BIG DATA : UNE NOUVELLE FORME D'INTELLIGENCE COLLECTIVE

Mini-série de Billets #3 : Des moteurs de recherche au Big Data. Et la résolution des conflits ?

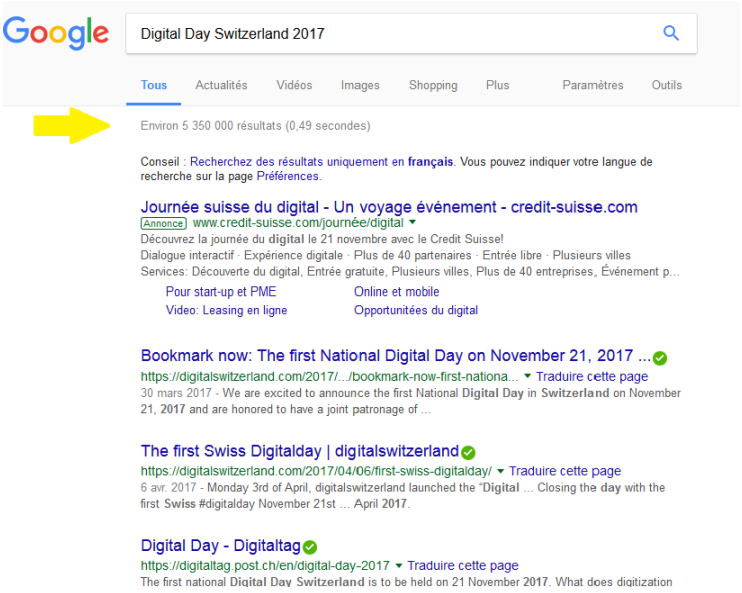
Billet #3C Hadoop – des logiciels : MapReduce et Hadoop Distributed File System, un nouveau paradigme de programmation

Souvenez-vous, qu'initialement, loin de toute idée Big Data dans son acception actuelle, le problème à régler était celui de l'indexation des pages Web afin d'améliorer l'efficacité des moteurs de recherches. La solution trouvée par Google, et implémentée par Doug Cutting avec Hadoop, consistait à découper le problème d'indexation des pages web en sous-problèmes distribués pour exécution dans un cluster d'ordinateurs. Nous avons vu au Billet #3B ce qu'est un cluster du point de vue architectural, physique. D'un point de vue logiciel, un cluster Hadoop se définit tout simplement comme un cluster sur lequel Hadoop a été installé, soit deux éléments : premièrement, l'ensemble des classes JAVA d'implémentation de l'algorithme MapReduce et, deuxièmement, le *Hadoop Distributed System File* (HDSF) comme système de fichiers sur l'ensemble des disques durs des nœuds du cluster[1]. On quitte ainsi les architectures centralisées pour aller vers les architectures distribuées. Dans une perspective logicielle, cela consiste à paralléliser l'exécution du travail, à le répartir entre un ensemble d'ordinateurs interconnectés considérés comme un tout.

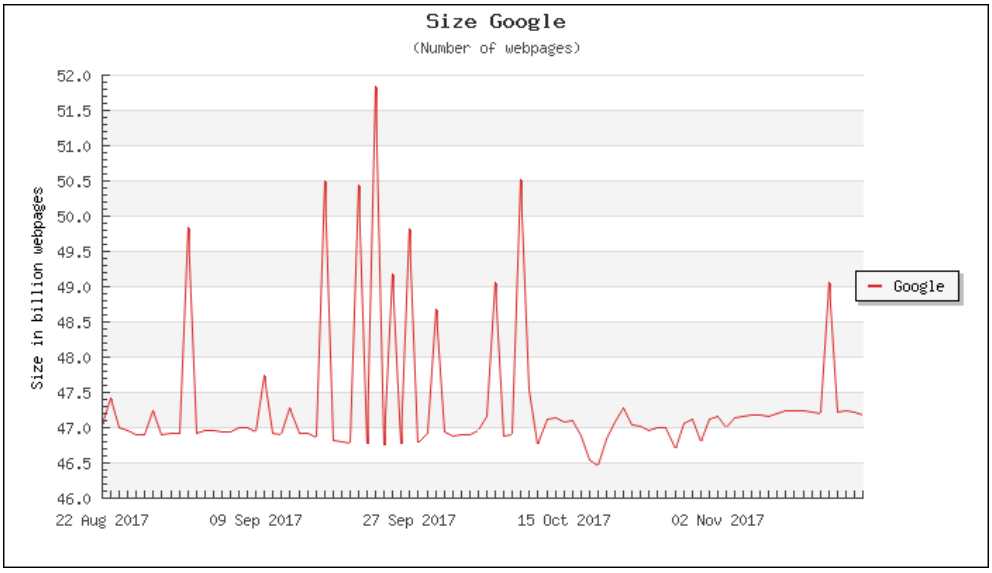
Toujours dans la perspective de réaliser le double objectif de cette mini-série de Billets #3 - cerner la notion de Big Data, d'une part, et clarifier l'impact de ces innovations sur les modes résolution des conflits, d'autre part – ce Billet #3C portera sur les notions suivantes :

- Que signifie « paralléliser l'exécution du travail » ?
- Qu'est-ce que l'algorithme MapReduce ?
- Qu'est-ce que le *Hadoop Distributed System File* (HDSF)?

Au spectacle cela représenterait les activités en coulisses. A l'opposé, voici par un exemple tiré de l'actualité récente, ce qui s'offre à nous sur scène. Le résultat en image de ma recherche faite le 20 novembre 2017, veille du *Digital Day Switzerland 2017* se présente ainsi :



Le moteur de recherche a trouvé environ 5.35 millions de résultats parmi l'ensemble des pages Web indexées par Google. Je n'ai pas trouvé le nombre précis de pages indexées par Google à ce jour mais selon une estimation, la taille de l'index de Google s'élève à environ 4.25 milliards de pages ! Le lundi 20 novembre 2017, toujours de manière estimée, le Web indexé contenait globalement au moins 4.49 milliards de pages[ii].



Source : DE KUNDER Maurice, *Daily estimated of the World Wide Web*, http://www.worldwidewebsize.com/ (20/11/2017)

Le temps de traitement de ma requête a été ultra rapide : en 0.49 secondes. Efficace, non ?! Allons voir maintenant ce qui se passe en coulisses.

Le parallélisme : pierre angulaire du passage de la centralisation vers la distribution

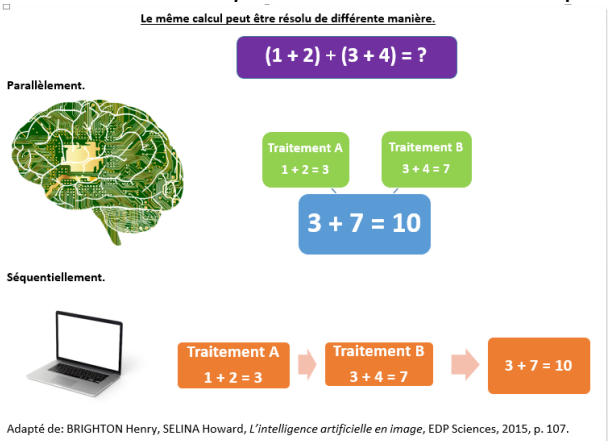
Le parallélisme massivement parallèle (indépendant) - Définition.

Le parallélisme peut-être de trois types, fonctions de l'infrastructure utilisée, du niveau de divisibilité et des contraintes du problème à traiter : asynchrone simultané, pipeline, massivement parallèle (indépendant). Le sens donné au mot parallèle peut varier d'un type de parallélisme à l'autre. Par conséquent, afin de ne pas compliquer inutilement le propos, Hadoop fonctionnant avec le dernier type de parallélisme et constituant le fil rouge de cette mini-série de Billets, je me limiterai au *parallélisme massivement parallèle*[iii]. Il se définit comme la division du « *traitement en tâches COMPLETEMENT indépendantes, qui sont exécutées de façon parallèles (…) sur les nœuds d'un cluster en architecture shared-nothing* »[iv].

Parallèle versus séquentiel.

De manière générale, le *Larousse*, donne de l'adjectif *parallèle* les définitions suivante : «*1) Se dit de droites coplanaires ou de plans sans point commun ou confondus ; 2) Qui est dirigé selon une droite ou un plan parallèle : Mettez vos skis bien parallèles ; 3) Qui se développe dans la même direction que quelque chose d'autre, semblable : Action politique parallèle de deux partis ; 4) Se dit d'activités, d'organismes qui se développent en marge d'un cadre légal ou officiel : Marché parallèle de l'or. Police parallèle* »[v]. *Séquentiel* signifie « *qui appartient, se rapporte à une séquence, à une suite ordonnée d'opérations* » [vi]. On voit de ces définitions que la notion d'indépendance, ou l'absence indépendance, participe à leur signification.

Dans le domaine informatique, parallèle s'oppose à séquentiel, on parle de calcul séquentiel en référence à « *l'exécution d'un traitement étape par étape, où chaque opération se déclenche que lorsque l'opération précédente est terminée, y compris lorsque les deux opérations sont indépendantes* » [vii]. Schématiquement, ça donne cela :



Adapté de: BRIGHTON Henry, SELINA Howard, *L'intelligence artificielle en image*, EDP Sciences, 2015, p. 107.

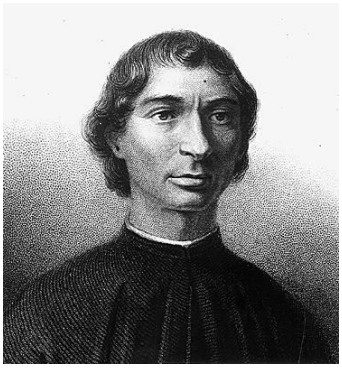
Dans la première variante, les traitements A et B sont exécutés en même temps de manière totalement indépendante alors que dans la seconde variante, le traitement B ne s'exécute qu'une fois le traitement A achevé.

Ce schéma appelle une petite parenthèse humaine, rafraichissante au milieu de cette technologie : notre cerveau humain fonctionne de manière massivement parallèle, c'est pour cela par exemple que l'on peut reconnaître extrêmement rapidement – en un dixième de seconde - nos proches[viii].

Le calcul séquentiel est fréquemment utilisé du fait de sa compatibilité avec tout type d'opération, la puissance des calculateurs constituant toutefois sa limite, là où le calcul parallèle n'est limité que par le nombre de calculateurs[ix]. Nous avons vu dans le Billet #3B, au paragraphe sur la *scalabilité horizontale,* que « passer à l'échelle, monter en charge » en augmentant le nombre de nœuds dans un cluster représente un concept central pour comprendre l'évolution allant de l'amélioration des moteurs de recherche au Big Data, ainsi que la popularité et le choix des clusters pour traiter la déferlante de données à laquelle nous assistons aujourd'hui.

Diviser pour régner : gros volumes de données, clusters et tâches indépendantes

Le traitement de gros volumes de données de formats variés[x] requiert l'utilisation d'un cluster d'ordinateurs. Cette utilisation - et donc Hadoop - est subordonnée à une condition *sine qua non* : le problème à résoudre est divisible en tâches indépendantes, tâches qui n'ont pas besoin du résultat d'une autre tâche pour continuer. Un problème divisible. Eh oui, la devise « Diviser pour régner », chère à l'homme politique et écrivain italien Niccolò Machiavelli (Machiavel, 1469 – 1527)[xi], entre autres, s'applique également en informatique[xii].



Source : http://www.larousse.fr/encyclopedie/litterature/Machiavel/175016

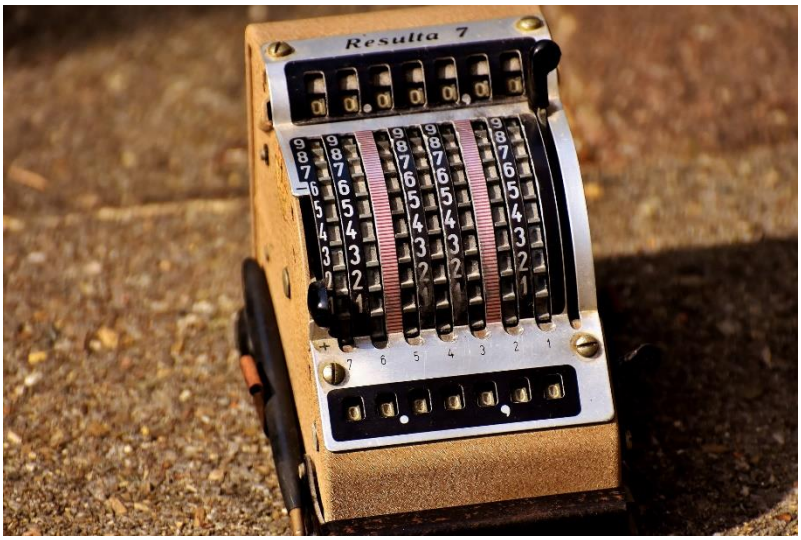
Avant de voir plus en détails comment Hadoop incarne cette devise « machiavélique » dans une approche nouvelle du stockage (HDFS) et du traitement des données (MapReduce), soulignons que le nouveau paradigme proposé par Hadoop repose sur un lien nécessaire entre des éléments logiciels (HDFS et MapReduce) et une architecture particulière : un cluster shared-nothing maître/esclave qui rend l'indépendance des tâches effectivement possible. *Shared-nothing* - rien n'est partagé : chaque nœud possède son propre disque dur, sa propre mémoire, son propre processeur. L'indépendance est complète, la totale « parallélisabilité » des tâches constitue une possibilité, certes, une nécessité, assurément. Impérativement, le problème doit être parallélisable. Dans une telle architecture, faute de communication entre les nœuds, il devient obligatoire de découper le problème à traiter en plusieurs sous-tâches

qui peuvent s'exécuter parallèlement, de manière indépendante, isolée sur un nœud du cluster. Nous retombons ainsi sur la condition *sine qua non* au traitement de gros volumes de données posée ci-dessus, et la boucle est bouclée[xiii] !

Nota bene – parallélisme dans le cluster n'est pas parallélisme à l'intérieur d'un nœud.
Hadoop fonctionne avec le parallélisme massivement parallèle entre les nœuds du cluster. Les tâches sont répliquées sur tous les nœuds du cluster et chaque nœud reçoit un bloc de fichier, une partie des données, qui y sont traitées. Il en va différemment à l'intérieur des nœuds, où les tâches ne sont pas exécutées en parallèle, mais séquentiellement.[xiv].

MapReduce, le cœur d'Hadoop[xv]

Il fut un temps où traiter des données ça donnait cela …



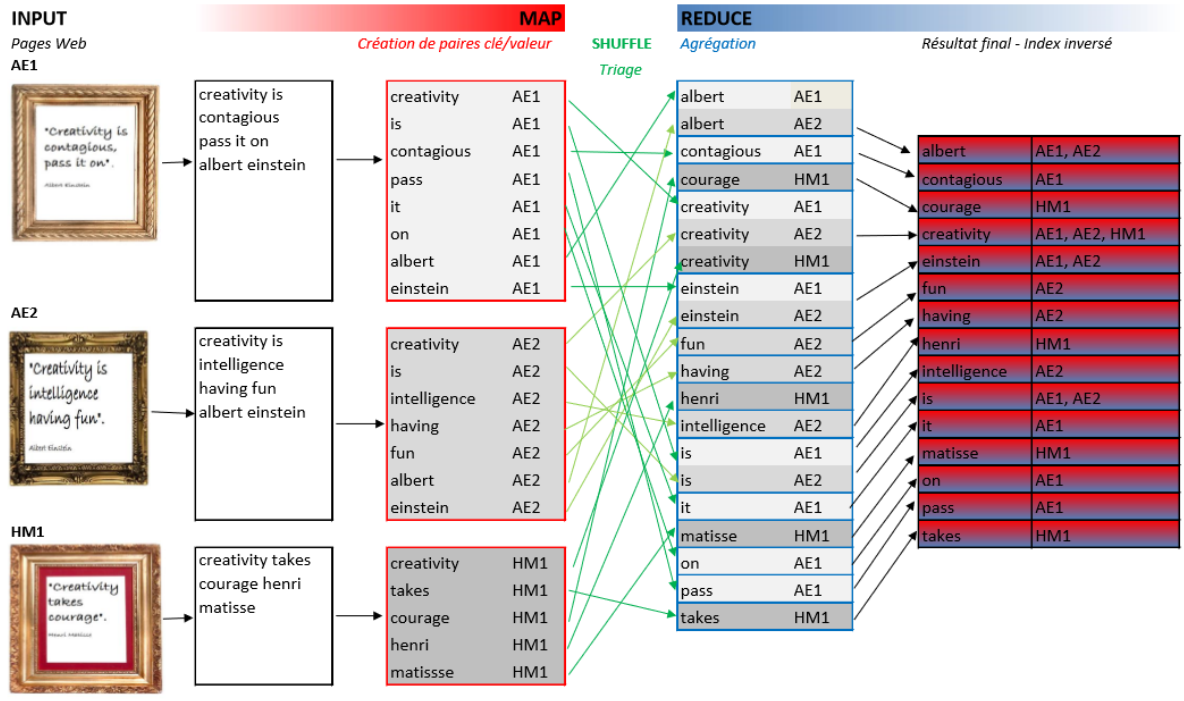
Source : Pixabay

Tempi passati…. L'objectif initial de Google au moment de la conception du MapReduce était le suivant : *« We designed a new abstraction that allows us to express the simple computations we were trying to perform but hides the messy details of parallelization, fault-tolerance, data distribution and load balancing in a library »*[xvi]. On se souvient que le problème à résoudre originellement était celui de l'indexation des pages Web et de l'amélioration des moteurs de recherches, et cela uniquement !

On retrouve Machiavel. Le MapReduce suit une approche de programmation de type « divide and conquer » où tout problème est divisé en tâches, chacune d'entre elles étant isolée dans un nœud pour traitement[xvii]. Expliqué, sous la plume de Jeffrey Dean et Sanjay Ghemawat, pères du MapReduce de Google, ça donne cela : *« MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate pairs, and reduce function that merges all intermediate values associated with the same intermediate key »*[xviii]. Ainsi, la solution apportée par Google, fondée sur deux uniques fonctions Map et Reduce, était la suivante : découper le problème d'indexation des pages web en sous-problèmes distribués dans l'ensemble du cluster pour y être exécutés. Pour ce faire, un index

inversé est construit par mot-clé apparaissant dans chaque page du Web. Un index inversé correspond à l'index d'un livre, à savoir selon le *Larousse* à la « *liste alphabétique d'auteurs, de matières, de mots clés, etc., apparaissant dans un ouvrage, avec des références permettant de les retrouver* »[xix] ; cela est extrêmement pratique, convenons-en. Ce principe de recherche sur le Web perdure de nos jours[xx]. Selon ses pères, un index inversé se définit ainsi : « *The map function parses each document,and emits a sequence of (word, document ID) pairs. The reduce function accepts all pairs for a given word, sorts the corresponding document IDs and emits a (word, list(documentID)) pair. The set of all output pairs forms a simple inverted index. It is easy to augment this computation to keep track of word positions* »[xxi].

En schéma, la construction de l'index inversé par Google avec MapReduce, ça donne cela :



Construction d'un index inversé avec MapReduce

Et en mots, ceci :

1) La phase *MAP* : création de paires clé/valeur.

Les données (pages Web, *input*), préalablement découpées en blocs distribués à l'ensemble des nœuds du cluster, sont transformées en paires clé/valeur.

- une clé = un mot d'une page Web,
- une valeur = un indice référençant la page où le mot se trouve.

Dans le schéma ci-dessus les indices sont AE1, AE2, HM1. Chaque page a un indice unique. La tâche d'attribuer à chaque mot (clé) d'une page web un indice (valeur) s'exécute parallèlement, en même temps dans tout le cluster sur les nœuds qui le composent.

2) La phase *Shuffle* : triage des paires clé/valeur.

Durant cette phase, il est procédé au triage par clé (mot) de toutes les paires générées pendant la phase Map.

3) La phase *REDUCE* : agrégation.

Les valeurs des clés sont agrégées, regroupées, consolidées et l'index inversé est finalisé.

Quelle prouesse ! Cela ne repose que sur deux fonctions : Map et Reduce. L'utilisateur les définit, l'orchestrateur – le chef d'orchestre du cluster - les assigne à chaque nœud, planifie les modalités d'exécution, le cluster apparaissant ainsi comme un unique ordinateur aux yeux des utilisateurs. MapReduce permet également de compter des occurrences de mots dans un texte, un livre par exemple, ou de joindre deux tables relationnelles. Il se révèle donc précieux dans des applications de recherche web, dans toutes les activités fondées sur de grandes quantités de texte, dans de nombreux types de traitement des données tels que les fouilles de données, les graphes, etc[xxii].

Un changement de paradigme de programmation

Nous avons vu que le défi d'amélioration des moteurs de recherche a apporté un changement de paradigme infrastructurel : centralisé vers distribué. Ce défi induit également un changement de paradigme de programmation : séquentielle vers parallèle et distribuée. « *Hadoop implements a computational paradigm named Map/Reduce, where the application is divided into many small fragments of work, each of which may be executed or re-executed on any node in the cluster* »[xxiii].

Nouveau modèle algorithmique consistant à découper un problème en tâches indépendantes, l'adoption du MapReduce implique de changer la question initiale à se poser pour résoudre une tâche. Pour calculer la somme de dix nombres consécutif, le raisonnement traditionnel s'exprime ainsi : « *Comment sommer dix nombres de manière séquentiel ?* ». Avec MapReduce, la question initiale devient : « *Comment transformer dix nombres en paires de clé/valeur et agréger les valeurs de façon à obtenir leur somme consécutive* »[xxiv].

La distinction entre séquentiel versus parallèle se traduit donc par deux manière fondamentalement différente d'aborder un problème.

Le stockage des données : le Hadoop Distributed File System

Hadoop c'est également l'implémentation du *Google Distributed File System*[xxv] sous le nom de *Hadoop Distributed File System* (HDFS).

Commençons par le commencement.

Il fut un temps où stocker des données ça donnait ça…

Source : Pixabay

D'aucuns disent que c'était mieux avant, vraiment ?!

De manière générale, un système de fichier c'est….

« *… une manière d'organiser le stockage des données sur le disque dur d'un ordinateur* »[xxvi]. Le disque dur quant à lui est l'élément de tout système informatique qui stocke les données dans la durée. Donnée – notion aussi ancienne que l'écriture[xxvii], parfois simples signaux numériques pour le disque dur et toutes autres mémoires de masse (clé USB, disque dur externe, par exemple[xxviii]), suites de O et de 1 dépourvues de toute valeur. Dépourvues de toute valeur, intéressant ! Je croyais que les données étaient le pétrole du XXIe siècle, objet de toutes les convoitises[xxix] ? A cet égard, la fonction du système de fichier s'avère donc essentielle car il confère un sens à ces suites de 0 et de 1, les organise et présente des fichiers sous une forme exploitable par les utilisateurs. Bien que techniquement le système de fichiers soit perçu comme « *une abstraction d'organisation du stockage des documents sur une mémoire de masse* »[xxx], sa mission d'intermédiaire, d'interface entre le disque dur et l'utilisateur, c'est-à-dire de lien entre la machine et l'humain, n'en demeure pas moins très concrète[xxxi]. Patience et persévérance, avant de révéler l'entier de leur valeur, ces données devront encore être agrégées, croisées, analysées.

Google - De la centralisation à la distribution des systèmes de fichiers

Revenons aux percées technologiques de Google, inspiratrices de Doug Cutting. Pour la construction de l'index inversé, le stockage des pages Internet n'était pas regroupé vers un seul serveur, approche inenvisageable eu égard au volume colossal des données générées, celui-ci excédant amplement la capacité d'un serveur traditionnel. Améliorer leur moteur de recherche via la construction d'un index inversé demandait le stockage de millions (devenus

des milliards !) de pages Web, de données de formats variés. Et surtout, un impératif : gérer les pannes du système. Il doit être disponible pour répondre efficacement aux requêtes des internautes. Ça doit marcher ! Le *Google File System* (GFS) fut la réponse apportée par la firme californienne à cet immense défi. Le système de fichier, tout comme l'architecture et le traitement devait opérer un mouvement de la centralisation vers la distribution. Ce système de fichier distribué (*distributed file system*) installé sur un cluster assure la distribution, le stockage ainsi que la redondance des fichiers sur de multiples disques durs, et non pas sur un seul disque dur partagé[xxxii]. Là se loge la nouveauté. Un changement d'optique a façonné le premier pas de cette innovation : « *While sharing many of the same goals as previous distributed file systems, our design has been driven by observations of our applicatiom workloads and technological environment, both current and anticipated, that reflect a marked departure from some earlier file system assumptions. (…). We have reexamined traditional choices and explore radically different points in the design space. (…). First, component failures are the norm rather than the exception. (…). Therefore, constant monitoring, error detection, fault tolerance, and automatic recovery must be integral to the system. (…). Second, files are huge by traditional standards. (…). Third, most files are mutated by appending new data rather than overwriting existing data. (…). Fourth, co-designing the applications and the file system API benefits the overall system by increasing our flexibility* »[xxxiii].

Une des idées centrales du GFS, que l'on retrouve chez Hadoop, est la division des données en blocs de taille unique et leur réplication (redondance) pour traitement sur les différents nœuds du cluster. Pour éviter toute redondance - langagière, cette fois- inutile, je présenterai cet élément simultanément avec Hadoop.

Hadoop Distributed File System – Noyau de l'efficacité d'Hadoop

Parce qu'Hadoop implémente un modèle de calcul dans un cluster, il a besoin d'un *Distributed File System* pour fonctionner : le *Hadoop Distributed File System* (HDFS), système de fichiers distribué d'Hadoop et implémentation du Google File System. Le HDFS constitue un système de fichiers global du cluster. Il apporte une *abstraction* (couche logicielle) permettant de voir et de gérer les disques durs individuels des nœuds du cluster comme s'il s'agissait en réalité d'un disque dur unique. Le cluster Hadoop est de type share-nothing : disque dur, mémoire, processeur, rien n'est partagé entre les nœuds. A chacun ses ressources propres. Néanmoins, ils partagent virtuellement leur disque dur à travers une abstraction de leur système de fichiers : le système de fichiers distribué[xxxiv]. Abstraction technique, partage virtuelle, mais fonction concrète essentielle pour les utilisateurs.

Les inventeurs du GFS, nous explique comment les données sont préalablement divisées et distribuées sur les différents nœuds du cluster : « *Files are divided into fixed-sized chunks. Each chunk is identified by an immutable and globally unique 64 bit chunk handle assigned by the master at the time of the chunk creation. Chunkservers store chunks on local disks as Linux files and read or write chunk data specified by a chunk handle and byte range. For reliability, each chunk is replicated om multiple chunkservers. By default, we store three replicas, though user scan designate different replication levels for different regions of the file namespace* »[xxxv].

Sur la base de ces propos, on constate donc que le HDFS remplit deux fonctions essentielles spécifiques qui lui confèrent le premier rôle dans l'efficacité d'un cluster Hadoop. :

1) La gestion du stockage distribué de gros volumes de données.
On parle de volumes de données considérables - au-delà des « *frontières du gigantisme* »[xxxvi] - de données stockées via une multitude d'ordinateurs. Avec Hadoop, système *in situ*, les traitements MapReduce vont trouver les données dans les nœuds. La localisation des données se fait grâce au HDFS, qui divise le fichier d'entrée en blocs de taille fixe (*chunks*) de 64 Mo par défaut (Mo = 10^6), blocs de fichier de taille plus grande que les systèmes de fichiers classiques[xxxvii].

2) La garantie de la tolérance aux pannes
Un cluster doit fonctionner malgré les pannes ou les défaillances. Il doit être disponible et performant. Pour assurer la résilience du système, le HDFS procède en répliquant plusieurs copies (trois par défaut) du même bloc dans différents nœuds. Cette réplication des blocs de données explique l'expression *stockage redondant des données*, clé de cette tolérance aux pannes cruciale. En effet, de la sorte les tâches en cours d'exécution ne subiront pas la gêne d'une tâche incomplètement traitée. Lorsqu'un nœud tombe en panne, l'astuce est la suivante : le système passe le relais à un autre nœud qui contient une copie des données identiques à celles traitées par le nœud défaillant, en cas de manque de ressources la prise de relais par un ordinateur doté de ressources supplémentaires est assurée[xxxviii].

La tolérance aux pannes et la haute disponibilité constitue deux des quatre avantages et caractéristiques phares d'un cluster à côté de la scalabilité horizontale et de la scalabilité linéaire[xxxix]. Or, il s'avère que ces avantages contribuent aussi aux limites du HDFS[xl], tout comme le MapReduce possède les siennes, n'étant pas adapté aux analyses interactives, aux jobs itératifs et au streaming[xli]. Dépasser ces limites pour répondre aux nouvelles exigences du Numérique, de sa déferlante de données et aux nouveaux usages des données, a conduit au développement d'améliorations. Aujourd'hui, Hadoop représente un véritable écosystème, une histoire qui continue au-delà de l'amélioration réussie de l'efficacité des moteurs de recherche ; cette évolution constituera en partie l'objet du Billet #3D.

Conclusion

Arrivés à la fin de ce Billet, on comprend pourquoi le parallélisme constitue la pierre angulaire du traitement des données dans le Numérique et la popularité de cette approche, certes nécessaire car seule à même de permettre le traitement de grands volumes de données, mais également possible grâce à la baisse du coût des ordinateurs autorisant la constitution de clusters à un prix accessible.

Au niveau logiciel (virtuel), on retrouve également le mouvement de découpage, de décentralisation vu préalablement au Billet #3B sur le plan architectural (physique).

En synthèse des Billets #3B et #3C dédiés à l'architecture (cluster) et aux logiciels (MapReduce et HDFS) d'Hadoop, on peut dire que : « *La notion d'architecture distribuée permet alors d'accéder à la fois à de grandes quantités de données et de mobiliser de la ressource de calcul distribuée, c'est-à-dire située là où se trouve l'information, elle-même distribuée* »[xlii].

A propos, cette distinction fonctionnelle entre architecture (élément matériel) et logiciels (éléments immatériels) ne vous évoque-t-elle pas quelque chose – ou quelqu'un devrais-je dire – de familier ? Mais oui, cela nous renvoie à vous, à moi, à nous les humains : « Le cerveau est comme le matériel informatique : c'est un dispositif physique. L'esprit correspond aux logiciels : il a besoin du matériel physique pour fonctionner mais, n'ayant aucune masse, il n'est pas matérialisé »[xliii].

A bientôt pour le Billet #3D : *Hadoop – une évolution : un écosystème, le Big Data et la résolution des conflits !* La résolution des conflits ? Oui, la résolution des conflits. Quel lien y a-t-il avec le Numérique ? Souvenez-vous qu'il existe un lien entre justice et société, un modèle de société engendre un mode de régulation correspondant[xliv]. Vers quoi allons-nous ? A suivre.

Nota bene :

Tous les billets de cette série seront publiés sur LinkedIn – au moins sous une forme édulcorée - mais également disponibles en format pdf avec les références détaillées sur le blog de mon site (www.medialien.ch).

An English version of this series of posts - Big Data: a new form of collective intelligence - will follow in a while.

Références :

[i] CHOKOGOUE Juvénal, *Hadoop : devenez opérationnel dans le monde du Big Data*, St-Herblain, ENI, 2017, p. 63 (https://m.editions-eni.fr/livre/hadoop-devenez-operationnel-dans-le-monde-du-big-data-9782409007613#).

[ii] DE KUNDER Maurice, *Daily estimated of the World Wide Web*, http://www.worldwidewebsize.com/ (20/11/2017)

[iii] CHOKOGOUE Juvénal, *op. cit.*, pp. 57 à 72.

[iv] CHOKOGOUE Juvénal, *op. cit.*, p. 61.

[v] *Parallèle*, http://www.larousse.fr/dictionnaires/francais/parall%C3%A8le/57927#5Ofc0xmKC8XfawVm.99

[vi] *Séquentiel*, http://www.larousse.fr/dictionnaires/francais/s%C3%A9quentiel_s%C3%A9quentielle/72230

[vii] *Calcul séquentiel*, https://fr.wikipedia.org/wiki/Calcul_s%C3%A9quentiel

[viii] BRIGHTON Henry, SELINA Howard, *L'intelligence artificielle en image*, EDP Sciences, 2015, pp. 106-107 (https://www.payot.ch/Detail/lintelligence_artificielle-henry_brighton-9782759817726?cId=0). Relevons que nous n'avons pas tous le même hémisphère dominant ; cette différence induit une manière spécifique de traiter l'information selon que l'on est neuro-droitier ou neuro-gaucher. Restons-en là !

[ix] *Calcul séquentiel*, https://fr.wikipedia.org/wiki/Calcul_s%C3%A9quentiel

[x] La question du format des données sera abordée au Billet #4 « Big au sens de Big Data ».

[xi] *Nicolas Machiavel*, http://www.larousse.fr/encyclopedie/litterature/Machiavel/175016

[xii] *Diviser pour régner (informatique)*, https://fr.wikipedia.org/wiki/Diviser_pour_r%C3%A9gner_(informatique)

[xiii] CHOKOGOUE Juvénal, *op. cit.*, pp. 62, 64, 75 ; voir également le Billet 3#B.

[xiv] CHOKOGOUE Juvénal, *op. cit.*, pp. 62 et 65.

[xv] CHOKOGOUE Juvénal, *op. cit.*, p. 65.

[xvi] DEAN Jeffrey, GHEMAWAT Sanjay, *MapReduce: Simplified Data Processing on Large Clusters*, 12/2004, p. 1 (http://static.googleusercontent.com/media/research.google.com/en/us/archive/mapreduce-osdi04.pdf).

[xvii] CHOKOGOUE Juvénal, *op. cit.*, p. 75.

[xviii] DEAN Jeffrey, GHEMAWAT Sanjay, *op. cit.*, p. 1.

[xix] *Index*, http://www.larousse.fr/dictionnaires/francais/index/42560

[xx] CHOKOGOUE Juvénal, *op. cit.*, p. 21.

[xxi] DEAN Jeffrey, GHEMAWAT Sanjay, *op. cit.*, p. 3.

[xxii] CHOKOGOUE Juvénal, *op. cit.*, pp. 21 à 23, 74 à 79, 85 ; *MapReduce*, https://fr.wikipedia.org/wiki/MapReduce; voir Billet #3B, paragraphe « Quelle communication à l'intérieur du cluster : y a-t-il un pilote dans l'avion Hadoop ? » ; pour un exemple de calcul d'occurrences dans un texte ou la jointure de deux tables « Films, Producteurs » voir : Réalisez des calculs distribués sur des données massives, 13/10/2017 (https://openclassrooms.com/courses/realisez-des-calculs-distribues-sur-des-donnees-massives/parcourez-les-principaux-algorithmes-mapreduce); autre calcul d'occurrences dans un texte : Santosh KONDA, Big Data Tutorial 1 : MapReduce, NYU, 27/09/2017 (*https://wikis.nyu.edu/display/NYUHPC/Big+Data+Tutorial+1%3A+MapReduce*); autres exemples in : DEAN Jeffrey, GHEMAWAT Sanjay, *op. cit.*, p. 2.

[xxiii] *Apache Hadoop*, https://wiki.apache.org/hadoop#Setting_up_a_Hadoop_Cluster

[xxiv] CHOKOGOUE Juvénal, *op. cit.*, p. 83.

[xxv] GHEMAWAT Sanjay, GOBIOFF Howard, LEUNG Shun-Tak, *The Google File System*, octobre 2003 (https://static.googleusercontent.com/media/research.google.com/fr//archive/gfs-sosp2003.pdf)

[xxvi] CHOKOGOUE Juvénal, *op. cit.*, p. 124.

[xxvii] Billet #2 : Données – Des tablettes mésopotamiennes à nos tablettes numériques (http://www.medialien.ch/blog-fr170.html).

[xxviii] BRAESH Christian, *Les mémoires de masse*. Définition sur : http://www.christian.braesch.fr/page/les-memoires-de-masse.

[xxix] Billet #1 : Voyage en terra Data (http://www.medialien.ch/blog-fr170.html).

[xxx] CHOKOGOUE Juvénal, *op. cit.*, p. 142.

[xxxi] CHOKOGOUE Juvénal, *op. cit.*, pp. 142 et 146 ; BRAESH Christian, *Les mémoires de masse* (http://www.christian.braesch.fr/page/les-memoires-de-masse).

[xxxii] CHOKOGOUE Juvénal, *op. cit.*, p. 133.

[xxxiii] GHEMAWAT Sanjay, GOBIOFF Howard, LEUNG Shun-Tak, *op. cit.,* p. 1.

[xxxiv] CHOKOGOUE Juvénal, *op. cit.*, pp. 110, 133 à 136 ; *Hadoop*, https://fr.wikipedia.org/wiki/Hadoop.

[xxxv] GHEMAWAT Sanjay, GOBIOFF Howard, LEUNG Shun-Tak, *op. cit.,* p. 2.

[xxxvi] ABITEBOUL Serge, PEUGEOT Valérie, *Terra Data*, Paris, Le Pommier, 2017, p. 27 (https://www.editions-lepommier.fr/terra-data).

[xxxvii] CHOKOGOUE Juvénal, *op. cit.*, pp. 109 et 136 ; *HDFS,* https://wiki.apache.org/hadoop/HDFS?action=show&redirect=DFS

[xxxviii] CHOKOGOUE Juvénal, *op. cit.*, p. 33 ; *HDFS Architecture Guide* (https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html), *Apache Hadoop*, https://wiki.apache.org/hadoop#Setting_up_a_Hadoop_Cluster; Billet #3B – paragraphe « La tolérance aux pannes ».

[xxxix] Billet #3B - Hadoop – une architecture distribuée, un nouveau paradigme infrastructurel (http://www.medialien.ch/blog-fr170.html).

[xl] Pour plus de détails à ce sujet, voir CHOKOGOUE Juvénal, *op. cit.*, pp. 136-137.

[xli] *MapReduce*, https://wiki.apache.org/hadoop/MapReduce, CHOKOGOUE Juvénal, *op. cit.*, p. 148.

[xlii] BABINET Gilles, *Big Data : penser l'homme et le monde autrement*, Paris, Le Passeur, 2015, p. 28 (http://www.eyrolles.com/Informatique/Livre/big-data-penser-l-homme-et-le-monde-autrement-9782368904923).

[xliii] BRIGHTON Henry, SELINA Howard, *op. cit*, p. 41.

[xliv] BONAFE-SCHMITT Jean-Pierre, *La médiation : une justice douce.* Paris, Syros-Alternatives, 1992, p. 180.