UNIVERSITÉ DU QUÉBEC À MONTRÉAL

L'ANALYSE DE DONNÉES EN TANT QUE SERVICE POUR LA MISE EN CACHE DES VIDEOS DANS LES CDNS

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INFORMATIQUE

PAR

EMIRA BEN ABDELKRIM

NOVEMBRE 2016

REMERCIEMENTS

Mes premiers remerciements s'adressent à toute ma famille et particulièrement à mes parents et à mon frère. Leur amour, leur soutien indéfectible et la confiance qu'ils ont toujours placée en moi n'ont cessé de me porter.

Merci à toi Maman de m'avoir toujours accompagné dans tous mes choix.

Je voudrais aussi exprimer ma profonde reconnaissance pour mon encadrant de rechercher Professeur. Halima Elbiaze pour m'avoir offert son savoir et son encadrement.

Je remercie de tout cœur la Fondation de l'UQAM, pour les bourses d'excellence qui m'ont été attribuées ; ces bourses ont grandement facilité mes études.

Je tiens aussi à remercier tous mes amis et collègues du TRIME pour la bonne ambiance qui y régnait. Je citerai notamment Mahdi, Rami, Zakaria et Zoubair mais je pense que les autres se reconnaîtront dans mes propos. Travailler en leur compagnie a été pour moi un vrai plaisir.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ALGORITHMES

LISTE DES ABRÉVIATIONS

LISTE DES ABRÉVIATIONS

CDN	*Content Delivery Network* ou réseau de diffusion de contenu.
IaaS	*Infrastructure as-a service* ou infrastructure en tant que service, est un modèle de cloud computing destiné aux entreprises.
PaaS	*Platform as-a service* ou Plate-forme en tant que service.
SaaS	*Software as-a service* ou logiciel en tant que service est un modèle d'exploitation des logiciels dans le *Cloud.*
AaaS	*Analytics as-a service* ou Analyse de données en tant que service.
LRU	*Least Recently Used* est un algorithme de remplacement de caches.
LFU	*Least Frequently Used* est un algorithme de remplacement de caches.
UGV	*User Generated Video* ou vidéo générée par les utilisateurs.
REST	*REpresentational State Transfer* est un style d'architecture pour les systèmes distribués.
URI	*Uniform Resource Identifier*, identifiant d'une ressource dans un réseau informatique.
API	*Application Programmable Interface*, est un ensemble de fonctions permettant d'accéder aux services d'une application.
R^2	R Carré est un indicateur basé sur la régression linéaire qui calcule le taux de corrélation entre les observations et la droite de régression linéaire.
MSE	pour *Mean Squared Error* ou l'erreur quadratique moyenne est très utile pour comparer plusieurs modèles de régression.

Cp de Mallows	Permet de choisir entre plusieurs modèles de régression.
AHR	Average Hit Ratio ou Taux de succès de cache moyen.

RÉSUMÉ

Ce mémoire explore l'analyse de données en tant que service, AaaS de l'anglais Analytics as-a Service, pour la mise en cache des vidéos dans les réseaux de distribution de contenu (CDN). La capacité de prédiction des popularités futures des vidéos, couplées avec les récents développements dans le domaine du *Big Data*, permet de réduire considérablement le nombre de défauts de caches dans la demande des utilisateurs en servant de manière pro-active les vidéos en fonction de leur demandes (popularité) potentielles. Pour montrer la faisabilité d'une telle approche, nous aborderons le problème sous différent ongles à savoir architectural et algorithmique.

Dans la première partie de ce mémoire, nous introduirons l'architecture Cloud pour énoncer le gain résultant de l'utilisation du AaaS dans un CDN. Nous nous focaliserons en particulier sur 1)- Le domaine du fournisseur de service CDN dans lequel les serveurs cache ont une capacité de stockage et de calcul limitée. 2)- Le domaine du Cloud. Dans ce dernier, nous utiliserons un IaaS (Infrastructure en tant que service) pour nous garantir les capacités nécessaires en traitement, en stockage et en mise en réseau. Par-dessus l'IaaS, nous réutiliserons tout PaaS (Plateforme en tant que service) existant qui supporte les *Frameworks* requis par les applications AaaS. 3)- Les interfaces régissant les échanges entre le CDN et le Cloud. Nous y caractérisons les ressources et les protocoles utilisés.

Dans la seconde partie, nous nous focaliserons sur un cas d'utilisation pratique de l'AaaS pour CDN. Nous nous concentrerons sur la prédiction des popularités des vidéos et de l'aspect algorithmique. En particulier 1)- Nous présenterons dans un premier temps les modèles de régressions linéaires et leur utilité pour la tâche de prédiction.2-) Nous proposerons ensuite un algorithme hybride qui utilise la régression statique, puis l'adapte de façon continue, similaire à une régression dynamique dans le but d'avoir une meilleure prédiction des popularités des vidéos. 3)- Pour l'estimation de la popularité des vidéos en pratique nous utiliserons un jeu de données de vidéos de YouTube faisant plus de 150 millions d'observations.

Nos résultats et analyses fournissent une nouvelle approche pour le déploiement de l'AaaS dans les CDNs. L'application proposée peut être implémentée pour une meilleure gestion du cache dans un CDN.

Mots clés : CDN, *Cloud Computing*, Analytics as-a-service, régression linéaire.

INTRODUCTION

0.1 Mise en contexte

Le trafic Internet a augmenté de façon exponentielle ces dernières années. D'après le rapport Cisco *Visual Networking Index* (VNI), 47% de ce trafic provient de la vidéo en 2014. Cette part devrait atteindre 90% d'ici 2019[1].

En effet, des sites de partages de vidéo comme YouTube[2] deviennent très vite achalandés avec toujours plus de visiteurs, ce qui entraîne une baisse de performance et de qualité de l'expérience utilisateur. Le besoin de CDN répond à cette complexification. En effet, un CDN permet aux utilisateurs d'accéder rapidement au contenu, quel que soit son positionnement dans le monde et donc de réduire les temps de latence liés à la distance séparant le serveur d'hébergement et l'utilisateur final. Ainsi, au lieu d'être adressées au serveur source, les requêtes des utilisateurs sont plutôt transmises à un serveur relais (intermédiaire) qui est choisi selon des règles de routage bien établies. On s'entend toutefois que la taille de ces serveurs soit limitée et donc qu'ils ne puissent pas contenir toutes les vidéos. Seule une partie de ces vidéos sera accessible aux utilisateurs depuis ces serveurs. Par conséquent, il devient indispensable de déterminer les vidéos qui sont les plus avantageuses (bénéfiques), c'est-à-dire les plus populaires.

Les algorithmes de remplacement de cache effacent le contenu le moins utilisé dans une fenêtre temporelle bien déterminée pour faire de la place aux contenus

---

1. Cisco Visual Networking Index : Forecast and Methodology, 2014–2019

2. `https://www.youtube.com`

les plus populaires. Par exemple, les algorithmes Least-Recently Used (LRU) et Least-Frequently Used (LFU) présument que les vidéos les plus regardées dans le passé continueront d'être populaires dans l'avenir (Famaey *et al.*, 2011). Ces stratégies de remplacement de cache ont montré d'excellents résultats en termes de succès de cache dans la mesure où elles traitent du contenu statique. Cependant, la popularité en tant que telle est dynamique et fluctue considérablement en fonction des interactions des utilisateurs avec ces vidéos (Zhou *et al.*, 2015). Les vidéos les plus populaires sont forcément celles qui sont les plus vues et les plus partagées. Par conséquent, les algorithmes de remplacement de cache doivent désormais être adaptés pour tenir en compte des statistiques dynamiques liées à ces vidéos. Jusqu'ici, les CDNs fournissent des services intermédiaires aux sites de partages de vidéos comme YouTube ou Facebook et ne recueillent toutefois pas ce genre de statistiques sans accord préalable du fournisseur du contenu. Cela ouvrirait de nouvelles perspectives intéressantes pour un certain nombre de CDN en intégrant ces statistiques comme le nombre de vues d'une vidéo, le temps de vue d'une vidéo ou le nombre de partage d'une vidéo dans leurs calculs de popularité des vidéos.

0.2 Problématique

Jusqu'à récemment, il était difficile, voire impossible de traiter des données de l'ordre de quelques gigaoctets. Le *Big Data* a levé cet obstacle de calcul et a permis la conservation et l'exploitation de grandes masses de données. Cette avancée technologique a fortement contribué au développent et à la vulgarisation de nombreux algorithmes de fouille de données ou *Data mining* qui avec le temps, sont devenus très courant. Aujourd'hui, ces algorithmes sont adoptés dans beaucoup de domaines qui ont à faire face au traitement de grande quantité de données. Les CDNs ne devraient pas échapper à cette tendance. En effet, les serveurs dans un CDN produisent un certain nombre de journaux de traces communément appelés

logs dans lesquels sont consignées les activités de tous les utilisateurs transitant par le CDN. Ces *logs* représentent à eux seuls une vraie mine d'information. Un CDN peut utiliser ces données afin de prédire quelle vidéo sera populaire en se basant sur leurs statistiques comme le nombre de vues ou encore le nombre de partages. Les résultats obtenus peuvent servir de base pour un algorithme de remplacement de cache.

Malheureusement, en dépit de tous ces avantages, un CDN doit d'une part, recueillir et conserver d'impressionnant volume de données (*logs*) provenant de sources hétérogènes, c'est-à-dire les serveurs et d'autre part, investir dans du stockage coûteux, du matériel de calcul assez puissant et dans des logiciels d'analyse. Du coup, il serait particulièrement difficile de mettre en place une telle architecture en pratique. *L'analytics as-a service (AaaS)* réponds à ce besoin en donnant aux CDNs la possibilité de lancer et tester plus facilement leurs applications "analytiques" sans avoir à se préoccuper ni de l'infrastructure ni de la plateforme logicielle.

0.3 Objectifs projetés

L'objectif principal de ce travail est l'amélioration de la mise en cache des vidéos en identifiant les vidéos populaires sur la base des traces *logs* recueillies dans un CDN. Ces performances sont exprimées en fonction de l'erreur de prédiction et du taux de succès de cache.

L'erreur de prédiction est centrale pour évaluer la capacité de prédiction des popularités des vidéos et se définit comme étant la différence entre la valeur réelle et la valeur prédite. Elle permet ainsi de sélectionner le meilleur modèle de prédiction des popularités des vidéos parmi une collection de modèles et donne une mesure de confiance à accorder au meilleur modèle (Seber et Lee, 2012).

On parle de succès de cache lorsqu'une vidéo est chargée depuis le réseau du CDN puis envoyé à l'utilisateur. A défaut, la vidéo sera chargée depuis le serveur d'origine du client ou le fournisseur du CDN (par exemple le site web qui est hébergé dans le CDN). Il est évident qu'un défaut de cache impliquerait un délais supplémentaire dans le temps de chargement de la vidéo. Le taux de succès de cache est le rapport du nombre des demandes de vidéos qui ont donné un succès de cache par le nombre totale de toutes les demandes (Stamos *et al.*, 2008).

Notre objectif projeté est de réduire l'erreur de prédiction c'est à dire définir un modèle de prédiction qui prédit avec le moins d'erreur la popularité des vidéos. Ce modèle sera utilisé dans un algorithme de mise en cache (de remplacement de cache) de manière à maximiser le taux de succès de cache.

Cet objectif est divisé en différents sous-objectifs. D'abord, nous proposerons une architecture en *Cloud* qui permettra le développement d'applications d'analyse de données pour CDN. Ensuite, nous développerons un algorithme de régression hybride, compromis entre la régression statique et dynamique afin de prédire la popularité des vidéos. Enfin, nous comparerons et nous validerons les résultats par des simulations.

0.4 Contributions

Ce mémoire étudie le problème de mise en cache des vidéos dans un réseau CDN. Afin de résoudre ce problème, les contributions suivantes seront présentées.

— Nous présentons les éléments théoriques d'une architecture en *Cloud* pour l'analyse des traces de *logs* recueillies dans un réseau CDN. En particulier, les CDNs fournissent ces *logs* au logiciel AaaS et récupèrent en retour des analyses spéciales qui permettent de comprendre la dynamique des popularités des vidéos, très utile pour leurs mise en cache. Cette architecture n'a pas été implémenté dans le cadre de ce projet de maîtrise et ne constitue en

soi qu'une proposition pour accélérer l'usage de l'analyse de données par les CDNs (pour améliorer, en autres, la gestion de leurs caches).

— Nous concevrons un algorithme hybride basé sur la régression linéaire multiple pour la mise en cache des vidéos dans les CDNs.

— Nous évoluerons l'algorithme hybride en comparant ces performances à ceux des régressions statique et dynamique. Dans l'approche statique, les données historiques des vidéos sont fixées. Nous proposons une approche hybride qui est un compromis entre la régression statique et dynamique. L'algorithme hybride utilise initialement les données historiques des vidéos pour déterminer la droite de régression de la popularité. Cette droite est mise à jour en fonction des données vidéos les plus récentes comme dans la régression dynamique. Cependant, nous utilisons un facteur de lissage afin d'écarter les données vidéo qui deviennent obsolètes. Nous soulignons aussi que notre méthode hybride s'exécute dans des fenêtres de temps (tranche horaire, journée, semaine, etc.) quelle qu'en soit la durée.

— Nous montrerons que l'algorithme hybride donne de meilleurs résultats en termes de prédiction par rapport aux algorithmes statique et dynamique. Nous évaluerons la méthode hybride en utilisant un vaste jeu de données vidéo. Nous montrons des améliorations significatives dans la prédiction, avec une amélioration moyenne de 14% dans l'erreur quadratique moyenne, couplée à une légère augmentation en temps d'exécution et en espace mémoire utilisé.

— Et finalement, nous démontrerons l'efficacité de l'algorithme proposé pour la mise en cache des vidéos en le comparant aux algorithmes de *caching*, les plus forts connus actuellement. Nos résultats indiquent que notre algorithme hybride permet une importante amélioration du taux de sucée de cache. Nous avons mesuré une augmentation de succès de cache entre 5% et 74%.

0.5 Organisation

Le reste de ce mémoire est organisé comme suit :

Le chapitre 1 exposera le cadre théorique. Tout d'abord, nous y présenterons la problématique d'explosion des vidéos dans un CDN et nous soulignerons le bénéfice potentiel que peut être obtenu à partir de l'analyse de données en tant que service (AaaS). Puis dans un second temps, nous élargirons la recherche sur un cas d'application de AaaS pour soutenir les stratégies de mise en cache au sein d'un CDN.

Le chapitre 2 présentera l'architecture de l'AaaS et mettra en lumière les différents participants et les interfaces correspondantes. Concrètement, nous présentons d'abord le domaine du fournisseur de services CDN. Ensuite nous présenterons le domaine du *Cloud Computing*. Enfin, les interfaces AaaS-PaaS et CDN-PaaS y seront détaillées.

Le chapitre 3 décrira l'application de prédiction de popularité. Nous donnerons d'abord le schéma de prétraitement des *logs* (donnés vidéos). Ensuite, nous introduirons les modèles de régression statique et dynamique. La méthode hybride que nous proposerons est un compromis entre la régression statique et dynamique.

Le chapitre 4 quand à lui, sera axé sur l'implémentation de l'algorithme hybride. Nous démontrerons que notre algorithme conduit à de meilleures performances en maximisant le taux de succès du cache. En outre, nous montrerons que des gains très visibles pourront être obtenus en réduisant la fenêtre de prédiction.

Enfin, la conclusion fera office de bilan de ce mémoire.

CHAPITRE I

CONCEPTS ET DÉFINITIONS

Ce premier chapitre introduit les aspects théoriques liés à notre projet. Dans un premier temps, une description des réseaux CDNs sera effectuée pour mettre en évidence les concepts de base liée à ce type de réseau. Les notions de mise en cache et de distribution de contenu en se basant sur l'évolution du critère de popularité y seront alors introduites. L'explosion des contenus du type vidéo ainsi que l'attente des utilisateurs finaux en matière de qualité de service ont souligné la nécessité d'améliorer la distribution de ce type de contenu. Un état de l'art sur les travaux de recherche menées dans ce sens y sera alors détaillé. Dans une seconde partie, l'usage de l'analyse de données sera étudié. Nous nous intéresserons dans un premier temps à son usage actuel dans CDN. Puis dans un second temps, nous introduirons l'analyse de données en tant que service pour améliorer la distribution et la mise en cache des vidéos dans un CDN à coûts réduits.

1.1 Les réseaux de distribution de contenu (ou CDN pour Content Delivery Network)

Au cours des dernières décennies, nous avons été témoins de la croissance et de la maturité d'Internet entraînant une croissance massive du trafic réseau. La nature évolutive de l'utilisation d'Internet impose de nouveaux défis quant à la gestion et la livraison de contenu aux utilisateurs. Les éditeurs de sites Web font face souvent à des goulots étranglement en raison des grandes demandes posées sur leurs contenus. Une hausse soudaine des demandes de contenu Web (par exemple celle produite lors des attaques du 11 septembre) est souvent qualifiée d'effets *flashs crowds* (Arlitt et Jin, 2000) ou *Slashdot* (Cronin, 2001).

Composer avec de telles demandes imprévues imposerait d'importantes surcharges sur un serveur web et éventuellement ce dernier serait totalement envahi par l'augmentation soudaine du trafic. Ainsi le site web en question deviendrait temporairement indisponible pour cause de cette surcharge dans les demandes.

Les CDNs viennent pallier ce problème en offrant les mécanismes et l'infrastructure de façon à fournir un contenu et des services très performants tout en améliorant l'expérience Web des utilisateurs (Pathan *et al.*, 2008).

1.1.1 Concepts de base

Un CDN est formé d'un ensemble de serveurs cache dispersé à travers le monde. Ces serveurs sont situés en bordure du réseau **(la bordure s'entendant ici comme la frontière virtuelle entre le réseau du CDN et les connexions utilisateurs)** et auxquels les utilisateurs sont connectés. Le CDN répartit les contenus sur les serveurs cache afin de faire parvenir le contenu aux utilisateurs finaux de façon fiable et rapide. Les contenus, qui font (ou feront) l'objet de requêtes répétées, sont dupliqués sur les différents caches soit sur demande (de manière réactive) ou à l'avance (de manière proactive). Les contenus Web sont transmis aux utilisateurs à partir du serveur cache le plus proche.

Dans *Content Delivery Network*, l'appellation *Content Delivery* ou l'équivalent en français diffusion de contenu fait référence à l'action de servir du contenu Web sollicité par les utilisateurs. Le mot *content* ou sa traduction française contenu concerne toute ressource numérique. Il comporte notamment des données multimédias statiques, dynamiques et continues (Exemples. audio, vidéo, documents, images et page Web).

Les trois entités principales qui interagissent directement avec le système CDN sont le fournisseur de contenu, le fournisseur du CDN et les utilisateurs finaux. Le

fournisseur de contenu est celui qui mandate les adresses URL (Uniform Resource Locator) des ressources Web à repartir. Le serveur d'hébergement ou d'origine du fournisseur de contenu détient la collection de tous les contenus. Le fournisseur CDN est une entreprise qui apporte l'infrastructure nécessaire aux fournisseurs de contenu afin de leur permettre de livrer du contenu en temps voulu et de manière fiable. Les utilisateurs finaux ou clients accèdent au contenu au travers le site Web du fournisseur de contenu. Le CDN fonctionne comme un intermédiaire entre le fournisseur de contenu et l'utilisateur final.

Dans un CDN, les contenus sont diffusés de manière à ce que tous les serveurs cache partagent les mêmes contenus et URL. Le fournisseur CDN redirige les demandes des utilisateurs sur la base d'un algorithme de routage vers le serveur cache qui est capable de servir le contenu le plus rapidement. En outre, les serveurs cache envoient des données relatives au trafic du réseau au fournisseur CDN via le système de facturation pour des raisons de facturations, de contrôle de performance et de surveillance (Pathan *et al.*, 2008).

1.1.2 Architecture

La figure 1.1 donne l'architecture générale du CDN. On y retrouve ces trois composants principaux (Pathan *et al.*, 2008) :

— Le *système de redirection* responsable de la redirection des demandes des utilisateurs vers le serveur cache le plus adapté (par exemple le plus proche et le moins charge).
— Le *système de distribution* interagit avec le système de redirection de manière à tenir à jour les contenus stockés dans les serveurs caches. Aussi, il permet d'alimenter les serveurs cache en avance de façon optimale. En autres il peut utiliser des protocoles traditionnels de *caching*.
— Le *système de facturation* maintient la traçabilité des transactions des accès

aux contenus. Ces données sont utilisées pour des raisons de facturation, de maintenance et de contrôle.

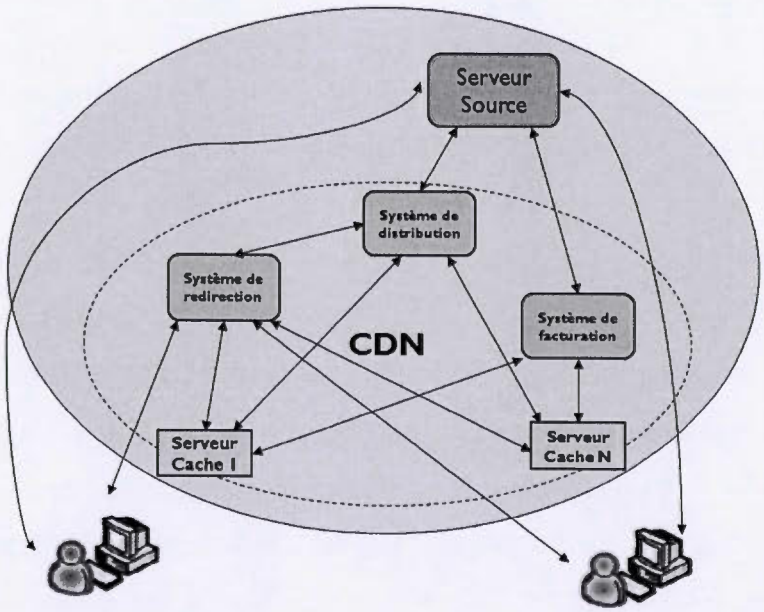

Figure 1.1 Architecture d'un CDN.

1.1.3 Modèle d'affaire

Le fournisseur de contenu s'entend avec le fournisseur de services CDN pour voir son contenu répliqué sur plusieurs serveurs caches. Dans la pratique, le CDN héberge du contenu statique (par exemple les pages HTML statiques, des images, des documents) et des ressources média (par exemple audio ou vidéo).

Les consommateurs typiques du CDN sont les médias, les sociétés publicitaires, les centres de données, les fabricants de matériel électronique grand public et d'autres sociétés de transports, etc. Chacun de ses fournisseurs de contenu souhaite publier et distribuer ses contenus de manière à optimiser le mécanisme de transmission.

Finalement, les utilisateurs finaux interagissent avec le CDN en spécifiant le contenu au travers de leur *smart phone*, ordinateur portable ou ordinateur de bureau (Pathan *et al.*, 2008).

Les fournisseurs de services CDN facturent leurs consommateurs en fonction du contenu livré (à savoir le volume du trafic) aux utilisateurs finaux par leurs serveurs périphériques. Comme énoncé plus haut, le CDN s'appuie sur un mécanisme de facturation qui collecte et retrace l'historique des transactions des utilisateurs finaux. Ce mécanisme s'exécute en temps réel et concerne chaque composant du réseau CDN. Les coûts estimatifs liés au service CDN sont assez élevés et souvent hors de la portée pour de nombreuses petites et moyennes entreprises et des organisations à but non lucratif. Les facteurs les plus influents affectant le prix du service CDN sont entre autres la quantité de la bande passante utilisée, la taille du contenu et la taille du CDN, à savoir le nombre de serveurs cache qu'il déploie (Pathan *et al.*, 2008).

1.1.4 Les techniques de mise en cache

La popularité des CDNs provient en grande partie de leur potentiel à proposer efficacement du contenu sur tout le Web. Par conséquent, les besoins en diverses techniques de mise en caches dans CDN sont importants pour améliorer la diffusion des contenus sur le Web.

Les techniques de mise en cache décident quels contenus sont à supprimer du cache pour en mettre des nouveaux (Stamos *et al.*, 2008). Dans ces techniques, on associe une valeur, communément appelée Content Utility Value (CUV), à chaque objet ou contenu. L'objet avec le plus petit CUV sera le premier à être éliminé du cache. Les techniques de mise en caches sont en général divisées en deux catégories : ceux dépendant de l'utilisation des données comme LRU (Least Recently Used), LFU (Least Frequently Used), JW-LFU (Jumping windowed LFU),

SW-LFU (Sliding Windowed LFU), FIFO-LFU, etc. Et d'autres, indépendants de l'utilisation des données comme l'algorithme aléatoire et FIFO (First In First Out) (Podlipnig et Böszörmenyi, 2003).

Brièvement, LFU garde un compteur qui s'incrémente à chaque fois qu'un contenu est consulté. Le contenu à remplacer sera celui dont le compteur est le plus petit. L'algorithme JW-LFU garde une trace des fréquences d'accès de chaque vidéo. JW-LFU introduit une fenêtre adaptative pour observer les contenus dont la popularité décline. S'il y a un défaut de cache, JW-LFU élimine le contenu le moins fréquemment utilisé. Semblable à JW-LFU, SW-LFU remplace le contenu avec la plus grande fréquence de demandes utilisateurs. Cependant SW-LFU utilise une fenêtre glissante de dimension n qui contient les plus récents et fréquents objets. À chaque fois que la fenêtre glissante est déplacée vers l'avant les nombre de demandes pour les différents contenus sont mis à jour. L'algorithme LRU consiste à supprimer le contenu qui n'a pas été consulté depuis le plus longtemps.

L'algorithme FIFO-LFU fonctionne différemment des quatre dernières stratégies de mise en cache. Ils stockent les contenus selon l'âge et le nombre de vues. Sur un défaut de cache, le contenu le plus ancien est supprimé avec une probabilité p et le moins fréquemment demandé est éliminé avec une probabilité $1 - p$.

L'algorithme aléatoire est le plus simple. Le contenu évincé est choisi au hasard. Pour FIFO, les contenus sont effacés dans l'ordre où ils sont arrivés dans le cache.

Les serveurs cache dans un CDN gardent des copies de tous les contenus des fournisseurs de contenu. Cependant, la distribution des contenus demeure statique. Un expert se charge de sélectionner (manuellement) les contenus qu'il juge potentiellement populaire à un moment donné. Seul ces contenus seront, par la suite, chargés dans les serveurs cache du CDN (Stamos *et al.*, 2008). Toutefois, cette approche demande passablement de temps, est assez coûteuse et n'est pas

efficace car les popularités des contenus peuvent varier considérablement dans le temps (Stamos *et al.*, 2008). Ceci représente une limitation qui restreint un usage efficace de l'infrastructure du CDN. Dans le sens où à un moment donné un contenu qui perd sa popularité va se retrouver encore dans les serveurs cache du CDN alors que d'autres populaire (qui ont gagné de la popularité entre temps) devraient être chargé depuis les serveurs d'origines du fournisseur du contenu ce qui engendre un délais additionnel dans le temps de chargement.

1.1.5 Distribution de contenu sous évolution de la popularité

Dans cette section, nous considérons le problème de distribution de contenus dans un réseau CDN. Les utilisateurs peuvent consulter des contenus numériques publiés par un fournisseur en se connectant aux serveurs cache détenus par le fournisseur de services CDN. Si le contenu ne se trouve pas dans les serveurs cache au moment de la requête alors, l'utilisateur, sera redirigé vers le serveur d'origine qui détient le contenu. De même ce dernier sera redistribué à nouveau dans les serveurs cache (Buyya *et al.*, 2008).

Dans ce scénario, le goulot d'étranglement est souvent la connexion entre les serveurs cache et le serveur d'origine. Afin d'utiliser efficacement ce lien tout en déchargeant le serveur d'origine du fournisseur de contenu, il devient impératif de choisir le bon contenu à livrer aux utilisateurs. Plutôt que de considérer une distribution statique des contenus (Pathan et Buyya, 2008), nous souhaitons étudier ce problème en adoptant un modèle de prédiction de popularité qui varie dans le temps et qui prend en compte les interactions des utilisateurs avec les contenus.

1.1.5.1 Les types de contenus Web

Le contenu est le coeur de tout site Web. Il peut prendre plusieurs formes, par exemple s'il est textuel, un article ou un texte descriptif sur un produit spéci-

fique, etc. S'il est visuel, ça sera une photo, une image ou une vidéo crée par les utilisateurs ou en anglais UGV pour User-Generated Video. Ces derniers représentent à eux seuls maintenant quelques 90% de tout le trafic Internet. YouTube [1] est le service de partage de vidéos le plus répandu (et utilisé) sur Internet avec 100 transferts de vidéos par minutes et plus de 1 trillion de visites estimées par année [2], a été le centre de plusieurs études. On y trouve de tout, des vidéos de création artistique aux segments d'émissions télévisées et clip. Les utilisateurs peuvent indiquer s'ils aiment ou pas une certaine vidéo, publier des commentaires sur la vidéo ou la chaîne qui la détient ou même la partager avec ses amis sur YouTube ou sur d'autres réseaux sociaux. Certaines vidéos sont vues plusieurs millions de fois ce qui permet d'avoir une appréciation du niveau d'engagement des utilisateurs envers ces vidéos.

L'étude de l'évolution des popularités des vidéos sur YouTube est difficile au vu du nombre de vidéos qui devient plus conséquent. Sans mentionner les différentes fonctionnalités auxquelles les platesformes permettent d'accéder (recommandation, recherche interne ou encore le réseautage social en ligne) ainsi que les difficultés à disposer d'un échantillon représentatif de toutes ces vidéos(Borghol *et al.*, 2011).

La popularité des vidéos sur YouTube, communément exprimée par le nombre de vues pour une vidéo, peut être décrite par plusieurs distributions (fonctions mathématiques) selon les caractéristiques de l'ensemble des données et la méthode d'ajustement utilisées. Dans (Cha *et al.*, 2009) et (Avramova *et al.*, 2009), la popularité des vidéos y est décrite par une distribution exponentielle alors que

---

1. `http://www.youtube.com`

2. Hernandez, B. A. (2011). YouTube in 2011 : How Its Busy Year Affects You. Consulté le 21 Avril 2016, dans `http://mashable.com/2011/12/31/youtube-in-2011`

dans (Cheng *et al.*, 2008) montre que cette dernière suit une distribution de Weibull. Pas très loin, (Borghol *et al.*, 2011) prouve que la popularité peut suivre une distribution Log-normal. Tout autrement, une distribution Gamma décrit la popularité sur YouTube selon (Cheng *et al.*, 2013).

Des analyses plus détaillées des popularités des vidéos montrent des schémas encore plus complexes et divers. Par exemple (Crane et Sornette, 2008) ont constaté que, tandis que l'activité autour de la plupart des vidéos YouTube peut être décrite par un processus poissonnien, de nombreuses vidéos montrent un comportement semblable autour des zones pic qui peuvent être décrites avec précision selon trois modèles d'évolution de popularité. Des modèles similaires ont été observés par (Figueiredo, 2013) et d'autres formes (distributions) ont été rapportées par (Gürsun *et al.*, 2011).

La popularité des vidéos a été examinée, en plus de YouTube, sur d'autres platesformes de vidéos en ligne (par exemple, Daum (Cha *et al.*, 2009), Dutch TV (Avramova *et al.*, 2009), DailyMotion (Carlinet *et al.*, 2012; Mitra *et al.*, 2011), Yahoo! video (Mitra *et al.*, 2011), Veoh (Mitra *et al.*, 2011), Metacafe (Mitra *et al.*, 2011), Vimeo (Ahmed *et al.*, 2013)) mais sur une plus petite échelle et aucune différence significative pour ce qui est des fonctions de popularité n'a été observée dans ces travaux.

1.1.5.2 Qu'est ce qui fait qu'un contenu Web devient populaire ?

Il n'y a pas de formule magique pour découvrir ce qui rend un contenu Web plus populaire qu'un autre. Seulement il reste possible d'identifier les éléments qui pourraient y contribuer. Les métadonnées pour un contenu web (par exemple le sujet, la qualité ou encore les commentaires des utilisateurs pour les UGVs) peuvent contribuer de manière considérable à cette popularité. Identifier les facteurs qui influenceraient la popularité d'un contenu Web est donc important dans

la construction des modèles de prédiction qui sont représentatifs de toutes les variables influant la dynamique des popularités des contenus en ligne (Tatar *et al.*, 2014).

L'émotion humaine est l'un des facteurs les plus importants qui influencent l'audience sur Internet. Les vidéos qui accompagnent des moments importants de la vie de chacun et qui suscitent des émotions intenses ont plus de chance d'être partagées sur ces plateformes (Guadagno *et al.*, 2013). De même, les contenus qui provoquent des troubles physiques et émotionnels (comme de la peur ou l'anxiété) chez les utilisateurs sont largement diffusés sur Internet et suscitent l'intérêt d'une large partie d'utilisateurs (Berger et Milkman, 2012; Berger, 2011). La qualité du contenu (Salganik *et al.*, 2006) et sa pertinence (Ratkiewicz *et al.*, 2010) sont également très fortement corrélées avec sa cote de popularité. D'un autre coté, il y a des variables qui ont pour effet de désavantager la popularité d'un contenu. Une de ces variables est l'existence de plusieurs versions du même contenu qui tendrait à limiter la popularité individuelle de chacune des copies (Cha *et al.*, 2009). Les filtres de contenu ont également une forte incidence sur la popularité des contenus web (Borghol *et al.*, 2012). Les services Internet les plus courants sur le web tel que les outils de recherche, les systèmes de recommandation et les sites de réseaux sociaux permettent d'étendre la visibilité du contenu et d'augmenter sa popularité. Par exemple, le moteur de recherche interne au site de YouTube compte pour la majeure partie des vues sur le site et donc joue un rôle important sur la façon dont une vidéo deviendrait probablement très populaire auprès des utilisateurs. Un autre exemple montre que les UGVs figurant dans les *related list* des vidéos populaires ont tendance à avoir plus de visites (Davidson *et al.*, 2010; Chatzopoulou *et al.*, 2010). De même, plus la vidéo est haut placée dans la liste plus elle recevra de vues (Zhou *et al.*, 2010). Un système de recommandation fait le lien entre les contenus similaires qui s'influencent entre eux en termes de

popularité(Zhou *et al.*, 2011).

Les réseaux sociaux permettent aussi de partager du contenu et servent de catalyseur, attirant ainsi un grand public sur Internet. Diffuser des vidéos sur les réseaux sociaux comme Facebook[3] ou Twitter[4] engendre un surcoût d'attention sur de courtes durées de temps (Broxton *et al.*, 2013). De même les relations sociales que les utilisateurs entretiennent sur ces sites expliqueraient comment un contenu deviendrait populaire. Par exemple, il a été observé que dès le début de la publication d'un contenu, plus étendue est le cercle sociale de l'éditeur du contenu, plus grande sera la montée de la popularité du contenu.

1.1.5.3 La régression pour la prédiction des popularités des contenus Web

Les méthodes de régression sont très fréquemment utilisées pour prédire la popularité des contenus en ligne (Tatar *et al.*, 2014). Pour l'essentiel, ces méthodes tentent de modéliser des données continues par une fonction, en minimisant le taux d'erreur, afin de prédire des valeurs futures.

(Tatar *et al.*, 2011) ont appliqué une régression linéaire simple en utilisant les premiers commentaires afin de prévoir le nombre final de commentaires pour les nouveaux articles publiés. Les auteurs font observer qu'il n'y a aucune amélioration sensible en utilisant l'heure de publication et la catégorie des articles. Par ailleurs, (Marujo *et al.*, 2013) ont étudié la prédiction de nombre de clics que les articles de *blog* enregistrent durant une heure. Quatre méthodes de régression ont été mises à l'essai. Ces méthodes sont respectivement la Régression linéaire multiple, les *regression-based trees* et le modèle additif de régression. Les auteurs montrent, qu'en combinant différents algorithmes de régression, il est possible d'obtenir de

---

3. `https://www.facebook.com`

4. `https://twitter.com`

bons résultats dans la prévision du nombre de clics accueillis par les nouveaux articles publiés durant une heure. Similairement, (Kim *et al.*, 2011) ont utilisé la régression linéaire, sur une échelle logarithmique, afin de prédire les classes de popularité des billets de *blogs* politiques. Les auteurs montrent qu'en examinant le nombre réel de vues lors des 30 premières minutes, on pourrait classer les articles en trois principales classes de popularité avec un pourcentage de précision de l'ordre de 86%. Une approche différente a été proposée par (Tatar *et al.*, 2012). Les auteurs étudient les performances de trois méthodes de régression, à savoir la régression linéaire simple, la régression log-linéaire et la régression à *constant scaling*, afin de publier les articles sur la base de leur nombre future de commentaires. À partir de données d'articles et de commentaires, les auteurs indiquent que la régression linéaire simple donne de meilleurs résultats pour la prédiction.

Ces méthodes de prédiction sont exécutées hors ligne sur un ensemble fixe de données. En tant que telles, elles ne sont pas adéquates pour prédire la popularité des vidéos. En outre, la demande des vidéos fluctue considérablement au fil du temps (Szabo et Huberman, 2010). Par conséquent, il est impératif pour un modèle de prédiction de s'adapter à cette dynamique. Par ailleurs, toute modification des données doit déclencher la mise à jour du modèle de prédiction afin de suivre le nouvel état de la popularité des vidéos. La toute nouvelle prédiction doit refléter une mise en balance de l'historique ainsi que l'état actuel des données.

1.1.6 La problématique de performance face à l'explosion des contenus vidéo

Les contenus de type vidéo font partie des ressources Web les plus problématiques pour un CDN. En 2015, les données vidéo représente la moitié du trafic Internet. En 2016, 1.2 million de minutes de vidéos, soit l'équivalent de 833 jours

de visionnage, circulent sur Internet chaque seconde[5]. Les sites de partage de vidéos et les réseaux sociaux, toujours plus nombreux, ne sont pas les seuls facteurs auxquels l'augmentation du volume du trafic vidéo peut être imputée. Les téléphones intelligents ou *Smatphone* deviennent eux aussi plus abordable financièrement et la qualité des réseaux à travers le monde s'améliore. En outre ; les résolutions plus élevées et la meilleure qualité d'image démocratisent l'utilisation des vidéos sur les sites web.

Se plaignant de l'explosion du trafic vidéo qui ont des difficultés à monétiser, les fournisseurs de contenu sont nombreux à louer les services des CDNs qui leur permettraient de répliquer leurs vidéos au plus près des utilisateurs et ainsi leur éviter la congestion du réseau. Le but étant de permettre à l'utilisateur final de regarder une vidéo sans que l'image se fige et sans arrêt.

Dans un CDN les serveurs cache ont une capacité de stockage limitée. Ainsi, seule une partie des vidéos sera accessibles aux utilisateurs depuis ces serveurs. Par conséquent, il est essentiel de déterminer les vidéos qui sont les plus avantageuses, c'est-à-dire les plus populaires à distribuer dans les différents serveurs. Un certain nombre d'études ont porté sur les stratégies de mise en cache des vidéos. (Cha *et al.*, 2009) présentent les conclusions d'une vaste étude de traces de vidéos recueillies sur plusieurs jours à travers deux systèmes de partage de vidéos, YouTube et Daum. Ils ont analysé certaines caractéristiques des vidéos ainsi que l'évolution de leur popularité dans le temps tout en considérant la faisabilité du *caching* versus une approche pair-à-pair pour améliorer la distribution des vidéos.

Prédire la popularité des contenus produits par les utilisateurs eux-mêmes, en particulier les vidéos, dépend du comportement des utilisateurs sur Internet. (Zink *et al.*, 2009) ont analysé les caractéristiques spécifiques au trafic de YouTube

5. Cisco Visual Networking Index (VNI) pour la période 2011-2016

pour en déduire les tendances d'évolution des popularités des vidéos, du *watch time*, des taille des vidéos, du *playback*, du *bitrate*, des requêtes des utilisateurs finaux et enfin du volume du trafic. Par ailleurs, ils ont discuté l'utilisation de ces données statistiques pour la mis en cache des vidéos. De manière similaire, (Abhari et Soraya, 2010) ont examiné la dynamique des popularités sur YouTube et celle des demandes des utilisateurs finaux dans une large quantité de données recueillies par *crawling*, c'est a dire par collecte automatisée de sites Web. Ils ont développé un générateur de données vidéos qui peut être utilisé aux fins d'analyse comparative pour les stratégies de mise en cache. D'autre part, (Zhou *et al.*, 2015) ont proposé un algorithme de mise en cache pour les vidéos. Les auteurs montrent que la popularité des vidéos évolue au cours du temps pour différents types de vidéos. De plus, (Borghol *et al.*, 2011) indiquent que les popularités des plus veilles vidéos peuvent refléter (impacteraient) leurs popularité dans un avenir immédiat. Toutefois, ceci n'est pas vrai pour les vidéos qui sont plus récentes.

Notre démarche contraste radicalement avec ce qui est proposé jusqu'ici dans la littérature. Nous, nous mettons l'accent sur les interactions des utilisateurs finaux avec les vidéos afin d'aider à prédire leur popularité dans le temps. Le but étant d'optimiser la distribution des vidéos dans un CDN en prédisant celles qui sont les plus demandées pour les placer au plus près des utilisateurs.

1.2 Recourir à l'analyse de données : une nécessité aujourd'hui ?

L'analyse de données est une famille de méthodes statistiques et structurelles. Ces méthodes peuvent être vues en fonction de l'objectif fixé. Usuellement, les méthodes statistiques sont employées soit pour explorer les données (nommée statistique exploratoire) ou soit pour prédire un comportement (nommée statistique prédictive). Les méthodes d'analyse de données ont commencé à être développées dans les années 50 poussées par le développement de l'informatique et du stockage

qui depuis n'a cessé de croître (Ott et Longnecker, 2015).

Aujourd'hui, les méthodes d'analyse de données sont adoptées dans beaucoup de domaines. Par exemple, l'analyse de données est répandue en marketing pour la gestion des relations avec la clientèle. De fait, les éditeurs de solutions CRM ou *Customer Relationship Management*, en français Gestion des Relations avec les Clients (GRC) proposent de plus en plus d'outils d'analyse de données marketing dans leurs offres logicielles. Le but étant de personnaliser les messages transmis aux consommateurs.

En général, on peut avoir recours à ces méthodes dans tous les domaines qui gèrent et/ou génèrent de grandes quantités de données de type varié. Les CDNs ne devraient pas échapper à cette tendance. Les routeurs et serveurs cache, prévus pour proposer du contenu, produisent un certain nombre de journaux de traces, plus communément appelé *logs* dans lesquels sont consignés les traces des activités de tous les utilisateurs transitant par le CDN. Pour optimiser l'utilisation des ressources dans CDN, notamment avec l'explosion d'utilisation des vidéos, nous serons tentés d'examiner et d'explorer ces *logs*.

1.2.1 Usage de l'analyse de données dans CDN

Les fournisseurs de services CDN sont des prestataires intermédiaires et ne sont pas autorisés à utiliser les traces de *logs* des utilisateurs transitant par leurs réseaux pour leur usage en interne[6](Wang *et al.*, 2015). Cependant, la plupart des CDNs fournissent des outils statistiques aux fournisseurs de contenu. Ces derniers, utilisent les données récoltées par les serveurs de relais pour établir des données statistiques comme le nombre de visiteurs uniques par jour ou encore le pourcen-

---

6. `https://www.akamai.com/us/en/privacy-policies/privacy-statement-for-akamai-sites.jsp`

tage de trafic généré par chaque contenu. Ces statistiques apportent un meilleur aperçu des interactions avec les sites Web (les fournisseurs de contenu) et de leurs visiteurs.

Par exemple, *Amazon CloudFront Rapports & Analytics*[7] offre des rapports détaillés de données statistique qui comprennent le nombre total des requêtes, le pourcentage des nouvelles requêtes, le total des données transférées en octets ainsi que la liste des contenus les plus populaires offerts par la distribution Amazon CloudFront.

Les analyses vidéo d'Akamai [8] donnent aux fournisseurs de contenu une meilleure compréhension de la manière dont leurs visiteurs consomment divers types de médias à différents moments de la journée. Les statistiques en question incluent des mesures de l'audience (les téléspectateurs, visites, visionnage, durée et plus), de la qualité de service (la disponibilité de la vidéo et plus) et des consommateurs des vidéos (profil de l'utilisateur, historique des visites, et plus).

L'offre "analytique" de Limelight [9] propose aux fournisseurs de contenus, de multiples rapports sur les habitudes de transmission des données, les informations géographiques, le stockage et l'utilisation des contenus pour les aider dans une prise de décision sur leurs sites Web basée sur des données précises et fiables.

Le constat est qu'aujourd'hui les fournisseurs de services CDN ne tirent pas pleinement profit de ces ces statistiques (nombre de partage d'une vidéo, le nombre de vues d'une vidéo, etc pour leur usage interne. Cependant, on constate que de plus en plus de fournisseurs de contenu sont en train de construire leur propre

---

7. `http://ww.aws.amazon.com/cloudfront`

8. http ://www.akamai.com

9. `http://www.limelight.com`

réseau de distribution de contenu. Par exemple, Youtube désormais possède son propre CDN, Google Cloud CDN [10], pour la diffusion de vidéo. Netflix fournit également des données en streaming en utilisant son propre CDN (Wang *et al.*, 2015). Cette nouvelle tendance ouvrirait de nouvelles possibilités en matière de gestion et de remplacement de contenu. Dans ce cas, les règles de confidentialité auxquelles sont soumis les fournisseurs CDN (qui eux même deviennent fournisseur du contenu) n'ont plus lieu d'être. Désormais, un CDN peut intégrer les *logs* existants des utilisateurs pour augmenter son taux de succès de cache. Rares sont les travaux de recherche qui discutent cette approche. Récemment (Wang *et al.*, 2015) ont introduit un CPCDN, qui fait référence à un CDN alimenté par l'intelligence au niveau du fournisseur de contenu (puisque les informations de l'utilisateur sont détenues par les fournisseurs de contenu). Ils présentent, en outre, les bénéfices obtenus en utilisant les informations des utilisateurs dans la détermination des contenus à reproduire à proximité des utilisateurs intéressés par ces contenus.

1.2.2 Vers une architecture "analytique" basée sur le *Cloud Computing*

L'analyse de données permet de créer une nouvelle manière d'optimiser la réplication (distribution) des vidéos dans un CDN. Cependant, elle nécessite un certain niveau d'expertise, ainsi que de non négligeable capacité de traitement et de stockages des traces de *logs*. Le *Cloud Computing*, ou l'informatique en nuage est une solution pour faciliter l'implémentation et l'exécution des applications d'analyse de données. Par *Cloud Computing*, on entend un modèle dans lequel les ressources sont fournies comme un service sur Internet au lieu d'être détenues et gérées par les prestataires de services eux-mêmes (Armbrust *et al.*, 2010).

---

10. Google se lance discrètement dans le CDN. (2015). Consulté le 25 Avril 2016, dans `http://www.silicon.fr/google-se-lance-dans-le-cdn-133772.html`

En effet, *le Cloud Computing* donne la possibilité aux CDNs, qui n'auraient pas forcément les ressources nécessaires ni les moyens financiers, de mettre en place une architecture complète d'analyse de données. Pour ces CDNs, le modèle en *Cloud* leur permettrait de lancer et tester plus facilement leurs nouvelles applications (d'analyse de données) sans se soucier de la plateforme ni de l'infrastructure. Selon une étude récente réalisée par O'Reilly(Barlow, 2014), 40% des participants du secteur de l'analyse de données utiliseraient aujourd'hui une solution *Cloud*. Parmi le reste, une grande majorité penserait à migrer leurs produits vers le *Cloud Computing*.

1.2.2.1 Les services du *Cloud Computing*

Le *Cloud Computing* peut être subdivisé en 3 modèles de services (voir figure 1.2) (Zhang *et al.*, 2010) :

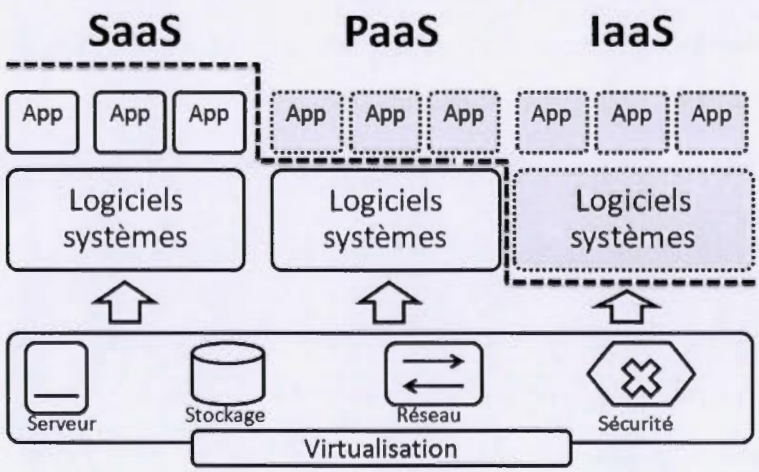

Figure 1.2 Les services du *Cloud Computing*.

— *Software as a Service (SaaS)* ou en français logiciel à la demande. Un SaaS est accessible à toutes les entreprises et est facturé en nombre d'utilisateurs. Une entreprise loue des applications auprès du fournisseur de services. Ces

applications sont accessibles via plusieurs interfaces. On retrouve de nombreuses applications accessibles en mode SaaS depuis le Cloud. L'e-mail est sans doute l'application la plus utilisée en mode SaaS. Il y a aussi les logiciels de gestion de la relation client (CRM) et les progiciels de gestion intégrée (ERP). Les outils de collaboration comme le partage de documents s'utilisent aussi en mode SaaS.

— *Platform as a Service (PaaS)* est un environnement qui permet à l'entreprise de déployer ses propres applications en dehors de son périmètre. L'entreprise loue l'environnement (les logiciels systèmes) et est facturée à la consommation. L'utilisateur du PaaS a le contrôle seulement sur les applications qu'il déploie et peut configurer l'environnement applicatif. En revanche, il n'a pas le contrôle sur l'infrastructure sous-jacente.

— *Infrastructure as a Service (IaaS)*, en français Puissance de calcul et de stockage à la demande. Un IaaS met à disposition des ressources des machines virtuelles facilement configurables et hautement disponibles. Dans le cas d'un IaaS, l'entreprise loue les capacités de traitement, de stockage et de mise en réseau qu'elle peut gérer de façon autonome côté logiciel.

1.2.2.2 L'analyse de données en tant que service (AaaS)

L'AaaS est un logiciel complet d'analyse de données, configurable et extensible en fonction des besoins de chaque CDN. Un AaaS est fourni comme un service dans le *Cloud* où plusieurs outils d'analyse de données sont déjà disponibles et peuvent être configurés pour traiter et analyser d'énormes quantités de données comme c'est le cas des traces de *logs* recueillies dans un CDN (Talia, 2013). De manière plus précise, les CDNs fournissent ces *logs* au logiciel AaaS et récupèrent en retour des analyses spéciales qui permettent de comprendre des questions complexes qui améliorent la qualité du service. Ces analyses spéciales sont produites par des applications d'analyse de données ou "analytique" (Raja-

gopalan et Krovi, 2002)(Chattamvelli, 2011). Ces dernières, traitent les flux des données et y exécutent des algorithmes complexes d'apprentissage-machine pour en tirer des connaissances approfondies.

Dans ce sens les AaaS offrent un fort potentiel pour optimiser la diffusion de contenu dans les CDNs. En effet, l'analyse des données des utilisateurs qui utilisent le CDN reflète le niveau de popularité des contenus consultés. Seulement l'analyse de ces données nécessiterait l'ajout de nouveaux éléments (matériels et logiciels) dans le réseau du CDN (Talia, 2013).

En exploitant les AaaS (Analytics as-a-service), les CDNs bénéficieront de plateforme spécialisée et distribuée pour la collecte des données venant de différents serveurs caches et d'en faire une analyse fine pour notre cas prédire les vidéos les plus populaires pour les charger au plus près des utilisateurs.

À l'heure où on réclame des débits toujours plus importants et une garantie sur le qualité d'accès aux données, bâtir un système d'analyse de données en interne du réseau du CDN n'est pas un choix judicieux (Chen *et al.*, 2011). Il est plutôt préférable de se tourner vers une architecture spécialisée dans le Cloud qui garantit à moindre coût un traitement et un stockage optimisé des données en cluster via Hadoop ou Spark. Cette approche apporte non seulement des performances en temps de l'installation de la plateforme (PaaS) et du matériel (IaaS) mais aussi une scalabilité pour faire face à l'augmentation des données et leur réutilisation. En effet les AaaS savent répondre aux besoins supplémentaires (en stockage et en traitement) sans coût supplémentaires extravagant.

L'objectif étant pour un CDN d'avoir de la capacité en infinie, une disponibilité non-stop et une élasticité, afin de déployer des nouvelles applications (AaaS) dans les délais les plus brefs pour une meilleure distribution de leurs contenus et garantir alors une utilisation efficace de leur réseau (stockage au niveaux de serveurs cache).

De plus, un AaaS serait pré-installée sur les serveurs du fournisseur Cloud donc garantit un temps de déploiement très faible. Le CDN n'a pas à se soucier des mises à jour qui seront réalisés par le fournisseur du Cloud. La fourniture de capacité d'analyse de données suffisantes à travers des applications spécialisées d'analyse de données en tant que service accélérerait son adoption par beaucoup de CDNs. En plus des économies importantes en termes de coûts, l'autre avantage principal de l'AaaS est d'aider les CDNs à créer de nouveaux types d'avantages concurrentiels pour leurs clients et notamment dans la gestion de leurs contenus.

CHAPITRE II

PROPOSITION DES ÉLÉMENTS THÉORIQUES D'UNE ARCHITECTURE D'ANALYSE DE DONNÉES EN TANT QUE SERVICE POUR LA MISE EN CACHE DES VIDÉOS DANS CDN

La prédiction des vidéos est fortement influencée par les interactions des utilisateurs avec ces vidéos. Dans le chapitre précédent, nous avons argumenté notre choix d'utiliser un modèle de service "analytique" dans le Cloud. En effet avec le AaaS (Analytics as-a service), le CDN n'a plus besoin de disposer de locaux pour entreposer ces serveurs. L'installation et la maintenance de la plateforme "analytique" sont gérées par le fournisseur du Cloud.

Nous présentons, dans ce chapitre, les éléments théoriques d'une architecture pour gérer les traces de *logs* acheminées vers le AaaS pour analyse. Cette architecture ne constitue en soi qu'une simple proposition pour supporter l'usage de l'analyse des données pour la gestion de cache dans un CDN. Les éléments d'une telle architecture peuvent être divisés en deux parties : les éléments du domaine CDN. Ces éléments assurent le service de diffusion des vidéos entre les différents serveurs cache en bordure du réseau. Puis les éléments du domaine du Cloud qui sont chargés de la gestion des données vidéo collectées depuis les serveurs caches et nécessaires pour l'analyse. Avant de décrire les éléments de l'architecture, nous présenterons d'abord le modèle d'affaires et les relations entre les différents éléments du système.

2.1 Modèle d'affaires

Notre modèle d'affaires comprend les acteurs suivants : les utilisateurs finaux, le fournisseur de contenu, le fournisseur CDN et le fournisseur *Cloud Computing*(voir figure 2.1).

L'utilisateur final est l'entité qui consomme la vidéo (le contenu). Le fournisseur de contenu est l'entité qui héberge les vidéos. Ces vidéos sont accessibles aux utilisateurs finaux via le réseau du CDN. Le fournisseur CDN possède les serveurs de relais et fournit aux utilisateurs finaux un accès fiable et rapide aux vidéos. Le fournisseur *Cloud* offre le service AaaS pour le déploiement et la gestion d'applications "analytiques" comme par exemple des applications de prédiction de popularité ou de corrélation entre vidéos. Le fournisseur CDN loue simplement le service AaaS offert par le fournisseur de *Cloud* en utilisant un modèle tarifaire à la carte. Comme dans tout modèle d'affaires, le même acteur peut jouer plusieurs rôles. Le CDN peut à la fois fournir le service de gestion de vidéos et le service *Cloud*.

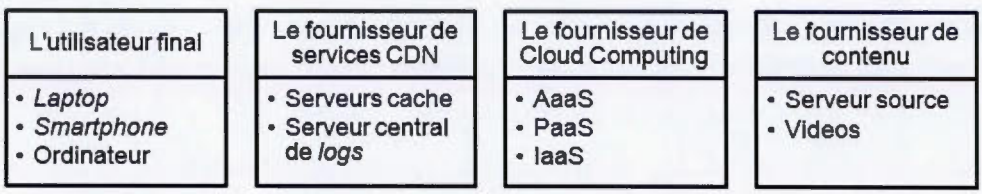

Figure 2.1 Les acteurs du modèle d'affaires.

2.2 Vue d'ensemble

Nous proposons un modèle de prestation de services "analytique" dans le *Cloud*, AaaS. Ce modèle permet aux fournisseurs de services CDN de construire et de consommer des applications d'analyse de données selon leur utilisation sans frais généraux de maintien de l'architecture ni de la plateforme.

Une vue d'ensemble de l'architecture est donnée dans la figure 2.2. Notre architecture se compose des utilisateurs finaux, du fournisseur du service CDN et du fournisseur du Cloud.

Les fournisseurs de services CDN peuvent utiliser le modèle dans le Cloud pour

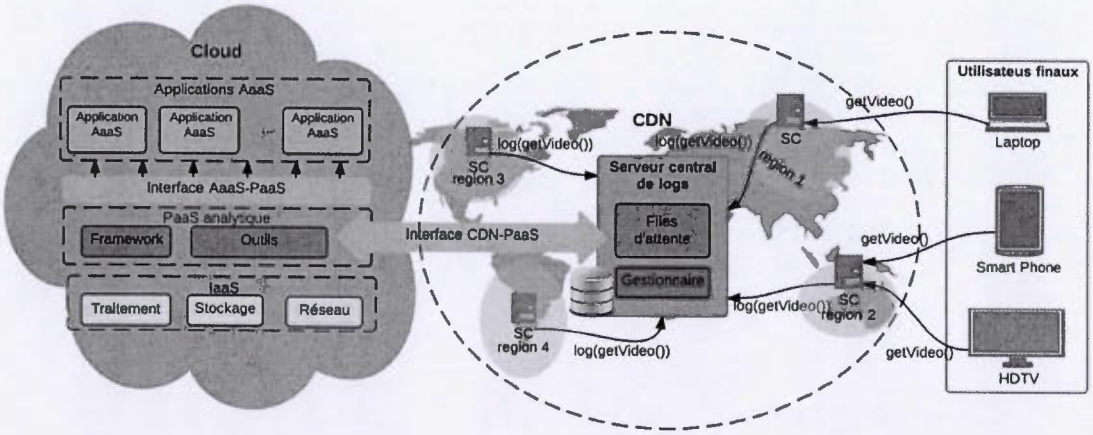

Figure 2.2 Vue d'ensemble.

déployer des applications "analytiques" à la demande. Le CDN représente le client du Cloud et dispose de ressources sur l'infrastructure de l'hébergeur. Chaque fournisseur de services CDN dispose de ses propres administrateurs, applications, scripts et outils. L'hébergeur Cloud fournit aux fournisseurs de services CDN l'environnement qui englobe tous les outils et *Frameworks* d'analyse de données. Les utilisateurs finaux envoient des demandes dans le domaine du fournisseur de services CDN pour aller chercher, regarder et interagir avec des vidéos. Le domaine du fournisseur de services CDN comprend un ensemble de serveurs cache déployés dans différentes régions géographiques et communiquent avec un serveur central de *logs*. L'ensemble interagit avec le domaine du Cloud au travers d'interfaces appropriées. Dans le domaine du Cloud, les applications analytiques sont déployées par-dessus d'un PaaS étendu (spatialisé) et d'un IaaS.

Plus loin, nous détaillons les éléments de chaque domaine ainsi que les interfaces entre le fournisseur CDN et les modules déployés dans le Cloud.

2.3 Le fournisseur de service CDN

Un CDN diffuse les vidéos aux utilisateurs par l'intermédiaire des serveurs cache. Les serveurs cache sont reparties dans une vaste zone géographique, permettant aux utilisateurs de regarder des vidéos sans que l'image se fige et sans arrêt.

2.3.1 Les serveurs cache

Les utilisateurs finaux sont servis avec les vidéos à partir du serveur cache le plus proche. Chaque fois qu'un utilisateur communique avec un serveur cache, ce dernier conserve cette information en la stockant sous un format défini (une entrée de *log*) en local. Afin de traiter toutes ces données (traces de *logs*), il est obligatoire de les centraliser dans un serveur dédié. Ce qui implique la nécessité de collecter ces *logs*. Pour cela il est possible d'utiliser des protocoles comme SYSLOG (Gerhards, 2009) pour relayer ces *logs* vers un serveur dédié.

2.3.2 Serveur central de *logs*

Le serveur central de *logs* est conçu pour stocker les traces de *logs* reçues depuis tous les serveurs cache. On se base sur un modèle de messagerie distribuée, en mode *publish-subscribe*. Le serveur central de *logs* conserve les *logs* qu'il reçoit dans des files d'attente. Chaque file contient une séquence de *logs*, ordonnée commençant du plus ancien au plus récent. Les serveurs cache publient les *logs* directement dans les files. L'application AaaS est l'entité qui va consommer les *logs*. Plus précisément le serveur cache se connecte d'abord au serveur central et collecte les métadonnées sur les files disponibles (allouées). Cela permettrait au serveur cache de mettre directement les entrées de *logs* dans les files allouées à tour de rôle (*round-robin*) de manière à équilibrer automatiquement les charges qui pèsent sur le serveur.

Le nombre de files à déployer au niveau du serveur est configuré par le gestion-

naire de *logs*. Ce dernier est aussi responsable du maintien des files. Chaque file est continuellement étendue avec les entrées de *logs* crées au niveau des serveurs cache. Le gestionnaire de *logs* conserve toutes les entrées de *logs* publiées pendant une période de temps configurable. Par exemple, si on décide de conserver toutes les entrées pendant une journée alors ces *logs* seront disponibles 24 heures après leurs publications, après quoi ces entrées seront retirées des files d'attente afin de libérer de l'espace sur le disque du serveur central de *logs*.

2.4 Le fournisseur du Cloud

Dans le domaine du Cloud, nous utilisons un IaaS pour nous garantir les capacités nécessaires en traitement, en stockage et en mise réseau. Par-dessus l'IaaS nous réutiliserons tout PaaS existant qui supporte les *Frameworks* et outils requis par les applications AaaS.

2.4.1 IaaS

Le niveau IaaS recouvre l'ensemble des composants matériels et logiciels qui apportent les éléments essentiels du Cloud. L'IaaS apporte aux niveaux PaaS et AaaS le matériel serveur, les systèmes d'exploitation, le stockage et le réseau. Plus précisément, l'IaaS permet aux CDNs de louer l'infrastructure pour héberger leurs propres applications AaaS.

2.4.2 PaaS "analytique"

Le PaaS vient à un niveau plus haut. Le PaaS dispose des ressources et des logiciels nécessaires à l'hébergement et au déploiement des applications de l'AaaS.

Dans le cadre de ce mémoire, il est nécessaire d'étendre un PaaS existant (à faire) afin que premièrement, il supporte de nouvelles capacités d'analyse (par exemple acquisition des *logs* et les algorithmes de *Data mining*) et deuxièmement,

il expose ses capacités de calcul vers l'AaaS à travers une interface appropriée. L'AaaS s'exécute par-dessus ce PaaS.

Nous décrivons ci-après les différents éléments fonctionnels d'un PaaS "analytique" comme le montre la figure 2.3.

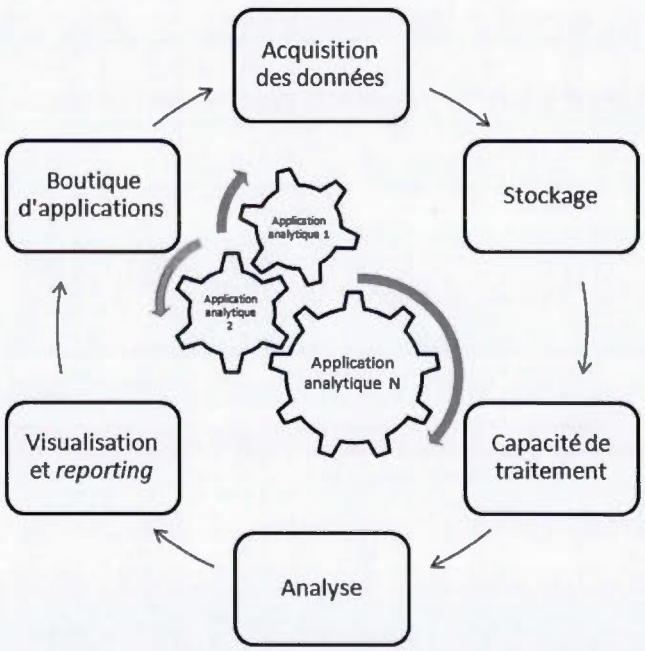

Figure 2.3 Les éléments fonctionnels d'un PaaS spécialisé pour l'analyse de données.

1. *L'acquisition des données* représente l'interface entre le Cloud et le CDN (client du Cloud). Cette interface permet de recueillir automatiquement les *logs* (activités des utilisateurs) provenant des serveurs caches afin de les analyser.
2. *Le stockage dans un entrepôt de données (Data Warehouse)* (Kimball et Ross, 2011). Il s'agit de base de données dédiée au stockage des données utilisées dans le cadre de l'analyse. Un entrepôt de données est alimenté en données depuis le client du Cloud, dans notre cas le CDN.

3. *La capacité de traitement.* Afin de gérer et traiter un plus gros volume de données, il est essentiel de distribuer les traitements sur plusieurs ordinateurs de manière à ce que différents algorithmes d'analyse de données peuvent être exécutés en parallèle. Aujourd'hui, la solution la plus populaire pour ce qui est de l'analytique à grande échelle est Hadoop[1] (White, 2012). Hadoop se base sur le modèle de programmation *mapReduce* (implémentation des fonctions map et reduce). Les fonctions *map* et *reduce* sont utilisées pour implémenter des opérations de base sur les données comme le tri, le filtrage, la projection ou l'agrégation. À partir d'un couple clé/valeur, la fonction *map* retourne un ensemble de nouveaux couples clé/valeur. Cet ensemble peut être vide, d'une cardinalité 1 ou plusieurs. À partir des groupes de valeurs associées à une clé, la fonctions *reduce* retourne généralement une valeur ou aucune (Dean et Ghemawat, 2008). Les applications adoptant ce modèle sont automatiquement parallélisées et exécutées sur des clusters d'ordinateurs. Cependant d'autres alternatives encore plus rapides comme Spark[2] et Storm[3] commencent à gagner du terrain.
4. *L'analyse de données* est la fonctionnalité coeur de la plateforme et regroupe une multitude d'algorithmes de *data mining* (Han *et al.*, 2011). En outre ces algorithmes permettent de traiter des problématiques de *Clustering*, de Classification et de régression etc.
5. *Les applications "analytiques"* répondent à un besoin spécifique. En particulier une application "analytique" permet d'appliquer un ou plusieurs algorithmes de *data mining*. Une application "analytique" opère donc sur des

---

1. `http://hadoop.apache.org`
2. `http://spark.apache.org`
3. `http://storm.apache.org`

données et permet d'en extraire des connaissances utiles.

6. La visualisation et *reporting*. Même si une grande partie des fonctionnalités de la plateforme est accessible depuis des interfaces de service Web, une solution plus complète intégrera une interface utilisateur facilitant l'accès à l'information. Des outils comme Penthao[4] ou Tableau[5] existent et permettent de fournir ceci.
7. *La boutique d'applications* fournit les mécanismes pour gérer les applications "analytiques" tout au long de leurs cycles de vie : développement, déploiement et suppression. Les administrateurs du Cloud gardent une vision centrale sur les applications analytiques disponibles et déployées et leurs versions.

2.4.3 AaaS

Une application AaaS collecte, traite et analyse les *logs*. La figure 2.4 illustre les étapes de traitement des *logs*. Ces étapes sont répétées chaque fois que l'application AaaS requiert de nouvelles données vidéo pour l'étape d'analyse.

Les étapes de traitement des *logs* de la figure 2.4 sont les suivants :

— Étape d'ingestion des *logs* : Cette étape consiste à acquérir les traces de *logs* à partir du serveur central de *logs*.

— Étape de filtrage des *logs* : Cette étape consiste à filtrer un sous-ensemble d'entrées de telle sorte que seules les entrées de *logs* pertinentes et correspondant à certains critères sont sélectionnées.

— Étape d'extraction des *logs* : Cette étape consiste à extraire des champs présélectionner en utilisant les expressions régulières ou le nom de certains

4. `http://www.pentaho.com/fr`

5. `http://www.tableau.com`

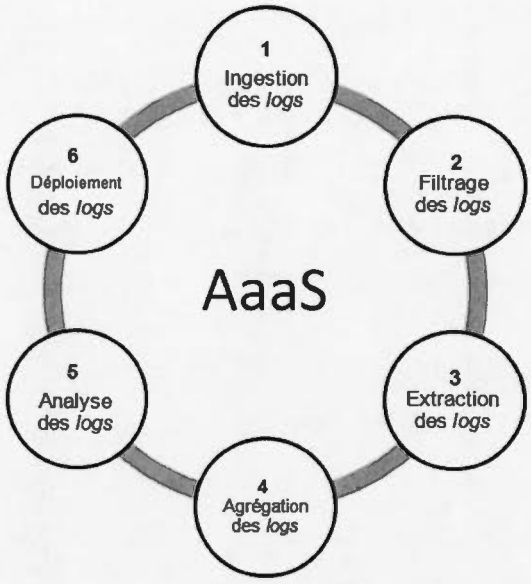

Figure 2.4 Les étapes de traitement des *logs* dans une application AaaS.

champs.

— Étape d'agrégation des *logs* : Cette étape consiste à agréger certaines données de façon à optimiser la collecte et le stockage des *logs*.

— Étape d'analyse des *logs* : Dans cette étape, nous nous intéressons à la sélection du modèle analytique (l'algorithme de *Data mining*) et aux meilleurs paramètres de configuration pour le modèle sélectionné.

— Étape de déploiement du modèle analytique : Cette étape consiste à exporter la sortie du modèle vers le fournisseur de services CDN.

2.5 Les interfaces

Ici nous proposons une première interface AaaS-PaaS pour les interactions entre le PaaS et l'application AaaS. Une deuxième interface PaaS-CDN régit les interactions (les échanges) entre le PaaS et le fournisseur de services CDN. Les spécifications de ces deux interfaces sont détaillées dans ce qui suit.

2.5.1 Interface AaaS-PaaS

L'interface AaaS-PaaS s'appuie sur l 'API REST, acronyme de Representational State Transfer (Fielding, 2000). Le REST est un style d'architecture qui repose sur le protocole HTTP. On accède à une ressource par son URI pour procéder à plusieurs opérations (GET, POST, PUT et DELETE).

L'interface AaaS-PaaS est conçue pour fournir une couche d'abstraction pour le PaaS "analytique" afin de gérer les applications AaaS d'une manière générique. Pour définir une nouvelle connexion entre le PaaS et l'application AaaS on peut simplement ajouter la propre spécification de l'application AaaS qu'on veut. Comme exemple d'applications AaaS, nous pouvons citer les applications de prédiction de popularité des vidéos ou encore les applications identifiant les corrélations entre vidéos.

2.5.1.1 La ressource : Application AaaS

L'interface AaaS-PaaS manipule principalement une seule ressource : Application AaaS. Une application AaaS réfère à tout logiciel ou programme "analytique" (qui manipule des données dont le but est d'en extraire des connaissances utiles) qui peut être déployé sur le PaaS. Le code source de ces applications devrait être fourni par le développeur dans un format spécifique (war, ear, zip, etc.). Pour déployer une application et l'exécuter via l'interface AaaS-PaaS, on doit suivre le scénario d'utilisation illustrée dans la figure 2.5.

2.5.1.2 Méthodes de gestion des ressources Applications AaaS

L'interface AaaS-PaaS prend en charge les méthodes HTTP suivantes pour la manipulation des ressources, Applications AaaS :

— La méthode POST (CREATE) crée une application AaaS

Figure 2.5 Scénario de déploiement d'une application AaaS.

— La méthode GET (READ) renvoie la représentation d'une application AaaS
— La méthode PUT (UPDATE) met à jour une application AaaS existante
— et la méthode DELETE supprime une application AaaS.

Le tableau 2.5.1.2 présente l'utilisation des méthodes HTTP sur les ressources, Applications AaaS par l'API REST.

Opération	**Méthode HTTP**	**Chemin d'accès**
Créer une application AaaS	POST	*/a3SApp*
Retourner une application AaaS	GET	*/a3SApp/{id}*
Mettre à jour une application AaaS	PUT	*/a3SApp/{id}/update*
Lancer une application AaaS	POST	*/a3SApp/{id}/start*
Arrêter une application AaaS	POST	*/a3SApp/{id}/stop*
Redémarrer une application AaaS	POST	*/a3SApp/{id}/restart*
Supprimer une application AaaS	DELETE	*/a3SApp/{id}*

Tableau 2.1 Spécification de l'interface AaaS-PaaS

2.5.2 Interface CDN-PaaS

L'interface CDN-PaaS est assez simple. Elle permet aux applications AaaS de communiquer directement avec le CDN. Techniquement l'interface de connexion CDN-PaaS est une adresse Internet qui combine l'adresse IP (celle de l'ordinateur

du serveur de distribution) et le numéro de port qu'on appel *socket*. Ces *sockets* assurent la communication entre le CDN et le PaaS, soit de manière fiable en utilisant la protocole TCP/IP, soit non fiable avec le protocole UDP.

Le serveur de distribution fonctionne sur une machine bien définie et liée à un numéro de port spécifique. Le serveur de distribution se met simplement à l'écoute d'un client (une application AaaS), qui demande une connexion. Une application AaaS connaît le nom de la machine sur laquelle le serveur de distribution est en exécution et le numéro de port sur lequel il écoute. L'application AaaS demandera alors une connexion au serveur en s'identifiant avec éventuellement une adresse IP ainsi que le numéro de port qui lui est lié. Une fois la connexion établie entre une application AaaS et le serveur de distribution, il sera alors possible d'envoyer la sortie de l'application vers le serveur de distribution.

CHAPITRE III

EXEMPLE D'UNE APPLICATION AaaS : VERS UNE RÉGRESSION HYBRIDE POUR PRÉDIRE LA POPULARITÉ DES VIDÉOS

Dans ce chapitre, nous présenterons un exemple d'une application AaaS : *une application AaaS pour prédire les futures popularités des vidéos.* La prédiction permet de comprendre ce qui rend une vidéo plus populaire qu'une autre, mais aussi d'observer et interpréter la dynamique de leurs popularités. En outre, l'application de prédiction de popularité permet de déterminer à l'avance les popularités des nouvelles vidéos mises en ligne. Ces popularités impactent la mise en cache des vidéos générées par les utilisateurs.

Dans le reste du chapitre, nous détaillerons d'abord la méthodologie ainsi que les étapes de prétraitement des *logs* pour le cas de notre application. Ensuite, nous introduirons le modèle de prédiction établie inspiré des modèles de régression linéaires. Finalement, nous discuterons le déploiement d'une telle application pour la mise en cache des vidéos dans un CDN.

3.1 Méthodologie

Évaluer la popularité des vidéos est plus complexe que simplement compter le nombre de vues des vidéos. Plusieurs paramètres sont à prendre en compte dans cette tâche. Il serait intéressant par exemple d'intégrer le *feedback* des utilisateurs après visionnages comme par exemple les commentaires ou les partages, etc. Ces retours d'expériences donnent une idée réaliste du niveau d'engagement des utilisateurs et fournissent des informations supplémentaires qui pourraient compléter le nombre de vues d'une vidéo.

En suivant le pipeline (les étapes) établi dans la sous-section 2.4.3, l'application de popularité peut être divisée en trois étapes différentes et interdépendantes : 1)- Étape de prétraitement des *logs*, 2)- Étape d'analyse de données ou encore modélisation des popularités et finalement 3)- l'Étape de déploiement du modèle de popularité. Dans la première étape, les *logs* sont nettoyés et transformés en un ensemble de statistiques représentant la dynamique des vidéos. Dans l'étape de modélisation, on se base sur les modèles de régression linéaires pour obtenir les valeurs cachées qui reflètent les popularités réelles des vidéos. Ces valeurs sont retournées au fournisseur de services CDN et peuvent être utilisées comme entrées pour d'autres applications telles que la mise en cache des vidéos.

3.2 Prétraitement des *logs*

L'étape de prétraitement des *logs* est importante et nécessaire avant d'appliquer un modèle de prédiction quelconque. Le but de cette étape est de récupérer un ensemble pertinent de paramètres qui traduisent les interactions des utilisateurs avec le contenu vidéo et donc leur réel niveau d'engagement. Cette étape comporte cinq sous étapes en partant de l'ingestion des *logs* jusqu'à leur agrégation (voir figure 3.1).

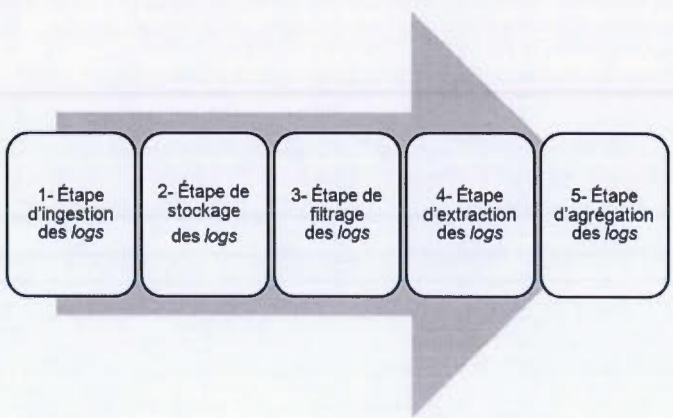

Figure 3.1 Les étapes de prétraitement des *logs*.

Le détail de ces étapes est donné ci-dessous.

3.2.1 Étapes d'ingestion et stockage des *logs*

3.2.1.1 Format des *logs*

Chaque demande d'affichage d'une vidéo de la part d'un utilisateur peut générer plusieurs requêtes. Des informations sur ces requêtes notamment le nom de la vidéo demandée et les réponses du serveur sont stockées dans les traces *logs* du serveur cache et relayées au serveur central de *logs* (voir détails dans la section 2.3.2).

Il existe plusieurs formats de *logs* mais dans le cas du CDN, rares sont les fournisseurs de services qui émettent publiquement leurs formats. Cloudfront[1] utilise un format proche du ECLF (Extended Common LogFile Format) (Hallam-Baker et Behlendorf, 1996). Ce format stocke plusieurs informations dont 1)-le nom ou l'adresse IP de la machine émettrice, 2)-l'heure et la date de la requête, 3)-la méthode utilisée dans la requête, 4)-le nom de la vidéo demandée, 5)-l'état (*status*) de la requête, 6)-la taille du fichier envoyé et 7)-le nom du serveur cache où la requête est survenue.

Voici une entrée typique de CloudFront à titre d'exemple :

```
2010-03-12 23:51:21 SEA4 192.0.2.222 play 3914 OK rtmp://
   shqshne4jdp4b6.cloudfront.net/cfx/st
   bfd8a98bee0840d9b871b7f6ade9908f key=value http://player.
   longtailvideo.com/player.swf http://www.longtailvideo.com/
   support/jw-player-setup-wizard?example=204 LNX%2010,0,32,18
   myvideo p=2&q=4 flv 1
```

Chaque partie de cette entrée de *log* est décrite dans ce qui suit :

1. **[2010-03-12 23 :51 :21]** Il s'agit de la date et l'heure à laquelle la requête a été reçue. Le format est le suivant : [ANNÉE-MOIS-JOUR HEURE :MI-

1. https://aws.amazon.com/fr/cloudfront

NUTES :SECONDES].

2. **[SEA4]** Le nom du serveur cache où la requête a été survenue. Chaque serveur cache est identifié par un code à trois lettres suivit d'un nombre arbitraire. Le code à trois lettres correspond généralement à celui de l'aéroport le plus proche du serveur cache. Ce code est attribué par l'IATA (Association internationale du transport aérien).
3. **[192.0.2.222]** Il s'agit de l'adresse IP de l'utilisateur (client distant). Elle n'indique pas nécessairement l'adresse IP de la machine devant laquelle l'utilisateur se trouve. Si un serveur *proxy* s'interpose entre le serveur cache et la machine de l'utilisateur alors cette adresse IP sera celle du serveur proxy et non celle de la machine à l'origine de la requête.
4. **[play]** Le type de l'événement survenu. Les événements sont *connect*, *disconnect*, *play*, *stop*, *pause*, *unpause*. Nous étendons les valeurs de ce champs pour inclure des événements de type *comment*, *share* ou *subscribe*.
5. **[3914]** La taille cumulée en octet de la vidéo retournée au client jusqu'au moment de l'événement. Si aucune donnée n'a été envoyé au client alors ce champ contiendra le symbole **"-"** indiquant l'absence de contenu.
6. **[OK]** C'est le code du statut que le serveur cache retourne à l'utilisateur. La valeur **"ok"** indique que la requête a fait l'objet d'une réponse positive.
7. **[rtmp ://shqshne4jdp4b6..]** La partie racine de l'URI, y compris l'application et l'instance de l'application.
8. **[bfd8a98bee084..]** Un identificateur utilisé pour différencier les clients. Cette valeur est unique pour chaque nouvelle connexion.
9. **[http ://player.longtailvideo..]** L'URL de la page à partir de laquelle le fichier SWF est lié.
10. **[http ://www.longtailvideo..]** Il s'agit du champ **Referer** de la requête.

Il spécifie la page web depuis laquelle le client a lancé sa requête. (C'est la page qui contient un lien vers la vidéo ou la vidéo).

11. **[LNX%2010,0,32,18]** Ce champ indique l'entête *User-Agent* de la requête et identifie le type de dispositif présentant la requête.
12. **[myvideo]** Le nom de la vidéo.
13. **[p=2&q=4]** La requête sous forme de chaînes de caractères.
14. **[flv]** Le type de la vidéo.
15. **[1]** L'identifiant du flux. Il est unique pour chaque nouvelle connexion.

3.2.1.2 Formalisation et stockage des *logs*

Ici on formalise le contenu des traces des *logs* dans un CDN.

Soit *Vidéos* l'ensemble $\{v_1, v_2, ..., v_{n_V}\}$ de toutes les URI des vidéos d'un site Web (le fournisseur de contenu) avec n_V est le nombre total des vidéos et $U = \{u_1, u_2, ..., u_{n_U}\}$, l'ensemble de tous les identificateurs des utilisateurs ayant accédé aux vidéos dans une période de temps donnée avec n_U est le nombre total des utilisateurs (Tanasa et Trousse, 2003). Nous pouvons alors formellement définir une entrée de log l comme suit :

$$l =< c, t, u, e, s, v > \tag{3.1}$$

où c représente le nom du serveur cache où la requête est saisie avec $server(l) = c$, t représente la date et l'heure de la requête avec $temps(l) = t$, $u \in U$ avec $user(l) = u$, e est l'évènement produit tel que $e \in$ {"play", "pause", "stop", "share", "subscribe", "comment", "rate"} et $event(l) = e$, s représente l'état de la requête tel que $stat(l) = s$ et $v \in$ *Vidéos* avec $video(l) = v$.

On définit l'ensemble de tous les lignes de logs *Logs* dans le cas d'un serveur cache tel que $Logs = \{l_1, l_2, ..., l_{n_{Logs}}\}$ avec n_{Logs} est la taille de *Logs*. Une fenêtre

w est définie par une date de début et une date de fin tel que $w = [t_0, t_1]$ avec $start(w) = t_0$ et $end(w) = t_1$. Ainsi la liste ordonnée de logs, $logs(w)$ s'écrit comme suit : $logs(w) = \{l \in Logs$ tel que $start(w) \leq temps(l) \leq end(w)\}$

Une fois le mécanisme d'ingestion mis en place, il faut stocker ces *logs*. Cela implique une variété de technologies et de systèmes comme HDFS (Borthakur, 2008), Amason S3 (Robinson, 2008), les bases de données SQL ou NoSQL tel ques HBase (George, 2011), Cassandra (Lacomme *et al.*, 2014) ou DynamoDB (Sivasubramanian, 2012).

3.2.2 Étape de filtrage des *logs*

Le filtrage des *logs* consiste à supprimer les requêtes qui ne font pas l'objet de l'analyse. De même que les requêtes provenant de robots Web.

Pour les CDNs et les sites Web populaires, la dimension des *logs* s'estime en Gigaoctet par heure. Par exemple, YouTube, le site de partage de vidéos le plus populaire à l'heure actuelle, reçoit plus de 4 billions de vues par jour[2]. Ainsi de nos jours, manipuler de telles quantités de données devient très difficile. C'est pour cela que filtrer ces données devient crucial.

3.2.2.1 Suppression des requêtes pour les ressources Web non analysées

Étant donné que notre objectif est de prédire les futures populaires des vidéos dans un CDN, nous choisissons de garder seulement les requêtes par vidéo visionnée et nous supprimons les autres requêtes comme celle pour les images ou les fichiers audio.

---

2. `http://www.statisticbrain.com/youtube-statistics`

3.2.2.2 Suppression des requêtes provenant des robots d'indexation

Les robots d'indexation ou en anglais *web crawler* ou *web spider* sont des logiciels qui balaient automatiquement un site web afin de collecter des ressources (pages web, images, vidéos, etc.) (Thelwall, 2001). Par exemple un moteur de recherche comme Google utilise ses robots afin d'extraire toutes les ressources web d'un site web et ainsi mettre à jour ses index de recherche (Najork, 2009). Le nombre de requêtes initiées par un robot d'indexation est en général bien plus supérieur à celui des utilisateurs "humains". La suppression des entrées *logs* produites par ces robots simplifiera l'étape d'analyse de *logs* qui suivra.

Généralement, un robot d'indexation est identifié en utilisant le champ "User-Agent" dans le format *log*. Seulement, aujourd'hui on compte chaque jour de nouveaux agents qui représentent des robots d'indexation ce qui rend presque impossible de tous les connaître. Il est possible néanmoins d'utiliser une des trois méthodes suivantes pour identifier les entrées de *logs* produites par des robots d'indexation (Tanasa et Trousse, 2003) :

- Reconnaître les adresses IP qui font une requête vers la page `"/robots.txt"`
- Utiliser une base de "*User-Agent*" connue comme étant des robots d'indexation
- Utiliser un seuil de vitesse de navigation ou en anglais *Browsing speed*. Au-delà de ce seuil, toutes les requêtes seront identifiées comme venant d'un robot d'indexation.

3.2.3 Étape d'extraction et d'agrégation des *logs*

Dans l'étape d'extraction et d'agrégation des données *logs*, nous présentons tout d'abord les définitions nécessaires et qui sont suggérées par le travail de W3C[3] sur

3. `https://www.w3.org/standards`

la terminologie de spécification du Web. Ensuite, nous élaborons six autres sous étapes qui sont respectivement : l'identification des vidéos, l'identification des vues par vidéo, l'identification des partages par vidéo, l'identification des abonnées à la chaîne par vidéo, l'identification de l'âge par vidéo et l'identification du temps de visionnage par vidéo qui traduisent tous en chiffre le niveau d'interaction des utilisateurs avec les vidéos. (Il s'agit des statistiques dynamiques pour une vidéo)

3.2.3.1 Définitions

- **Vidéo** - est identifiée par un URI et accessible via le protocole HTTP ou un protocole similaire.
- **Serveur cache** - serveur qui donne accès aux vidéos dans un CDN.
- **Requête** - une requête pour une vidéo faite par un utilisateur à un serveur cache.
- **Vue de vidéo** - le fait d'afficher une partie de la vidéo dans l'environnement du client à un moment précis.
- **Session utilisateur** - est un ensemble de requêtes utilisateurs (entrées de logs) sur un ou plusieurs serveurs caches.
- **Visite(s)** - est un sous-ensemble de clics utilisateurs pendant une session utilisateur. Les clics sont regroupés en plusieurs visites en estimant la distance temporelle entre deux requêtes consécutives. Si cette distance est supérieure à un certain seuil, on compte une nouvelle visite.
- **Épisode** - est un sous-ensemble de clics liés sur un ou plusieurs serveurs caches qui font partie d'une session utilisateur. Par exemple pendant une session sur YouTube, l'utilisateur a visionné une partie de la vidéo, s'est abonné à la chaîne de la vidéo et ensuite a partagé la vidéo avec son cercle d'amis.

3.2.3.2 Identification des vidéos

L'identification des vidéos à partir des lignes de *logs* est assez triviale. Le but est d'identifier toutes les vidéos consultées dans une fenêtre de temps w. En effet en utilisant le format des *logs* dans 3.2.1.2, nous connaissons les URI des vidéos ainsi

$$Videos(w) = \{video(l) \setminus l \in Logs \wedge temps(l) \in w\} \tag{3.2}$$

3.2.3.3 Identification de nombre de vues par vidéo (*views*)

On peut identifier le nombre de vues par vidéo en utilisant le champ *event* (ou évènement). Ce qui revient à compter toutes les entrées de *logs* contenant une action de type **"play"** pour une vidéo identifiée par l'URI v durant la fenêtre w.

$$V(v) = |\{video(l) \setminus l \in Logs \wedge temps(l) \in w \wedge (event(l) == "play")\}| \tag{3.3}$$

3.2.3.4 Identification de nombre de partage par vidéo (*shares*)

Même chose pour le nombre de partages par vidéo. Cette statistique indique combien de fois la vidéo a été partagée. Elle est obtenue par sommation de toutes les entrées de *logs* dont le champ action a une valeur **"share"**. Soit $SH(v)$ le nombre de partage pour la vidéo identifiée par l'URI v alors $SH(v)$ s'écrit :

$$SH(v) = |\{video(l) \setminus l \in Logs \wedge temps(l) \in w \wedge (event(l) == "share")\}| \tag{3.4}$$

3.2.3.5 Identification du nombre d'abonnés à la chaîne par vidéo (*subscribers*)

Le nombre d'abonnés à la chaîne par vidéo indique le total de personnes qui se sont abonnées ou désabonnées par vidéo durant la fenêtre de temps w. D'une manière générale, les utilisateurs devenus abonnés d'une chaîne interagissent plus

avec le contenu de la chaîne et consultent ses vidéos plus souvent. On a :

$$SB(v) = |\{video(l) \setminus l \in Logs \wedge temps(l) \in w \wedge (event(l) == "subscriber")\}| \tag{3.5}$$

3.2.3.6 Identification de l'âge par vidéo (*age*)

L'âge d'une vidéo est calculé en soustrayant la date du dernier visionnage par rapport à la date de mise en ligne de la vidéo. On suppose que l'âge d'une vidéo est toujours initialisé par une valeur très grande. Ce qui reflète que la vidéo est vieille (dans le sens de sa popularité) si jamais elle n'est pas visionné durant la fenêtre de temps w. Soit $A(v)$ l'âge de la vidéo avec l'URI v durant la fenêtre w alors

$$A(v) = t_{max}(v) - Upload(v) \tag{3.6}$$

avec

$t_{max}(v) = max_{temps(l) \in w}(temps(l) \setminus l \in Logs \wedge event(l) == "play" \wedge (video(l) == v))$

et $Upload(v)$ est la date de mise en ligne de la vidéo d'URI v.

3.2.3.7 Identification du temps de visionnage par vidéo (*watchtime*)

Le temps de visionnage par vidéo correspond au total du temps de vues d'une vidéo par tous les utilisateurs durant la fenêtre w. Cette donnée indique quelles sont les vidéos que les utilisateurs regardent réellement, par opposition aux vidéos sur lesquelles les utilisateurs cliquent mais ne regardent que quelques secondes. Ce nombre est calculé sous la forme de la moyenne de la somme des durées de tous les épisodes. Conceptuellement, un épisode commencerait par un événement de type "play" et se termine soit par un événement "pause" ou "stop". Seuls des évènements de type "comment", "share" ou "rate" peuvent composer un épisode. On note $\zeta(v)$ l'ensemble de tous les épisodes pour une vidéo v durant la fenêtre

de temps w alors :

$$W(v) = \frac{\sum \Delta\zeta(v)}{\mid \zeta(v) \mid} \tag{3.7}$$

avec $\Delta\zeta(v)$ est la durée de l'épisode $\zeta(v)$ associé à v.

3.3 Analyse des *logs* : Les modèles de régression linéaires

Une fois nos données prêtes, il faut les explorer. Cette étape regroupe plusieurs techniques pouvant être utilisées seules ou en compléments avec d'autres dans un but prédicatif.

Les modèles de régression sont utilisés pour cette tâche. Leur utilité principale réside dans la simplicité de leurs algorithmes d'estimation et de test qui permet de poser des modèles à plusieurs centaines de paramètres. Cette simplicité leur donne aussi une grande souplesse ainsi que la possibilité de s'adapter à plusieurs situations. Les modèles de régression sont fréquemment utilisés pour prédire la popularité. Plus particulièrement dans notre cas, prédire la dynamique de la popularité des vidéos devient alors possible. En effet, des variables comme celles évoquées dans 3.2.3 peuvent expliquer cette popularité.

Un modèle de régression tente de modéliser les données continues par une fonction (linéaire ou pas) en minimisant le taux d'erreur afin de prédire les valeurs futures. Dans notre cas, la valeur cible est la popularité à prédire.

En règle générale, les modèles de régression peuvent être divisés en deux grandes catégories : statique et dynamique. Dans l'approche statique, les modèles de régression utilisent un ensemble de données d'entraînement fixes qui ne change jamais. Contrairement à la régression statique, la régression dynamique utilise les données d'entraînement qui évoluent en fonction d'une fenêtre glissante dans le temps. La taille de la fenêtre est fixe et prédéterminée.

Dans notre cas, nous proposons un modèle de régression hybride qui commence par un modèle de régression statique, puis adapte le modèle de régression de façon continue, similaire à la régression dynamique, sur la base de nouvelles données disponibles dans la fenêtre sautante. Dans ce qui suit, nous allons discuter brièvement les approches de régression statique et dynamique et présenter notre modèle hybride de régression.

3.3.1 Régression linéaire statique

3.3.1.1 Modélisation

La régression linéaire statique tente, sur la base de données d'observations (fixe dans le temps) de rechercher une liaison linéaire entre une variable continue y et une ou plusieurs variables prédicatrices x_j, $1 < j < k$ également continues où k est le nombre total des variables de prédictions (Cornillon et Matzner-Lober, 2007). L'équation générale de la régression linéaire avec k variables prédicatrices est la suivante :

$$y_i = \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i} + ... + \beta_k x_{k,i} + \epsilon_i \tag{3.8}$$

où l'indice i représente la nième observation dans les données de traitement. $1 \leq i \leq n$, n étant le nombre d'observations. Les β_k sont les coefficients inconnus et β_0 est l'ordonnée à l'origine de la fonction linéaire. Le terme ϵ_i caractérise la marge d'erreur du modèle statique. Par conséquent notre objectif est d'utiliser le modèle de régression statique pour en déduire la relation entre les variables prédicatives x_j et notre variable de sortie, la popularité des vidéos.

On peut écrire matriciellement l'équation 3.8 de la manière suivante :

$$Y = X\beta + \epsilon \tag{3.9}$$

où

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \; X = \begin{bmatrix} 1 & x_{1,1} & x_{2,1} & \cdots & x_{k,1} \\ 1 & x_{1,2} & x_{2,2} & \cdots & x_{k,2} \\ 1 & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{1,n} & x_{2,n} & \cdots & x_{k,n} \end{bmatrix}, \; \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{bmatrix} et \; \epsilon = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

Y désigne le vecteur à expliquer de taille $n \times 1$, X est la matrice explicative de taille $n \times (k+1)$, et ϵ est le vecteur d'erreurs de taille $n \times 1$.

3.3.1.2 Estimation par les moindres Carres Ordinaires

Le problème consiste à calculer les coefficients de régression β_k. Il est évident qu'on ne peut calculer des valeurs exactes de β_k mais seulement des estimations que nous notons $\widehat{\beta_k}$. Plus précisément il s'agit de déterminer la droite de façon à ce que les termes d'erreurs de la forme $\epsilon_i = y_i - \widehat{y_i}$ soient les plus petits possible donc les plus proches de 0 avec $\widehat{y_i}$ est la valeur prédite par l'équation 3.8 lorsque $x = (x_{1,i}, x_{2,i}, \cdots, x_{k,i})$ (Cornillon et Matzner-Lober, 2007).

Le principe des moindres carres consiste à calculer les valeurs des paramètres $\widehat{\beta_k}$ qui minimisent la somme des carrés des résidus.

$$min \sum_{i=1}^{n} \widehat{\epsilon_i}^2 = min_{\widehat{\beta_0},\ldots,\widehat{\beta_k}} \sum_{i=1}^{n} (y_i - \widehat{\beta_0} - \widehat{\beta_1} x_{i,1} - \ldots - \widehat{\beta_k} x_{i,k}) \tag{3.10}$$

Ce qui revient à chercher les solutions de :

$$\frac{\partial \sum \widehat{\epsilon_i}^2}{\partial \widehat{\beta_j}} = 0 \tag{3.11}$$

En passant l'opérateur de dérivation dans la somme, on a $\forall j = 0, \ldots, k$:

$$\sum_{i=1}^{n} x_{i,j} (y_i - \widehat{\beta_0} - \widehat{\beta_1} x_{i,1} - \ldots, \widehat{\beta_k} x_{i,k}) = 0 \tag{3.12}$$

Il suffit alors d'écrire cette dernière relation 3.12 sous forme vectorielle :

$$X'(Y - X\widehat{\beta}) = 0 \tag{3.13}$$

Suite à une simplification, nous aurons :

$$\widehat{\beta} = (X'X)^{-1}X'Y \tag{3.14}$$

Dans la suite du rapport, on définit $\widehat{\beta}_{statique}$ comme étant le vecteur des paramètres de régression résultat de la régression statique.

3.3.1.3 Conditions d'application de la régression linéaire

Avant de procéder aux prédictions, il est important de remplir les conditions d'application de la régression linéaire. Si ces conditions ne sont pas remplies en partie il sera tout de même possible d'utiliser la régression par normalisation ou transformation des données. Voici donc les conditions d'application de la régression linéaire :

1. La relation entre x et y est linéaire
2. Les valeurs résiduelles sont distribuées normalement. C'est-à-dire pour chaque valeur de x donnée, l'ensemble des valeurs de y a une distribution normale.
3. Présence d'homoscédasticité Cette dernière condition suppose que les variances des résidus sont constantes pour toutes les valeurs de x et donc indépendantes des valeurs de x et de y.

3.3.1.4 Normalisation des données

Si une ou plusieurs des conditions plus haut ne sont pas remplies, il est possible de corriger le modèle par des normalisations ou des transformations des données. Il est possible de transformer la variable dépendante donc à prédire, les variables

de prédiction ou les deux au même temps. Plusieurs types de normalisations et de transformations peuvent être utilisés tels que le logarithmique, la racine carre ou encore les transformations exponentielles.

3.3.2 Régression dynamique

Contrairement à un modèle de régression linéaire statique, la régression linéaire dynamique utilise les données dans une fenêtre de temps prédéterminée (Nadungodage *et al.*, 2011). Par conséquent, le jeu de données change tout le temps. À chaque fois de nouvelles données arrivent les données précédentes qui ne coïncident pas avec la fenêtre courante seront éjectées. Ainsi la fonction de régression du modèle dynamique sera recalculée pour les nouvelles données capturées dans la fenêtre la plus récente. De cette façon, le modèle dynamique arrive à appréhender la dynamique du comportement des utilisateurs ainsi que leurs interactions avec les vidéos.

3.3.3 Vers une régression hybride

Comme nous avons pu voir précédemment il y a principalement deux catégories de régressions qu'on retrouve d'ailleurs dans la quasi-totalité des études scientifiques abordant la régression linéaire. Pourtant, il existe un compromis entre le modèle statique et le modèle dynamique donnant une meilleure prédiction des données. Contrairement à la régression dynamique, le modèle statique a une connaissance globale des données. Par conséquent, les modèles de régression dynamique ne permettent pas de prédire avec précision tous le flux de données. En conséquence nous proposons un modèle de régression hybride qui s'inspire des deux modèles présentés ci-haut et qui permet de prédire la dynamique des popularités des vidéos en tenant compte de tout le flux de données. La régression hybride utilise les interactions des utilisateurs avec les vidéos puis met dynamiquement à

jour la fonction de modèle de popularité.

Dans le modèle de régression hybride, nous commençons la prédiction sur la base d'un modèle statique puis dynamiquement , nous mettons à jour les coefficients de la régression au fur et à mesure que de nouvelles données sont transmises depuis le CDN (en déplaçant la fenêtre de prédiction dans le temps). Nous utilisons une estimation de la méthode EWMA (Brown et Meyer, 1961) pour mettre à jour le vecteur de coefficients $\widehat{\beta}_k$ dans la k^{ieme} fenêtre temporelle en utilisant le vecteur de coefficient de la fenêtre temporelle précédente $\widehat{\beta}_{k-1}$ comme suit :

$$\widehat{\beta}_k = \begin{cases} \alpha\widehat{\beta}'_{k-1} + (1-\alpha)\widehat{\beta}_{k-1} & \forall k \geq 1 \\ \widehat{\beta}_{statique} & k = 0 \end{cases}$$

(3.15)

où α est le facteur de lissage de telle sorte que $0 < \alpha < 1$ et $\widehat{\beta}_{k-1}$ est le vecteur des coefficients de la régression hybride calcule à partir de la $(k-1)^ieme$ fenêtre et $\widehat{\beta}_{statique}$ est le vecteur des coefficients de la régression statique (voir section 3.3.1). Le vecteur des coefficients de la régression hybride $\widehat{\beta}_k$ est essentiellement une combinaison pondérée de l'ensemble des données courante et précédente. En faisant varier le facteur de lissage α on peut ajuster la préférence quant aux données historiques. Par exemple assigner de grande valeur de α donnerait plus de poids aux données de la fenêtre en cours et donc moins de poids à ceux des fenêtres précédentes. De cette manière, nous pouvons contrôler l'impact des données historiques (précédentes) sur la popularité des vidéos les plus récentes.

3.4 Déploiement de la régression linéaire hybride

Comprendre la dynamique de popularité des vidéos est important afin d'améliorer la mise en cache des vidéos dans une infrastructure CDN. Ici nous proposons de déployer une stratégie de remplacement de cache pour les CDN hébergeant des

vidéos et qui utilise notre approche hybride de la régression. Ainsi nous sommes en mesure de prédire la popularité des vidéos en supprimant du cache les vidéos qui deviennent moins populaires au fil du temps et en gardant (dans le cache) celles qui sont toujours populaires de manière à avoir le taux de succès de cache le plus élevée possible.

Étant donné que la popularité des vidéos est très dynamique, LRU, LFU et FIFO-LFU (voir section 1.1.4) ne peuvent prédire avec précision la popularité des nouvelles vidéos mise en ligne ni s'adapter à la dynamique de la popularité. (Zhou *et al.*, 2015) montrent qu'une fois une vidéo perd de son attractivité, sa popularité diminuera. Par conséquent, il existe des vidéos populaires qui ne sont pas stockées par ces stratégies de mise en cache soient LFU et LRU. LFU garde les vidéos dans le cache pour une plus longue durée, même si elles n'ont pas été récemment sollicitées, puisque leurs fréquences d'accès resteront très élevées. Dans le but de tenter d'augmenter le taux de succès de cache, FIFO-LFU remplace les vidéos dans le cache selon l'âge et le nombre de vues. Par conséquent, nous proposons une stratégie de mise en cache qui élimine les vidéos les moins populaires. Dès qu'un défaut survient, la vidéo avec la popularité prédite la plus faible est évincée du cache. L'algorithme 1 illustre le pseudo code pour la mise en cache des vidéos dans un CDN avec notre approche hybride. Les définitions des paramètres utilisés sont présentées dans la table 3.1.

Dans l'algorithme 1, on applique la régression hybride aux fenêtres de logs $L = \{w_1, w_2 \ldots, w_N\}$ avec N le nombre totale de fenêtre collectée, C est la capacité du cache tel que $C \in N^+$, β est le tableau de tous les vecteurs des paramètres de régression et α est le facteur de lissage tel que $0 < \alpha < 1$.

Au départ l'entier i (indice de la fenêtre de logs) vaut 0 (voir ligne 3), *cache* vaut null (est vide) et $\beta[0]$, vecteur des paramètres de régression pendant la fenêtre de

logs w_0 vaut $\beta_{statique}$ ce qui correspond au vecteur des paramètres de régression du modèle statique (voir section 3.3.1).

Tant qu'on n'a pas atteint la fin de la liste $L(i < N)$, on calcule le nombre de hit (voir ligne 6). Ici on somme tous les nombres des vues des vidéos ayant eu un succès dans le cache. Dans la ligne 8, on prédit les nouvelles popularités des vidéos en utilisant l'équation 3.8. Au niveau de la ligne 9, on calcule la nouvelle valeur du vecteur des paramètres de régression en utilisant l'équation 3.15 de la régression hybride. Dans la ligne 10, on trie P par ordre décroissant des p_v (des popularités). Enfin dans la ligne 11, on met les C premières vidéos dans cache à partir de P (on remplit le cache avec les vidéos les plus populaires) et on incrémente l'indice i de la fenêtre de logs (voir ligne 15).

L'algorithme fait la mise en cache des vidéos en utilisant le modèle hybride et retourne le tableau des N nombres de hits.

$Videos$	Ensemble de toutes les URIs des vidéos
N	Nombre totale des fenêtres collectées
l	Une entrée de *logs* (voir définition 3.1)
w_i	Une fenêtre de *logs* d'indice i et de taille n_w
L	Ensemble de toutes les fenêtres de prédictions, $L = \{w_1, w_2, \cdots, w_N\}$. avec N est la taille de L
$V(v)$	Nombre de vues recueillies pour la vidéo avec l'URI v
X_i	Matrice des variables prédicatives pour toutes les vidéos collectées durant la fenêtre de logs w_i
$Videos(w_i)$	Liste de tous les URIs des vidéos reçues pendant la fenêtre w_i $Videos(w_i) = \{video(l) \setminus l \in w_i\}$
$cache$	Liste de toutes les vidéos stockées dans le cache avec $cache = \{v \setminus v \in Videos\}$.
H	Tableau de tous les *hits* de taille N
β_i	Vecteur des paramètres de régression dans la fenêtre w_i.
β	Tableau des vecteurs des paramètres de régression
α	Facteur de lissage, $0 < \alpha < 1$
C	Capacité du cache $C \in N^+$
p_v	La popularité prédite pour la vidéo avec l'URI v
P	Liste de toutes les popularités, $P = (p_v)_{v \in Videos}$

Tableau 3.1 Définition des entrées, sorties et des variables locales

Algorithme 1 Mise en cache des vidéos en utilisant la régression hybride

Entrée : $\{w_0, w_2, \cdots, w_{N-1}\}$: les fenêtres de logs tel que $0 \leq i \leq N-1$, C, Capacité totale du cache en nombre de vidéos, β : tableau des vecteur des paramètres de régression de taille N et α : facteur de lissage, $0 < \alpha < 1$

Sortie : H, tableau de tous les hits de taille N

Initialisation : $i \leftarrow 0$; $cache \leftarrow null$; et $\beta[0] = \beta_{statique}$

tant que $i < N$ **faire**

    /* Calcul des *hits* */

    $H[i] \leftarrow \sum_{v \in Videos(w_i) \cap cache} V(v)$

    /* Mise à jour du cache */

    $P \leftarrow predict(X_i, \beta[i])$ // En utilisant l'équation 3.8

    $\beta[i+1] \leftarrow update(X_i, \alpha)$ // En utilisant l'équation 3.15

    Trier P par ordre décroissant des p_v,

    Mettre dans *cache* les C premières vidéos à partir de P.

    $i \leftarrow i + 1$

fin tant que

retourner H

CHAPITRE IV

ÉVALUATION DES PERFORMANCES

Dans ce chapitre nous présenterons les performances de l'algorithme hybride pour la mise en cache des vidéos dans CDN. Nous commencerons par présenter le jeu de données qui nous a servis à illustrer nos résultats tout au long de ce chapitre. Nous y détaillerons quelques considérations spécifiques à propos de l'implémentation. La sélection des variables de régression et le choix du modèle se feront dans une deuxième étape. Puis quelques résultats numériques sur le jeu de données seront présentés. Finalement, nous comparerons l'approche qu'on propose dans ce mémoire à des stratégies bien connues de mise en cache.

Les résultats présentés ci-après ont été déjà publié à « *IEEE Global Communications Conference Exhibition Industry Forum, 2016* ».

4.1 Jeu de données

Dans la mesure où nous souhaitons prédire la future popularité des vidéos, il nous était nécessaire de disposer d'un grand nombre de vidéos. Dans cette section, nous décrirons comment notre jeu de données a été construit.

YouTube est devenu en quelques années le site de partage de vidéos le plus connu au monde (Burgess et Green, 2013). L'un des aspects des vidéos sur YouTube est leurs popularités décrites par leur nombre de vues. Entre autres, une vidéo est accompagnée d'un ensemble de métadonnées d'intérêt comme le titre, la date d'ajout, le nombre de partages ou le nombre d'abonnés. La page Web

de la vidéo affiche parfois quelques statistiques qui ne peuvent être récupérées que par autorisation du propriétaire du contenu. Pour avoir accès à une partie de ces statistiques, YouTube met à disposition deux interfaces de programmation. La première est l'API YouTube Data [1] qui permet de récupérer des statistiques disponibles pour tout utilisateur (comme le nombre de vues ou le titre d'une vidéo,etc.). La deuxième est l'API YouTube Analytics [2] qui permet sous autorisation du propriétaire du contenu de récupérer plus de statistiques comme la dynamique du nombre de vues pour une vidéo.

Comme une partie des données qui nous intéresse ne peut être accessible à travers de ces APIs, nous avons utilisé un jeu de données de plus de 24 millions de vidéos extraits de manière aléatoire à partir de YouTube. Le jeu de données est téléchargeable à partir du site CONGAS [3] en deux fichiers :

— le *Original* contient plus de 7 millions de vidéos extraits de manière aléatoire à partir de YouTube entre le 31 août 2013 et le 19 novembre 2013.

— et le *Extended* comporte plus de 17 millions de vidéos. La période de la collecte s'étend du 31 août 2013 au 22 novembre 2014.

4.1.1 Obtention et traitement du jeu des données

Les deux fichiers de données sont des fichiers ZIP [4] dans lesquels sont comprimées plusieurs entrées de vidéo sous format JSON. L'exemple ci-dessous présente une

---

1. `https://developers.google.com/youtube/v3`

2. `https://developers.google.com/youtube/analytics`

3. `http://www.congas-project.eu/youstatanalyzer-database`

4. Les fichiers Zip permettent de conserver des fichiers ensemble et rendent plus rapides et plus efficaces le transport, l'envoi par messagerie électronique, le téléchargement et le stockage de données et de logiciels.

partie de la définition d'une vidéo. Une vidéo est un objet composé de membres qui sont les attributs et de plusieurs tableaux.

```
{ id: u'9eToPjUnwmU',
    title: u'Traitor Compilation # 1 (Trouble ...',
    ...
    publishedDate: u'2012-10-09T23:42:12.000Z',
    author: u'ServilityGaming',
    duration: u'208',
   ...
    },
    views: {
        ...
        daily: {
            data: [15.0, 10.0, 1.0, 0.0, ..]
        }
    },
    shares: {
       ...
        daily: {
            data: [0.0, 0.0, 0.0, 0.0, ...]
        }
    },
    watchtime: {
       ...
        daily: {
            data: [22.5666666667, 13.95, 0.166666666667, 0.0, ...]
        }
    },
    subscribers: {
        ...
        daily: {
            data': [-1.0, 0.0, 0.0, 0.0, ...]
        }
    },
    day': {
        data: [1349740800000.0, 1349827200000.0, 1349913600000.0,
           1350000000000.0, ...]}
}
```

Ces données en leur état brut, ne nous sont pas très intuitives. Le but est de les convertir en format CSV (Shafranovich, 2005), abréviation de Comma-Separated

Values. Prenons par exemple une partie du fichier JSON de l'exemple précédent. Pour chaque jour (*day*) on crée (si le fichier n'existe déjà pas) un fichier avec comme nom la date du jour et on y inscrit l'entrée suivante :

```
id,watchtime,age,shares,subscribers
```

Cette ligne indique l'identifiant de la vidéo (*id*), la durée de visionnage (*watchtime*), son âge (*age*), le nombre de partages (*shares*) et le nombre d'abonnés (*subscribers*) correspondants .

À l'issue de cette étape, on obtient un premier répertoire de 814 fichiers en format CSV. Chaque fichier regroupe toutes les entrées de vidéos collectées durant une journée. Pour construire un répertoire de toutes les semaines de notre jeu de données, il nous suffit alors de regrouper tous les 7 jours consécutifs en partant du premier jour de la collecte dans un même fichier et avec comme nom le numéro de la semaine. Le total des fichiers alors obtenu est 118 et est égal aux nombres de semaines dans notre jeu de données.

4.1.2 Chargement et analyse du jeu de données

Arrivée à cette étape, il était important pour nous de choisir l'outil adéquat pour implémenter notre régression hybride et l'appliquer sur le jeu de données dont on dispose. Pour cela il existe une panoplie de *Framework* Open source, accessible et adaptée à ce type de traitements à l'exemple des logiciels SAS (Littell, 2006) et RStudio (Team, 2014). Comme on a à manipuler un nombre de données assez volumineux, notre recherche s'est vite tournée vers les technologies dites *Big Data*. Le terme *Big Data* en soi fait référence à toutes les problématiques reliées à l'usage des données dont la taille et l'étendue dépassent les capacités de traitement des outils traditionnels de gestion et d'analyse de données (Marz et Warren, 2015). C'est pour faire face au jeu de données conséquent dont nous disposons que notre

choix s'est arrêté sur le *Framework* Apache Spark[5].

4.1.2.1 Apache Spark

4.1.2.1.1 Présentation

Apache Spark est un Framework de calcul distribué apparu en 2010. C'est l'une des technologies qui marque le début de l'année 2015 et est connu pour être le successeur du pattern d'architecture MapReduce (Dean et Ghemawat, 2008) initié chez Google en 2002. Contrairement à MapReduce, Spark essaye de stocker le plus possible de données intermédiaires en mémoire ainsi sa performance peut-être plusieurs fois plus rapide que d'autres technologies *Big Data*.

Écrit en Scala, Spark est pensé pour être utilisé en Scala. Néanmoins, Spark propose des API additionnelles Java, Python, R et Clojure.

4.1.2.1.2 Les APIs

Plusieurs projets se greffent au-dessus de Spark :

— *Spark SQL* qui permet d'exécuter des requêtes SQL sur des RDD (Resilient Distributed Datasets). En autres Spark SQL permet d'extraire, transformer et charger des données sous différents formats comme JSON, Parquet, etc.

— *Spark Streaming* pour le traitement temps réel des flux de données. La librairie Streaming de Spark s'appuie sur un mode de traitement en "micro-batch" et utilise pour les données temps réel les DStream c'est-à-dire une série de RDD

— *GraphX* pour l'exécution d'algorithmes de graphes. GraphX étend les RDD de Spark en introduisant le Resilient Distributed Dataset Graph, un multi-graphe orienté avec des propriétés attachées aux nœuds et aux arrêtes. De plus, GraphX inclut une collection toujours plus importante d'algorithmes

5. `http://spark.apache.org`

pour simplifier les tâches d'analyse de graphes.

— et *MLlib*, la librairie de machine learning. Elle contient tous les algorithmes et utilitaires d'apprentissage classiques, comme la classification, la régression, le *clustering*, le filtrage collaboratif, la réduction de dimensions, en plus des primitives d'optimisation sous-jacentes.

4.1.2.1.3 Les Resilient Distributed Datasets

Un RDD est une structure de données, immuables, itérable et complètement paresseuses (*lazy*). Cette structure représente un graphe acyclique ordonnée des différentes opérations à appliquer aux données chargées par Spark. Il s'agit en quelque sorte d'un plan d'exécution.

Deux concepts s'appliquent au RDD (voir figure 4.1) : les transformations et les actions.

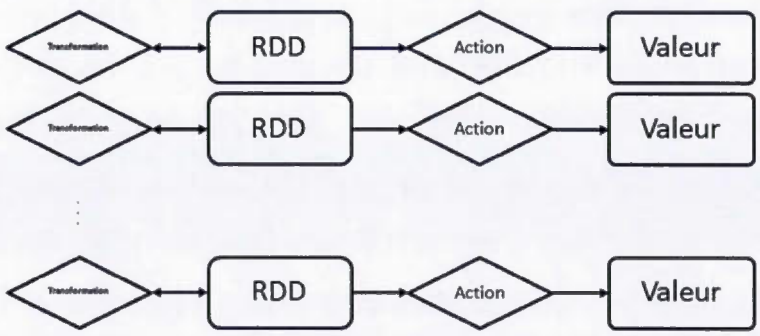

Figure 4.1 Les concepts s'appliquant aux RDDs.

Les transformations sont empilables sur les RDD. Une transformation crée un nouveau RDD. Les transformations sont paresseuses. Cela signifie qu'elles ne sont pas évaluées tout de suite. Les actions évaluent un RDD et retournent une nouvelle valeur.

Dans l'API Spark, le nom des méthodes ne permettent pas de déterminer si

on à faire à une opération ou à une action. Pour cela, il faut regarder le type de retour : s'il s'agit d'un RDD, la méthode est une transformation, sinon il s'agit d'une action.

4.1.2.2 Création et mise à jour de la régression hybride

Nous avons ici choisi d'utiliser le *Framework* Apache Sapark qui permet de traiter des problèmes de type *Big Data*. Comme énoncé plus haut Spark permet le développent d'algorithmes itératifs où la mise en cache est nécessaire. Une fois l'algorithme développé la génération de code et le lancement des calculs se font de manière transparente que ce soit sur une machine locale ou distribuée.

Nous utilisons l'API Java du *Framework* Apache Spark. Nous avons choisi ce langage pour implémenter l'algorithme de régression hybride que nous proposons pour sa simplicité et les nombreuses bibliothèques qu'il offre simplifiant ainsi le calcul scientifique.

L'important est de comprendre que notre régression hybride est paramétrée par un *LabledPoint*. Un *LabeledPoint* est composé d'un vecteur qu'on associe à un *label*. Le *label* est un *double* et représente la valeur de la popularité à prédire pour une vidéo. Le vecteur regroupe toutes les variables indépendantes. L'algorithme de régression hybride prend en entrée un *DStream[LabelPoint]* et retourne un *StreamingLinearRegressionWithEWMA* qui va pouvoir prédire la popularité de nouvelles vidéos grâce à la méthode *predict()* comme le montre la figure 4.2

Pour résumer la démarche globale pour prédire la popularité basée sur la régression hybride est la suivante :

1. Charger les données à traiter dans un *DStream*
2. Transformation des données pour obtenir un *DStream[Vector]* ou *DStream[LabeldPoint]* utilisable par l'algorithme de régression hybride.

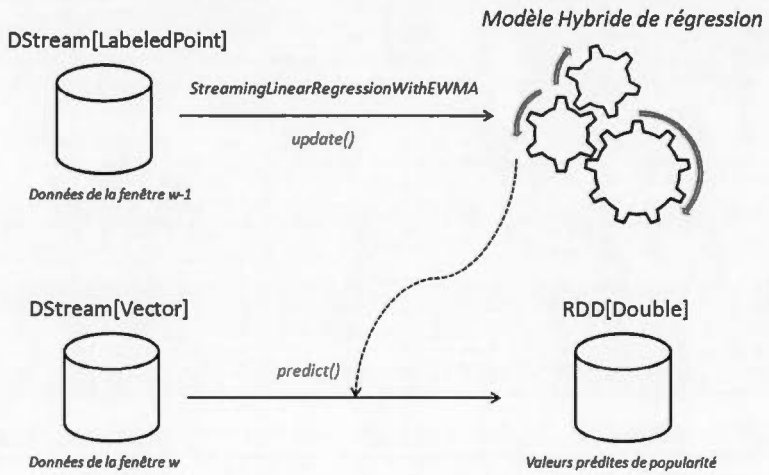

Figure 4.2 Schéma d'exécution de la Régression hybride.

3. Prédiction des nouvelles données grâce à la méthode *predict()* du modèle résultant de l'itération précédente
4. Mise à jour du modèle à l'aide de la méthode *update()* à partir du DStream crée
5. Une étape supplémentaire est nécessaire dans notre cas afin d'évaluer les performances du nouveau modèle. Pour cela, on utilise la méthode *evaluate()*. L'erreur de prédiction est calculée entre les valeurs prédites et les valeurs réelles de popularité.

4.2 Sélection des variables de régression

Le but de cette section est de construire un modèle de régression statique "solide" qui :

— décrit la relation entre les variables de façon à ce que l'on puisse prédire Y c'est-à-dire la valeur de popularité d'une vidéo à partir des valeurs connues de X = {*watchtime*, *age*, *shares*, *subscribers*}

— et ne contient pas plus de variables X que nécessaires pour obtenir une bonne

prédiction. Typiquement, les modèles comportant un petit nombre de variables indépendantes (bien sélectionnées évidemment) donnent de meilleurs résultats en pratique.

4.2.1 Description des données

On dispose d'un échantillon de taille 1 million de vidéos choisis aléatoirement à partir du jeu de données *Original* (voir section 4.1). Sur chacune des vidéos, nous récupérons :

— Le nombre de vues par jour ou *views*

— Le temps de visionnages par jour ou *watchtime*

— L'âge par jour, *age*

— Le nombre d'abonnées par jour ou *subscribers*

— et le nombre de partages par jour ou *shares*

Un extrait des données est présenté ci-dessous :

```
id,views,age,watchtime,subscribers,shares
BtWsPa5pGCk,96,11,72.0166666667,1,0
hQpygKgtZlk,56,84,120.95,0,0
mdo_9daI3LA,59,50,158.45,0,0
k2IViXcS8EM,0,34,0,0,0
3gGCedWUkk8,164,171,332.866666667,0,0
fqk-uc56TsQ,634,4,92,864.716666667,10.05,0,0,1,0
dHuLmDanSn0,29,350,33.7833333333,0,0
oHhK08-kRw8,6,57,0.25,0,0
LNcNMTrg4m8,97,92,109.216666667,0,0
cLhbVDTRCGY,0,39,0,0,0
HUuyrgCZBxg,21,179,31.0833333333,0,0
sVJpXbpFDEM,102,0,158.7,0,0
d_cqiYTM60E,3,49,0.483333333333,0,0
Cn3ZeTwOQo8,0,160,0,0,0
dR8bvDlZUBU,44,0,66.1166666667,0,0
hq7dgm0b1fQ,0,74,0,0,0
```

Le tableau 4.3 donne quelques statistiques élémentaires sur l'ensemble de l'échantillon.

	Popularité	*Watchtime*	*Age*	*Subscribers*	*Shares*
Total	1000000	1000000	1000000	1000000	1000000
Maximum	1396.84	3718.99	116.87	1.51582	2.67142
Minimum	0	0	0	-530.0	0
Moyenne	3.84093E7	1.04762E8	441.0	37305.0	71226.0

Tableau 4.1 Les statistiques de base du Jeu de données

4.2.2 Ajustement des données

Afin d'ajuster notre modèle de régression aux données, nous utilisons l'évolution journalière du nombre de vues *views* en fonction du temps. Nous désignons une observation d'une vidéo par $(y_i, W_i, A_i, SB_i, SH_i)$. avec y_i est le nombre de vues de la vidéo i, W_i est la durée de visionnage en minutes, A_i est son âge en nombre de jours, SB_i est le nombre d'abonnées et SH_i est le nombre de partages.

4.2.3 Corrélation et sélection des variables

L'un des objectifs d'un modèle de régression linéaire est de prédire la relation de causalité entre la variable réponse y_i et une ou plusieurs variables indépendantes. L'idée est de décrire le rôle de telle ou telle variable dans l'explication des valeurs prises par y_i. Si l'on trouve qu'une variable n'est d'aucune utilité, il est fortement conseillé de l'éliminer.

4.2.3.1 Analyse de la Corrélation

Un bon point pour commencer à élaborer un modèle de régression linéaire est l'analyse de la corrélation. Cinq variables prédicatrices possibles ont été sélectionnées, en plus du nombre de vues. Ces variables sont :

— X_1 : *watchtime*

— X_2 : *age*

— X_3 : *subscribers*

— et X_4 : *shares*

La figure 4.3 affiche la matrice des nuages des points des graphiques X-Y pour chaque pair de variables. La première ligne est très intéressante. Elle affiche le

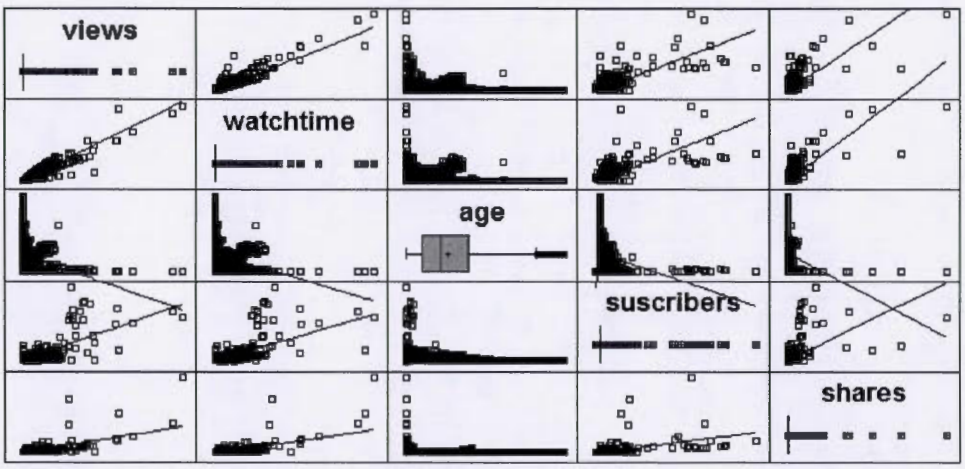

Figure 4.3 Matrice des nuages de points des points graphiques X-Y pour chaque pair de variables.

nombre de vues par jour c'est-à-dire la popularité par rapport à chacune des 4 variables prédicatrices potentielles. Toutes les variables sont clairement corrélées avec le nombre de vues par jour. Seule la variable âge par jour est non linéairement corrélée avec la variable réponse qui est le nombre de vues par jour. De même, il y a une corrélation importante entre les variables indépendantes (multicolinéarité entre les variables) ce qui laisse penser que plusieurs combinaisons peuvent être intéressantes pour prédire le nombre de vues par jour.

Le tableau 4.2 affiche une matrice des coefficients de corrélation estimés pour chaque paire de variables dans l'analyse. Le coefficient de corrélation r quantifie le degré de liaison linéaire entre deux variables continues X et Y (Seber et Lee,

2012). Il est défini par

$$r = \frac{\sum(x - \bar{x}).(y - \bar{y})}{\sqrt{\sum(x - \bar{x})^2}.\sqrt{\sum(y - \bar{y})^2}} \tag{4.1}$$

Où x et y représentent respectivement les valeurs de la première et la deuxième distribution. $\bar{x}$ et $\bar{y}$ représentent respectivement les moyennes de la première et la deuxième distribution. r est un nombre compris entre -1 et 1 et mesure la force

	Y	**X_1**	**X_2**	**X_3**	**X_4**
Y	-	**0.9432**	**-0.0493**	0.7852	0.7112
X_1	0.9432	-	-0.0465	0.7260	0.7083
X_2	-0.0493	-0.0465	-	-0.0385	-0.0319
X_3	0.7852	0.7260	-0.0385	-	0.5500
X_4	0.7112	0.7083	-0.0319	0.5500	-

Tableau 4.2 Matrice de corrélation X-Y pour chaque pair de variables.

de la relation linéaire entre deux variables. Plus la corrélation est proche de -1 ou de +1, plus la relation de linéarité est proche. Le signe de la corrélation indique le sens de la relation. Une valeur négative indique Y diminue lorsque X diminue.

4.2.3.2 Sélection des variables

La ligne de haut du tableau 4.2 indique la corrélation entre le nombre de vues par jour et les 4 variables prédicatrices. La plus forte corrélation est avec la variable *watchtime* et vaut 0.9432. Le signe positif indique que lorsque la variable *views* augmente, la variable *watchtime* croit elle aussi, ce qui n'est pas surprenant. La tableau 4.2 montre qu'il existe une corrélation linéaire faible entre les valeurs de popularités et la variable *age*, A. La valeur mesurée est proche de zéro et est égale à -0.0493. Il n'existe alors aucune nécessité de la variable A dans le calcul de la popularité. Nous pouvons ainsi l'exclure de notre modèle.

4.2.4 Choix du modèle

Cette approche consiste à produire toutes les combinaisons possibles de variables indépendantes, puis de choisir la régression qui optimise certains critères de qualité. Le nombre de cas à évaluer est égal à 2^{p-1} avec p le nombre de variables indépendantes. Dans notre cas p est égal à 3 (*watchtime, subscribers* et *shares*) ce qui nous donne 7 combinaisons possibles.

4.2.4.1 Les statistiques (critères) impliquées dans le choix du meilleur modèle

Les critères coefficient de détermination (R^2), Cp de Mallows et MSE pour racine du carré moyen de l'erreur sont calculés pour déterminer le meilleur sous modèle et peuvent servir de critères de comparaison.

4.2.4.1.1 R^2

Le R^2 exprime la part de variance expliquée par le modèle. C'est le premier critère qu'on regarde dans une régression. On essaye de trouver la combinaison de variables qui maximise le R^2. Il est défini comme 1 moins le ratio entre l'erreur avec les valeurs prédites et la variance des données :

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{4.2}$$

où y_i sont les valeurs des mesures, $\hat{y}_i$ sont les valeurs prédites, $\bar{y}$ est la moyenne des mesures et n la taille de l'échantillon (nombre des vidéos). En réalité R^2 ne convient pas seul. En effet, R^2 augmente de manière normale avec le nombre de variables. Plus on ajoute de variables, meilleur il devient, même si ces variables ne sont pas forcément pertinentes. Par exemple le modèle complet à 3 variables prédicative, incluant les variables *watchtime*, *subscribers* et *shares*, aura toujours une valeur de R^2 supérieure à un sous modèle de seulement 2 variables prédicatives. Il est possible de savoir à l'avance que le modèle comportant toutes les variables p

candidates est la solution optimale. Du coup le R^2 conviendrait uniquement pour comparer des régressions ayant le même nombre de variables.

4.2.4.1.2 *MSE*

L'erreur quadratique moyenne de prédiction ou en anglais MSE pour Mean square error est calculée sur la base de n vidéos par la formule suivante :

$$MSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2} \tag{4.3}$$

4.2.4.1.3 Cp de Mallows

La statistique de Mallows est donnée par la formule suivante :

$$C_p = \frac{\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2}{MSE(full)} - (n - 2p) \tag{4.4}$$

avec MSE(full) est la moyenne des carrés des résidus après la régression sur l'ensemble complet de p paramètres.

Un modèle est bien ajusté aux données si et seulement si la valeur de Cp est environ égale à p qui est le nombre de paramètres du modèle. Une valeur faible de Cp indiquera que le modèle est relativement précis dans son estimation des coefficients de régression et dans la prédiction des réponses future (dans notre cas popularité future). Cette précision sera peu améliorée par l'ajout de variables indépendantes supplémentaires. Par contre si la valeur du Cp est supérieure à p le modèle est jugé inadéquat.

4.2.4.2 Comparaison entre modèles

Chacune des lignes du tableau 4.3 indique un modèle différent. Il est clair dans ce cas-là que le meilleur modèle incluant les variables *watchtime* et *shares* convient le mieux pour nos données avec un Cp très proche de p dans ce cas égal à 3 et une

valeur de R^2 élevée (90.7623) alors que sa valeur de MSE est égale à 0.0102173. Le modèle contenant toutes les variables à la valeur de R^2 la plus élevée (92.9895), un Cp de Mallows faible (4.0) et une value de MSE la plus faible (0.010173). Le meilleur modèle à une seule variable inclut la variable *watchtime*. Il a un Cp très loin de p = 2 et une valeur de R^2 (92.5578) un peu plus faible que celle du modèle complet (*watchtime*, *subscribers* et *shares*).

Variables	**R^2**	**MSE**	**Cp**
W et SH	90.7623	0.0102173	**2.12938**
W, SB et SH	92.9895	0.010173	4
W et SB	90.3260	0.04786	4.88552
W	92.5578	0.027865	6.54608
SB et SH	0.023456	0.15987	1.67546E7
SB	0.026754	0.15876	1.67543E7
SH	0.028765	0.15765	1.324654E7

Tableau 4.3 Choix du modèle - Comparaison des résultats issus du jeu de données principal

Pour nous assurer que le modèle incluant les deux variables : *watchtime* et *shares* obtenu par la procédure de sélection ci-haut n'est pas ajusté uniquement au jeu de données (principal) qu'on a utilisé, nous avons refait le même traitement en utilisant 6 autres (différents) jeux de données. Le table 4.4 fournit la synthèse des résultats du choix de régression sur 6 nouveaux jeux de données.

Dans la table 4.4 nous nous limitons aux modèles de régression et leurs paramètres qui nous donnent le meilleur R^2 et un MSE et un Cp les plus petit possible. Il nous parait évident d'après le même tableau que les paramètres : *watchtime* et *shares* sont les deux paramètres prédominants qui nous donnent le meilleur mo-

dèle. Par conséquent, nous avons choisi d'utiliser ces deux paramètres dans notre stratégie pour prédire la popularité des vidéos.

Jeu de données	**Paramètres**	**MSE**	R^2	**Cp**
J-D1	W et SH	0.0107805	92.5271	2.00788
J-D2	W et SH	0.0102173	92.9895	2.12938
J-D3	W et SH	0.0100581	93.0515	2.56315
J-D4	W, SB et SH	0.0105582	92.7393	4.0
J-D5	W et SH	0.0102943	92.8941	2.00048
J-D6	W et SH	0.0133633	92.9911	2.33781
J-D7	W et SH	0.0337238	92.6732	2.92337

Tableau 4.4 Choix du modèle - Comparaison des résultats issus de plusieurs jeu de données

4.2.4.3 Le modèle final (optimal)

A cette étape, seules deux variables explicatives forment le modèle optimal : *watchtime* et *shares*. La variable à expliquer est bien évidemment le nombre de *views* c'est-à-dire la popularité d'une vidéo. Le modèle est le suivant :

$$V = \beta_0 + \beta_1 \times W + \beta_2 \times SH \tag{4.5}$$

La régression estimée est alors :

$$V = 0.052835 + 0.922079 \times W + 0.0237382 \times SH, \tag{4.6}$$

dénotée par le vecteur $\widehat{\beta}_{offline} = [0.052835\ 0.922079\ 0.0237382]$.

4.3 Résultats numériques

Les résultats numériques énoncés plus loin ont été mis en œuvre avec le langage de programmation Java SE 8[6] et Apache Spark 1.6[7] . Des expériences furent réalisées pour comparer les performances du modèle hybride avec celle du modèle dynamique et du modèle statique. L'algorithme 1 est ensuite comparé à d'autres algorithmes bien connus de mise en cache. Les expériences ont été réalisées avec l'outil Java Microbenchmark Harness (JMH)[8] et exécutées sur un processeur Intel inside à quatre coeurs avec une fréquence d'horloge de 2.5 GHz et 6 GB de mémoire RAM. Nous utilisons Eclipse 4.4.2 Luna[9] (32-bit) sur un système Linux Ubuntu 12.04.1 LTS.

4.3.1 Comparaison des performances entre les trois modèles : la régression statique, la régression dynamique et la régression hybride

Nous démarrons notre analyse par une analyse des performances de notre modèle face à la régression statistique et celle dynamique. Pour chaque modèle, nous évaluons le coefficient de détermination, R^2 et l'erreur quadratique moyenne, MSE ainsi que le temps d'exécution et l'espace mémoire utilisé.

Le facteur de lissage α prend des valeurs entre 0 et 1 et permet de donner un poids quant aux données historiques inclus dans la prédiction des nouvelles popularités (voir section 3.3.3). Si ce facteur a une grande valeur (donc proche de 1), le modèle hybride donne plus de poids aux données de la fenêtre en cours

---

6. http ://www.oracle.com/technetwork/java/javase/overview/index.html

7. https ://spark.apache.org/releases/spark-release-1-6-0.html

8. http ://openjdk.java.net/projects/code-tools/jmh/

9. http ://www.eclipse.org/downloads/packages/release/Luna/SR2

et donc moins de poids à ceux des fenêtres précédentes. Si ce facteur a de faible valeurs (donc proche de 0), les données historiques (des fenêtres précédentes) auront plus de poids. Dans le but de voir l'impact de ce facteur dans les résultats de prédictions (et alléger notre travail car il nous était impossible d'essayer toutes les valeurs de l'intervalle), on donne au facteur de lissage α les valeurs suivantes 0.25, 0.5 et 0.75. Pour une valeur de 0.25, les données historiques ont plus de poids que les données de la fenêtre courante. Pour une valeur de 0.5, les données historiques et ceux de la fenêtre courante ont des poids égaux. Pour une valeur de 0.75, les données actuels ont plus de poids que les données historiques.

Les logs des utilisateurs qui transitent pas le CDN sont traités dans des fenêtres temporelle successives. La fenêtre de temps a une taille prédéfinie (donc fixe). Cette taille peut prendre des valeurs de 1 heure, 1 jour, 1 semaine ou plus. Pour comparer notre modèle hybride aux deux modèles statique et dynamique, on a pris des tailles de fenêtre de 1 jour. Cette valeur est un compromis acceptable : 1 heure reste pas trop grand pour voir la dynamique d'une vidéo changée, les valeurs de 1 semaine ou 1 mois n'étant pas suffisamment petit pour détecter un changement de popularité d'une vidéo. La mise à jour de notre modèle hybride est faite en utilisant les données des logs de chaque fenêtre de temps. Ainsi la construction de notre modèle hybride est incrémentale : le modèle hybride est mis à jour en continue avec les données de logs d'une fenêtre de temps.

4.3.1.1 Coefficient de détermination (R^2)

Pour illustrer la qualité de notre modèle hybride, on représente l'évolution de R^2 dans le temps. En abscisse, la valeur zéro correspond au premier jour de la prédiction. Comme on le voit sur la figure 4.4, le modèle hybride prend environ 100 jours pour converger vers une valeur optimale de R^2 proche de 1. Notre modèle hybride présente des valeurs de R^2 très proches de 0.98, ce qui signifie que plus de

0.98 de la variation de la popularité des vidéos est expliquée par les deux variables prédicatrices *watchtime* et *shares*. La régression statique réalise également des coefficients de détermination qui ont des valeurs proches de notre modèle hybride. La régression dynamique réalise les valeurs les plus faibles de R^2, il reste cependant bon avec en moyenne un R^2 égal à 0.87. Chose intéressante, en comparaison avec la régression dynamique, le modèle statique atteint des valeurs plus élevées de R^2, avec à peu près 600 jours pour atteindre le niveau optimal (proche de 1), ce qui implique que les données historiques ont un impact évident dans le calcul de la popularité.

Malheureusement, les valeurs de R^2 seules ne sont pas pertinentes[10]. Ce qui souligne l'intérêt d'étudier les MSE des différents modèles en vue de dresser un portait plus clair des performances réelles de chacune des régressions.

4.3.1.2 Erreur quadratique moyenne

La figure 4.5 montre l'évolution de l'erreur quadratique moyenne pour la durée de 814 jours. Les six courbes correspondent respectivement aux modèles hybrides (en rouge, cyan et magenta), dynamique (en noire) et statique (en bleue). Nous observons que les gains apportés par la régression hybride en matière de MSE sont plus importants. De même comme nous le pensions, l'efficacité de notre approche hybride s'améliore au fur et à mesure que l'historique des données se remplit. En effet, la régression hybride s'appuie sur les données des vidéos précédentes afin d'estimer la popularité future de celles-ci. Il est clair que l'erreur induite par la régression hybride est beaucoup plus petite. La figure 4.5 montre bien que l'erreur diminue avec le temps. Notez qu'en utilisant un facteur de lissage α égal à 0.25, le modèle hybride supprime rapidement les données des jours précédents et favorise

10. `http://data.library.virginia.edu/is-r-squared-useless`

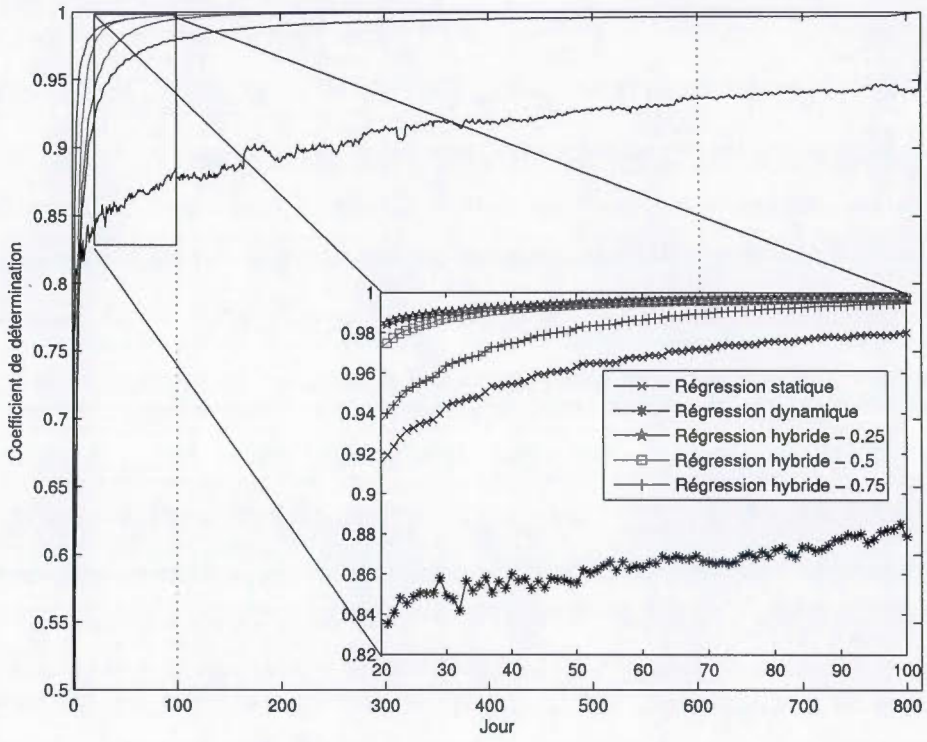

Figure 4.4 Évolution du R^2 dans le temps.

La régression hybride prend 100 jours pour converger vers un R^2 proche de 1, contre 600 jours approximative pour la régression statique.

donc les données les plus récentes. Pour des valeurs de α plus importantes, l'erreur augmente entraînant des queues plus longues et plus larges.

L'analyse de l'évolution de l'erreur quadratique moyenne est comparable avec les résultats précédemment mesurés pour le R^2. La popularité des vidéos est très sensible à la fois au *watchtime* et aux *shares*. Étant donné que le modèle hybride met à jour en continu la popularité précédemment prédite par les données historiques des vidéos, ce modèle hybride est capable de prédire la popularité avec une

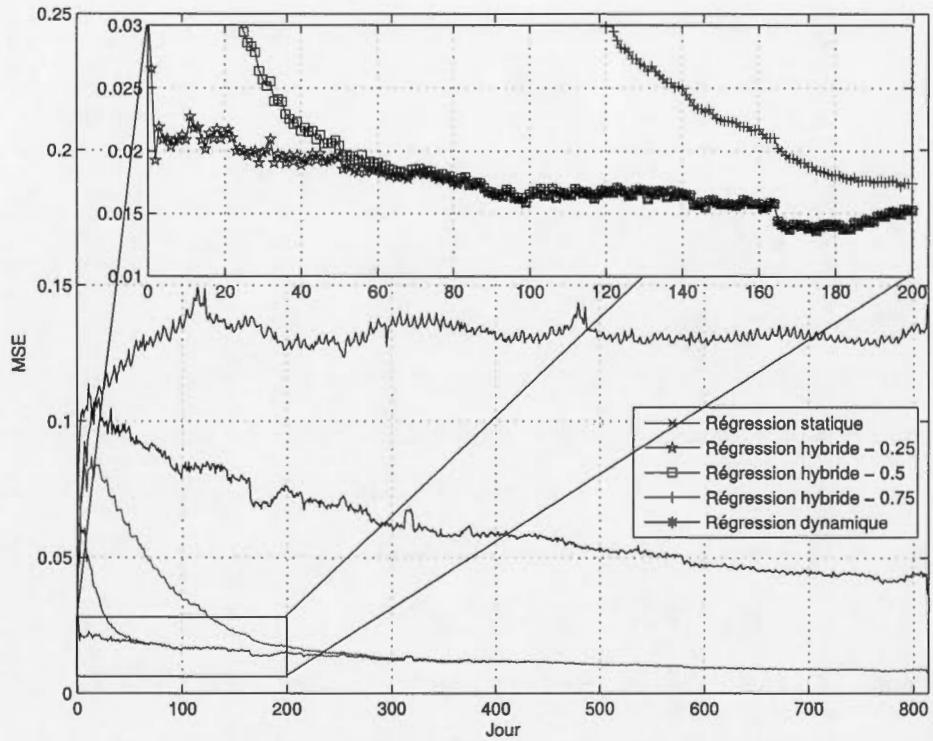

Figure 4.5 Évolution du MSE dans le temps.

La régression hybride a le MSE le plus bas, avec une augmentation maximale de 14%, par rapport à la régression statique

très grande précision. En revanche, la régression dynamique calcule les nouveaux coefficients de régression en utilisant les données des vidéos de la journée la plus récente, par conséquent, il ne peut pas atteindre la précision nécessaire obtenue en utilisant l'historique des données vidéos.

Comme l'illustre la Figure 4.5, l'erreur de prédiction pour la régression dynamique se situe entre 4% et 12%. D'autre part, le modèle statique prédit la popularité des vidéos en se basant sur un échantillon fixe de données vidéo (qui ne

changent pas au cours du temps). Par conséquent, les paramètres du modèle statique peuvent rapidement devenir obsolètes, ce qui explique que le modèle statique a les plus grandes valeurs de MSE. Ces valeurs varient entre 4% et 15%.

4.3.1.3 Coûts en temps d'exécution et en mémoire

Les figures 4.6 et 4.7 comparent les performances en termes de temps d'exécution et d'espace mémoire entre la régression statique, dynamique et hybride. La régression statique a les temps d'exécution les plus faibles, étant donné qu'elle nécessite un nombre fixe d'opérations pour prédire la popularité des vidéos. En revanche, on observe une augmentation dans les temps d'exécution pour les régressions dynamique et hybride comme ces modèles mettent à jour en permanence les paramètres de régression.

Comme l'illustrent les figures 4.6 et 4.7, aussi bien la régression dynamique que hybride tend à augmenter les temps d'exécution et l'espace mémoire utilisé. Ces augmentations découlent principalement des calculs supplémentaires de mise à jour du modèle, mais au prix d'un surcoût en temps d'exécution et en espace mémoire. Comme le montre la figure 4.6, les régressions statiques, dynamique et hybride ont 80%, 77% et 64% de chance qu'ils aient respectivement un temps d'exécution inférieure à 2500ms. De même respectivement les régressions statique, dynamique et hybride ont dans l'ordre 31%, 29% et 27% de chance que leur utilisation de la mémoire aient une valeur inférieure à 150 Mo (voir figure 4.7).

Veuillez noter que la régression hybride prend plus de temps d'exécution et d'espace mémoire en comparaison avec les deux régressions statique et dynamique. Cependant, un compromis existe inévitablement entre exactitude et surcout. Notre modèle hybride améliore la précision des prédictions avec un écart dans l'erreur allant jusqu'à 14% ce qui contrebalance le surcout produit en termes de temps d'exécution et de mémoire utilisés. Au vu de ces résultats, on est tenté d'utiliser

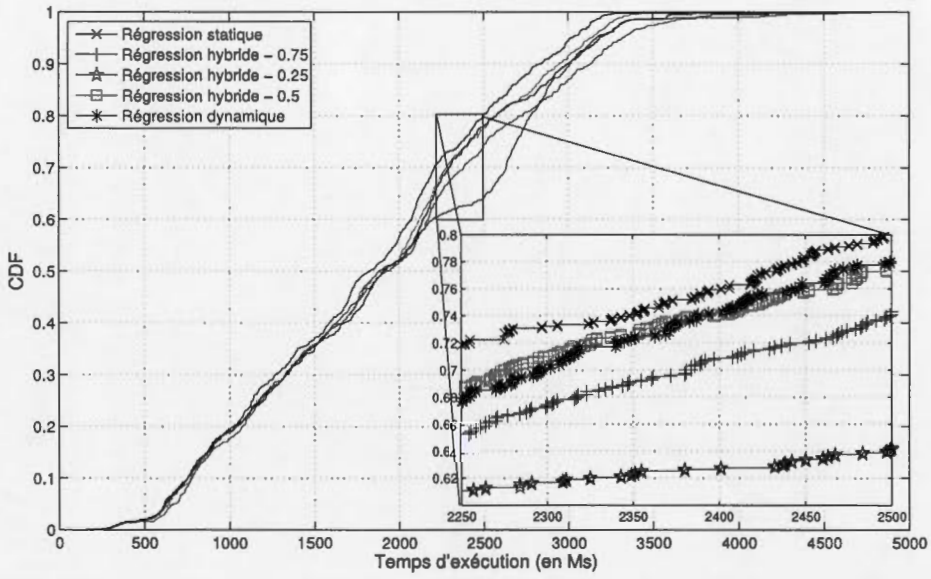

Figure 4.6 La fonction cumulative pour le temps d'exécution.

La régression hybride augmente le temps d'exécution de 16%, par rapport à la régression statique.

la régression hybride pour la mise en cache des vidéos.

4.3.2 Taux de succès cache

Nous évaluons la performance de l'algorithme hybride en termes de taux de succès cache moyen ou anglais AHR pour Average Hit Ratio calculé selon la formule suivante :

$$AHR = \frac{1}{L} \sum_{i=0}^{|L-1|} \frac{H_i}{R_i} \tag{4.7}$$

avec $|L|$ le nombre de fenêtres de prédiction, H_i le nombre de *Hits* durant la fenêtre i et R_i est le total des requêtes vidéos.

De manière générale, l'objectif est de maximiser le AHR. Un AHR élevé indique

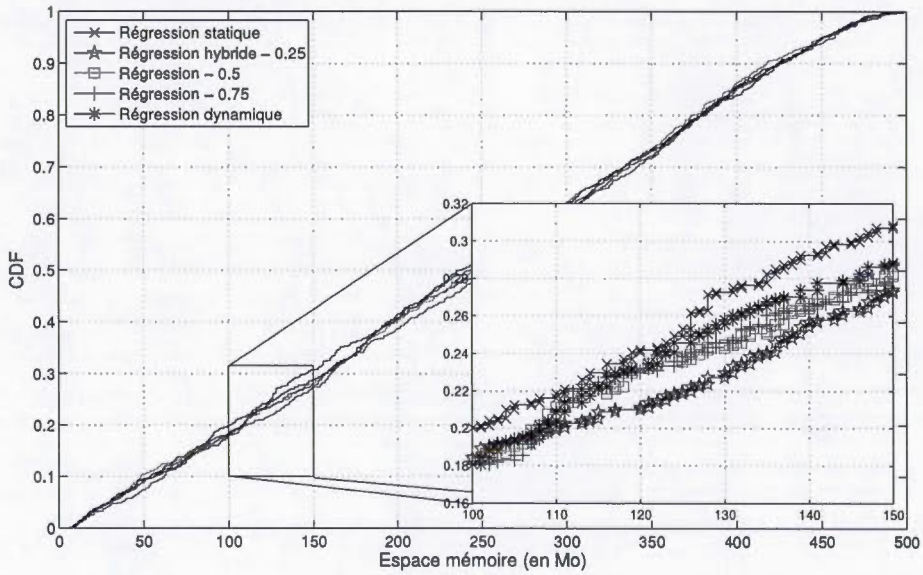

Figure 4.7 La fonction cumulative pour l'espace mémoire.

La régression hybride augmente l'utilisation de la mémoire de 4%, par rapport à la régression statique.

que les vidéos répliquées dans le cache sont populaires et réduiront par conséquent le temps de réponse moyen des demandes vidéos.

Nous comparons notre stratégie hybride de remplacement de cache, en utilisant la régression hybride avec une fenêtre de prédiction d'une taille d'une semaine et un facteur de lissage α égal à 0.5, avec les stratégies FIFO-LFU, LRU, JW-LFU et SW-LFU. La capacité de stockage du cache varie entre 1k et 30k en nombre de vidéos. Pour toutes les stratégies, les résultats sont présentés dans la figure 4.8. La stratégie hybride se positionne comme la meilleure solution parmi les autres algorithmes. Le AHR de l'algorithme hybride est respectivement 5%, 11% et 13% plus élevé que LRU, FIFO-LFU et JW-LFU. Alors que SW-LFU a la plus mauvaise

performance.

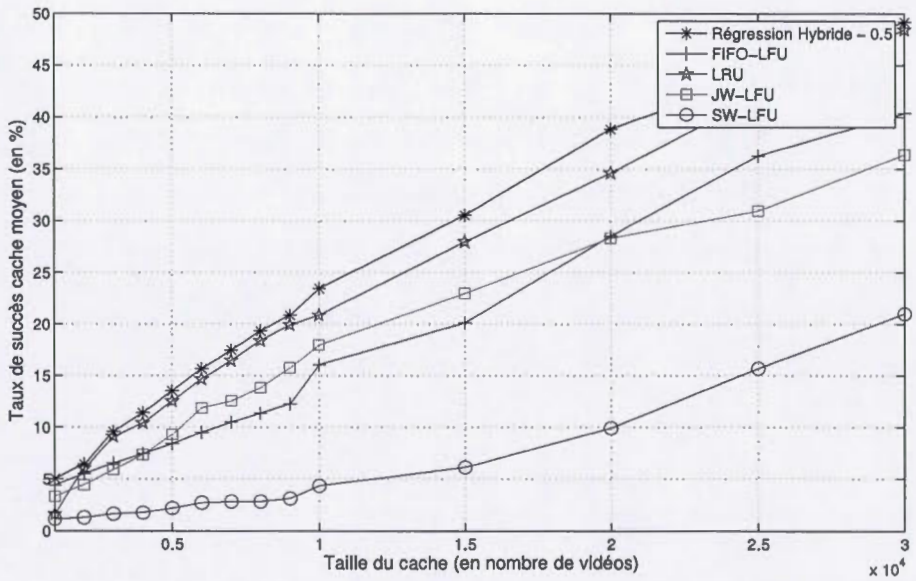

Figure 4.8 Évolution du taux de succès cache moyen en fonction de la taille du cache - Les prédictions se font à distance d'une semaine

Le taux de cache moyen pour la régression hybride est 5% plus élevé que celui de LRU et 11% plus grands que celui de FIFO-LFU.

La figure 4.9 montre le AHR journalier de toutes les stratégies de remplacement de cache citées plus haut. Il est évident que notre stratégie a la meilleure performance en termes de succès de cache comparé aux stratégies connues. La figure 4.9 montre pour chaque stratégie le taux de succès de cache en fonction de la taille du cache pour une fenêtre de prédiction de la taille un jour. Nous constatons un gain non négligeable. On relève un AHR de plus de 90% pour une taille de cache représentant 2% de la taille totale des données. En outre, le AHR du modèle hybride réalise un gain de plus de respectivement 7%, 74% et 80% que FIFO-LFU, LRU et JW-LFU.

Notre stratégie hybride utilise un modèle de prédiction de popularité qui est non seulement sensible à la fois aux variables *watchtime* et *shares*, mais s'adapte également aux vidéos les plus récentes. Par conséquent, il fonctionne bien pour les petites et grandes tailles de fenêtre de prédiction. Cependant, les fenêtres de petite taille génèrent des AHR plus élevés pour notre stratégie de remplacement de cache, comme il prédit la popularité avec une plus grande précision pour les vidéos qui changent rapidement de popularité. Remarque, FIFO-LRU, réalise le deuxième plus haut AHR journalier. Il est clair que la prédiction de popularité des vidéos est plus sensible aux variables *watchtime* et *shares* que la variable *age*. Bien que, LRU réalise le troisième meilleur AHR journalier, il reste nettement faible par rapport à celui de l'algorithme hybride. Les AHRs des algorithmes SW-LFU et JW-LFU sont les plus bas comme ces algorithmes gardent les vidéos même après qu'elles perdent leurs popularités dans le cache.

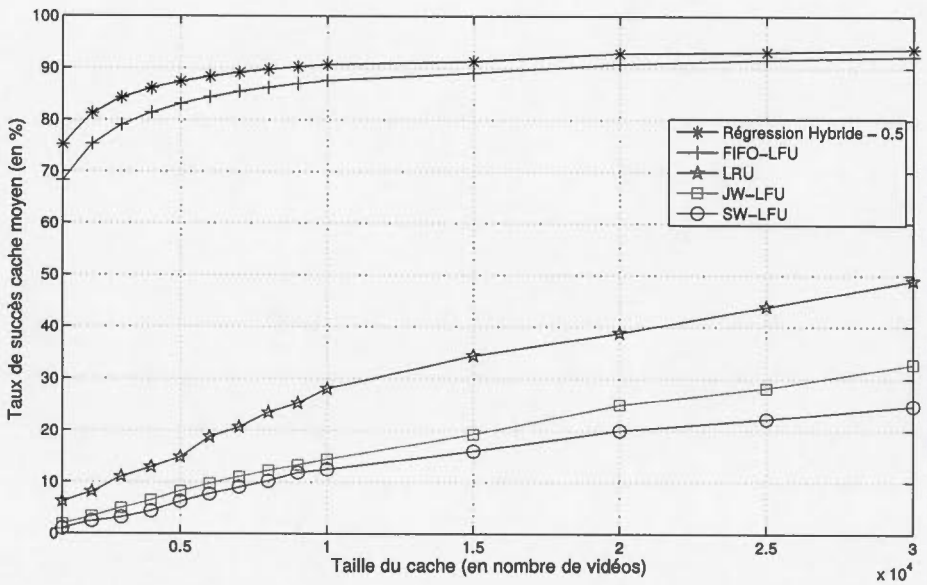

Figure 4.9 Évolution du taux de succès cache moyen en fonction de la taille du cache - Les prédictions se font tous les jours.

Le taux de cache moyen pour la régression hybride est 7% plus élevé que celui de FIFO-LFU et 74% plus grands que celui de LRU.

CONCLUSION

La diversification des services du *Cloud Computing* et la grande popularité qu'ont connues les vidéos, ces dernières années, ont donné naissance à de nombreuses problématiques de recherche. Dans ce mémoire, nous nous sommes particulièrement intéressés aux stratégies de remplacement de cache de vidéos et leur distribution dans un CDN, afin de garantir une qualité d'expérience satisfaisante aux utilisateurs finaux.

La distribution des vidéos ne peut plus être traitée par les stratégies traditionnelles. Le plus souvent ces stratégies présument que les vidéos les plus regardées dans le passé continueront d'être populaires dans le futur. Cependant, aujourd'hui, la popularité en tant que telle est dynamique et fluctue considérablement en fonction des interactions des utilisateurs avec ces vidéos.

Le *Big Data* et le *Data mining* répondent à ce besoin. En effet, ces outils sont l'occasion pour les CDNs de croiser un nombre conséquent de données vidéo et de les analyser. Ces outils mettent à profit des algorithmes et des techniques d'analyse pour transformer les données collectées en informations utiles. Les CDNs peuvent mettre en oeuvre ces outils dans leurs réseaux pour augmenter le taux de succès de cache par l'analyse de l'historique des traces de *logs* recueillies depuis leurs serveurs périphériques.

Dans ce contexte, la première contribution de ce mémoire a été de proposer une architecture *Cloud Computing* pour l'analyse de données en tant que service, AaaS. Cette dernière a pour but de faciliter le déploiement d'applications analytiques, dans lesquelles les différents acteurs (fournisseurs de services et/ou

de contenus) collaborent pour une distribution efficace des vidéos. Les travaux de recherche menés dans ce mémoire se fondent en partie sur la vue globale de l'architecture et plus particulièrement, sur les interfaces entre le CDN et le AaaS.

la deuxième contribution de ce mémoire est la conception d'un algorithme hybride pour prédire les futures popularités des vidéos. Nous avons montré le bon rendement de l'ensemble des données avec notre algorithme et la prédiction optimisée entre notre algorithme hybride et les données observées. Notre approche hybride permet de caractériser la dynamique des popularités des vidéos selon deux propriétés, qui sont le *watchtime* et les *shares*, sur une fenêtre de temps de durée prédéterminée. Nous découvrons que pour des fenêtres de temps respectivement égales à une durée d'un jour et une semaine, notre modèle hybride conduit à de meilleures performances en maximisant le taux de succès de cache alors que d'autres algorithmes réagissaient moins rapidement à l'évolution de la popularité. En outre, nous montrons qu'un gain non négligeable peut être obtenu en utilisant une fenêtre de prédiction d'une durée d'un jour pour le remplacement des vidéos.

À long terme, nous visons d'abord à améliorer notre algorithme hybride en matière de complexité temporelle et spatiale. Ensuite, nous allons étudier et résoudre le problème de corrélation entre vidéos. Enfin, nous allons nous concentrer sur l'aspect pratique de nos solutions. En d'autres termes, nous allons implémenter nos algorithmes sur des plateformes réelles de *Big Data*.

RÉFÉRENCES

Abhari, A. et Soraya, M. (2010). Workload generation for youtube. *Multimedia Tools and Applications*, *46*(1), 91–118.

Ahmed, M., Spagna, S., Huici, F. et Niccolini, S. (2013). A peek into the future : predicting the evolution of popularity in user generated content. Dans *Proceedings of the sixth ACM international conference on Web search and data mining*, 607–616. ACM.

Arlitt, M. et Jin, T. (2000). A workload characterization study of the 1998 world cup web site. *Network, IEEE*, *14*(3), 30–37.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I. *et al.* (2010). A view of cloud computing. *Communications of the ACM*, *53*(4), 50–58.

Avramova, Z., Wittevrongel, S., Bruneel, H. et De Vleeschauwer, D. (2009). Analysis and modeling of video popularity evolution in various online video content systems : power-law versus exponential decay. Dans *Evolving Internet, 2009. INTERNET'09. First International Conference on*, 95–100. IEEE.

Barlow, M. (2014). *Migrating Big Data Analytics into the Cloud*, volume 1. O'Reilly Media.

Berger, J. (2011). Arousal increases social transmission of information. *Psychological science*, *22*(7), 891–893.

Berger, J. et Milkman, K. L. (2012). What makes online content viral ? *Journal of marketing research*, *49*(2), 192–205.

Borghol, Y., Ardon, S., Carlsson, N., Eager, D. et Mahanti, A. (2012). The untold story of the clones : content-agnostic factors that impact youtube video popularity. Dans *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 1186–1194. ACM.

Borghol, Y., Mitra, S., Ardon, S., Carlsson, N., Eager, D. et Mahanti, A. (2011). Characterizing and modelling popularity of user-generated videos. *Performance Evaluation*, *68*(11), 1037–1055.

Borthakur, D. (2008). Hdfs architecture guide. *HADOOP APACHE PROJECT http ://hadoop. apache. org/common/docs/current/hdfs design. pdf.*

Brown, R. G. et Meyer, R. F. (1961). The fundamental theorem of exponential smoothing. *Operations Research*, *9*(5), 673–685.

Broxton, T., Interian, Y., Vaver, J. et Wattenhofer, M. (2013). Catching a viral video. *Journal of Intelligent Information Systems*, *40*(2), 241–259.

Burgess, J. et Green, J. (2013). *YouTube : Online video and participatory culture.* John Wiley & Sons.

Buyya, R., Pathan, M. et Vakali, A. (2008). *Content delivery networks*, volume 9. Springer Science & Business Media.

Carlinet, Y., Huynh, T. D., Kauffmann, B., Mathieu, F., Noirie, L. et Tixeuil, S. (2012). Four months in daily motion : Dissecting user video requests. Dans *Wireless Communications and Mobile Computing Conference (IWCMC), 2012 8th International*, 613–618. IEEE.

Cha, M., Kwak, H., Rodriguez, P., Ahn, Y.-Y. et Moon, S. (2009). Analyzing the video popularity characteristics of large-scale user generated content systems. *IEEE/ACM Transactions on Networking (TON)*, *17*(5), 1357–1370.

Chattamvelli, R. (2011). *Data mining algorithms.* Alpha science international.

Chatzopoulou, G., Sheng, C. et Faloutsos, M. (2010). A first step towards understanding popularity in youtube. Dans *INFOCOM IEEE Conference on Computer Communications Workshops, 2010*, 1–6. IEEE.

Chen, Q., Hsu, M. et Zeller, H. (2011). Experience in continuous analytics as a service (caaas). Dans *Proceedings of the 14th International Conference on Extending Database Technology*, 509–514. ACM.

Cheng, X., Dale, C. et Liu, J. (2008). Statistics and social network of youtube videos. Dans *Quality of Service, 2008. IWQoS 2008. 16th International Workshop on*, 229–238. IEEE.

Cheng, X., Liu, J. et Dale, C. (2013). Understanding the characteristics of internet short video sharing : A youtube-based measurement study. *Multimedia, IEEE Transactions on*, *15*(5), 1184–1194.

Cornillon, P.-A. et Matzner-Lober, E. (2007). *Régression : : théorie et applications.*

Crane, R. et Sornette, D. (2008). Robust dynamic classes revealed by measuring the response function of a social system. *Proceedings of the National Academy*

of Sciences, *105*(41), 15649–15653.

Cronin, B. (2001). Bibliometrics and beyond : some thoughts on web-based citation analysis. *Journal of Information Science*, *27*(1), 1–7.

Davidson, J., Liebald, B., Liu, J., Nandy, P., Van Vleet, T., Gargi, U., Gupta, S., He, Y., Lambert, M., Livingston, B. *et al.* (2010). The youtube video recommendation system. Dans *Proceedings of the fourth ACM conference on Recommender systems*, 293–296. ACM.

Dean, J. et Ghemawat, S. (2008). Mapreduce : simplified data processing on large clusters. *Communications of the ACM*, *51*(1), 107–113.

Famaey, J., Wauters, T. et De Turck, F. (2011). On the merits of popularity prediction in multimedia content caching. Dans *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*, 17–24. IEEE.

Fielding, R. (2000). Representational state transfer. *Architectural Styles and the Design of Netowork-based Software Architecture*, 76–85.

Figueiredo, F. (2013). On the prediction of popularity of trends and hits for user generated videos. Dans *Proceedings of the sixth ACM international conference on Web search and data mining*, 741–746. ACM.

George, L. (2011). *HBase : the definitive guide.* " O'Reilly Media, Inc.".

Gerhards, R. (2009). The syslog protocol.

Guadagno, R. E., Rempala, D. M., Murphy, S. et Okdie, B. M. (2013). What makes a video go viral ? an analysis of emotional contagion and internet memes. *Computers in Human Behavior*, *29*(6), 2312–2319.

Gürsun, G., Crovella, M. et Matta, I. (2011). Describing and forecasting video access patterns. Dans *INFOCOM, 2011 Proceedings IEEE*, 16–20. IEEE.

Hallam-Baker, P. M. et Behlendorf, B. (1996). Extended log file format. *WWW Journal*, *3*, W3C.

Han, J., Kamber, M. et Pei, J. (2011). *Data mining : concepts and techniques.* Elsevier.

Kim, S.-D., Kim, S.-H. et Cho, H.-G. (2011). Predicting the virtual temperature of web-blog articles as a measurement tool for online popularity. Dans *Computer and Information Technology (CIT), 2011 IEEE 11th International Conference on*, 449–454. IEEE.

Kimball, R. et Ross, M. (2011). *The data warehouse toolkit : the complete guide*

to dimensional modeling. John Wiley & Sons.

Lacomme, P., Phan, R. et Aridhi, S. (2014). *Bases de données NoSQL et Big Data.* Ellipses Marketing.

Littell, R. C. (2006). *Sas.* Wiley Online Library.

Marujo, L., Bugalho, M., Neto, J. P. d. S., Gershman, A. et Carbonell, J. (2013). Hourly traffic prediction of news stories. *arXiv preprint arXiv :1306.4608.*

Marz, N. et Warren, J. (2015). *Big Data : Principles and best practices of scalable realtime data systems.* Manning Publications Co.

Mitra, S., Agrawal, M., Yadav, A., Carlsson, N., Eager, D. et Mahanti, A. (2011). Characterizing web-based video sharing workloads. *ACM Transactions on the Web (TWEB)*, *5*(2), 8.

Nadungodage, C. H., Xia, Y., Li, F., Lee, J. J. et Ge, J. (2011). Streamfitter : a real time linear regression analysis system for continuous data streams. Dans *Database Systems for Advanced Applications*, 458–461. Springer.

Najork, M. (2009). Web crawler architecture. In *Encyclopedia of Database Systems* 3462–3465. Springer.

Ott, R. et Longnecker, M. (2015). *An introduction to statistical methods and data analysis.* Nelson Education.

Pathan, M. et Buyya, R. (2008). A taxonomy of CDNs. In *Content delivery networks* 33–77. Springer.

Pathan, M., Buyya, R. et Vakali, A. (2008). Content delivery networks : State of the art, insights, and imperatives. In *Content Delivery Networks* 3–32. Springer.

Podlipnig, S. et Böszörmenyi, L. (2003). A survey of web cache replacement strategies. *ACM Computing Surveys (CSUR)*, *35*(4), 374–398.

Rajagopalan, B. et Krovi, R. (2002). Data mining algorithms. *Data warehousing and web engineering*, p. 77.

Ratkiewicz, J., Flammini, A. et Menczer, F. (2010). Traffic in social media i : paths through information networks. Dans *Social Computing (SocialCom), 2010 IEEE Second International Conference on*, 452–458. IEEE.

Robinson, D. (2008). *Amazon Web Services Made Simple : Learn how Amazon EC2, S3, SimpleDB and SQS Web Services enables you to reach business goals faster.* Emereo Pty Ltd.

Salganik, M. J., Dodds, P. S. et Watts, D. J. (2006). Experimental study of inequality and unpredictability in an artificial cultural market. *science*, *311*(5762), 854–856.

Seber, G. A. et Lee, A. J. (2012). *Linear regression analysis*, volume 936. John Wiley & Sons.

Shafranovich, Y. (2005). Common format and mime type for comma-separated values (csv) files.

Sivasubramanian, S. (2012). Amazon dynamodb : a seamlessly scalable non-relational database service. Dans *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, 729–730. ACM.

Stamos, K., Pallis, G. et Vakali, A. (2008). Caching techniques on cdn simulated frameworks. In *Content Delivery Networks* 127–153. Springer.

Szabo, G. et Huberman, B. A. (2010). Predicting the popularity of online content. *Communications of the ACM*, *53*(8), 80–88.

Talia, D. (2013). Toward cloud-based big-data analytics. *IEEE Computer Science*, 98–101.

Tanasa, D. et Trousse, B. (2003). Le prétraitement des fichiers logs web dans le "web usage mining" multi-sites. *Journées Francophones de la Toile (JFT'2003)*, 113–122.

Tatar, A., Antoniadis, P., De Amorim, M. D. et Fdida, S. (2012). Ranking news articles based on popularity prediction. Dans *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*, 106–110. IEEE Computer Society.

Tatar, A., de Amorim, M. D., Fdida, S. et Antoniadis, P. (2014). A survey on predicting the popularity of web content. *Journal of Internet Services and Applications*, *5*(1), 1–20.

Tatar, A., Leguay, J., Antoniadis, P., Limbourg, A., de Amorim, M. D. et Fdida, S. (2011). Predicting the popularity of online articles based on user comments. Dans *Proceedings of the International Conference on Web Intelligence, Mining and Semantics*, p. 67. ACM.

Team, R. (2014). Rstudio : integrated development for r. *RStudio, Inc., Boston, MA. URL http ://www. RStudio. com/ide.*

Thelwall, M. (2001). A web crawler design for data mining. *Journal of Information Science*, *27*(5), 319–325.

Wang, Z., Zhu, W., Chen, M., Sun, L. et Yang, S. (2015). CPCDN : Content Delivery Powered by Context and User Intelligence. *Multimedia, IEEE Transactions on*, *17*(1), 92–103.

White, T. (2012). *Hadoop : The definitive guide.* " O'Reilly Media, Inc.".

Zhang, Q., Cheng, L. et Boutaba, R. (2010). Cloud computing : state-of-the-art and research challenges. *Journal of internet services and applications*, *1*(1), 7–18.

Zhou, R., Khemmarat, S. et Gao, L. (2010). The impact of youtube recommendation system on video views. Dans *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, 404–410. ACM.

Zhou, R., Khemmarat, S., Gao, L. et Wang, H. (2011). Boosting video popularity through recommendation systems. Dans *Databases and Social Networks*, 13–18. ACM.

Zhou, Y., Chen, L., Yang, C. et Chiu, D. M. (2015). Video popularity dynamics and its implication for replication. *Multimedia, IEEE Transactions on*, *17*(8), 1273–1285.

Zink, M., Suh, K., Gu, Y. et Kurose, J. (2009). Characteristics of YouTube network traffic at a campus network–measurements, models, and implications. *Computer Networks*, *53*(4), 501–514.