# Les calculs sur données-client massives sont-ils trop importants pour être laissés aux analystes marketing ?


Michel CALCIU
Maître de conférences, Université Lille 1, IAE, RIME Lab EA 7396
Francis SALERNO
Professeur des Universités, Université Lille 1,
IAE, Lille Economie Management (UMR CNRS 9221)
Jean-Louis MOULINS
Professeur des Universités, Cret-Log, Aix Marseille Université


## Introduction

*La loi de Moore et les limites du microcosme CPU poussent à explorer le macrocosme CPU*. Après de nombreuses années d'augmentation de puissance de calcul grâce à la miniaturisation des circuits, les progrès à rythme exponentiel du microcosme CPU (Unité à Processeur Central), connus sous le nom de loi de More, semblent atteindre un goulot d'étranglement en raison des limites physiques. Tandis que les processeurs continuent d'être de plus en plus rapides nos ensembles de données ne cessent de grandir à un rythme encore plus rapide. L'intérêt semble par conséquent se déplacer progressivement vers l'exploration du macrocosme des processeurs par le calcul parallèle et distribué. Il devient habituel d'avoir des ordinateurs portables ayant 2 ou 4 cœurs au sein de la même CPU et les serveurs ayant 8, 32 ou plusieurs noyaux sont monnaie courante.

Le *High Performance Computing* (HPC) se réfère plus généralement à la pratique de l'agrégation de la puissance de calcul d'une manière offrant des performances beaucoup plus élevées que celles d'un ordinateur de bureau ou d'une station de travail typique pour résoudre de grands problèmes en science, en ingénierie ou en gestion. Le HPC utilise habituellement des superordinateurs et / ou des clusters (grappes) d'ordinateurs.

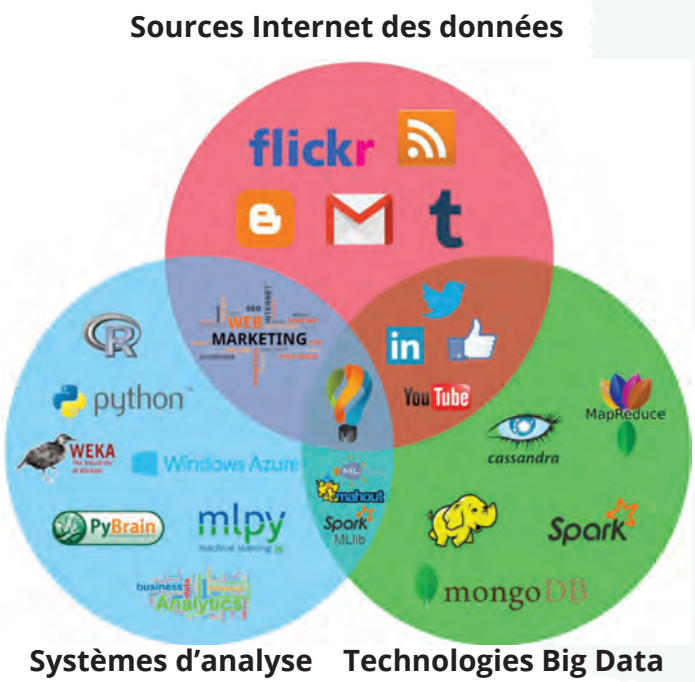
Bien qu'en développement continu, ce traitement d'énormes quantités de données à l'aide des supercalculateurs ou de grappes d'ordinateurs est considéré par la plupart des utilisateurs, mais aussi par la plupart des scientifiques de la donnée et notamment par certains analystes marketing, comme quelque chose dont il ne vaut pas la peine de se soucier, ou de trop coûteux, ou même comme un mythe, comme quelque chose d'inaccessible.

*La disponibilité d'une énorme puissance de calcul ou la fascination d'un mythe se transformant en réalité*. L'accès démocratisé à une immense puissance de calcul, souvent par cloud computing basé sur l'Internet, génère une nouvelle fascination, celle de pouvoir utiliser l'incroyable pouvoir de l'informatique ; l'impossible devenant possible, les mythes se transforment en réalité.

*La Fascination avec le pouvoir*. « De l'eau, de l'eau, partout de l'eau, Et pas une seule goutte à boire », cet extrait du poème Samuel Taylor Coleridge intitulé « La complainte du vieux marin » peut aussi être paraphrasée. On doit se sentir bien d'être en mesure d'aider les managers

“naufragés” qui voient « des données, des données, des données partout, et pas un seul octet à utiliser ». La sensation devient plus grande lorsqu’on est capable d’utiliser des grappes d’ordinateurs ou superordinateurs pour faire face au tsunami de données. La Figure 1 montre les principales sources Internet de données client massives ainsi que les systèmes d’analyse et technologies de traitement de Big Data.

Figure 1 - Big Data: Sources Internet, Technologies et Systèmes d’analyse




*Source : adapté de Bello-Orgaz & al. (2016)*

Les chercheurs et analystes en marketing, cantonnés dans les paradigmes du marketing transactionnel classique où la connaissance du client se limitait aux études à base d’échantillons et panels, se sont fait surprendre par l’avalanche de données comportementales issues des nouvelles techniques du marketing relationnel et digital au point que les informaticiens se sont emparés d’une partie de leur corps de métier. En paraphrasant une citation célèbre de Clemenceau “La guerre est une chose trop importante pour être laissée aux militaires”, nous soutenons que “les Big Data sont trop importantes pour être laissées aux informaticiens”. Les statisticiens et les analystes marketing devraient jouer un rôle actif et contribuer aux nouvelles approches qui conduisent à des changements révolutionnaires dans la science des données. Les utilisations sont déjà nombreuses en marketing pour le e-commerce et dans les réseaux sociaux (Tableau1).

## Tableau 1 – Exemples d'analyses de données massives en e-commerce et pour les réseaux sociaux

**Personnalisation**

- Gestion du portefeuille de relations clients. Identification des clients ayant les plus grandes rentabilités et fidélités potentielles. Augmentation de la probabilité qu'ils souhaitent l'offre de produit ou service ; maintenir leur fidélité
- Marketing direct/interactif ; marketing relationnel. Machine de recommandation pour générer « Vous aimerez sans doute aussi » et développer ainsi du cross-selling
- Emails personnalisés. Amélioration des taux de réponse par une personnalisation fondée sur l'analyse de la BDD clients ou l'intégration de plusieurs BDD clients.
- Individualisation (customisation) des offres de produit ou service. Produire des profils détaillés de clients, micro-segmenter et personnaliser les offres-produit et ainsi renforcer la fidélité attitudinale et comportementale.

**Prix dynamiques**

- Identifier le prix qui maximisera la marge ou le profit
- Dériver les prix précis des produits et services avec de fins calculs de la rentabilité client
- Optimiser les prix de millions d'items en un temps record
- Programmer les réductions progressives de produits périssables avant qu'ils ne se gâtent

**Gestion des produits et services**

- Détecter plus tôt les problèmes de qualité et les minimiser
- Analyser les choix des clients et leurs innombrables avis
- Simuler les emplacements de nouveaux produits dans les linéaires pour tester les effets de conception afin d'améliorer l'acceptabilité des produits après lancement
- Analyse au niveau des unités de gestion des stocks (SKU : Stock Keeping Unit) pour s'assurer que les assortiments de produits sont déjà disponibles

**Clusters, Analyses Prédictives**

- Intégrer systématiquement les analyses et les connaissances clients en utilisant les données du programme de fidélisation afin de mieux segmenter et cibler
- Développer des segmentations comportementales et des programmes de récompenses multi-niveaux en analysant les profils des clients, les changements des comportements des clients en temps réel et la profitabilité des clients
- Fin réglage des promotions mondiales pour chaque media et chaque région
- Connaître les parts de marché des concurrents locaux dans différentes zones

**Réseaux sociaux**

- Ciblage publicitaire sur Facebook : Mesures en temps réel pour détecter les utilisateurs les plus valables
- Twitter comme mécanisme de e-BAO (bouche-à-oreille électronique) : Analyse des sentiments
- LinkedIn. Génération de millions de nouvelles pages vues en utilisant « Les gens que vous pouvez connaître »
- Twitter pour la prévision des recettes box-office pour les films : Détection des sujets de conversation, analyse des sentiments
- Marketing viral dans les réseaux sociaux : Analyses de réseau, modèles de diffusion de l'information

*Source : d'après Akter et Wamda (2016) ; Bello-Orgaz et al. (2016)*

Les solutions open-source relativement récentes qui forment un écosystème autour du plus élégant système statistique, R, et le système de calcul distribué Hadoop (sorte de Linux pour les clusters d'ordinateurs) démocratisent le traitement des Big Data et offrent une opportunité exceptionnelle aux statisticiens et analystes marketing "d'opérer" de vraies usines à calcul.

## Quelles données sont "Big"? Solutions open-source sélectionnées

Certains auteurs prétendent qu'un fichier avec plus d'un million d'enregistrements peut être considéré comme Big Data. D'autres indiquent des tailles de plusieurs Tera ou même Peta octets. Une définition moins orthodoxe vient de Hadley Wickham (2015) qui considère que, dans l'analyse traditionnelle le temps de la cognition est plus long que le temps de calcul, tandis que pour les Big Data c'est l'inverse, le temps CPU de calcul prend plus de temps que le processus cognitif de conception d'un modèle.

Dans le choix des outils et des solutions utilisables par les scientifiques du marketing pour les calculs Big Data nous insistons sur des solutions open-source activement développées à la fois pour les tâches de calcul statistiques et parallèles et nous centrons nos illustrations et discussions autour du système statistique R et du système de calcul distribué Hadoop.

## Le système statistique R et son écosystème Big Data

Le système statistique R, qui devient progressivement l'outil préféré par la majorité des analystes de données et par un nombre croissant de scientifiques du marketing, a eu, il y a quelques années la réputation de ne pas être en mesure de traiter des Big Data. Cela reste vrai pour beaucoup d'autres solutions logicielles statistiques mais plus pour R qui a beaucoup changé et qui propose aujourd'hui une série de packages spécialisés formant un écosystème pour traiter les Big Data.

## Masquer la complexité du Calcul Parallèle Distribué avec MapReduce et Hadoop

Quelles données sont « Big » pour R et comment peuvent-elles être traitées? Wijffels (2013) suggère trois niveaux de taille des données. Le plus bas, est lorsque les données contiennent moins d'un million d'enregistrements et peuvent être traitées en utilisant le R standard. Le second s'entend lorsque les données contiennent entre un million et un milliard d'enregistrements. Elles peuvent également être traitées dans R mais ont besoin d'un effort supplémentaire en adoptant une ou plusieurs des cinq stratégies énumérées ci-dessous. Le troisième niveau, le plus élevé, est atteint lorsque les données contiennent plus d'un milliard d'enregistrements. Dans ce cas, les algorithmes peuvent être conçus avec R et traités avec des connecteurs pour Hadoop ou des solutions HPC alternatives. Les cinq stratégies qui peuvent être utilisées avec R pour faire face aux Big Data, sont : 1) l'échantillonnage, 2) des machines plus puissantes, 3) placer les objets mémoire sur disque, 4) l'intégration avec d'autres langages et 5) les interpréteurs alternatifs.

*L'échantillonnage* permet de réduire la taille des données. Bien que beaucoup de données soit préférable à peu de données, l'échantillonnage est acceptable si la taille des données franchit le seuil d'un milliard d'enregistrements.

*Des machines plus puissantes* peuvent être une solution lorsque les données deviennent importantes. Comme R conserve tous les objets en mémoire, une solution consiste à augmenter la mémoire de la machine. Sur les machines 64 bits, R peut traiter 8 To de RAM ; une énorme amélioration par rapport aux de 2 Go de mémoire vive adressables par des machines de 32 bits.

*Stocker des objets sur le disque dur et les analyser par morceaux* est une solution qui évite le stockage de données en mémoire. Le morcellement (*chunking*) conduit naturellement à la parallélisation et les algorithmes doivent être explicitement conçus pour traiter des types de données spécifiques au disque dur. "ff" et "ffbase" sont les packages CRAN les plus connus (open-source) qui suivent ce principe. Le package "scaleR" de Revolution R Enterprise est également une solution populaire.

*L'intégration avec des langages de programmation performants* comme le C ++ ou Java. Afin d'éviter les goulots d'étranglement et des procédures coûteuses en performance, des parties du programme sont déplacées de R à un autre langage. L'objectif est de combiner, lors du traitement des données, l'élégance de R avec la performance d'autres langages. Il est relativement facile d'externaliser le code de R vers ces langages en utilisant les packages Rcpp et Rjava.

*Les interpréteurs alternatifs* pourraient être plus rapides ou mieux adaptés dans certains cas que le R standard. Ceux-ci sont pqR (pretty quick R), Renjin qui réimplémente l'interpréteur R en Java et peut donc fonctionner sur la machine virtuelle Java (JVM) et TERR, un interpréteur basé sur C ++ créé par Tibco. Oracle R utilise la bibliothèque mathématique d'Intel pour obtenir de meilleures performances sans modifier le noyau de R.

Les cinq stratégies évoquées sont des palliatifs lorsque les données ne sont pas encore trop grandes pour R. Lorsque ce n'est pas le cas, des connecteurs à Hadoop ou des solutions HPC alternatives doivent être utilisés en conjonction avec R.

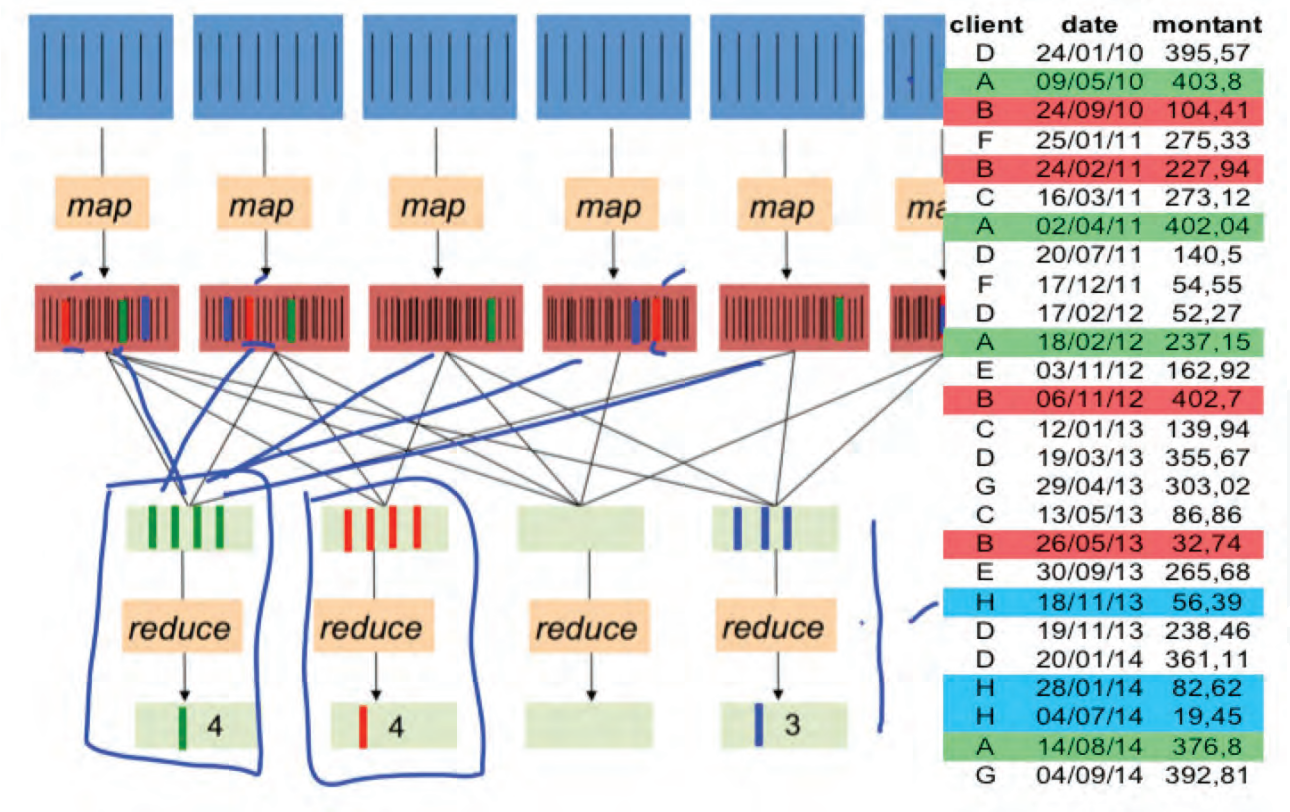
MapReduce est un modèle de programmation de haut niveau et une mise en œuvre associée pour le traitement de données en parallèle à grande échelle. Il a le mérite d'avoir fortement contribué à démocratiser le traitement des Big Data. A l'origine, le nom MapReduce faisait référence à une technologie propriétaire Google qui est devenue depuis une marque générique. Il est intégré dans Hadoop d'Apache, une plateforme logicielle open-source, écrite en Java, pour le stockage et le traitement distribué de très grands ensembles de données sur les grappes d'ordinateurs.

Il est utile pour les statisticiens et analystes marketing d'avoir une bonne compréhension de MapReduce afin d'être en mesure d'adapter leurs modèles et algorithmes au traitement parallèle distribué des ensembles de données volumineux. MapReduce est basé sur l'observation que la plupart des calculs peuvent être exprimés en termes d'une procédure Map, qui permet le filtrage et le tri, et d'une procédure Reduce, qui effectue une opération d'agrégation comme le comptage, la somme etc. Map et Reduce sont des fonctions d'ordre supérieur usuelles en programmation fonctionnelle à laquelle le système statistique R appartient.

## Utilisation de MapReduce dans les calculs d'agrégation des données massives

L'approche de MapReduce peut être facilement appliquée aux calculs d'agrégation. Dans le marketing relationnel on agrège les données sur les transactions des clients pour calculer des variables fondamentales pour la segmentation comportementale et le ciblage : la fréquence (F), la récence (R) et le montant (M) des achats.

Figure 2 - Exemple MapReduce: calculs RFM sur une base de clientèle




*Source: adapté de Howe B. eScience Institute, U. Washington, 2013*

Dans l'exemple ci-dessus chaque transaction client, comme on peut le voir sur le côté droit de la Figure 2, est un enregistrement qui contient l'identifiant du client, la date de la transaction et le montant dépensé à cette occasion. Le fichier que nous utilisons peut être considéré comme Big Data car il contient 343.766.402 transactions (enregistrements) effectués au cours de 78 semaines par 6.326.658 clients.

Ainsi, pour la *fréquence* d'achat, la fonction *Map* retourne *une paire clé / valeur* composée de l'identifiant du client et la valeur un. La fonction *Reduce* sera, soit la somme soit la longueur du vecteur contenant ces valeurs groupées par la clé.

Pour la *récence* qui, dans ce cas, est la date de la dernière transaction d'un client, la fonction Map doit retourner, en tant que valeur de la paire clé / valeur, la date de la transaction, tandis que la fonction Reduce fusionnera la date maximale par client.

Pour le *montant* qui pourrait être le montant total ou moyen dépensé par chaque client, la fonction Map doit retourner en tant que valeur du couple clé / valeur le montant dépensé par transaction, tandis que la fonction Reduce fusionnerait par client, soit la somme soit la moyenne du montant dépensé.

Si les données étaient trop grandes et devaient être traitées sur un cluster d'ordinateurs, le processus de MapReduce (voir Figure 2) lirait le fichier d'entrée et le diviserait (*split*) en plusieurs morceaux (*chunks*). Chaque morceau se voit associé à un exemplaire du programme Map et ces derniers sont exécutés en parallèle sur les nœuds du cluster pour regrouper les données d'un morceau par client. Le système prend alors la sortie de chaque programme Map et fusionne (*shuffle / sort*) les résultats pour le programme Reduce, qui calcule la longueur (ou la somme des valeurs) du vecteur par client pour obtenir la *Fréquence*, la valeur maximale pour la *Récence* et

la somme ou la valeur moyenne pour le *Montant*.

Comme R est également un langage fonctionnel, les fonctions Map et Reduce et / ou des fonctions d'ordre supérieur équivalentes peuvent aussi être utilisées sur une seule machine. Dans ce cas, une fonction qui applique l'approche de MapReduce que nous avons utilisée est *tapply*. Sa syntaxe est tapply (valeur, clé, FUN) où la valeur et la clé peuvent être les sorties d'une fonction Map tandis que FUN est une fonction Reduce de fusion comme *somme, moyenne, longueur*, etc.

Par conséquent, en R, appliquer l'approche MapReduce (mais pas le processus MapReduce) aux données de la Figure 2 consiste à utiliser :
tapply(rep(1, length(client)),client, sum) pour le calcul de la fréquence d'achats par client, F
tapply(date,client, max) pour le calcul de la récence des achats par client, R
tapply(montant,client, sum) pour le calcul du montant des achats par client, M

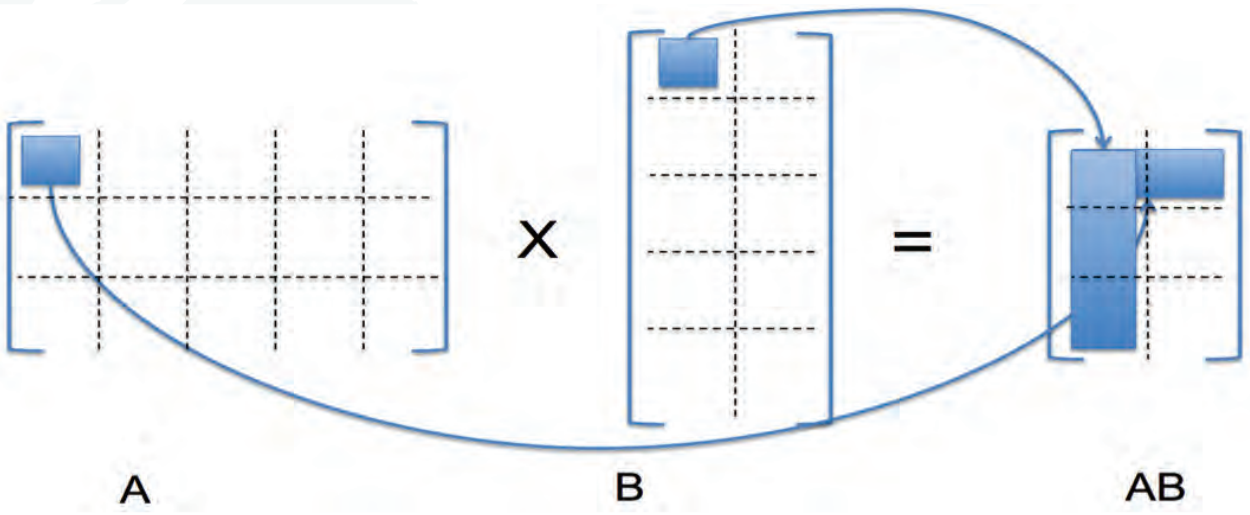
*Mettre en œuvre un processus de MapReduce avec R et Hadoop*. Lorsque les données sont trop grosses pour la mémoire d'un ordinateur, un processus MapReduce sur un cluster d'ordinateurs doit être mis en place. MapReduce par rapport aux approches HPC standard est un processus de type "partage rien".

Le processus de MapReduce peut être mis en œuvre après l'installation du système de Hadoop open-source sur chaque ordinateur de la grappe. Une série de packages R pour mettre en œuvre la solution RHadoop est disponible gratuitement auprès de *Revolution Analytics*. Ces packages gardent transparentes toutes les complexités du calcul parallèle pour les utilisateurs et sont donc hautement recommandables aux analystes marketing.

## MapReduce dans des calculs statistiques avancés utilisés en marketing

Alors que les calculs d'agrégation pour obtenir des variables RFM qui résument le comportement transactionnel du client dans la BDD marketing sont une application évidente de l'approche MapReduce, ce n'est pas toujours le cas des modèles de marketing quantitatif plus sophistiqués, mais beaucoup peuvent être adaptés pour MapReduce. De nombreux modèles statistiques utilisés en marketing quantitatif et analyse des données utilisent des calculs d'algèbre linéaire. Parmi ceux-ci la multiplication de matrices est un calcul très important qui peut être adapté à l'approche MapReduce (voir Figure 3 et 4)

Figure 3 - La multiplication de matrices adaptée pour MapReduce

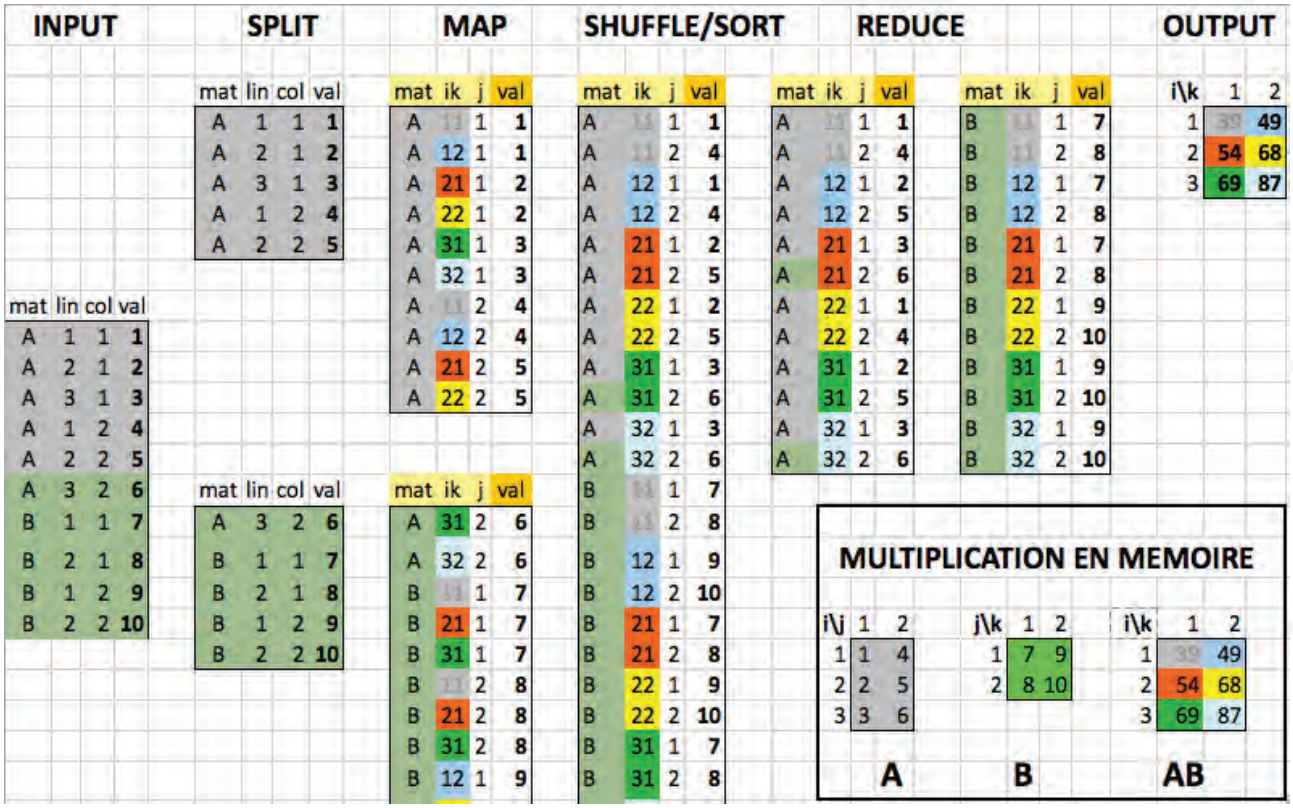


Il est bien connu que dans la multiplication matricielle la matrice résultat (AB) est la somme des produits des cellules de la ligne correspondante dans la première matrice (A) et des cellules de la colonne correspondante dans la deuxième matrice (B). Pour la phase "Reduce" toutes ces cellules doivent alors avoir la même clé, qui correspond à la position (ligne et colonne) de ce calcul dans la matrice résultat (AB).

La matrice résultat a les dimensions *IxK*, où I est le nombre de lignes de la première matrice et *K* est le nombre de colonnes de la deuxième matrice et *J*, le nombre de colonnes de la première matrice, doit être égal au nombre de lignes de la deuxième.

Comme le montre la Figure 3 chaque cellule (*i,j*) dans la matrice A est utilisée *K* fois pour le calcul des cellules de la ligne (*i*) dans la matrice résultat et chaque cellule (*j,k*) dans la matrice B est utilisée *I* fois pour le calcul des cellules de la colonne (*k*) de la même matrice (AB).
C'est la raison pour laquelle (voir Figure 4) dans la phase "Map" chaque valeur (cellule) de A doit être dupliquée *K* fois et se voit associée une clé (*i,k*) où *k=1..K* et chaque valeur (cellule) de B doit être dupliquée *I* fois et se voit associée une clé (*i,k*) où *i=1..I*.
La phase « reduce » calculera pour chaque cellule résultat (*i,k*) la somme de tous les produits des éléments des deux matrices associés à la même clé.

Figure 4 – Etapes du processus MapReduce de multiplication de matrices




La Figure 4 illustre le processus MapReduce comparé à la multiplication classique en mémoire. Dans la phase "*split*" le fichier qui représente les deux matrices regroupées est divisé en deux morceaux (*chunks*) et chaque morceau est attribué à un nœud (ordinateur). On remarque que la matrice A, étant "trop grande" pour le premier nœud, a vu son dernier élément passer dans le "*chunk*" destiné au deuxième ordinateur. La phase "*map*" se déroule séparément sur chaque nœud et les valeurs des éléments de chaque matrice sont dupliqués et associées à des clés, comme expliqué précédemment, pour pouvoir d'abord être triés sur chaque ordinateur. La phase "*shuffle/sort*" récupère ces données pour les regrouper et trier au niveau du "*cluster*".

Finalement la phase "*reduce*" calcule la somme des produits des valeurs des deux matrices qui ont la même clé.

Des modèles comme la régression linéaire, l'analyse factorielle ou l'analyse discriminante appliquent des algorithmes de calcul plus sophistiqués (inversion, diagonalisation) à une matrice symétrique d'assez petites dimensions qui dépendent du nombre de variables qu'elle résume et non du nombre d'observations. Cette petite matrice des sommes des produits croisés des variables est obtenue par la multiplication de deux matrices qui, elles, peuvent être des Big Data car elles contiennent les observations qui peuvent être très nombreuses.

Le modèle de programmation parallèle des données de MapReduce cache la complexité de la distribution et de la tolérance aux pannes. Sa philosophie principale est d'augmenter à l'échelle la puissance de calcul par l'ajout de matériel pour faire face à la taille des problèmes, mais aussi d'économiser les coûts de matériel, programmation et administration. MapReduce ne convient pas à tous les problèmes mais de nouveaux modèles et environnements de programmation sont encore en cours de création et renforcent ces idées. Comme nous avons pu le voir, les solutions de calcul doivent être adaptées à MapReduce car les programmes classiques ne peuvent faire le travail lorsque les données sont réparties entre plusieurs nœuds. Une solution prometteuse est l'approche DAG (graphes acycliques orientés), un style de programmation pour les systèmes distribués. Il peut être considéré comme une alternative à MapReduce. Alors que MapReduce comprend seulement deux étapes (Map et Reduce), DAG peut avoir plusieurs niveaux pouvant former une structure arborescente et DAG est donc plus souple avec davantage de fonctions comme map, filter, union, etc. Son exécution est aussi plus rapide grâce à la non écriture sur disque des résultats intermédiaires. Une des mises en œuvre les plus plébiscitées de DAG est le projet Spark de Apache. Son principal concept de RDD (Resilient Distributed Datasets) est expliqué dans Zaharia et al. (2012)[1] de l'Université de Berkley, à l'initiative de cette approche. Spark amène MapReduce à un niveau supérieur avec des remaniements moins couteux dans le traitement des données. Avec des fonctionnalités telles que le stockage de données en mémoire et le traitement en temps quasi-réel, la performance peut être plusieurs fois plus rapide qu'avec les autres technologies Big Data.

## D'autres solutions HPC pour les scientifiques (analystes) marketing

Alors que MapReduce et ses descendants cachent complètement la complexité du calcul distribué et parallèle et sont donc facilement implémentables comme services et accessibles à tous par le biais du *cloud computing*, il existe tout un ensemble de méthodes et facilités HPC également disponibles pour les universitaires scientifiques du marketing. De nombreuses universités ont ou participent à des projets de *clusters* ou *grids* d'ordinateurs. Les auteurs, membres de l'université de Lille qui occupe en France le cinquième rang du point de vue de la capacité de calcul de son cluster d'ordinateurs[2], ont pu utiliser ces installations et une série de packages R afin de tester des gains de performances sur la modélisation prédictive basées sur des variables RFM en faisant varier le nombre de noyaux et d'ordinateurs de la grappe. Pour une liste détaillée des paquets disponibles R on peut lire la page Web officielle de la CRAN task view concernant le calcul de haute performance et parallèle avec R[3].

1. En bref, les RDD sont des ensembles de données distribués qui peuvent rester en mémoire et se replier sur disque. Le RDD en cas de perte peut être facilement reconstruits à l'aide d'un graphe qui indique comment le reconstruire. Les RDD sont très bien quand il est nécessaire de garder un ensemble de données en mémoire et lancer une série de requêtes - cela fonctionne mieux que l'extraction de données à partir du disque à chaque fois. Un autre concept important RDD est qu'il existe deux types de choses qui peuvent être faites sur un RDD : 1) des *transformations* comme, map, filter qui génèrent un autre RDD et 2) des *actions* comme count (compter) qui retournent des valeurs. Un job Spark se compose d'un DAG de tâches qui exécutent des transformations et des actions sur des RDDs.
2. Le cluster informatique utilisé se compose de machines hétérogènes qui incluent 2110 cœurs de processeurs (CPU) et 7168 cœurs de processeurs graphiques (GPU). Le cluster a une capacité de disque d'environ 156 Tb et un débit théorique de 50 Tflops. Toutes les machines sont reliées par un réseau Infiniband 40 Gbps.
3. https://cran.r-project.org/web/views/HighPerformanceComputing.html

Nous introduisons une approche brute d'enregistrement de l'analyse de la performance du calcul parallèle et l'appliquons à des calculs qui évaluent plusieurs modèles prédictifs basés sur des variables RFM du comportement d'achat. Les deux ensembles de données utilisés proviennent d'une chaîne de magasins et d'un catalogue de vente par correspondance. Ils reflètent le comportement d'achat répété pendant plusieurs saisons pour des cohortes de clients. Le premier ensemble de données a déjà été utilisé dans le présent document pour illustrer l'approche de MapReduce pour calculer des variables RFM et réduire un grand fichier de 344 millions de transactions à un fichier plus petit mais toujours assez grand avec 6,3 millions de clients. Afin de mesurer l'impact des variables RFM sur l'incidence d'achat, nous utilisons ce fichier pour estimer plusieurs modèles prédictifs : linéaire, logit, probit, réseaux de neurones ou les arbres de classification (CART). Comme le fichier est assez grand, l'objectif principal est ici de mesurer le temps qu'il faut pour calibrer un modèle en utilisant un noyau de la CPU, ici le microprocesseur Intel i7 d'un ordinateur MacBook Pro. Le temps le plus court, 4-5 secondes, a été atteint avec la régression linéaire et le plus long avec les réseaux neuronaux a pris plus de 820 secondes ou 14 minutes et 320 itérations pour converger.

Comme la régression linéaire multiple peut être résolue facilement en mimant une approche MapReduce nous l'avons appliquée sur le même ordinateur MacBookPro, pour comparer des calculs sériels à des calculs parallèles implicites en faisant varier le nombre de cœurs du processeur de 2 à 4. La régression linéaire peut être résolue par des multiplications simples de la matrice contenant les valeurs des variables indépendantes et une première colonne de uns, appelée **X** et d'un vecteur contenant les valeurs de la variable dépendante, appelée **y**. En supposant que la mémoire est limitée, ces grandes tables de 6,3 millions enregistrements ont été réparties en fonction du nombre de cœurs dans 2 à 4 morceaux. Des *fonctions Map* ont été définies pour calculer des multiplications par morceau (*chunkwise*) **X'X** et **X'y** et appliquées à la fois en série et en simultané en utilisant le parallélisme implicite. Pour la dernière, la fonction *mcapply* du package *parallel* de R a été utilisée.

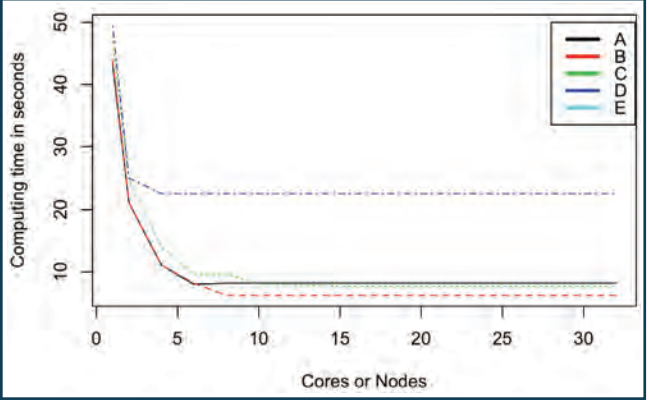
La *fonction Reduce* fait la somme des résultats par morceaux de ces multiplications particulières en exploitant simplement le fait que la somme des matrices est associative et commutative. Nous avons répété le calcul en série et en parallèle sur 2, 3 et 4 morceaux une centaine de fois. Le test t apparié a montré une différence positive importante en temps de calcul entre les calculs série et parallèle ($t = 40{,}2006$, $dl = 299$, p-valeur $<2{,}2e\text{-}16$). Alors que les calculs en série avaient une durée moyenne de 1,99 secondes, les calculs parallèles n'ont duré que 1,44 secondes et la différence moyenne était de 0,51 secondes. La même différence positive significative est observée en analysant les calculs avec coupure en 2 morceaux (0.60855s), 3 morceaux (0.21772s) et 4 morceaux (0.68903) séparément.

Le second fichier avec ses six mille enregistrements représentant l'incidence d'achat des clients en vente par correspondance est beaucoup plus petit et a été utilisé pour effectuer une validation croisée de type "*leave-one-out*" du modèle afin de limiter la surestimation et améliorer la validité prédictive. La validation croisée, comme le *boostrapping* est un problème dit « **embarrassingly parallel** » (aussi appelé *perfectly parallel* ou *pleasingly parallel*), c'est-à-dire facile à paralléliser. La validation croisée leave-one-out pour nos 6000 observations impliquait l'ajustement du modèle répété 6000 fois sur des ensembles obtenus en éliminant à chaque fois une observation. Ce calcul répétitif peut facilement être parallélisé aussi bien pour le parallélisme *implicite ou multicœurs*, signifiant le partage de la même mémoire par plusieurs noyaux, que pour le *parallélisme en cluster* qui regroupe plusieurs ordinateurs.

Tableau 2– Baisse de durée avec la parallélisation

| Cores/Nodes | A | B | C | D |
|---|---|---|---|---|
| 1 | 43.217 | 43.817 | 46.802 | 49.445 |
| 2 | 21.139 | 21.133 | 24.683 | 24.985 |
| 4 | 11.151 | 11.131 | 13.995 | 22.484 |
| 6 | 8.026 | 8.208 | 9.694 | 22.484 |
| 8 | 8.211 | 6.250 | 9.650 | 22.484 |
| 10 | 8.211 | 6.250 | 8.143 | 22.484 |
| 16 | 8.211 | 6.250 | 7.676 | 22.484 |
| 32 | 8.211 | 6.250 | 7.721 | 22.484 |

Figure 5: Gains de vitesse du calcul parallèle




A: linéaire, noeuds1/coeurs >0; B: logistic, noeuds1/coeurs >0; C: logistic, noeuds > 0/coeurs 1; D : logistic, interactive, noeuds 1/coeurs >0

Les trois premières colonnes du tableau 2 enregistrent des gains de performance de parallélisation à l'aide de calculs batch (par lots) sur des serveurs Dell PowerEdge ayant des CPU à 8 cœurs et appartenant au même cluster d'ordinateurs. La dernière colonne enregistre des gains de performances similaires sur un portable MacBook Pro avec CPU Intel I7 à 4 cœurs. Les colonnes A, B et D montrent des gains de performance obtenus en augmentant le nombre de noyaux, tandis que la colonne C analyse des gains de vitesse en augmentant le nombre de machines (nœuds) dans les clusters. On peut observer que lorsque le nombre maximum de noyaux est atteint, ici huit pour les serveurs et 4 pour l'ordinateur portable, aucune amélioration supplémentaire de la performance n'est possible. Les gains de performances tout en augmentant le nombre de machines ou noeuds (colonne C) peuvent être obtenus de façon constante mais, là aussi, il y a une limite lorsque les gains de performance deviennent moins importants.

## Conclusion

Le marketing repose de plus en plus sur la technologie de l'information, qu'il s'agisse du choix de canal, de la personnalisation et des systèmes de recommandation, de l'analyse des contenus générés par les utilisateurs, des commentaires en ligne et de l'influence sociale dans les réseaux sociaux en ligne (Tableau 1). Le marketing est maintenant considéré comme l'un des moteurs (*driver*) des technologies Big Data, tout comme l'a été la comptabilité pour les bases de données dans les années 80. La technologie a toujours transformé la science du marketing en suivant une tendance assez systématique et prévisible. Cela implique des changements dans la formulation des problématiques, dans les méthodes, et de mettre davantage l'accent sur les aspects analytiques du marketing. Les scientifiques du marketing ont de bonnes connaissances en statistiques, en économétrie et en recherche opérationnelle, mais semblent en avoir beaucoup moins en matière de programmation moderne et de technologie de l'information. Beaucoup ignorent encore des fonctionnalités qui profitent du contexte distribué et riche en données fourni par l'Internet, le cloud computing et le HPC. Négliger les facteurs qui favorisent la facilité d'usage des modèles (convivialité) risque de rendre ces derniers non pertinents et de limiter leur utilisation. En essayant de démystifier les approches Big Data cet article invite les statisticiens et analystes marketing à accorder plus d'attention aux évolutions technologiques, à participer davantage à l'élaboration d'éléments analytiques spécifiques et à ne pas abandonner le champ aux informaticiens. Démystifier les approches et technologies Big Data ne signifie pas les banaliser mais, au contraire, insister sur la haute importance et les changements de rupture qu'elles engendrent pour la société en général, pour la science statistique et pour la science du marketing en particulier.

## Références


- Akter S. et S.F. Wamda (2016), Big data analytics in E-commerce: a systematic review and agenda for future research, *Electronic Markets* 26:173–194
- Bello-Orgaza G., Jungb J. J., Camachoa D. (2016) Social big data: Recent achievements and new challenges, *Information Fusion* 28: 45–59
- Wickham H. (2015), R Packages. Sebastopol, CA: O'Reilly Media, Inc.
- Wijffels J., (2013), "ffbase: statistical functions for large datasets", The R User Conference, Albacete, Castilla-La Mancha
- Zaharia M., Chowdhury M., Das T., Dave A., Ma J., McCauley M., Franklin M.J., Shenker S., Stoica I. (2012) Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing, NSDI 2012, April.