Originally published in French (08/12/2017), this post is a translation made with the help of: www.DeepL.com/Translator

BIG DATA: A NEW FORM OF COLLECTIVE INTELLIGENCE (3C)

Mini-Series of Posts #3 : From search engines to Big Data. And, what about conflict resolution?

Post #3C – *Hadoop - softwares : MapReduce and the Hadoop Distributed File system - a new programming paradigm*

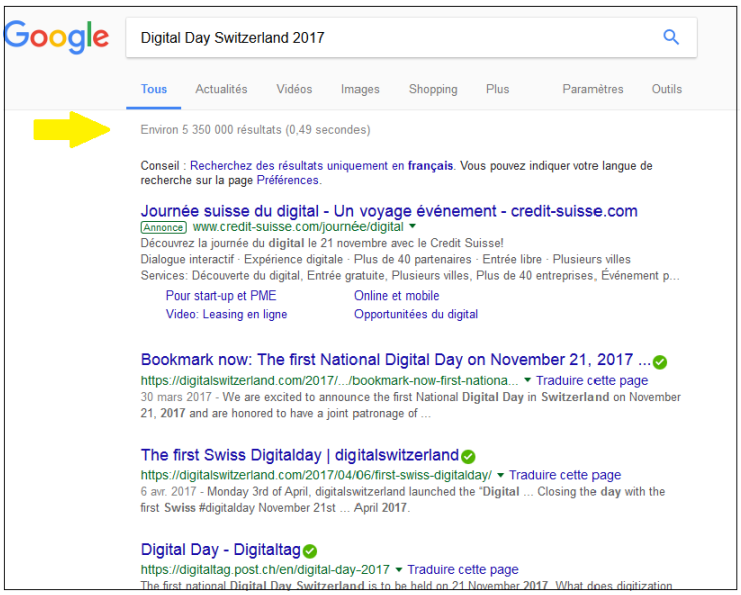
Remember, that initially, far from any Big Data idea in its current sense, the problem to be solved was the one of Web pages indexing in order to improve the efficiency of search engines. The solution found by Google, and implemented by Doug Cutting with Hadoop, was to split the problem of indexing web pages into sub-problems distributed for execution in a cluster of computers. We saw in Post #3B what a cluster is from an architectural, physical point of view. From a software point of view, a Hadoop cluster is simply defined as a cluster on which Hadoop has been installed, i.e. two elements: firstly, the set of JAVA implementation classes of the MapReduce algorithm and, secondly, the Hadoop Distributed System File (HDSF) as a file system on all of the hard disks of the cluster nodes[i]. We thus leave centralized architectures to go towards distributed architectures. From a software perspective, this consists of parallelizing the execution of the work, dividing it between a set of interconnected computers considered as a whole.

In keeping with the twofold objective of this mini-series of Posts #3 - to identify the notion of Big Data, on the one hand, and to clarify the impact of these innovations on conflict resolution methods, on the other hand - this Post #3C will focus on the following notions:

- What does it mean to « parallelize the work execution »?
- What is the MapReduce algorithm?
- What is the Hadoop Distributed System File (HDSF)?

At a spectacle it would represent the backstage activities.

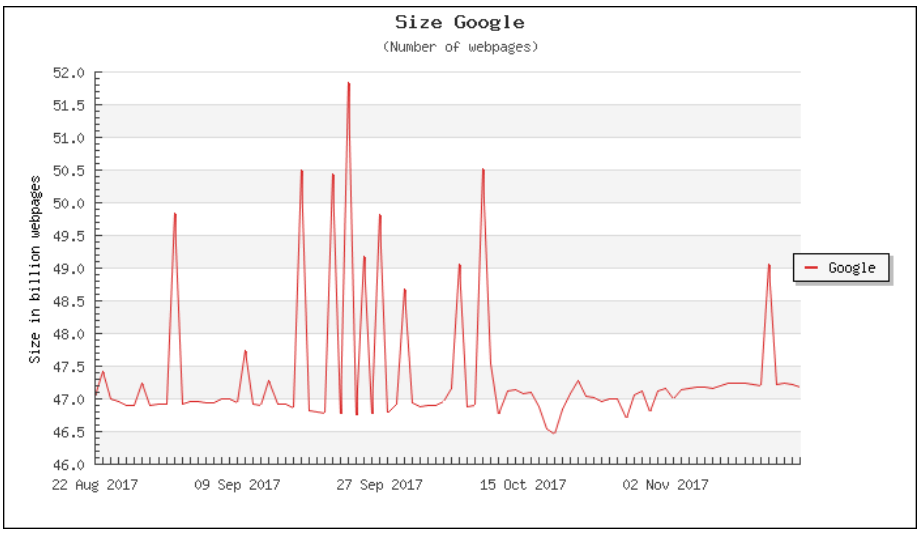
On the opposite, here is an example taken from recent news, showing what is visible on stage.



This screenshot results from my research done on November 20, 2017, on the eve of Digital Day Switzerland 2017.

The search engine found about 5.35 million results among all web pages indexed by Google.

I have not found the exact number of pages indexed by Google up to this day but according to one estimate, the size of Google's index is about 4.25 billion pages! On Monday, November 20, 2017, still in an estimated way, the indexed Web contained globally at least 4.49 billion pages [ii].



The processing time of my request was ultra fast: 0.49 seconds. Efficient, isn't it?! Now let us see what is going on backstage.

Parallelism: the cornerstone of the transition from centralisation to distribution

Massively parallel parallelism (independent) - Definition.

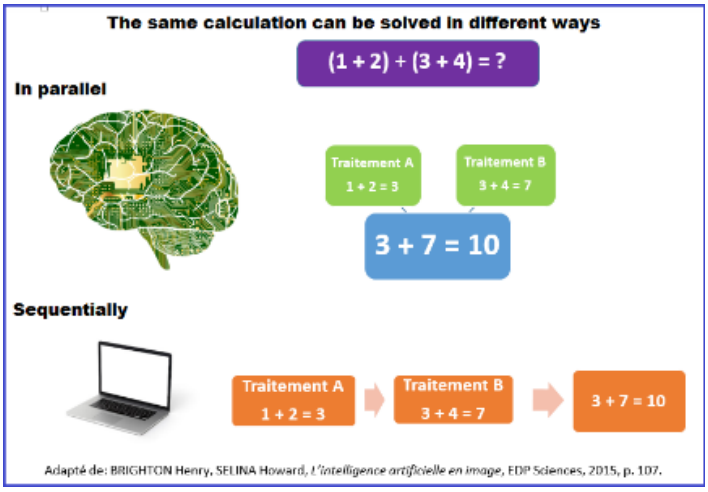
Parallelism can be of three types, depending on the infrastructure used, the level of divisibility and the constraints of the problem to be dealt with: simultaneous asynchronous, pipeline, massively parallel (independent). The meaning given to the word *parallel* can vary from one type of parallelism to another. Therefore, in order not to complicate the subject unnecessarily, Hadoop working with the latest type of parallelism and constituting the red line of this mini-series of Posts, I will confine myself to *massively parallel parallelism*[iii], defined as the division of the *« processing into COMPLETELY independent tasks, which are executed in parallel (...) on the nodes of a cluster in shared-nothing architecture »*[iv].

Parallel versus sequential.

Generally speaking, the French Larousse dictionary gives the following definitions of the adjective parallel: «*1) Said of coplanar straight lines or planes with no common or confused point ; 2) Which is directed according to a parallel line or plane: Put your skis parallel to each other ; 3) Which develops in the same direction as something else, similar: Parallel political action of two parties ; 4) Refers to activities, organizations that develop outside a legal or official framework: Parallel gold market. Parallel police* »[v]. Sequential means *« which belongs to, refers to a sequence, to an ordered chain of operations »*[vi]. We see from these definitions that the notion of independence, or lack thereof, contributes to their meaning.

In the computer field, parallel is opposed to sequential, we speak of sequential calculation in reference to *« the execution of a processing step by step, where each operation is triggered only when the previous operation is completed, including when the two operations are independent »*[vii].

This figure[viii] shows us what this difference schematically looks like.



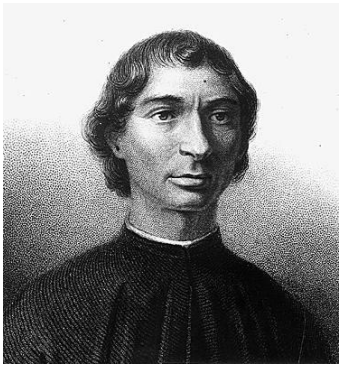
In the first variant, the processings A and B are carried out at the same time and completely independently, while in the second variant, processing B is only performed after processing A has been completed.

This scheme calls for a small human parenthesis, refreshing in the middle of this technology: our human brain works massively parallel, and that is

the reason why we can recognize our loved ones extremely quickly - in a tenth of a second - for example[ix].

The sequential calculation is frequently used because of its compatibility with any type of operation, the power of the computers however being its limit, where the parallel calculation is restricted only by the number of computers[x]. We saw in Post #3B, in the paragraph about *horizontal scalability*, that « scaling up » by increasing the number of nodes in a cluster represents a central concept for understanding the evolution from search engines improvement to Big Data, as well as the popularity and choice of clusters to handle the flood of data we are witnessing today.

Divide and conquer: large data volumes, clusters and independent tasks



Processing large volumes of data of various formats[xi] requires the use of a computer cluster. This use - and thus of Hadoop - is subject to a sine qua non condition: the problem to be solved being divisible into independent tasks that do not need the result of another one to continue. A divisible problem. Yes, the motto « *Divide and rule* », dear to the Italian politician and writer Niccolò Machiavelli (1469 – 1527)[xii], among others, also applies in computer science[xiii].

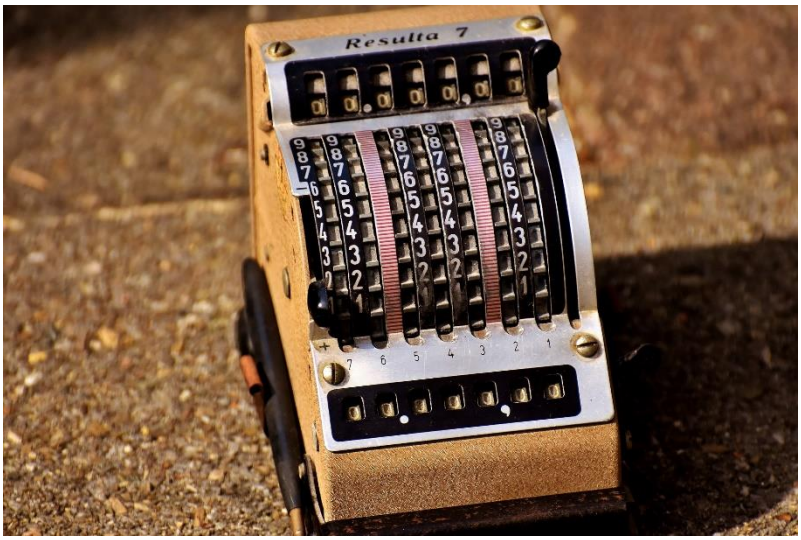
Before seeing in more details how Hadoop embodies this « machiavellian » motto in a new approach of storage (HDFS) and data processing (MapReduce), let us underline that the new paradigm proposed by Hadoop is based on a necessary link between software elements (HDFS and MapReduce) and a particular architecture: a master/slave shared-nothing cluster which makes the independence of tasks indeed possible. Shared-nothing - nothing is shared: each node is equipped with its own hard disk, its own memory, its own processor. Independence is complete, the total « parallelisability » of tasks is not only a possibility, it is a necessity. The problem must imperatively be parallelizable. In such an architecture, without communication between nodes, it becomes mandatory to split the problem to be addressed into several subtasks that can run in parallel, independently, isolated on a node of the cluster. We thus fall back on the *sine qua non* condition for the processing of large volumes of data presented hereabove, and the loop is closed[xiv] !

Nota bene - parallelism in the cluster is not parallelism within a node.

Hadoop works with massively parallel parallelism among the nodes of the cluster. The tasks are replicated on all cluster nodes and each node receives a file block, a portion of the data, which is processed there. The situation is different inside the nodes, where tasks are not performed in parallel, but sequentially[xv].

MapReduce, Hadoop heart[xvi]

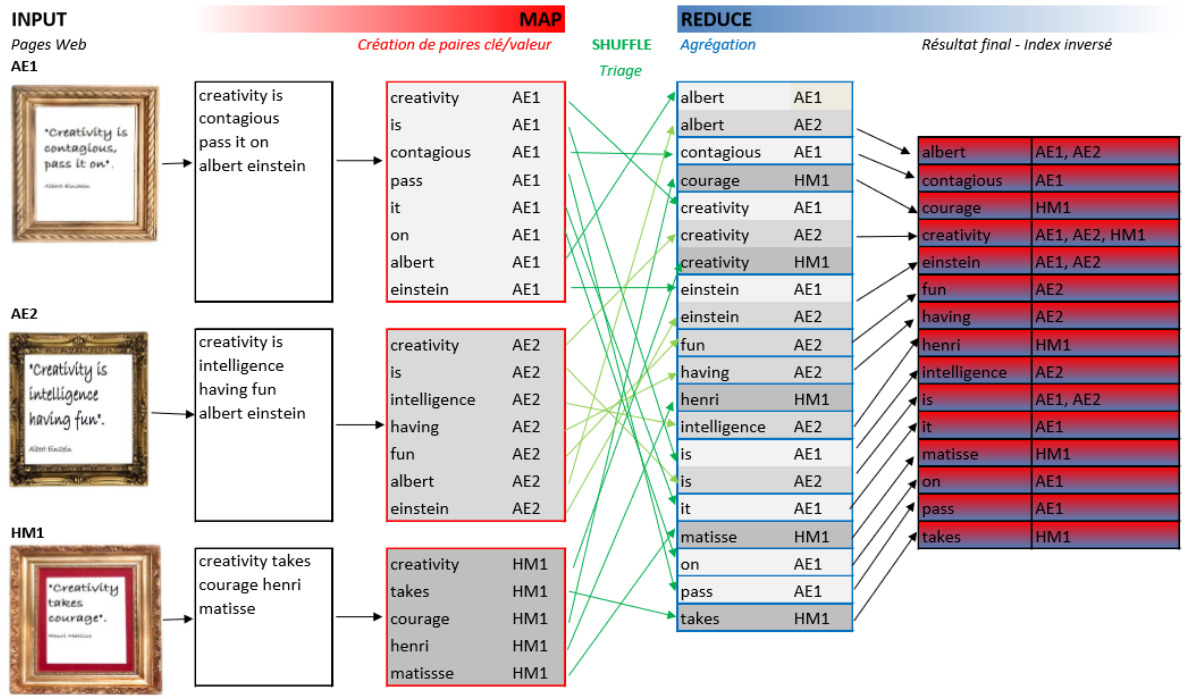
There was a time when processing data gave that...



Tempi passati…. Google's initial goal at the time the MapReduce was designed was the following: *« 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 »*[xvii]. Let's remember that the original problem to solve consisted in indexing Web pages as well as improving search engines, and that alone!

Machiavelli is back. MapReduce follows a « divide and conquer » programming approach where any problem is divided into tasks, each of them being isolated in a node for processing[xviii]. Explained under the pen of Jeffrey Dean and Sanjay Ghemawat, fathers of Google MapReduce, it gives that: *« 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 »*[xix]. Thus, the solution provided by Google, based on two unique functions Map and Reduce, was the following: to divide the problem of indexing web pages into sub-problems distributed throughout the cluster to be executed therein. To do this, an inverted index is constructed by keyword appearing in each page of the Web. An inverted index corresponds to the index of a book, i. e. according to French dictionary *Larousse* to the « *alphabetical list of authors, subjects, keywords, etc., appearing in a book, with references to find them* »[xx] ; convenient, isn't it ?! This principle of search on the Web persists today [xxi]. According to his fathers, an inverted index is defined as follows : *« 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 »*[xxii].

Schematically, the construction of the inverted index by Google with MapReduce looks like this :



Construction d'un index inversé avec MapReduce

And in words, like that:

1) The *MAP* phase: creating key/value pairs

The data (Web pages, input), previously cut into blocks distributed to all the nodes of the cluster, are transformed into key/value pairs.

- a key = a word on a web page,
- a value = an index referencing the page where the word is located.

In the diagram above the indexes are AE1, AE2, HM1. Each page has a unique index. The task of assigning to each word (key) of a web page an index (value) runs in parallel, at the same time throughout the cluster on the nodes that form it.

2) The *Shuffle* phase : sorting key/value pairs.

During this phase, all pairs generated during the Map phase are sorted by key (word).

3) The *REDUCE* phase : aggregating

The key values are aggregated, grouped, consolidated and the inverted index is finalized.
What a feat! This is based on only two functions: Map and Reduce. The user defines them, the orchestrator - the cluster conductor - assigns them to the nodes, plans the execution modalities, the cluster thus appearing as a single computer to the users. MapReduce also allows to count word occurrences in a text, a book for example, or to join two relational tables. It is therefore valuable in web search applications, in every activity based on large amounts of text, in many types of data processing such as data mining, graphs, etc[xxiii].

A paradigm shift in programming

We have seen that the challenge of improving search engines has brought a change of infrastructural paradigm: centralized to distributed. This challenge also induces a paradigm shift in programming: sequential to parallel and distributed. « *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* »[xxiv].

New algorithmic model consisting in cutting a problem into independent tasks, the adoption of MapReduce implies to change the initial question to be asked in order to solve it. For instance, in order to calculate the sum of ten consecutive numbers, traditional reasoning is expressed as follows: *« How to sum ten numbers sequentially? ».* With MapReduce, the initial question becomes *« How to transform ten numbers into key/value pairs and aggregate the values to get their consecutive sum »*[xxv].

The distinction between sequential versus parallel therefore translates into two fundamentally different ways of approaching a problem.

Data storage: the Hadoop Distributed File System

Hadoop is also the implementation of the *Google Distributed File System*[xxvi] under the name of *Hadoop Distributed File System* (HDFS).

Let's start from the beginning.

There was a time when storing data looked like this...

Some people claim it was better before, really?!

Generally speaking, a file system is....

« ... a way to organize the data storage on the hard disk of a computer »[xxvii]. The hard disk is the element of any computer system that stores data over time.

Data - a notion as old as writing[xxviii], sometimes simple digital signals for the hard disk and all other mass memories (USB key, external hard disk, for example[xxix]), sequences of worthless O's and 1's. Worthless, interesting! I thought data was the oil of the 21st century, the object of all lusts[xxx] ? In this respect, the function of the file system is therefore essential because it gives meaning to these sequences of 0 and 1, organizes them and presents files in a form usable by users. Although technically the file system is perceived as *« an abstraction of the organization of the storage of documents on a mass memory »*[xxxi], its mission as an intermediary, as an interface between the hard disk and the user, i.e. as a link between the machine and the human being, nevertheless remains very concrete[xxxii]. Patience and perseverance, before revealing their full value, these data must still be aggregated, cross-referenced and analysed.

Google - From centralization to file system distribution

Let's go back to Google's technological breakthroughs, which inspired Doug Cutting. For the construction of the inverted index, the Internet pages storage was not concentrated towards a single server, an unthinkable approach considering the colossal volume of the generated data, greatly exceeding the capacity of a traditional server. Improving their search engine through the construction of an inverted index required the storage of millions (now billions!) of Web pages, data of various formats. And above all, an imperative: to manage system failures. It must be available to respond effectively to requests from Internet users. It has to work! The Google File System (GFS) was the California firm's answer to this huge challenge. The file system, as well as the architecture and processing, had to move from centralization to distribution. This distributed file system installed on a cluster ensures distribution, storage and redundancy of files on multiple hard disks, not on a single shared hard disk.[xxxiii]. There lies the novelty. A change of perspective shaped the first step of this 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* »[xxxiv].

One of the central ideas of the GFS, which we also find in Hadoop, is the division of data into blocks of a single size and their replication (redundancy) for processing on the various nodes of the cluster. To avoid any unnecessary redundancy - of language this time - I will present this element simultaneously with Hadoop.

Hadoop Distributed File System – Core of Hadoop's efficiency

Because Hadoop implements a computing model in a cluster, it needs a Distributed File System to run : the *Hadoop Distributed File System* (HDFS), Hadoop distributed file system and implementation of Google File System. The HDFS is a global cluster file system. It provides an *abstraction* (software layer) to view and manage the individual hard disks of the cluster nodes as if they were actually composing a single hard disk. The Hadoop cluster is of a share-nothing type: hard disk, memory, processor, nothing is shared between the nodes. Each node owns its own resources. Nevertheless, they virtually share their hard disk through an abstraction of their file system: the distributed file system[xxxv]. Technical abstraction, virtual sharing, but concrete and essential function for users.

The inventors of the GFS, explain us how the data is previously divided and distributed on the different nodes of the 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 »[xxxvi].

On the basis of these remarks, it can therefore be seen that HDFS fulfils two specific key functions which give it the first role in the effectiveness of a Hadoop cluster:

1) Managing distributed storage of large volumes of data.
We are talking about considerable volumes of data - beyond the « borders of gigantism » [xxxvii] - stored on a multitude of computers.

With Hadoop, *in situ* system, MapReduce processes will find the data in the nodes. Data localization is done through HDFS, which divides the input file into fixed size blocks (*chunks*) of 64 MB by default (MB = 10^6), file blocks larger than traditional file systems[xxxviii].

2) The guarantee of fault tolerance
A cluster must function despite faults or breakdowns. It must be available and high-performing. To ensure system resiliency, HDFS proceeds by replicating multiple copies (three by default) of the same data chunk in different nodes. This data block replication explains the term *redundant data storage*, key element of this critical fault tolerance. Indeed, in this way the tasks in progress will not suffer the inconvenience of a task incompletely processed. When a node fails, the trick is as follows: the system passes on the duty to a different node containing a copy of the data, identical to those processed by the failed node. In case of lack of resources the taking over of the duty by a computer with additional resources is ensured[xxxix].

Fault tolerance and high availability are two of the four key advantages and features of a cluster, together with horizontal scalability and linear scalability[xl]. However, it appears that these advantages also contribute to the limitations of the HDFS[xli], just like MapReduce has its own ones, not being suitable for interactive analysis, iterative jobs and streaming[xlii]. Exceeding these limits to meet the new requirements of the Digital, its flood of data and new uses of data, has led to the development of improvements. Today, Hadoop is a true ecosystem, a story that continues beyond the successful improvement of search engine efficiency. This evolution will be a part of Post #3D.

Conclusion

Coming to the end of this Post, we understand why parallelism constitutes the cornerstone of data processing in the digital era as well as the popularity of this approach. Indeed, a necessary approach because it alone can enable the processing of large volumes of data, but also an approach made possible due to the reduction in the cost of computers allowing the building of clusters at an affordable price.

At the software (virtual) level, we also find the splitting and decentralization movement previously seen in Post #3B, on the architectural (physical) level.

In synthesis of the Posts #3B and #3C dedicated to the architecture (cluster) and software (MapReduce and HDFS) of Hadoop, we can say that : *« The notion of distributed architecture then makes it possible to access both large quantities of data and to mobilize the distributed computing resource, i.e. located where the information, itself distributed, is located »*[xliii].

By the way, doesn't this functional distinction between architecture (hardware element) and software (immaterial elements) remind you of something - or should I say : of someone someone - familiar? Of course, it sends us back to you, to me, to us - human beings : *« The brain is like computer equipment: it is a physical device. The mind corresponds to software: it needs physical hardware to function but, having no mass, it is not materialized »*[xliv].

See you soon for Post #3D : Hadoop - an evolution : an ecosystem, Big Data and conflict resolution ! Conflict resolution? Yes, conflict resolution. What is the link with the Digital? Remember, there is a link between justice and society, a model of society generating a corresponding mode of regulation[xlv]. Where are we heading to, in this respect? To be continued.

Anne-Sylvie Weinmann
www.medialien.ch

References & Notes :

[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 to 72.

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

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

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

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

[viii] Adaptation from BRIGHTON Henry, SELINA Howard, *L'intelligence artificielle en image*, EDP Sciences, 2015, p. 107.

[ix] BRIGHTON Henry, SELINA Howard, *op. cit.*, pp. 106-107 (https://www.payot.ch/Detail/lintelligence_artificielle-henry_brighton-9782759817726?cId=0). It should be noted that we all do not all have the same dominant hemisphere; this difference induces a specific way of processing information depending on the dominant hemisphere : the right one, or the left one.

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

[xi] The data format topic will be addressed subsequently in Post #4A.

[xii] *Nicolas Machiavel*, http://www.larousse.fr/encyclopedie/litterature/Machiavel/175016 (picture included).

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

[xiv] CHOKOGOUE Juvénal, *op. cit.*, pp. 62, 64, 75 ; see also Post #3B.

[xv] CHOKOGOUE Juvénal, *op. cit.*, pp. 62 and 65.

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

[xvii] 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).

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

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

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

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

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

[xxiii] CHOKOGOUE Juvénal, *op. cit.*, pp. 21 to 23, 74 to 79, 85 ; *MapReduce*, https://fr.wikipedia.org/wiki/MapReduce; see Post #3B, paragraph « What communication inside the cluster : is there a pilot in the Hadoop plane ? » ; for an example of occurrences calculation in a text or the joining of two tables « Films, Producers » see : *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); other calculation of occurrences in a text : Santosh KONDA, Big Data Tutorial 1 : MapReduce, NYU, 27/09/2017 (*https://wikis.nyu.edu/display/NYUHPC/Big+Data+Tutorial+1%3A+MapReduce*); other examples in : DEAN Jeffrey, GHEMAWAT Sanjay, *op. cit.*, p. 2.

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

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

[xxvi] 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)

[xxvii] Translation of : CHOKOGOUE Juvénal, *op. cit.*, p. 124.

[xxviii] See on this topic Post #2 : Data – Des tablettes mésopotamiennes à nos tablettes numériques (http://www.medialien.ch/blog-fr170.html).

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

[xxx] Post #1 : A trip in Terra Data (http://www.medialien.ch/blog-fr170.html).

[xxxi] Translation of : CHOKOGOUE Juvénal, *op. cit.*, p. 142.

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

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

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

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

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

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

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

[xxxix] 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 ».

[xl] Post #3B - Hadoop – a distributed architecture, a new infrastructural paradigm. (http://www.medialien.ch/blog-fr170.html).

[xli] For more details on this topic, see CHOKOGOUE Juvénal, *op. cit.*, pp. 136-137.

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

[xliii] Translation of : 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).

[xliv] Translation of : BRIGHTON Henry, SELINA Howard, *op. cit*, p. 41.

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