SPARK

Valorisez vos données en temps réel avec Spark ML et Hadoop

SPARK

Valorisez vos données en temps réel avec Spark ML et Hadoop

Romain Jouin

DUNOD

Direction artistique : Élisabeth Hébert
Conception graphique : Pierre-André Gualino
Image de couverture : © andresr – iStock

11 rue Paul Bert, 92240 Malakoff
www.dunod.com
ISBN 978-2-10-079432-4

TABLE DES MATIÈRES

AVANT-PROPOS

◆ *The sexiest job of the XXIe century*

D'après le titre d'un article de Thomas Davenport[1] publié en octobre 2012 dans la très prestigieuse *Harvard Business Review*[2], **data scientist** serait le job le plus sexy du XXIe siècle. Cet article décrit l'émergence de l'importance de l'analyse des données dans la Silicon valley, et le nécessaire lien avec le business. Il décrit le data scientist comme « un hybride de pirate de données, d'analyste, de communicateur et de conseiller de confiance. La combinaison est extrêmement puissante et rare. » C'est à partir de cet article que le métier de data scientist a gagné en visibilité sur la scène internationale.

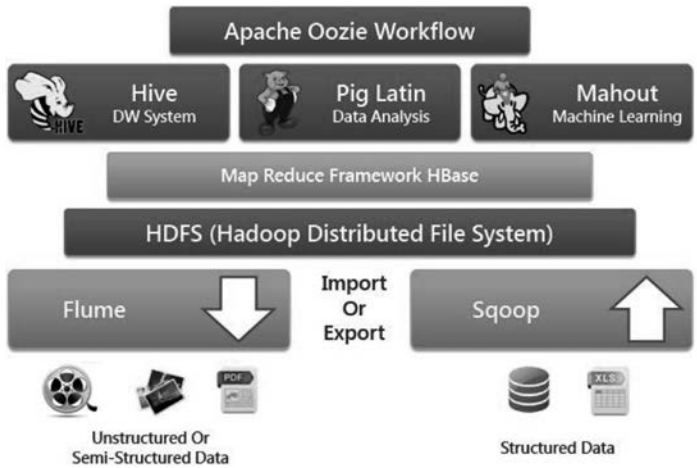
◆ *Spark : l'outil de base pour le big data*

C'est IBM qui en 2012 a donné la définition de base du big data, avec « *les 3 V* » :

1. ***V****olume* : plus d'information que ce qui tient sur un seul ordinateur.
2. ***V****ariété* : des informations de tout type : mail, texte, vidéo, images, tweets, cours de bourse, informations structurées, non structurées, semi-structurées…
3. ***V****itesse* : des informations qui arrivent en grande quantité, en temps réel.

À la fin des années 2000, tous les acteurs majeurs du web (Facebook, LinkedIn, Yahoo !, Google…) ont développé leur propre outil pour gérer et analyser les données. Ces outils s'appelaient Hive, Hadoop, Pig, Mahout…

Figure I.1 – Principaux outils de l'écosystème Hadoop 1.0.



1. http://www.tomdavenport.com/.
2. https://hbr.org/2012/10/data-scientist-the-sexiest-job-of-the-21st-century.

Cette multiplication des outils était la conséquence de la lutte technologique entre différents acteurs économiques qui voulaient créer leur avantage en développant leur propre outil big data. L'inconvénient de cette diversité d'outils était la nécessaire formation des programmeurs.

Une solution plus élégante et intégrée est venue du monde universitaire. Développé depuis 2009 par l'université de Berkeley, Spark propose un certain nombre de fonctionnalités standard intégrées dans un seul outil.

Il remplace notamment Hive (pour le SQL), Pig (pour la programmation), Mahout (pour les algorithmes d'apprentissage automatique) et Map-Reduce (pour la distribution des calculs) par différentes briques intégrées :

- ✓ **Spark Core** pour remplacer Map-Reduce
- ✓ **Spark SQL** pour le SQL
- ✓ **Spark MLlib** pour l'apprentissage automatique
- ✓ **Spark Streaming** pour l'analyse de données en temps réel
- ✓ **Spark GraphX** pour l'analyse de données représentant des graphes
- ✓ Des interfaces en Python, Java, Scala ou R pour la programmation

◆ *Le big data en France*

La conférence « big data, big value ? » de Télécom ParisTech en décembre 2012[3] marque l'émergence du concept de big data dans le débat public français. S'en sont suivis de nombreux investissements, tant financiers que politiques :

- ✓ Le big data devient l'un des sept piliers de « la nouvelle France industrielle »[4] en septembre 2013
- ✓ Les formations en big data apparaissent
- ✓ Les articles de presse se multiplient
- ✓ La Banque publique d'investissement (BPI) investit massivement dans les big data

Les premières formations en septembre 2013 hésitent encore dans le choix des technologies à enseigner. Plus que de big data, l'accent est mis sur le *machine learning* avec des outils comme ***R*** et ***Python***. Les concepts de *bases de données distribuées* (MongoDB, Cassandra…) et de *cloud computing* (Amazon Web Services, Google Cloud Platform…) étaient encore trop récents pour la formation en France.

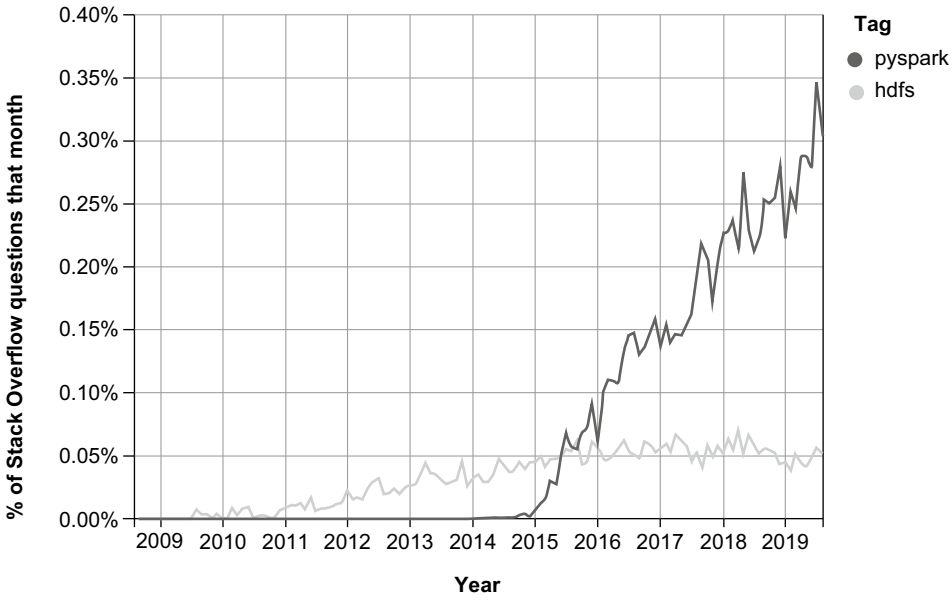
Finalement la simplicité d'usage et l'unification des fonctions proposés par Spark en a fait un outil de choix pour les formations et les équipes de data scientists dans le monde entier. Spark est devenu le point d'entrée de base pour toute personne voulant tester rapidement des concepts de big data.

---

3. https://www.telecom-paristech.fr/formation-continue/telecom-paristalks-les-entretiens-du-numerique/dec-2012-big-data-big-value/avis-experts.html.

4. https://www.economie.gouv.fr/files/files/PDF/nouvelle-france-industrielle-sept-2014.pdf.

Figure I.2 – Comparaison de l'évolution du nombre de recherches pour « pyspark » et « HDFS » sur Stack Overflow.



Source : https://insights.stackoverflow.com/trends?tags=pyspark%2Chdfs

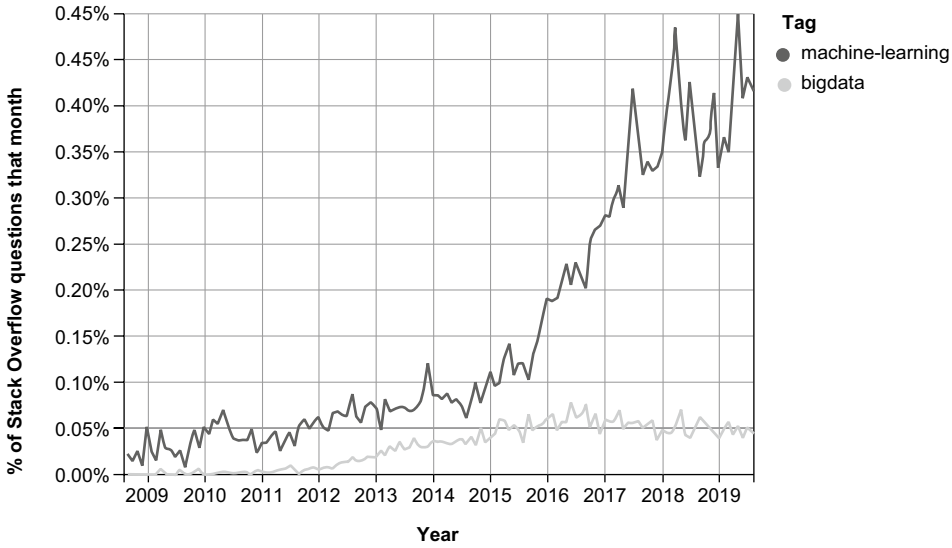
◆ *Big data* vs *machine learning* : quelles différences ?

En simplifiant, on pourrait dire que **le big data est l'ensemble des techniques qui permettent d'agréger plusieurs ordinateurs pour en créer un gros qui cumule leurs puissances. Tandis que l'apprentissage automatique (*machine learning*) est l'ensemble des techniques qui permettent d'analyser des données pour en extraire de l'information pertinente pour résoudre un problème.**

Depuis 2012 les entreprises ont souvent commencé leur aventure big data en achetant de gros ordinateurs, des « data lakes » et en recrutant des data scientists. Ensuite elles ont essayé de trouver des cas d'usage, ce qui s'avère nettement plus difficile. Acheter de la technique est plus simple que d'analyser les enjeux business... Or il est bien plus pertinent de commencer par réfléchir à quoi pourrait servir les données, pour remettre l'intérêt financier et stratégique de l'entreprise au centre de la pensée data. C'est une partie qui est souvent mise de côté par les data scientists dans les datalabs.

Pour profiter de la vague du *machine learning* il est donc nécessaire de penser aux cas business qui sont associés à l'analyse de données. C'est pourquoi nous allons mettre l'accent dans ce livre sur le lien entre l'analyse des données et leur intérêt business, en nous appuyant notamment sur la partie « machine learning » de Spark.

Figure I.3 – Comparaison de l'évolution du nombre de recherches pour « machine learning » et « big data » sur Stack Overflow.



Source : https://insights.stackoverflow.com/trends?tags=machine-learning%2Cbigdata

◆ *À qui s'adresse ce livre ?*

Ce livre est une introduction à Spark. Il s'adresse à des data scientists, data architects, ingénieurs DevOps, programmeurs et chefs de projets qui ont besoin d'acquérir les compétences fondamentales pour installer, configurer et utiliser Spark.

Il présuppose une connaissance en programmation objet, et notamment en Python. Le code Python étant très simple à lire ; il est possible de suivre la logique des codes sans mettre les mains au clavier pour un chef de projet qui voudrait juste avoir un aperçu des méthodologies des projets data.

Une certaine aisance avec des notions d'administration système sera la bienvenue pour configurer le réseau (cluster) d'ordinateurs.

◆ *Remarque sur le machine learning*

Ce livre n'est pas une reformulation de la documentation de Spark. **Il a pour vocation de vous aider à valoriser vos données dans votre entreprise. Or cela ne saurait se limiter à un enjeu technique.** Aussi avons-nous particulièrement développé la partie sur la méthodologie du *machine learning via* l'analyse d'une entreprise virtuelle de location de vélos en libre-service (chapitre 10).

Ainsi, dans ce chapitre, nous vous proposons un code de simulation d'utilisateurs de vélos, et de création de parcours. Nous nous mettrons ensuite dans la peau d'un data scientist embauché dans cette entreprise pour valoriser cette donnée :

- ✓ Que faire ?
- ✓ Comment regarder les données ?
- ✓ Quelles questions poser ?
- ✓ Comment valoriser toutes ces données ?
- ✓ À quoi peuvent-elles servir ?
- ✓ …

Ce chapitre est donc conçu pour vous apprendre Spark, et en même temps pour **vous guider dans les processus d'un projet data et sa valorisation au sein d'une entreprise**. Vous en sortirez avec **un outil et une méthode**.

◆ *Comment lire ce livre ?*

Après la mise en contexte proposée par les chapitres 1 et 2, le livre est structuré ainsi :

- ✓ Une présentation générale, orientée infrastructure/installation, de Spark et de HDFS : ce sont les chapitres 3 à 5.
- ✓ Une présentation technique de la programmation orientée Spark, qui vous entraîne à travers les bases de Spark : Spark core/SQL et Spark Streaming. Ce sont les chapitres 6 et 7.
- ✓ Ensuite nous attaquons les enjeux du machine learning, avec une présentation du concept et de ses enjeux au chapitre 8.
- ✓ Enfin nous présentons les implémentations du machine learning dans Spark dans le chapitre 9, ce qui nous permet *in fine* de mettre en place un algorithme de prédiction en temps réel en liant Spark ML et Spark Streaming.

À noter que nous vous proposons un environnement virtuel avec Docker qui vous permettra de tester Spark dans des conditions simplifiées, sur un seul ordinateur. En effet la beauté de Spark réside aussi dans le fait qu'un code qui fonctionne sur un seul ordinateur fonctionnera également sur un cluster de plusieurs ordinateurs. Vous pouvez donc vous entraîner à connaître la syntaxe et les objets Spark avant même d'installer un cluster. Nous vous expliquons la mise en place de Docker dans le chapitre 6.

◆ *Remarques techniques*

Il est courant d'utiliser Linux dans les environnements big data. Le livre est donc basé sur un système d'exploitation compatible avec Linux, plus précisément Ubuntu et Mac OS. Les utilisateurs Windows devront sans doute adapter les informations, notamment sur la gestion de l'infrastructure.

Le livre est basé sur l'API Python de Spark. Cette dernière ne comprend pas la librairie d'analyse de graphes, GraphX. Nous ne la présentons donc pas dans ce livre.

◆ *Remerciements*

Merci à Mikaël Dautrey pour sa relecture attentive.

Merci à Jean-Luc Blanc, pour m'avoir permis de faire ce livre, et à Maxine Pouzet de m'avoir soutenu pendant tout ce projet.

1 Spark et le big data

Objectifs

Dans ce chapitre, nous allons découvrir comment le big data a émergé, quelles sont les nouvelles problématiques qu'il pose, et pourquoi les technologies du XXe siècle n'arrivent pas à y faire face :

- **La croissance des volumes de données liée à Internet et la numérisation**
- **L'avancée des algorithmes de calculs distribués**
- **La montée en puissance des réseaux Ethernet**
- **La hiérarchie entre l'utilisation de la mémoire RAM et celle du disque dur**
- **La fin de la loi de Moore**

Il s'agira aussi de savoir quelles sont les réponses apportées par Hadoop, et pourquoi dans ce contexte Spark s'impose comme une bonne solution pour commencer à analyser les données. Nous explorerons donc autant les enjeux techniques que logiques qui sont apparus avec les grandes quantités de données qui apparaissent au début du XXIe siècle.

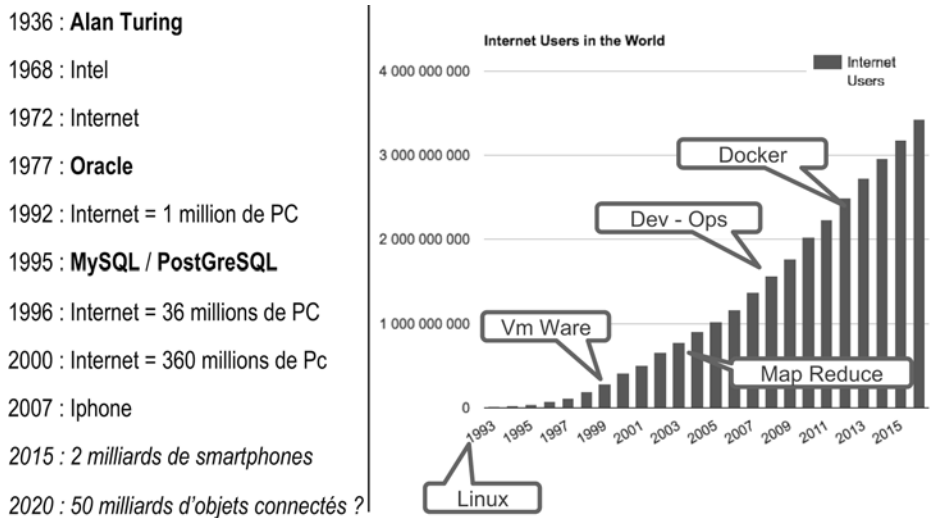
1.1 L'ÉMERGENCE D'INTERNET

La figure 1.1 nous montre les grandes étapes de l'informatique, depuis sa théorisation par Alan Turing entre les deux guerres, jusqu'à l'émergence de l'Internet des objets aujourd'hui. Les entreprises historiques de l'informatique (Intel, Microsoft, Oracle…) ont été créées au milieu du XXe siècle et les bases de données les plus populaires (MySQL/PostgreSQL...) avant l'expansion d'Internet. Les outils historiques se sont trouvés débordés par la croissance exponentielle du nombre d'utilisateurs et de données générés par Internet. Un certain nombre de nouvelles technologies et de nouveaux concepts ont émergé pour faire face à cette nouvelle situation et sont à la base du big data :

- ✓ **Linux**, système d'exploitation utilisé dans 80 % des serveurs Internet

- ✓ Les **systèmes de virtualisation** comme **VmWare** et **Docker**
- ✓ L'algorithme **MapReduce** qui propose un nouveau paradigme de programmation : en passant de logiciels pensés pour une seule machine de Turing/un seul ordinateur à des logiciels pensés pour plusieurs machines de Turing/plusieurs ordinateurs
- ✓ L'iPhone et l'émergence des **app-stores** qui mettent en avant la **rapidité de développement** et **l'agilité** dans le business modèle des éditeurs logiciels

Figure 1.1 – 90 ans d'informatique.



1. 2 OÙ CONSERVER LES DONNÉES ?

Comprendre les big data nécessite de comprendre l'architecture des ordinateurs. Les données sont stockées sur des disques durs. Les disques durs sont l'un des points de faiblesse des ordinateurs, parce qu'ils sont très lents, ce qui pose problème lorsqu'on a beaucoup de données à lire et à stocker. Un disque dur fonctionne à la manière d'une platine de lecture vinyle : une tête de lecture se déplace pour aller lire de l'information sur des pistes qui contiennent les fichiers. Le déplacement de la tête de lecture et la vitesse de rotation du disque rendent le processus de lecture assez lent, comparé à d'autres supports sans parties mobiles (RAM, disques SSD).

Sur les meilleures cartes mères d'ordinateurs de bureau on peut mettre une dizaine de disques durs, pour environ 500 €. Les disques les plus gros contiennent environ 6 To et coûtent environ 300 €. Pour près de 3 500 € on peut donc obtenir un ordinateur avec 60 To de stockage. Mais quel serait le temps nécessaire pour lire ces 60 To de manière linéaire ? Imaginons que les disques soient en interface SATA :

- ✓ 60 To = 60 000 Go = 60 000 000 Mo
- ✓ Débit SATA = 600 Mo / sec
- ✓ Durée de lecture : 60 000 000 Mo/600 Mo/sec = 100 000 sec = 27 heures

Or cette durée théorique de 27 heures peut facilement être multipliée par deux ou trois en pratique. Avec 3 500 € on peut donc acquérir 60 To de stockage, mais il nous faudra **2 à 3 jours** pour en lire le contenu… Voilà le problème auquel le big data cherche à répondre.

Figure 1.2 – Éléments d'un disque dur.

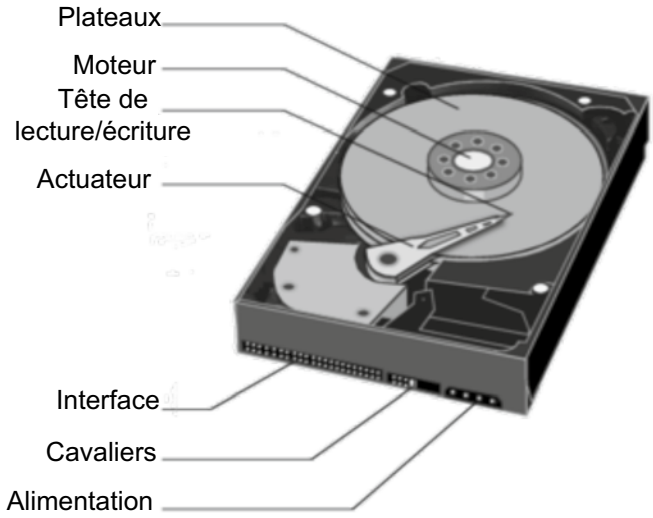


Figure 1.3 – Différentes interfaces de disques durs, avec leur débit théorique.

IDE-ATA
Débit de 133 Mo/s

SCSI
Débit de 5 à 600 Mo/s

S-ATA
Débit de 150 à 600 Mo/s

Une solution propriétaire, comme le système de stockage DS8870 d'IBM propose des débits beaucoup plus importants, avec un temps de lecture d'une heure, mais pour 2 millions d'euros…

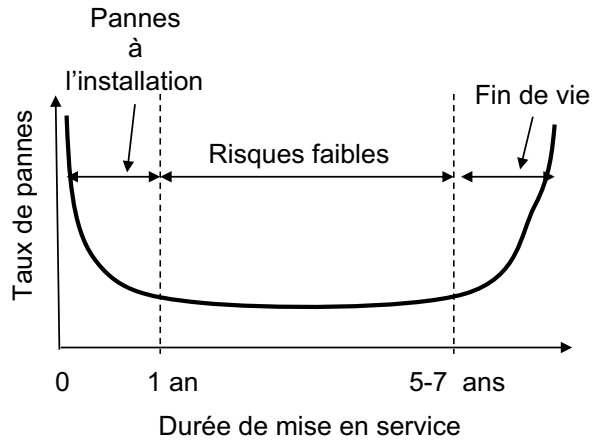
Ce système possède 500 disques durs et permet une lecture en parallèle des données, ce qui explique la vitesse de lecture accélérée.

Figure 1.4 – La solution de stockage DS8870 proposée par IBM : 70 To, 80 minutes de temps de lecture – 2 M d'euros.

1.3 LE MTBF : MEAN TIME BEFORE FAILURE

Le problème quand on a 500 disques durs, c'est qu'ils risquent de tomber en panne…

Figure 1.5 – Taux de panne selon la période de vie d'un disque dur. Les disques tombent en panne à l'installation, ou bien après quelques années.



Avec une durée de vie moyenne de cinq ans, un administrateur technique qui gère 500 disques devrait voir 100 disques tomber en panne par an, soit deux par

semaine, soit un disque à changer tous les trois jours… Cela engendre des coûts et des risques :

- ✓ coût matériel,
- ✓ coût d'arrêt de services,
- ✓ instabilité du système,
- ✓ perte d'information.

C'est pourquoi il fallait historiquement faire des sauvegardes et avoir des systèmes RAID. Aujourd'hui, une nouvelle solution existe : le stockage distribué.

1.4 STOCKAGE DISTRIBUÉ ET HDFS

Hadoop résout le problème lié au MTBF en dupliquant la donnée (par défaut trois copies). ***HDFS (Hadoop Distributed File System)*** est un outil qui s'occupe de copier les informations à enregistrer d'un ordinateur (on dira aussi un ***nœud***) vers d'autres, automatiquement. L'avantage de ce système est que si un disque dur tombe en panne, on peut récupérer l'information dans d'autres disques.

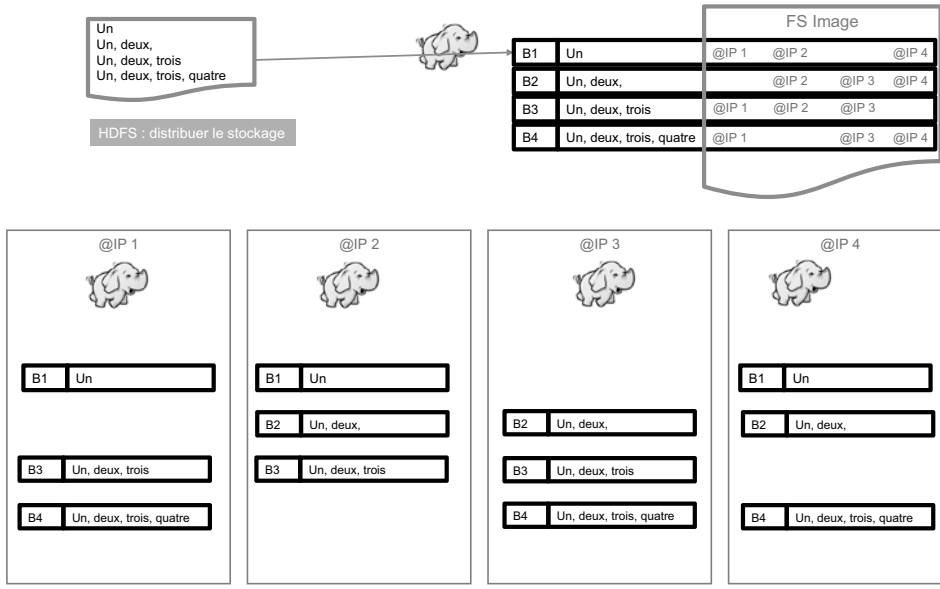
Les fichiers sont dupliqués sur HDFS en étant tout d'abord divisés en blocs.

Figure 1.6 – L'éléphant (jaune) est le logo d'Hadoop et de HDFS. Ce système permet de dupliquer des informations d'un seul disque dur vers plusieurs.

Imaginons un cluster HDFS de quatre machines (ci-dessous les quatre cadres avec des adresses IP), et un fichier de quatre lignes enregistré sur ce cluster :

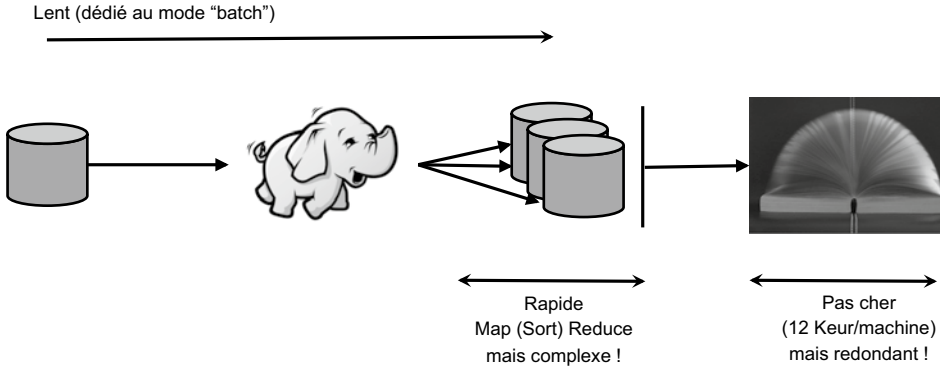
- ✓ Le fichier serait donc divisé en quatre blocs : B1, B2, B3, B4
- ✓ Chaque bloc serait dupliqué par le facteur de réplication, par défaut trois fois, sur le cluster
- ✓ Le nœud maître (***NameNode*** dans la terminologie HDFS) enregistre l'emplacement de chaque bloc (dans son fichier ***FS Image***)
- ✓ Chaque nœud esclave (***DataNode*** dans la terminologie HDFS) enregistre des blocs de fichiers

Figure 1.7 – Schéma d'un cluster HDFS avec un ***NameNode*** en haut et quatre ***DataNode*** en bas. Notez la **réplication des blocs** B1, B2, B3, B4 dans les différents DataNodes.



HDFS est gratuit. Et l'autre intérêt de ce système est qu'il permet de faire une **lecture parallélisée des fichiers**. On peut lire théoriquement avec une durée de lecture divisée par le nombre de blocs créés. Cette lecture parallélisée s'adapte parfaitement à l'algorithme MapReduce publié par Google en 2003. Dans cette première version, les résultats des calculs intermédiaires étaient écrits sur disque dur, ce qui ralentit considérablement la vitesse des calculs. Spark change cela en enregistrant les résultats intermédiaires en mémoire.

Figure 1.8 – HDFS est lent à l'écriture mais rapide à la lecture.



La solution de stockage d'IBM est chère notamment parce qu'elle implémente une version physique du stockage distribué proposé par HDFS, qui en est une version logicielle. En changeant le physique en logiciel, on diminue les coûts.

1.5 LA FIN DES BASES DE DONNÉES SQL ?

Les **bases de données classiques** sont basées sur des modèles conceptuels de données. Dans ces systèmes le ***système de gestion des bases de données*** (SGBD) demande de préciser le *modèle conceptuel de données* (MCD) à utiliser avant d'enregistrer de la donnée. Cela afin de minimiser l'espace disque utilisé, **en garantissant la non-duplication des données (*3e forme normale*)** afin de diminuer les coûts de stockage.

Avec la baisse drastique des coûts de stockage, limiter l'espace disque utilisé n'est plus un enjeu. Dès lors on peut mettre en œuvre de nouveaux concepts d'enregistrement de la donnée, avec une duplication des informations. Cela donne naissance à la notion de **bases de données *dénormalisées*** *orientées documents*, dont certaines sont basées sur le formalisme JSON (*Javascript Simple Object Norm*). Dans ces bases *orientées documents* **les informations sont dupliquées** de telle manière que chaque document (JSON) contienne toutes les informations nécessaires à son utilisation.

Par exemple, dans une base d'achat on aurait classiquement une table avec le nom des acheteurs, une table avec le nom des produits uniques, et une table de liens (montrant qui a acheté quoi). En enregistrement orienté document, on aura un document JSON pour chaque enregistrement, dans lequel le nom de l'acheteur sera écrit pour chaque achat, ainsi que le nom du produit. **Il n'y a plus d'unicité des informations :** elles sont dupliquées. C'est possible grâce à la baisse du coût des disques durs.

Tableau 1.1 – Différences et vocabulaire autour des bases de données classiques (MCD - OLPT) et les nouvelles bases de données (JSON-OLAP).

Intitulé phare	MCD-OLTP : Online Transaction Processing	JSON - OLAP : Online Analytical Processing
Caractéristique	Transactions garanties	Pas de transactions
Optimisation	Lecture et écriture	Principalement en lecture
Schéma	Schéma défini	Sans schéma
Utilisateurs	Banques Systèmes de sécurité	Sites Web Applications non critiques
Exemple de bases de données majeures	Oracle/Access PostgreSQL/MySQL	MongoDB/CouchDB HBase/Cassandra
Vocabulaire	MCD/Schéma/Relationnel UML/MERISE Référentiel/SQL Triggers Dictionnaires de données Silos/Logiciels/Licences/BI	NoSQL/Schemaless machine learning/Prédictif Apprentissage (Non) Supervisé Partage/API/ Open Sources/Dashboard Data Visualisation

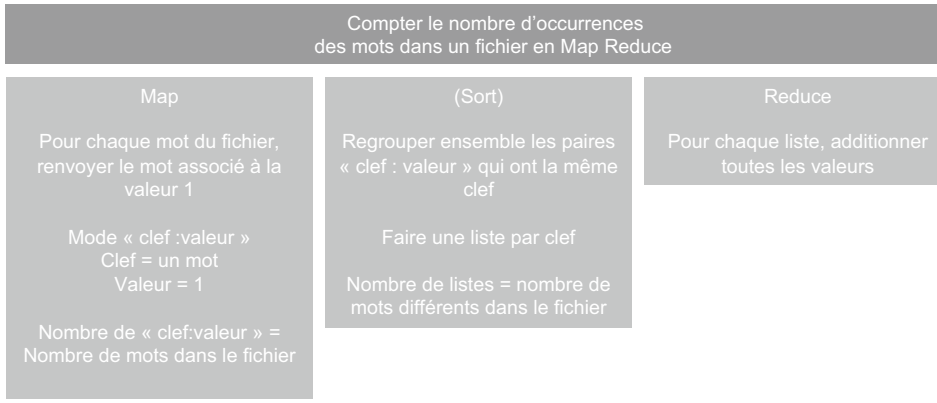
Cela permet d'envoyer des documents de données (JSON) qui sont autosuffisants en termes de données contenues, et qui permettent donc des calculs indépendants, parallélisés sur les différents CPU d'un ordinateur, ou distribués *via* le réseau sur un cluster d'ordinateurs. Ces idées sont à l'origine de la création de l'algorithme MapReduce.

1. 6 L'ALGORITHME MAPREDUCE

L'algorithme MapReduce est une évolution majeure dans l'histoire de l'informatique puisqu'il propose de penser les logiciels comme devant fonctionner sur plusieurs machines de Turing (plusieurs ordinateurs)'.

Quand on apprend *MapReduce* l'équivalent du « hello world » standard'consiste à compter le nombre d'occurrences des mots dans un fichier.

Figure 1.9 – Description de la logique algorithmique pour compter les mots dans un fichier en Map-Reduce.



MapReduce a été proposé par Google en 2003, puis mis en open source *via* Hadoop en 2009. La phase de ***Map*** est distribuée sur les différentes machines du cluster, chaque machine renvoyant au *master* une liste de résultats intermédiaires. C'est cette liste que le master va *réduire* en un seul résultat *via* la fonction ***Reduce*** qui n'est exécutée que de manière centrale.

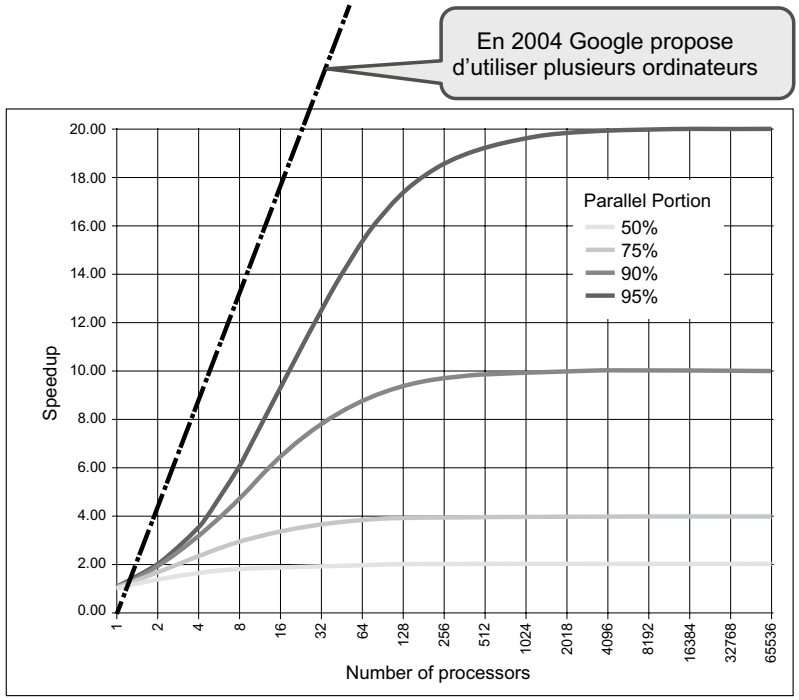
L'un des intérêts de la distribution du calcul, c'est la capacité à **dépasser les contraintes de la loi d'Amdahl**. Cette dernière spécifie que le gain en vitesse d'exécution d'un programme n'est pas proportionnel au nombre de CPU présents dans l'ordinateur : la nécessaire synchronisation entre les threads impose une limite au gain de vitesse possible.

C'est ce qu'on voit dans le schéma ci-dessous :

- ✓ Avec 2 CPU on peut théoriquement aller jusqu'à 2 fois plus vite
- ✓ Avec 4 CPU, on va entre 2 et 3.75 fois plus vite
- ✓ Avec 16 CPU on va entre 2 et 9 fois plus vite
- ✓ La limite est atteinte avec 1024 CPU où l'on ne va que 20 fois plus vite

En distribuant le calcul, on s'affranchit théoriquement de cette limite, en déportant le problème sur le réseau. On a alors un gain de vitesse qui n'est évidemment pas linéaire, mais qui n'est plus limité : il suffit de rajouter des ordinateurs pour aller plus vite. D'où l'intérêt des clusters.

Figure 1.10 – La loi d'Amdhal (1922-2015) indique la limite théorique du calcul parallélisé sur un seul ordinateur.



Nous présentons dans les trois schémas suivants le fonctionnement de MapReduce sur l'exemple du calcul du nombre d'occurences de chaque mot dans un fichier.

Figure 1.11 – Les calculs sont distribués sur plusieurs ordinateurs, c'est ce qu'on appelle le *mappage* de fonctions.

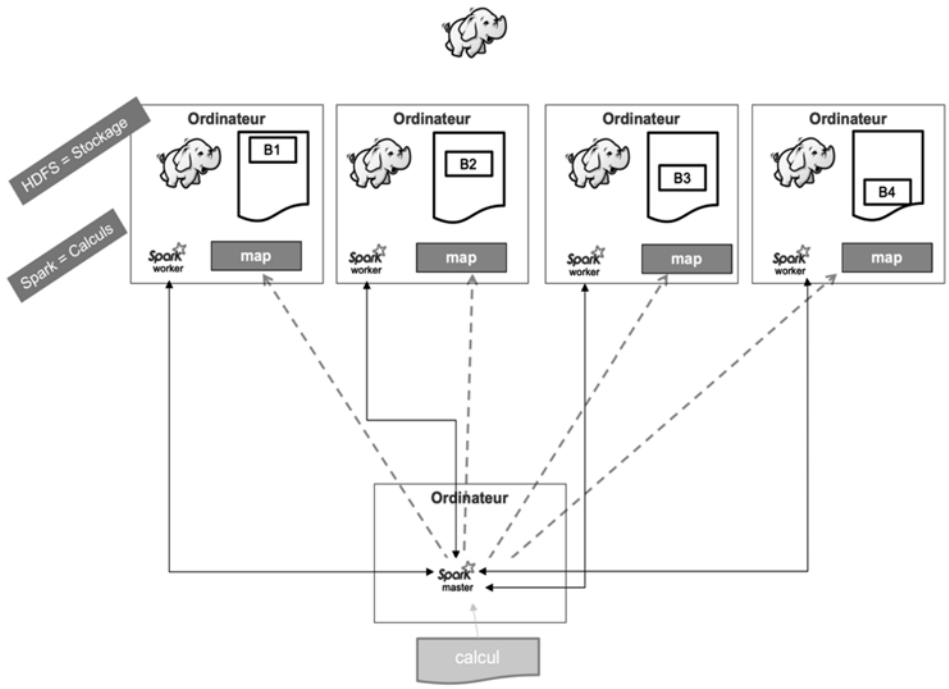


Figure 1.12 – Chaque fonction *mappée* renvoie des informations au processus centralisée sur le *master*.