets et le volume de données dans 2020 peut atteindre 40 zêta-octets.



Par la suite, les trois dimensions initiales sont élargies par deux autres dimensions des données Big Data (on parle aussi des « 5 V du Big Data») :

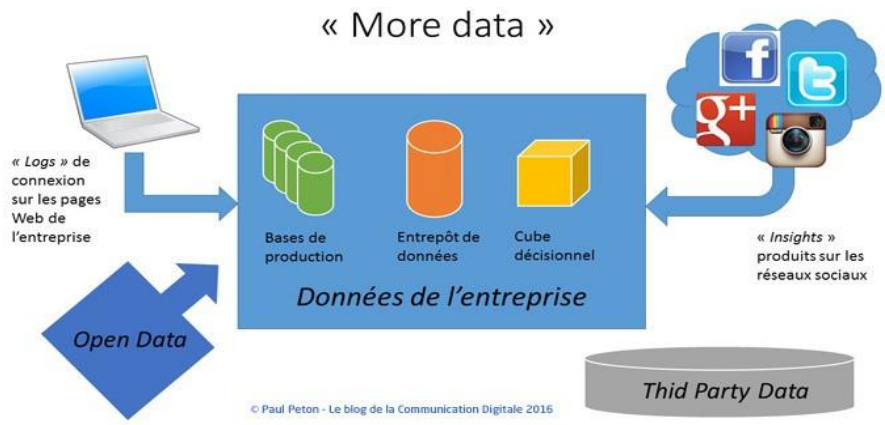
- **Véracité** : La véracité (ou validité) des données correspond à la fiabilité et l'exactitude des données, et la confiance que ces Big Data inspirent aux décideurs. Si les utilisateurs de ces données doutent de leur qualité ou de leur pertinence, il devient difficile d'y investir davantage.
- **Valeur** : Ce dernier V joue un rôle primordial dans les Big Data, la démarche Big Data n'a de sens que pour atteindre des objectifs stratégiques de création de va- leur pour les clients et pour les entreprises dans tous les domaines.



Une des raisons de l'apparition du concept de Big Data est le besoin de réaliser le défi technique qui consiste à traiter de grands volumes d'information de plusieurs types (structurée, semi structuré et non structurée) générée à grande vitesse. Le Big Data s'appuie sur quatre sources de données:

- Les logs (journaux de connexion) issus du trafic sur le site officiel de l'entreprise : Ces sources de données, sont les chemins pris par les visiteurs pour parvenir sur le site : moteurs de recherche, annuaires, rebonds depuis d'autres sites, etc. Les entreprises d'aujourd'hui disposent d'une vitrine sur le Web au travers de son site officiel. Ce dernier génère du trafic qu'il est indispensable d'analyser, ainsi ces entreprises disposent des trackers sur les différentes pages afin de mesurer les chemins de navigation, ou encore les temps passés sur chaque page,.etc. Ci- tons parmi les solutions d'analyse les plus connues : Google Analytics, Adobe Omniture, Coremetics.
- Les issus des médias sociaux «insights» : Une approche complémentaire, consiste à recueillir les commentaires aux publications et à y appliquer des algorithmes d'analyse de sentiment. Citons quelques pistes pour suivre nos différents comptes : Hootsuite, Radian6 ou encore les API mises à disposition et interrogées avec le complément Power Query pour Excel, IRaMuTeQ pour l'analyse de données tex- tuelles.
- Les données comportementales (third party data) Ces données sont toutes des données sur les internautes récoltées via des formulaires ou des cookies. Au- delà des classiques informations d'identité (sexe, âge, CSP, etc), il est maintenant beaucoup plus efficace de mesurer les comportements (navigation, configuration matérielle, temps passé sur les pages, etc). Pour cela, il existe des acteurs spécialisés du Web qui nous aident à collecter de l'information sur nos clients ou prospects et à améliorer ainsi les campagnes de communication. Quelques acteurs du domaine de la third party data : Bluekai, Exelate, Weborama, Datalogix, etc.
- Les données ouvertes et réutilisables «L'open data» sont toutes les données ouvertes et réutilisables, L'open data permet de mettre en ligne les données ouvertes, de fiabiliser

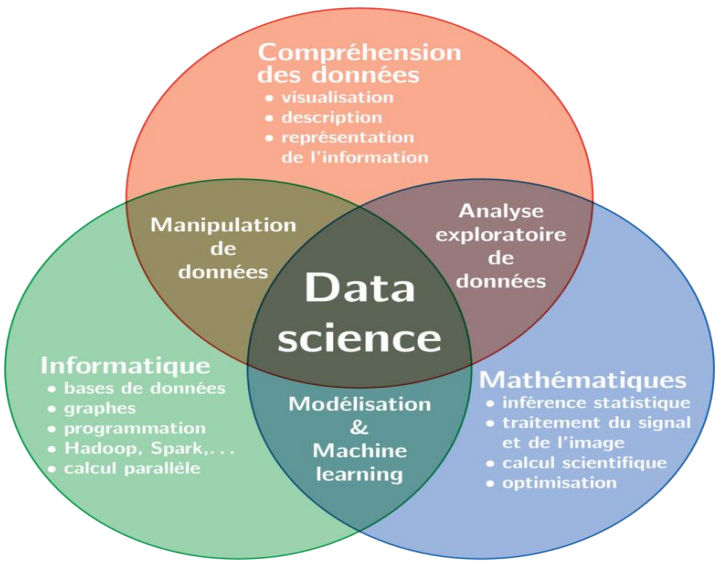
les données et de les rendre réutilisables et exploitables, où l'ouverture consiste à rendre une donnée publique : libre de droits, téléchargeable, réutilisable et gratuite. L'ouverture ne concerne pas les données à caractère privé, les informations sensibles et de sécurité, les documents protégés par des droits d'auteur, etc. Les données ouvertes et réutilisables ne sont pas encore légion même si une mission gouvernementale est très active sur le sujet manque de complétude, niveau de détail insuffisant, relative ancienneté sont les défauts actuels de nombreux jeux de données. Toutefois, c'est un champ d'investigation qu'il ne faut pas négliger, ne serait-ce que par son faible coût (celui du temps passé à chercher!) et son développement inéluctable.



Du statisticien au data scientist

Comparé au traditionnel statisticien du passé dans l'entreprise, le data scientist est probablement moins pointu en mathématique mais pratique la pluridisciplinarité. Il possède un niveau de développement informatique lui permettant de prototyper et tester facilement les algorithmes d'analyse de données qu'il conçoit. De plus, il s'implique dans le métier premier de l'entreprise afin de savoir mener des analyses de données pertinentes pour, par exemple, identifier de nouveaux marchés.

Enfin, une partie du travail du data scientist consiste à simplifier, synthétiser et présenter les résultats de ses analyses aux autres membres de l'entreprises. Cette dernière facette de son travail est primordiale car il doit convaincre ses collègues de la pertinence et de la solidité de ses conclusions sur l'évolution d'un process industriel ou de la stratégie de l'entreprise.



Le data engineer

Les couches logicielles d'un système Big Data sont devenues très complexes. Stocker et mettre à jour en permanence un Data Lake demande beaucoup de ressources matérielles, mais aussi une architecture logicielle efficace et tolérante aux pannes, qui dépasse de loin les simples serveurs de comptes et de données installés sur un petit réseau local. Un interfaçage avec des sources de données externes, ou avec des flux de données continus est aussi devenu une composante classique d'un système Big Data. De plus, analyser de gros volumes de données sur ces architectures logicielles et matérielles requiert également une expérience spécifique en algorithmique et en programmation, pour aboutir à des solutions efficaces à large échelle. Enfin, avant d'analyser des masses de données il convient de collecter, nettoyer et enrichir ces données, qui sont souvent hétérogènes, redondantes, mal identifiées. . .

Toutes ces tâches (et d'autres) constituent des travaux d'ingénieurs apparus avec les Big Data, et définissent le travail des Data Engineers. Il est habituel que 80% d'un projet Big Data soit constitué de Data Engineering (et de seulement 20% de Data Science

Les technologies de traitement et de stockage de Big Data

Les Big Data requièrent de redéfinir les systèmes de stockage et de traitement de données, qui peuvent supporter ce volume de données. En effet, plusieurs technologies ont été proposées afin de représenter ces données, ces technologies prennent au moins un axe parmi les deux soit l'amélioration des capacités de stockage, soit l'amélioration de la puissance de calcul:

- Amélioration de la puissance de calcul : le but de ces techniques est de permettre de faire le traitement sur un grand ensemble de données, avec un coût considérable et d'améliorer les performances de l'exécution comme le temps de traitement et la tolérance aux pannes. Avant l'apparition de la plateforme Hadoop on trouve plusieurs technologies comme le Could Computing, les architectures massivement parallèles MPP et les technologies In-Memory.
- Amélioration des capacités de stockage : l'amélioration du stockage vers des systèmes distribués, où un même fichier peut être réparti sur plusieurs disques durs, cela permet d'augmenter les volumes de stockage en utilisant un matériel de base. Ces technologies de stockage évoluent toujours pour offrir des accès plus rapides aux données comme le NoSQL, HDFS de la plateforme Hadoop, HBase, le Cloud Computing, etc.

MapReduce

Pourquoi MapReduce?

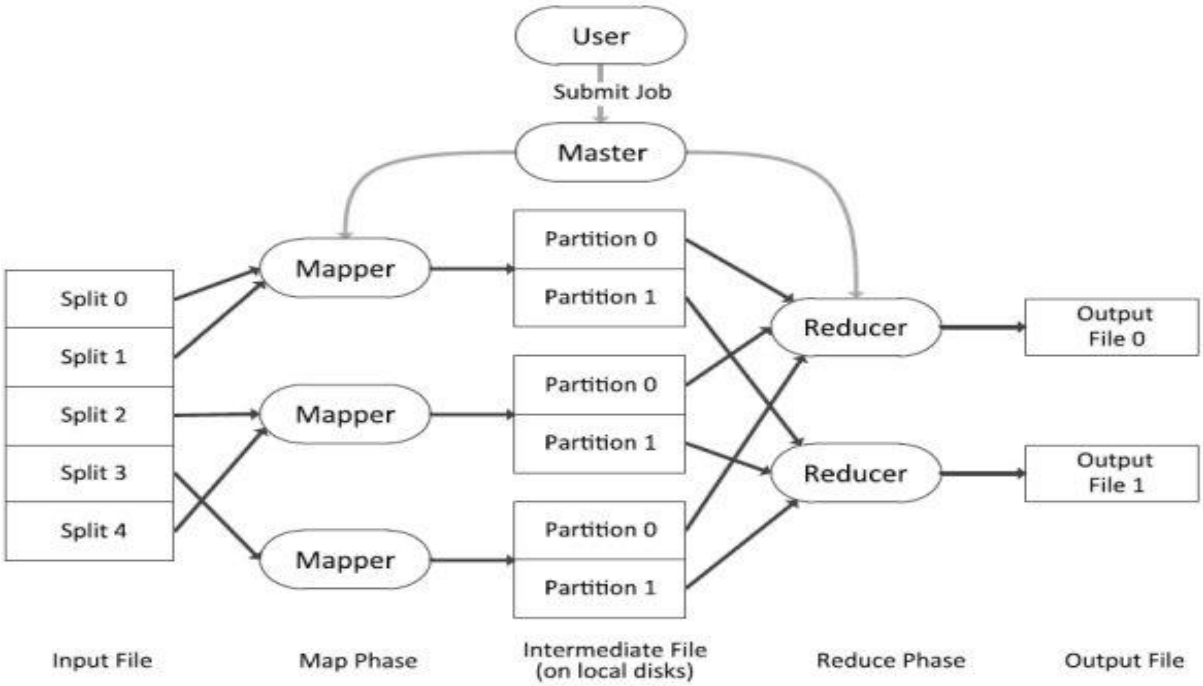
Les systèmes d'entreprise traditionnels disposent normalement d'un serveur centralisé pour stocker et traiter les données. Le modèle traditionnel n'est certainement pas adapté au traitement de gros volumes de données évolutives et ne peut pas être géré par des serveurs de base de données standard. De plus, le système centralisé crée trop de goulot d'étranglement lors du traitement simultané de plusieurs fichiers. Google a résolu ce problème de goulot d'étranglement à l'aide de modèle MapReduce.

Définition de modèle MapReduce :

Il a été conçu dans les années 2000 par les ingénieurs de Google. C'est un modèle de programmation conçu pour traiter plusieurs téraoctets de données sur des milliers de nœuds de calcul dans un cluster. MapReduce peut traiter des téraoctets et des pétaoctets de données plus rapidement et efficacement. Par conséquent, sa popularité a connu une croissance rapide pour diverses marques d'entreprises beaucoup de champs. Il fournit une plateforme très efficace pour l'exécution parallèle des applications, l'allocation de données dans des systèmes de bases de données distribuées, et communications réseau à tolérance de pannes. L'objectif principal de MapReduce est de faciliter la parallélisation des données, leur distribution et l'équilibrage de la charge dans une bibliothèque simple.

L'architecture de modèle MapReduce

Google a créé MapReduce pour traiter de grandes quantités des données non structurées ou semi-structurées, tels que les documents et journaux de demandes de pages Web, sur de grandes clusters de noeuds. Il a produit différents types de données telles que des indices inversés ou fréquences d'accès aux URL [60]. Le MapReduce com- porte trois parties principales, y compris Master, la fonction de Map et de reduce. Un exemple de ce flux de données est présenté dans la figure suivante.



Le Master est responsable de la gestion des fonctions Map et Reduce et de la mise à leur disposition des données et des procédures, il organise la communication entre les mappeurs et les réducteurs. La fonction map est appliquée à chaque enregistrement d'entrée et produit une liste d'enregistrements intermédiaires. La fonction Réduire (également appelée Réducteur) est appliquée à chaque groupe d'enregistrements in- termédiaires avec la même clé et génère une valeur. Par conséquent, le processus de MapReduce inclut les étapes suivantes :

- Les données d'entrée sont divisées en enregistrements.

- Les fonctions de Map traitent ces données et produisent des paires clé/valeur pour chaque enregistrement.
- Toutes les paires clé/valeur issues par la fonction Map sont fusionnées ensemble et regroupées par une clé, puis elles sont triées.
- Les résultats intermédiaires sont transmis à la fonction de réduction (Reduce), qui va produire le résultat final.

Les applications de MapReduce

L'application MapReduce facilite l'exécution de nombreuses applications parallèles des données. MapReduce est le facteur principal dans de nombreuses applications importantes et peut améliorer le parallélisme du système. Il reçoit une attention considérable, pour les applications gourmandes en données et en temps de calculs sur des clusters de machines. Il est utilisé comme un outil de calcul distribué, efficace pour résoudre des problèmes variables, tels que la recherche, le regroupement, l'analyse de journaux, différents types d'opérations de jointure, la multiplication de matrices, la correspondance de modèles et l'analyse de réseaux sociaux. Il permet aux chercheurs d'étudier dans différents domaines.

MapReduce est utilisé dans de nombreuses applications de Big Data tels que : l'exploration de messages courts, les algorithmes génétiques, k-means, algorithme de clustering, fragment d'ADN, système de transport intelligent, applications scientifiques dans le domaine de la santé, systèmes de classification à base de règles floues, environnements hétérogènes, la machine d'apprentissage extrême, forêt aléatoire, l'énergie proportionnelle, capteur mobile de données, web sémantique, etc.

Les plateFormes de traitement des Big Data

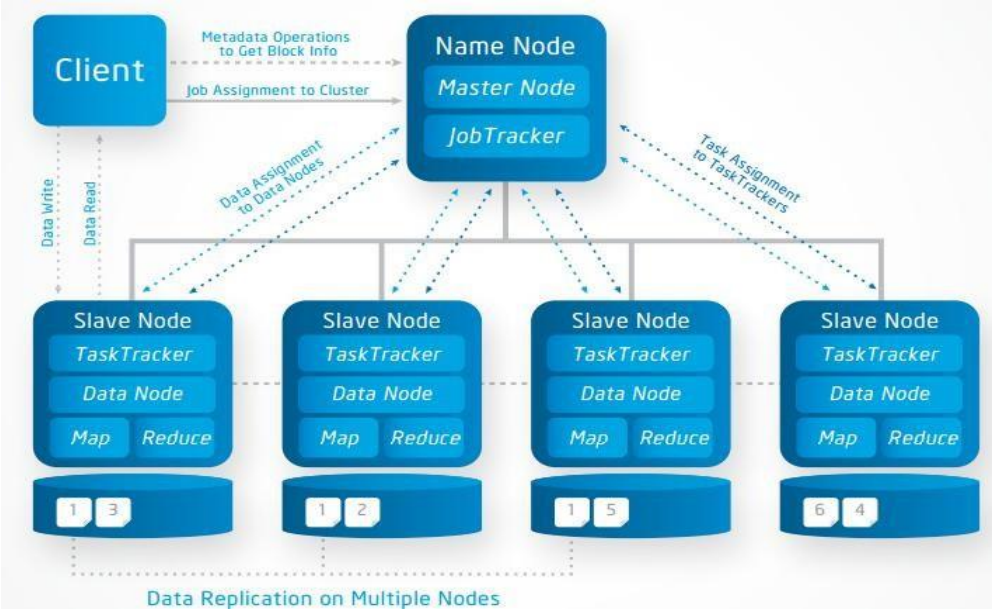
La plateforme hadoop pour le calcul distribué de Big Data

Tout d'abord, Hadoop est un Framework libre, écrit en java, créé et distribué par la fondation Apache, et destiné au traitement de données volumineuses (de l'ordre des péta-octets et plus) ainsi qu'à leur gestion intensive. Inspirée de plusieurs publications techniques rédigées par le géant Google, son objectif est de fournir un système de stockage et de traitement de données distribué, évolutif et extensible. Il permet de traiter un grand nombre de types de données (y compris les données non structurées). On dit qu'il est organisé sur un mode non-relationnel, il est plus général que NoSQL, on peut par exemple stocker les données avec deux types de systèmes HDFS (Hadoop Distributed File System) et HBase qui forment un système de gestion de bases de données orientées, colonnes projetées sur des serveurs distribués en clusters.

Hadoop parallélise le traitement des données à travers de nombreux nœuds faisant partie d'une grappe d'ordinateurs (clusters), ce qui permet d'accélérer les calculs et masquer la latence des opérations d'entrées et de sorties. Hadoop contient un système de fichiers distribué fiable qui garantit la tolérance aux pannes grâce à la réplication des données.

Hadoop contient deux composants de base, HDFS et MapReduce. Les deux sont liés au calcul distribué comme suit :

- HDFS est le responsable de stockage des données et MapReduce est le cœur de Hadoop qui effectue le traitement parallèle grâce aux deux fonctions Map et Reduce.
- Au niveau architectural, Hadoop contient deux types de nœuds.
- Les serveurs ou bien les maitres : le NameNode, le NameNode secondaire et le JobTracker.
- Les esclaves qui sont distribués dans le cluster : le DataNode et le TaskTracker.
- L'exécution des tâches se fait par les TaskTrackers qui sont déployés sur chaque machine selon les instructions du JobTracker.



L'architecture maître/esclave de la plateforme Hadoop

La plateforme Spark pour le calcul distribué de Big Data

Motivation de Spark

Depuis sa création, Hadoop est devenu une technologie importante pour le Big Data. Une des principales raisons de ce succès est sa capacité à gérer d'énormes quantités de données quelque soit leur type (structuré, semi structuré, non structuré). Toutefois, les utilisateurs ont été systématiquement plaignants du problème de la latence élevée avec Hadoop MapReduce indiquant que la réponse en mode batch pour toutes ces applications en temps réel est très douloureux quand il s'agit de données de traite- ment et d'analyse.

Historique de Spark

Spark est un cluster de calcul rapide développé par les contributions de près de 250 développeurs de 50 entreprises AMPLab de l'Université de Berkeley, pour faire l'ana- lyse de données plus rapide et plus facile à écrire et ainsi à courir.

Spark a commencé en 2009 comme un projet de recherche dans le Berkeley Lab RAD, qui deviendra plus tard l'AMPLab. Les chercheurs en laboratoire avaient déjà travaillé sur Hadoop MapReduce, et ont observé que MapReduce était inefficace pour des emplois informatiques itératifs et interactifs. Ainsi, depuis le début, Spark a été conçu pour être rapide pour les requêtes interactives et les algorithmes itératifs, apportant des idées comme le support de stockage en mémoire et la récupération efficace de fautes.

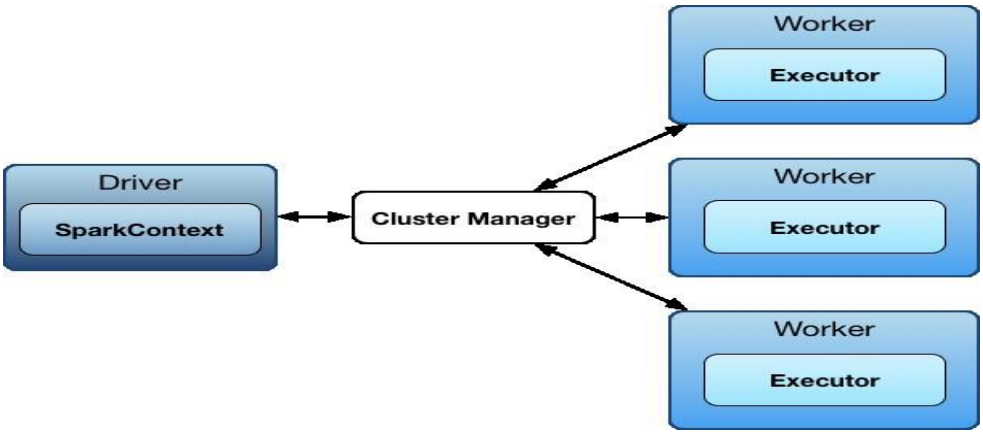
Les documents de recherche ont été publiés à propos de Spark à des conférences universitaires et peu de temps après sa création en 2009, il était déjà de 10 à 100 fois plus vite que MapReduce pour certains emplois.

Certains des premiers utilisateurs de Spark étaient d'autres groupes à l'intérieur de l'Université de Berkeley, y compris des chercheurs, tel que le projet mobile du Millénaire, qui a utilisé Spark pour surveiller et prévoir les embouteillages dans la baie de San Francisco Machine Learning. Dans un temps très court, cependant, de nombreuses organisations externes ont commencé à utiliser Spark, et aujourd'hui, plus de 50 organisations liste eux-mêmes sur la Page Spark PoweredBy, et des dizaines parlent de leurs cas d'utilisation lors d'événements.

En 2011, l'AMPLab a commencé à développer des composants de haut niveau sur Spark, comme Shark et Spark streaming. Ceux-ci et d'autres composants sont parfois appelés comme Berkeley Data Analytics Stack (ODB). Spark a été en open source en Mars 2010, et il a été transféré à Apache Software Foundation en Juin 2013, où il est maintenant un projet de haut niveau.

Définition

Apache Spark est un Framework open source de traitement, il est construit autour de la vitesse, la facilité d'utilisation et capacité de gérer des grands ensembles de don- nées qui sont de nature diverse (données de texte, données de graphes, etc.), Spark étend le modèle MapReduce pour soutenir efficacement plusieurs types de calculs, y compris le traitement itératif, les requêtes interactives et le traitement de flux.



Architecture de Spark

Avantages de Spark par rapport à Hadoop MapReduce

Spark est un fort Framework pour les futures grandes applications de données qui peuvent nécessiter des requêtes de faible latence, calcul itératif et traitement en temps réel. Spark a beaucoup d'avantages par rapport au Framework Hadoop MapReduce parmi eux on trouve:

- **La rapidité**

Spark est un environnement de calcul open source similaire à Hadoop, mais il a quelques différences utiles qui le rendent supérieur dans certaines charges de travail, il permet de charger l'ensemble de données en mémoire distribuée pour optimiser la charge de travail itératif et les requêtes interactives.

Spark peut exécuter les traitements de 10 à 100 fois plus rapidement que Hadoop MapReduce tout simplement en réduisant le nombre de lectures et écritures sur le disque.

- **Traitement itératif**

Il existe beaucoup d'algorithmes qui appliquent la même fonction à plusieurs étapes. Comme les algorithmes d'apprentissage, Hadoop MapReduce est basée sur un modèle de flux de données acyclique, c'est à dire que la sortie d'une tâche MapReduce précédente est l'entrée de la prochaine tâche MapReduce. Dans ce cas on perd beaucoup de temps dans l'opération d'E/S, alors dans Hadoop MapReduce entre deux opérations MapReduce, il existe une barrière de synchronisation et on a besoin de conserver les données sur le disque à chaque fois.

Mais avec Spark, le concept de RDD (Resilient Distributed Datasets) permet d'enregistrer les données sur la mémoire et préserver le disque seulement pour les opéra- tions de résultats. Ainsi il n'a pas tout une barrière de synchronisation qui éventuelle- ment pourrait ralentir le processus. Alors Spark permet de réduire le nombre de lecture/écriture sur le disque, donc le temps principal est lie au traitement des données car il n'existe pas un temps de E/S.

- **Requêtes interactives**

Pour le traitement dans les algorithmes d'extraction de données interactive où un utilisateur a besoin d'exécuter de multiples requêtes sur un même sous ensemble de données, Hadoop charge les mêmes données plusieurs fois à partir de disque selon le nombre de requêtes c'est à dire chaque requête a son propre lecteur de disque et son propre traitement MapReduce.

Mais Spark charge les données une seule fois, il stocke ces données dans la mé- moire distribué, ensuite il applique le traitement adéquat. Pour le traitement dans les algorithmes d'extraction de données interactive où un utilisateur a besoin d'exécuter de multiples requêtes sur un même sous ensemble de données, Hadoop charge les mêmes données plusieurs fois à partir de disque selon le nombre de requêtes c'est à dire chaque requête a son propre lecteur de disque et son propre traitement MapReduce.

- **Plus riche**

Spark fournit des API concises et cohérentes à Scala, Java et Python et Prend en charge plusieurs fonctions (actions et transformations), contrairement à Hadoop, on trouve seulement les deux fonctions Map et Reduce et un seul langage Java.

- **Facilité d'utilisation**

Spark vous permet d'écrire rapidement des applications en Java, Scala, ou Python avec des instructions simples et lisibles.

- **Généralité**

Du côté de la généralité, Spark est conçu pour couvrir un large éventail de charges de travail qui nécessitent au paravent des systèmes distribués distincts, y compris les applications de traitement en temps réel, les algorithmes itératifs, les requêtes interac- tives et le streaming. En soutenant ces charges de travail dans le même moteur, Spark est facile et peu coûteux de combiner les types de traitement différents, ce qui est sou- vent nécessaire dans les pipelines d'analyse de données de production.

- **Méthode streaming Real-Time de Spark à traiter des flux**

En cas de Hadoop MapReduce il est juste possible de traiter un flot de données stockées, mais avec Apache Spark il est ainsi possible de modifier les données en temps réel grâce à Spark streaming.

- **Traitement graphique**

Les développeurs peuvent maintenant aussi bien faire usage de Apache Spark pour le traitement graphique qui mappe les relations dans les données entre les diverses entités telles que les personnes et les objets.

- **Les algorithmes d'apprentissage**

Spark est livré avec une bibliothèque d'apprentissage appelé MLlib, elle fournit plu- sieurs types d'algorithmes d'apprentissage, notamment la classification, la régression, le regroupement et le filtrage collaboratif, ainsi que l'appui des fonctionnalités telles que l'évaluation du modèle et l'importation de données. Mais dans Hadoop il faut intégrer une bibliothèque d'apprentissage appelé Mahout.

- **Gestion rapide des données structurées**

Spark SQL est le module de Spark pour travailler avec des données structurées. Spark SQL permet d'interroger des données structurées comme un ensemble de don- nées distribuées (RDD) dans Spark, avec des API intégrées en Python, Scala et Java.

- **Généralité de stockage**

Spark utilise le système de fichier HDFS pour le stockage de données. Il fonctionne aussi avec n'importe quelle source de données compatible avec Hadoop y compris, HBase, Cassandra, etc.

- **Interactive**

Offre une console interactive pour Scala et Python. Ce n'est pas encore disponible en Java.

Déploiement

Exécuter des traitements lourds sur un cluster, piloter les nœuds esclaves, leur distribuer les tâches équitablement, et arbitrer la quantité de CPU et de mémoire qui sera allouée à chacun des traitements, tel est le rôle d'un gestionnaire de cluster. Spark offre pour l'instant trois solutions pour cela : Spark standalone, YARN et Mesos. Livré avec Spark, Spark Standalone est le moyen le plus simple à mettre en place. Ce gestionnaire de cluster s'appuie sur Akka pour les échanges et sur Zookeeper pour garantir la haute-disponibilité du noeud maître. Il dispose d'une console pour superviser les traitements, et d'un mécanisme pour collecter les logs des esclaves.

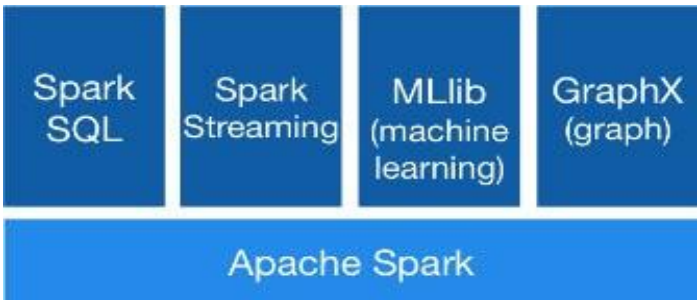
Autre possibilité, YARN le gestionnaire de cluster Hadoop, Spark peut s'exécuter dessus, et aux côtés de jobs Hadoop. Enfin, plus sophistiqué et plus généraliste, Mesos permet de configurer plus finement l'allocation des ressources (mémoire, CPU) aux différentes applications.

Composants de Spark

Parce que le moteur de base de Spark est à la fois rapide et polyvalent, il alimente de multiples composants de haut niveau spécialisés pour diverses charges de travail, tels que SQL ou l'apprentissage automatique. Ces composants sont conçus pour inter opérer étroitement, vous permettant de les combiner comme les bibliothèques dans un projet logiciel.

Spark Core : contient les fonctionnalités de base de Spark, y compris les compo- sants pour la planification des tâches, gestion de la mémoire, la reprise après incident, interaction avec les systèmes de stockage, et plus. Spark Core est également l'API qui définit les ensembles de données distribués élastiques (RDD), qui sont les principales abstractions de programmation de Spark. RDD représentent une collection d'objets répartis sur plusieurs nœuds de calcul qui peuvent être manipulés en parallèle. Spark Core offre de nombreuses API pour la construction et la manipulation de ces collections.

Autre que Spark Core API, il ya des bibliothèques supplémentaires qui font partie de l'écosystème de Spark et fournissent des capacités supplémentaires dans l'analyse des Big Data. Ces bibliothèques sont : Spark streaming, Spark SQL, Spark MLlib, Spark GraphX.



Les composants de Spark

Références

Müller, A. C., & Guido, S. (2016). *Introduction to machine learning with Python: a guide for data scientists*. " O'Reilly Media, Inc.".

Dean, J. (2014). *Big data, data mining, and machine learning: value creation for business leaders and practitioners*. John Wiley & Sons.

Ratner, B. (2017). *Statistical and Machine-Learning Data Mining:: Techniques for Better Predictive Modeling and Analysis of Big Data*. CRC Press.

Guy, H. (2015). Next generation databases: NoSQL, newSQL, and big data.

Stephenson, D. (2018). *Big Data Demystified: How to use big data, data science and AI to make better business decisions and gain competitive advantage*. Pearson UK.

Trovati, M., Hill, R., Zhu, S. Y., & Liu, L. (2015). *Big-data analytics and cloud computing*. Springer Berlin Heidelberg.

Provost, F., & Fawcett, T. (2013). Data science and its relationship to big data and data-driven decision making. *Big data*, *1*(1), 51-59.

Chen, Y., Chen, H., Gorkhali, A., Lu, Y., Ma, Y., & Li, L. (2016). Big data analytics and big data science: a survey. *Journal of Management Analytics*, *3*(1), 1-42.