UNIVERSITÉ DU QUÉBEC À MONTRÉAL

INTÉGRATION DE DONNÉES MASSIVES DANS UN ENVIRONNEMENT EN CODE OUVERT

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAITRISE EN INFORMATIQUE

PAR

ANTOINE JEAN ROGER BRIAND

JUILLET 2019

REMERCIEMENTS

Ce document est la synthèse de deux ans de travail au sein de l'équipe de Marie-Jean Meurs du LATECE (laboratoire de recherche multidisciplinaire qui s'intéresse aux aspects technologiques et organisationnels des applications de l'Internet) [1].

Je crois sincèrement qu'aucun travail ne se fait seul, même si le résultat final ne présente qu'un seul auteur. Aussi, j'aimerais remercier toutes les personnes qui m'ont soutenu tout au long de ce parcours, vous nommer tous est impossible tant vous êtes nombreux, mais je suis persuadé que vous vous reconnaîtrez. Évidemment, un grand merci à ma famille et mes parents, Patrick et Valérie, pour tout le soutien qu'ils m'apportent et continuent de m'apporter depuis toutes ces années. Merci aussi à mes amis, qui de près ou de loin se sont intéressés à ce que j'ai pu faire durant ces deux ans. Bien évidemment, je remercie toute l'équipe de Marie-Jean Meurs, pour son soutien professionnel ou personnel. Mentions spéciales pour :

— Sara Zacharie, avec qui j'ai partagé de longues journées sur ce travail ... et aussi de nombreux repas pour compenser !
— Marc Queudot, Diego Maupomé, Hayda Almeida, vous êtes des collègues et amis précieux et je suis très content que nos chemins se soient croisés.
— Marie-Jean Meurs, directrice de mes travaux, qui a une capacité à soutenir et pousser les membres de son équipe au-delà de leurs limites, sans jamais montrer un signe de fatigue.

---

1. http://www.latece.uqam.ca/

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

RÉSUMÉ

La gestion des données massives et leur exploitation sont des sujets importants de l'industrie et de la recherche. Leurs particularités ont nécessité la création de technologies et techniques spécialisées afin de faciliter leur exploitation. Toutefois, la tâche d'intégration des données massives est toujours complexe. Pour autant, les applications qui découlent de l'exploitation des données massives ont des impacts très importants sur le plan industriel, scientifique ou social.
Afin d'encourager la création de projets relatifs aux données massives, nous présentons ici une infrastructure type, pour le traitement de données textuelles possédant toutes les caractéristiques des données massives : volume, vélocité, variété, véracité, valeur.
Celle-ci a pour caractéristiques la simplification du passage à l'échelle, de la maintenance, ainsi que la maîtrise du coût. Cette infrastructure est entièrement construite sur des technologies libres. Nous verrons également comment il est possible d'utiliser une même plateforme pour plusieurs types d'applications, en mettant en oeuvre une infrastructure en tant que service et en tant que code.
Pour en démontrer l'efficacité, nous présenterons deux cas réels d'application.
La première consiste à appuyer la protection de la vie privée par l'identification des données sensibles dans le cadre d'un projet de conformité réglementaire. Nous montrerons comment explorer au travers d'une masse de données non structurée afin d'en extraire par la suite les informations pertinentes.
La seconde application consiste en la création d'un système de détection des utilisateurs potentiellement dépressifs sur un réseau social en se basant sur leur production textuelle. Nous verrons comment les approches à base d'apprentissage machine et de recherche d'information peuvent contribuer à résoudre ce problème. Puis, nous étudierons les possibilités offertes par ces techniques lorsqu'elles sont couplées à une infrastructure orientée données massives. Ce travail ayant été soumis à la conférence CLEF eRisk, nous comparons nos performances à celles des autres équipes ayant participé.

MOTS-CLÉS : code source ouvert, architecture distribuée, données massives,traitement automatique du langage naturel, indexation

INTRODUCTION

La production de données est en pleine croissance depuis des années, donnant naissance au domaine d'étude des données massives (ou mégadonnées). La raison principale de cette haute génération de données est tout d'abord attribuée au commerce électronique puis, plus récemment, aux réseaux sociaux et à la démocratisation des téléphones intelligents et des services en ligne.

En 2017, Cristos Goodrow, vice-président du département d'ingénierie chez YouTube, annonce qu'un milliard d'heures de vidéos sont visionnées chaque jour sur YouTube (Goodrow, Cristos, 2017). À titre comparatif, un milliard d'heures représentent plus de 114 000 ans. Facebook possède une base d'utilisateurs proche des 2 milliards, et leurs centres de données contiennent plus de 300 pétaoctets de données à travers le monde (Kevin Wilfong, 2014).

Les entreprises de toutes tailles emmagasinent de plus en plus de données (Schroeck et al., 2012). Cela est principalement dû à l'adoption massive des outils informatiques pour la gestion client (CRM), la communication interne (messagerie instantanée, courriels) ou externe (publipostage, courriel ciblé), la mesure d'indicateurs de performance, etc.

Pour autant, ces données sont peu exploitées par ces entreprises, en témoigne le rapport Vertias (Veritas, Inc, 2016) Global Databerg qui stipule que 85% des mégadonnées de l'industrie sont stockées ou traitées inefficacement (doublons, véracité, obsolescence ...).

À première vue, ces données ne sont que des vidéos consommées par les utili-

sateurs, des échanges de messages, des informations commerciales, des remontées techniques (*logs*). Toutefois, l'exploration et le raffinement de ces données peuvent permettre d'extraire d'autres informations, extraire des profils d'utilisateurs, des cibles commerciales et aider, par exemple, à la prise de décisions.

La collecte d'informations de navigation, des requêtes dans un moteur de recherche, du temps passé sur une page, permet au marché de la publicité ciblée de fournir à leurs cibles des propositions personnalisées. Les utilisations des données massives ont ainsi un autre intérêt que la simple consommation de contenu, il s'agit de la création de valeur (pécuniaire, stratégique, scientifique, etc.).

Selon un rapport d'Affaires Mondiales Canada[2] datant de 2016, le marché annuel des données massives au Canada représente 1,1 milliard de dollars et ceci excluant les sources de revenus qui sont indirectement liées à l'utilisation des données massives (Affaires Mondiales, Canada, 2016).

La valeur ajoutée des données massives provient de la création de nouvelles techniques et technologies en rapport avec le domaine. En effet, l'exploration et le raffinement des données massives nécessitent systématiquement la mise en place de stockage de masse distribué, la création de cadres applicatifs (*frameworks*, un ensemble d'outils et de pratiques visant à abstraire des tâches et des concepts pour faciliter l'utilisation ou la création de systèmes) et d'algorithmes capables d'ingérer des données de grandes tailles, aux formats et aux sources hétérogènes, et des techniques de programmation particulières. La baisse du coût de stockage et l'augmentation de la puissance de calcul des machines ont permis de grandes avancées et la création de nouvelles applications qui ont suscité l'intérêt croissant des scientifiques et des industriels en matière d'exploration des données massives.

---

2. `https://www.international.gc.ca/`

Au-delà de cet intérêt, nous sommes aussi témoins, souvent à nos dépens, des risques liés à la gestion de ces gigantesques entrepôts de données. La presse relate de nombreux exemples de fuites d'informations (comprendre vol d'information). L'une des fuites les plus importantes de ces dernières années est celle de Yahoo ! qui en 2013 a perdu un milliard de comptes utilisateurs et cinq cents millions en 2014. Ces comptes utilisateurs contiennent des données personnelles comme le nom, le prénom, les adresses postales, les adresses de courriels, des numéros de téléphone, mais aussi des mots de passe (chiffrés ou non) et des réponses aux questions secrètes (Goel,Vindu et Perlroth, Nicole, 2016).

Plus récemment, 145,5 millions de clients de la société d'assurance Equifax se sont fait dérober des données au travers des serveurs de la compagnie (United States Government Accountability Office, 2018). La fuite inclut des informations de crédit (comptes, cartes, revenus), de santé (évaluations, rapports), et des informations d'identification comme les noms, prénoms, numéros d'assurance sociale, etc.

Dans la même lignée, un vol de données a eu lieu entre 2014 et 2018 au sein de la compagnie Mariott Hôtels (Radio Canada, 2018). Cette brèche impacte plus de 500 millions de clients en dévoilant, par exemple, les cartes de crédit des clients. La dernière fuite en date provient du site de questions-réponses Quora, qui fait état de 100 millions de comptes utilisateurs volés (Zhong, Raymond, 2018).

Les failles techniques et les vols de données sont en général la cause des fuites. L'intervention humaine est aussi à l'origine de ces fuites d'informations.

Ce fut le cas par exemple de l'affaire très médiatisée de Facebook et Cambridge Analytica. Cambridge Analytica a obtenu l'accès à plus de 50 millions de comptes d'utilisateurs de Facebook. Cet accès permettant de récupérer toutes les activités de chacun de ces utilisateurs sur le réseau social (messages, publications, habitudes d'utilisation, etc.) (Rosenberg, Matthew, Confessore, Nicholas et Carole

Cadwalladr, 2018).

Plus que le volume, c'est la fréquence à laquelle ces données sont volées qui est inquiétante, passant de quelques exceptions par année à plusieurs annonces de failles massives tous les mois. L'infographie interactive proposée par le site *informationisbeautiful.net*[3] rend compte de ce nouveau fléau.

Il existe pourtant un certain nombre de lois pour protéger les données privées ou sensibles. On peut citer Privacy Act (Department of Justice, US Government, 1974) pour les États-Unis, The Personal Information Protection and Electronic Documents Act (PIPEDA) (Gouvernement du Canada, 2000) pour le Canada ou encore le Règlement Général sur la Protection des Données (RGPD) (Parlement Européen, 2016) pour l'Europe. D'autres textes plus spécifiques à certains types de données comme Health Insurance Portability and Accountability Act of 1996 (HIPAA) (U.S. Department of Health & Human Services, 1996) existent pour protéger des données sensibles particulières comme celles concernant l'état de santé des individus.

Il apparaît donc primordial qu'un effort global pour limiter les fuites d'informations soit fait, impliquant la mise en place de systèmes à plusieurs niveaux pour prévenir ou, à défaut, très rapidement identifier les brèches de données dans les systèmes.

Ces fuites de données massives sont en lien avec le mouvement des données massives qui a débuté dans les années 1990. La démocratisation d'internet, l'amélioration des technologies de stockage et l'omniprésence des technologies au coeur de la société sont autant de facteurs qui ont conduit des milliards d'individus à renseigner leurs informations personnelles sur les plateformes en ligne. Tous ces

3. `https://informationisbeautiful.net`

systèmes sont donc autant de sources critiques où sont stockées les données de nombreuses personnes.

Malgré l'émergence d'un écosystème logiciel et matériel orienté données massives en pleine expansion, les défis posés par les données massives sont encore présents. Si des géants de la technologie sont capables de traiter des pétaoctets[4] de données tous les jours, il reste pour autant une grande proportion de l'industrie qui n'utilise pas pleinement les données massives. Les raisons sont multiples, mais se ramènent souvent aux mêmes questionnements : que faire de ces données ? Comment ? Combien cela va-t-il me coûter ?

En effet, aujourd'hui nos ordinateurs personnels ont une capacité de stockage qui peut atteindre plusieurs téraoctets (To). Pourtant traiter des téraoctets ou même des centaines de gigaoctets de données sur un ordinateur classique est presque impossible. Les systèmes de fichiers classiques ne sont pas adaptés au traitement intensif de données. Les processeurs, bien que de plus en plus puissants, ne sont pas à eux seuls assez rapides pour que le traitement soit efficace et la programmation classique ne permet pas de mettre à profit toutes les ressources disponibles.

Les expérimentations et mises en oeuvre opérationnelles doivent donc être réalisées avec une infrastructure possédant la capacité de traiter des données massives, autant sur le plan logiciel que matériel.

L'exploration de données requiert aussi des compétences et connaissances spécifiques au domaine. Partant de ce constat, il est nécessaire de pouvoir construire une infrastructure évolutive, permettant de commencer par des petites expérimentations tout en conservant une possibilité d'évolution vers une infrastructure suffisamment robuste pour gérer un flot continu et intensif de données en produc-

---

4. un pétaoctet équivaut à 10^{15} octets, c'est-à-dire 1 000 téraoctet.

tion.

Ainsi, ce mémoire s'articule autour de la construction d'une infrastructure hautement distribuée et évolutive pour les applications au domaine des mégadonnées. Il sera construit comme suit :

Dans un premier temps, nous allons faire un état de l'art des technologies et techniques inhérentes aux données massives, à l'indexation et à l'annotation de textes puis nous exposerons la problématique de ce mémoire.

Par la suite, nous détaillerons dans le Chapitre 1 une infrastructure de calcul et stockage adaptée aux données massives, puis nous présenterons deux applications des données massives, supportées par cette infrastructure avec au Chapitre 2 une application à recherche d'entités d'intérêt au sein de large corpus et au Chapitre 3 une application de la recherche d'informations pour la classification de textes. Enfin nous conclurons ce mémoire.

CHAPITRE I

DÉFINITIONS ET ÉTAT DE L'ART

1.1 Données massives

Les données massives (*Big Data* en anglais) représentent un concept dont les définitions tendent à varier et se contredire (Chen et al., 2012; Kwon et al., 2014; Gandomi et Haider, 2015).

Communément, on se réfère à la définition des VVV (ou 3V) énoncée par Doug Laney, de la société Gartner (Laney, 2001). Ces 3V correspondent à :

- Volume
- Vélocité
- Variété

Le **Volume** correspond à la quantité de données d'un système.

Il n'existe pas de taille standard pour laquelle des données sont massives ou non. L'appréciation du volume est faite en fonction du domaine d'où proviennent les données et des moyens dont dispose l'organisation qui souhaite les exploiter. En 2012, IBM a conduit une étude (Schroeck et al., 2012), basée sur les réponses de 1144 professionnels de la donnée. Dans ce rapport, on peut lire que plus de la moitié des répondants qualifient de massives des données entre 1 téraoctet et 1

pétaoctet (2^{10}To = 2^{20}Go = 1048576Go).

Le problème induit par le volume, c'est l'incapacité de traiter ces fichiers avec des solutions dites "classiques". Sur un ordinateur personnel, ouvrir un fichier de quelques gigaoctets est une opération lourde et chronophage, qui ne peut être exécutée que par des éditeurs très optimisés. Même si l'on parvient à ouvrir quelques gigaoctets avec un éditeur, exploiter ce contenu n'est pas chose aisée.

En code ASCII (*American Standard Code for Information Interchange*), un caractère loge dans un octet, ce qui veut dire qu'un fichier ASCII d'un gigaoctet pourrait contenir un milliard de caractères. Pour extraire des informations pertinentes d'un tel volume, il faudrait exploiter des fonctions de recherche qu'un éditeur de texte classique ne fournit pas.

Ces limitations sont encore plus présentes si l'on utilise des fichiers entre un téraoctet et un pétaoctet. Des fichiers d'un tel volume ne pourraient pas être lus d'un seul coup sur un ordinateur classique même avec le meilleur logiciel qui existe. En effet, la lecture en un seul morceau d'un fichier revient à charger son contenu dans la mémoire vive. Aucun ordinateur actuel ne possède plus que quelques centaines de gigaoctets de mémoire vive.

La **Vélocité** correspond à la vitesse à laquelle les données changent (mise à jour, obsolescence, suppression, etc.). Là encore le seuil de vélocité n'est pas clairement défini. Aujourd'hui, certains services sont capables de gérer une vélocité très importante. WhatsApp (Facebook), un service de messagerie instantanée, traiterait chaque jour plus de 65 milliards de messages[1] à travers le monde.

Ce critère pose un certain nombre de problèmes : Flux de données constant, né-

---

1. `https://www.theverge.com/2018/5/1/17302462/facebook-f8-conference-2018-keynote\-highlights-summary-day-1`

cessité de travailler en (quasi) temps réel, mises à jour et traitements par lots.

La vélocité pose problème avec les systèmes traditionnels comme les bases de données relationnelles.

La majeure partie de ces systèmes a été conçue avant l'ère des données massives et a été pensée pour maximiser la consistance et la disponibilité des données.

La consistance est imposée par un système très rigide (à base de schéma, de transactions atomiques, etc.). Dans un monde où les systèmes doivent s'adapter aux données et à leur évolution constante, cette rigidité est une limite que l'on ne peut pas toujours se permettre.

La disponibilité implique que chaque requête ait une réponse. En base de données relationnelles, cela implique souvent de verrouiller les ressources nécessaires aux requêtes, une à une, afin d'en garantir la disponibilité. Le comportement final induit par ces mécanismes est un ralentissement des réponses et un passage à l'échelle complexe.

Concernant les bases de données, des systèmes alternatifs ont émergé afin de résoudre les problématiques. C'est le mouvement des systèmes de bases de données NoSQL (Not Only SQL). La figure 1.1 permet la visualisation des types de bases de données en fonction de leurs caractéristiques [2]. Elle met en opposition les systèmes de gestion de base de données relationnelles classiques et les systèmes NoSQL (Clé Valeur, Orientés colonne, Orientés document).

Cette représentation est basée sur le théorème CDP (CAP en anglais) (Gilbert et Lynch, 2002) qui stipule qu'un système informatique distribué ne peut pas garantir les trois caractéristiques suivantes en même temps :

---

2. source `http://blog.nahurst.com/visual-guide-to-nosql-systems`

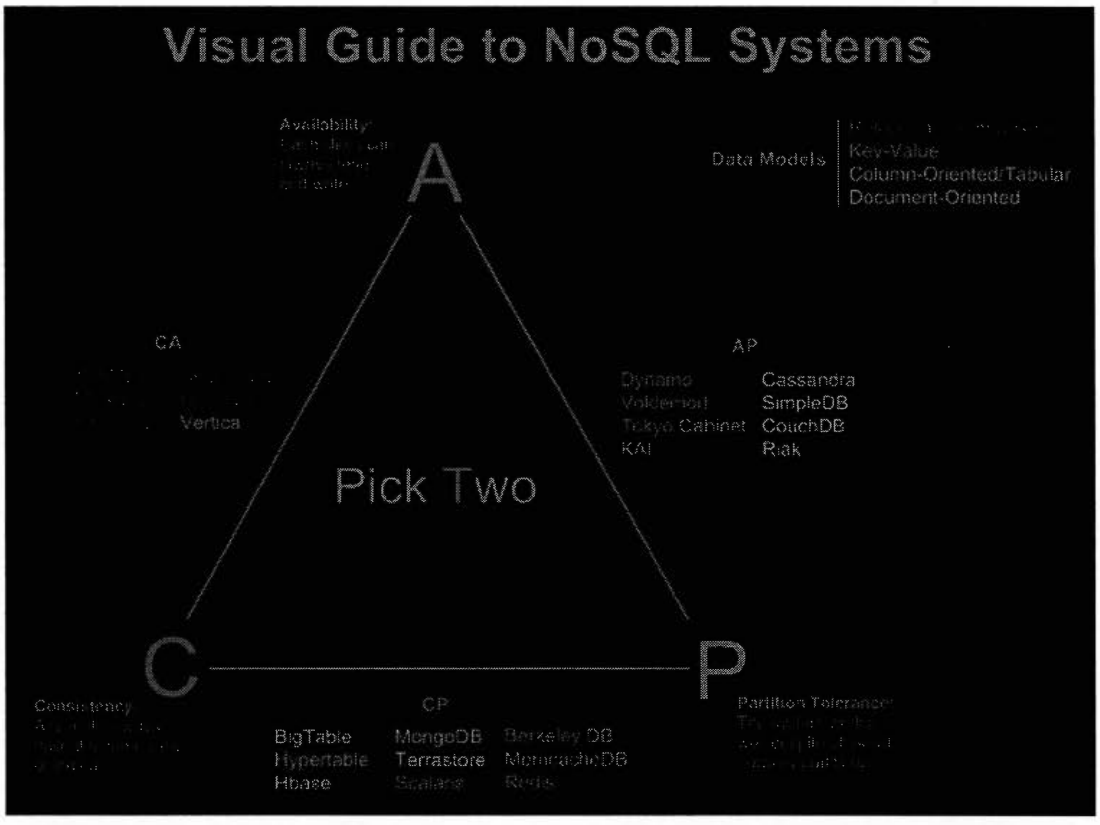

Figure 1.1: Représentation visuelle des systèmes de gestion de bases de données

— Cohérence (*Consistency* en anglais), la garantie que tous les nœuds du système soient constamment synchronisés.
— Disponibilité (*Availability* en anglais), la garantie que toutes les requêtes reçoivent une réponse.
— Tolérance au partitionnement (*Partition Tolerance* en anglais), la garantie qu'aucune panne inférieure à une interruption totale du système n'empêche le système de fonctionner.

Cette théorie s'étend à tous les systèmes distribués en général et n'est pas limitée aux systèmes de bases de données.

Enfin, la **Variété** caractérise des données qui proviennent de sources différentes

et/ou qui sont sous formats différents (données structurées / non structurées). La variété des données impose l'utilisation d'outils adaptatifs et compatibles avec de nombreux formats (données structurées, non structurées, textes, vidéos, etc.). Les entrepôts de données sont en général composés d'une grande variété de données, autour d'un même cœur de métier. Aussi, l'analyse de cette masse d'information doit tenir compte de chaque type de données disponibles, avec leurs particularités.

Une seule des caractéristiques n'est généralement pas suffisante pour qualifier des données de massives. L'accroissement de la capacité des supports de stockages et du réseau en général permet très facilement et à bas coût de stocker plusieurs téraoctets de données dont la vélocité est nulle, ou de gérer des données hautement véloces en faible quantité.

Cette définition des 3V a par la suite été étendue à 5V (Demchenko, 2013) pour inclure les notions de **Véracité** et de **Valeur**.

La **Véracité** des données est une caractéristique qui représente le degré de confiance que l'on peut accorder à ces données. Les données massives sont souvent fortement non structurées et proviennent de sources hétérogènes (internet, media sociaux, conversations, journaux, etc.). La qualité brute de celles-ci est donc bien souvent mauvaise.

La valeur est une caractéristique importante des données qui est définie par la valeur ajoutée que les données recueillies peuvent apporter à l'activité. C'est ce critère qui va influencer la manière dont les données sont recueillies, stockées (durée, support), etc. Les critères de **volume** et de **variété** peuvent donc être conditionnés par la **valeur**.

La figure 1.2[3] représente les VVVVV (ou 5Vs) que nous venons de présenter.

---

3. Extraite et traduite depuis (Demchenko, 2013)

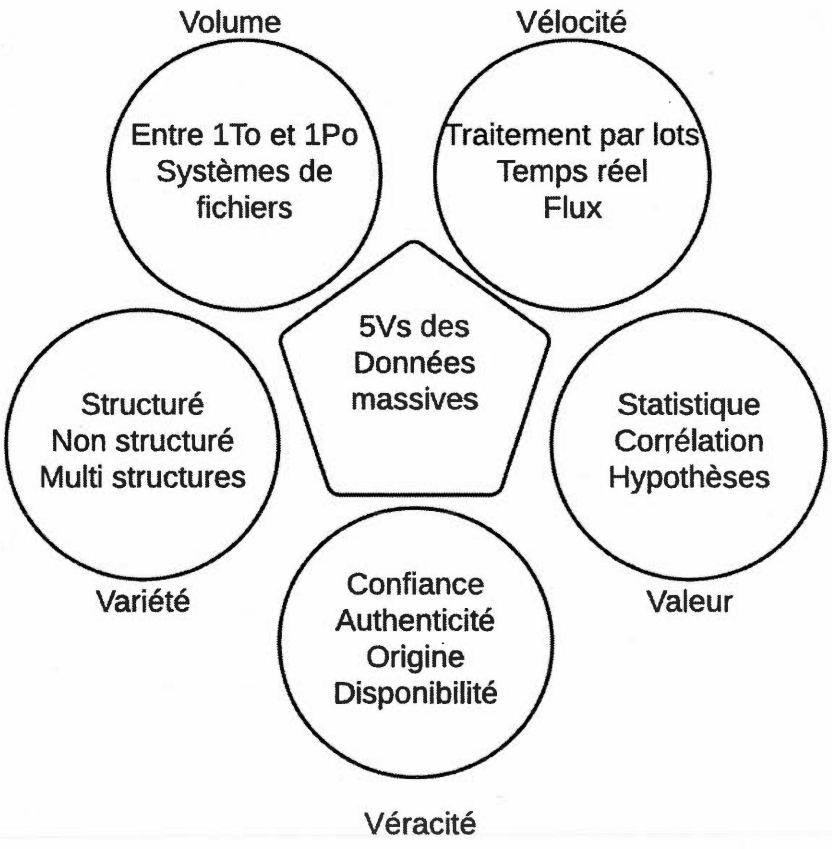

Figure 1.2: Le VVVVV des données massives

Pour répondre aux problématiques de stockage et d'exploitation de données massives, des technologies et techniques ont dû être développées. C'est le cas de l'écosystème Apache Hadoop, un cadre applicatif permettant de résoudre un grand nombre des problèmes posés par les données massives.

Cet écosystème est distribué sous licence libre (Apache) et maintenu par une très grande communauté, d'indépendants et d'industriels.

Ce cadre applicatif est un ensemble de projets interopérables et complémentaires dont voici la liste :

— Hadoop Common : Les librairies et utilitaires nécessaires au fonctionnement des autres modules Hadoop.
— Hadoop Distributed File System (HDFS ™) : Un système de fichiers distribués qui gèere le stockage et procure un accès haut-débit aux données.
— Hadoop Yarn : Une plateforme de gestion des ressources de calcul en grappes et de l'affectation des tâches (planification)
— Hadoop Mapreduce : Une implémentation du modèle de programmation MapReduce pour le traitement des gros volumes de données
— Ambari : Un outil web pour la gestion des grappes de calcul Hadoop.
— Avro : Un système de sérialisation des données
— Cassandra : Un système de gestion de base de données orienté colonnes distribuée conçu pour gérer les données massives sur de multiples serveurs, sans maître, avec réplication asynchrone.
— Chukwa : Un système dédié à la collecte et l'analyse des données de journaux (logs) nécessaires à la surveillance des grands systèmes distribués.
— HBase : Un système de gestion de base de données distribué conçu pour gérer les tables de très grande taille.
— Hive : Une infrastructure d'entrepôt de données permettant le réesumé, l'envoie de requêtes et l'analyse des données.
— Mahout : Des librairies d'apprentissage automatique et de fouille de données qui passent à l'échelle.
— Pig : Une plateforme pour l'analyse des données massives qui inclut un langage de haut niveau adapté au calcul parallèle, pour l'écriture de programmes d'analyse et une infrastructure pour leur évaluation.
— Spark : Un engin de calcul générique pour le traitement des données Ha-

doop incluant un modèle de programmation adapté à de nombreuses applications, depuis l'apprentissage automatique jusqu'à la fouille de graphes.

— Tez : Un framework d'exécution de tâches qui permet de rouler les tâches modélisées par des DAG (graphes orientés acycliques) et est plus rapide que MapReduce qu'il tend à remplacer.

— ZooKeeper : Un service de coordination haute performance pour les applications distribuées dont l'architecture permet la haute disponibilité des données grâce à des services redondants.

— Flume : Système distribué pour collecter, agréger et déplacer efficacement de grandes quantités de journaux (logs).

Cette liste évolue avec les avancées technologiques et les propositions de la communauté Apache. La majeure partie de ces projets est écrite en Java.

Le socle commun à tous ces projets est *Hadoop Distributed Filesystem* (HDFS) (Shvachko et al., 2010)(Borthakur, 2007). HDFS est un système de gestion de fichiers, distribué et hautement évolutif. Il permet la répartition de quantités de données importantes (pétaoctets) sur une grappe ou un ensemble de grappes de stockage de masse.

Pour y parvenir, HDFS repose sur une architecture précise composée de nœuds d'espaces de noms (*Name nodes*), nœuds de métadonnées (*MetaData nodes*) et des nœuds de données (*Data Nodes*). Un nœud étant une machine virtuelle ou un serveur physique. Tous ces services fonctionnent en synergie autour de la même interface de programmation (API) facilitant les communications et le passage à l'échelle. La figure 1.3 représente cette architecture.

L'espace de nom et les métadonnées sont le point central d'HDFS. Cette partie permet de stocker les informations autour des données stockées sur la grappe : nom, taille, type et surtout leurs positions dans la grappe. En effet, HDFS ne

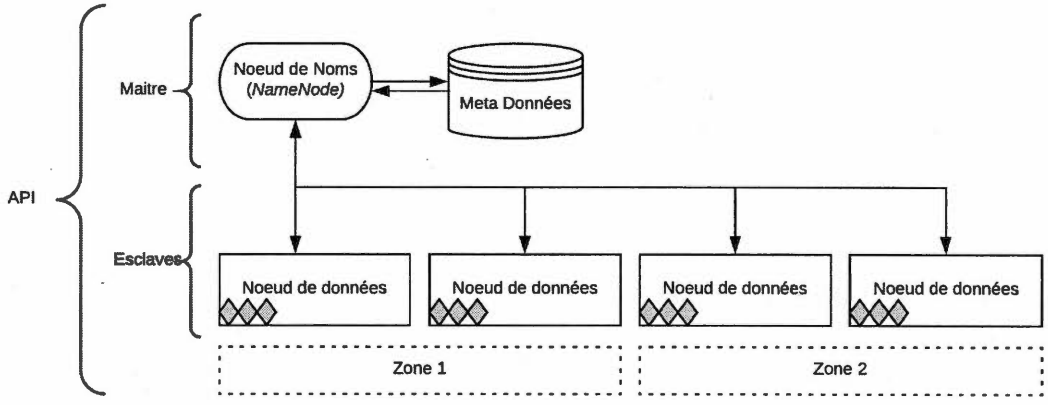

Figure 1.3: Architecture HDFS

stocke pas les fichiers comme un système de fichiers standard. L'objectif principal est d'être capable de traiter de grands volumes de données au global ou au niveau d'un seul fichier. Pour faire cela, HDFS s'appuie sur la distribution des données au sein de sa grappe avec un découpage à taille fixe des fichiers (par défaut 128 mégaoctets). Ainsi, un fichier d'un gigaoctet est stocké en huit morceaux (blocs) qui sont répartis au sein des nœuds de données. Ces morceaux peuvent être répliqués pour permettre d'accéder en parallèle au même fichier sans compromettre les performances.

Tous ces blocs doivent être répertoriés et retrouvés facilement afin de pouvoir exploiter de nouveau le fichier qu'ils représentent.

Le processus d'écriture au sein d'une grappe HDFS est schématisé par la figure 1.4.

Le processus de lecture est similaire, le client demande un fichier aux nœuds responsables de l'espace de noms, récupère le fichier morceaux par morceaux et le retourne. Il n'est toutefois pas obligatoire de retourner tout le fichier, surtout si celui-ci est très gros, le client HDFS permettant de travailler sur les morceaux

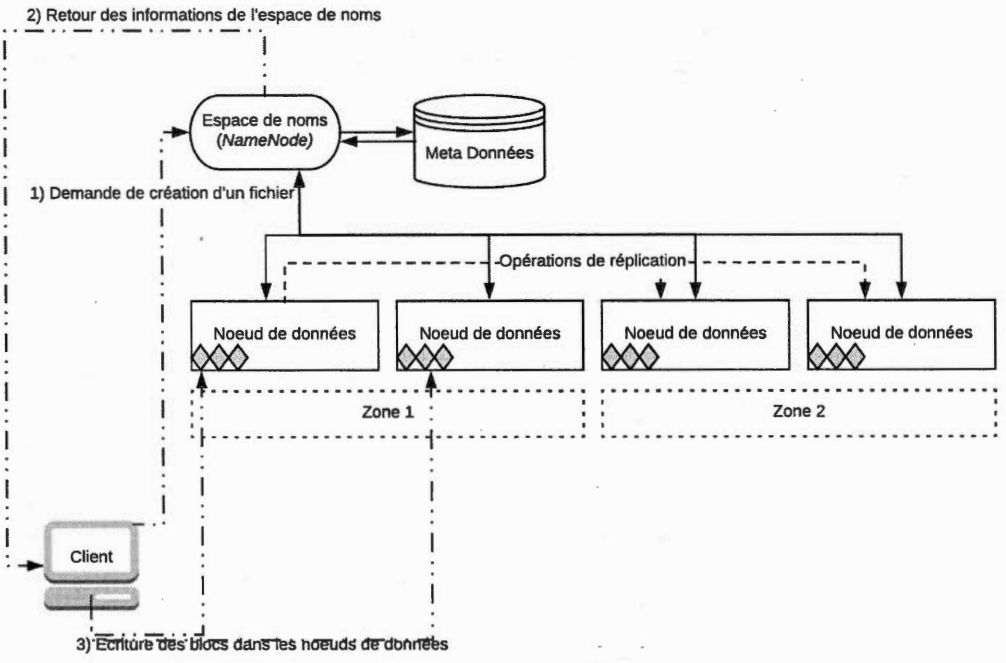

Figure 1.4: HDFS Écriture

du fichier.

En raison de son architecture, HDFS se veut tolérant aux pannes. Aussi, tous les services et nœuds sont constamment surveillés. La perte de données est évitée par réplication au sein de la grappe. Dans le cas où un nœud de donnée deviendrait indisponible, il est facilement possible de redistribuer les morceaux manquants à partir de leurs réplicas, disponibles sur les autres nœuds. C'est encore une fois l'espace de nom ou *Namenode* qui permet de localiser les réplicas et d'assurer la redondance.

Grâce à l'effort de la communauté, il est possible de travailler avec HDFS depuis de nombreux langages de programmation comme Java, Python[4], R, etc.

4. `https://pypi.org/project/hdfs/`

Dans la catégorie des systèmes de fichiers distribués HDFS n'est pas le seul projet existant. Ceph (Weil et al., 2006a), par exemple, est un projet libre créé à l'université de Californie, Santa Cruz par le *Storage Systems Research Center* et notamment par Sage A. Weil. Bien que plus récent que HDFS, Ceph est un projet mature utilisé par de grands groupes comme Red Hat [5], le CERN [6], etc [7].

Ceph est un système comparable à HDFS pour sa capacité à travailler avec des données massives. Son architecture quant à elle diffère complètement. La figure 1.5 présente l'architecture de Ceph [8].

Ceph est basé sur les principes suivants :

— Tous les composants doivent passer à l'échelle horizontale. [9]
— Pas de point individuel de défaillance.
— Fonctionne indépendamment du matériel.
— Réduction des opérations de maintenance (gestion automatisée).
— Code source sous licence libre.

Pour respecter ces principes, Ceph repose entièrement sur un système de stockage uniforme appelé RADOS (Reliable Autonomic Object Storage) (Leung et Maltzahn,). RADOS est capable de gérer le placement des données, leur réplication

---

5. `https://www.redhat.com/en/technologies/storage/ceph`

6. `https://ceph.com/community/new-luminous-scalability/`

7. `https://ceph.com/users/`

8. source `http://docs.ceph.com/docs/master/architecture/`

9. Passer à l'échelle horizontale signifie ajouter de nouveaux serveurs à une grappe. Dans le cas de Ceph, passer à l'échelle horizontale signifie être capable de prendre en compte les nouveaux serveurs d'une même grappe, automatiquement.

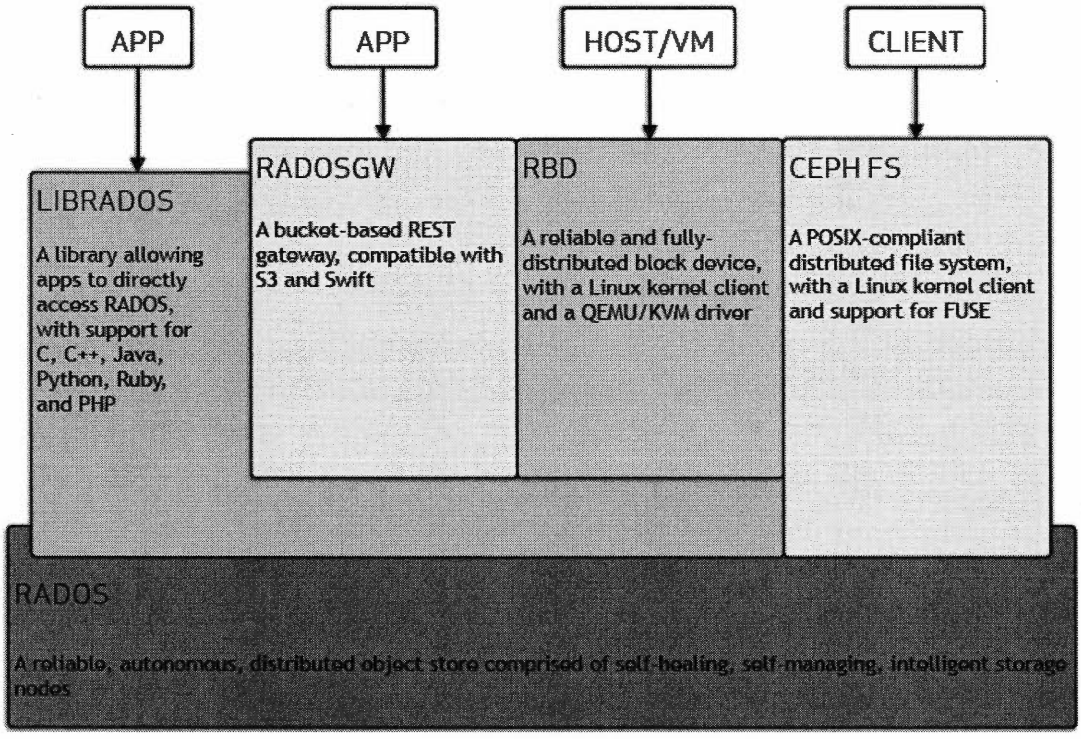

Figure 1.5: Ceph - Architecture

et leur récupération, sur l'ensemble de la grappe de stockage. Contrairement à HDFS, Ceph est un système totalement décentralisé, il n'existe pas de "maître" unique, chaque nœud pouvant être interrogé directement.

La figure 1.5 montre que Ceph propose plusieurs accès pour plusieurs types de données : Objets, Blocs (Disques virtuels) et fichiers. Toutes ces portes d'accès sont une abstraction à l'utilisation de RADOS.

RADOS fonctionne avec le système d'OSD (*Object Storage Deamon*). Ces OSD sont responsables d'unité de stockage physiques (disques de stockage, SSDs, ..). L'utilisation d'un système unifié pour la gestion du stockage physique permet à Ceph d'être très performant sur la récupération des données et la tolérance aux pannes.

Avec HDFS, les données sont stockées sur les nœuds de données avec un ordre imposé par le système d'espace de noms. Avec Ceph, c'est l'algorithme CRUSH(*Controlled Replication Under Scalable Hashing*) (Weil et al., 2006b) qui est responsable de cette tâche.

Cet algorithme tend à résoudre plusieurs problèmes inhérents au stockage distribué d'objets. Ces systèmes proposent des journaux centralisés qui stockent des métadonnées. Bien qu'efficace, cette solution crée un point de défaillance unique sur le journal de métadonnées. Si ce journal n'est plus disponible, la grappe est paralysée. De plus, les systèmes de répartition des objets organisent ceux-ci en fonction de l'utilisation des nœuds à un instant T. Dans le monde réel, les nœuds de stockage sont sans cesse en mouvement (ajouts, faille (suppression d'un nœud), etc.), et leurs caractéristiques varient. Au final, ces variations alliées à une répartition basée sur la charge tendent à rendre la grappe de stockage déséquilibrée.

Pour résoudre le problème de répartition, CRUSH utilise un système pseudo aléatoire qui distribue les données aléatoirement sur la totalité de la grappe de stockage. Le déplacement de données induit par le mouvement des nœuds se produit alors de la même façon, une quantité de données est de nouveau répartie aléatoirement sur toute la grappe, permettant de conserver un niveau de charge équilibré.

Ces systèmes de stockage distribués, hautement disponibles et tolérants aux pannes, sont les fondations solides des systèmes d'analyse distribués. Pour traiter des données massives, nous devons adapter nos techniques de programmation pour que celles-ci passent à l'échelle.

En programmation séquentielle, nous utilisons un seul fil d'exécution pour traiter les instructions une à une jusqu'à l'obtention d'un résultat. C'est un modèle de programmation qui convient aux applications simples. Avec les processeurs actuels, il est possible de tirer parti de la programmation parallèle qui consiste à faire

de multiples traitements en même temps, en général un traitement par cœur de processeur. Cette technique permet de grandement accroître les performances d'un système à l'échelle locale.

Lorsque l'on s'adresse à des quantités de données très importantes, ou que l'on doit répondre à des problématiques en temps réel, la programmation séquentielle est hors concours. La programmation parallèle est utilisée pour supporter le passage à l'échelle verticale et l'utilisation maximale des ressources disponibles.

Toutefois, pour exploiter pleinement les systèmes comme HDFS ou Ceph, on utilise la programmation distribuée et parallèle. Ce type de programmation implique de distribuer les traitements d'un programme sur le maximum de nœuds de calcul possibles.

De la même façon que la programmation parallèle nécessite des techniques particulières (gestion de la concurrence, des multicoeurs, etc.), la programmation distribuée demande l'utilisation de technologies nécessaire à la synchronisation des nœuds, le partage des tâches, la récupération des résultats, etc.

Apache Hadoop propose aussi des solutions à cette problématique. La plus connue est Hadoop MapReduce. MapReduce est un modèle de programmation publié par Google en 2004 (Dean et Ghemawat, 2008) qui est inspiré par la programmation fonctionnelle.

Hadoop MapReduce est un modèle qui propose d'effectuer les traitements avec deux opérations *map* et *reduce*.

Avec *map* les opérations sont divisées en plusieurs sous-opérations qui sont traitées indépendamment sur l'ensemble de la grappe de calcul. Les résultats de toutes les *map* sont ensuite agrégés par des tâches *reduce* pour au final donner un résultat.

La figure 1.6 [10] illustre le processus de traitement avec Hadoop MapReduce.

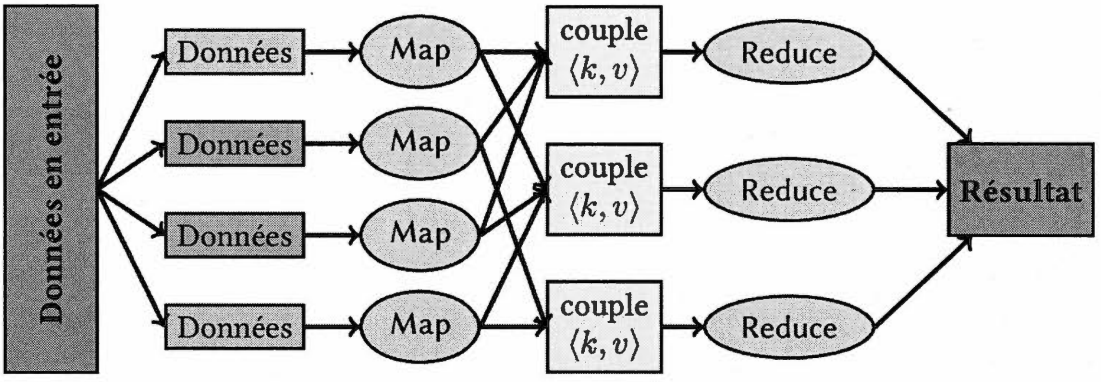

Figure 1.6: MapReduce

La programmation MapReduce est en parfaite adéquation avec HDFS et Ceph. En effet, ces systèmes conservent les données en morceaux, facilitant la distribution du calcul et le morcellement du traitement sur plusieurs nœuds. De plus, ces systèmes de stockage proposant par défaut la réplication des données, ils sont adaptés à l'accès concurrentiel, augmentant ainsi les performances du traitement.

Hadoop MapReduce n'est pas la seule solution de calcul distribué. Apache Spark est une solution de calcul distribuée proposant une sérieuse alternative à MapReduce.

La différence principale entre Apache Spark et Hadoop MapReduce n'est pas dans le modèle de programmation, mais dans la manière de gérer les données.

Les résultats intermédiaires des opérations *map*, effectuées au sein d'Hadoop MapReduce, sont stockées dans HDFS et donc, sur un disque. Apache Spark, qui utilise aussi les opérations *map* et *reduce*, se sert de la mémoire vive des nœuds de calcul pour stocker ces opérations. Cette mémoire étant beaucoup plus rapide

10. Illustration par Clém IAGL sous licence CC BY-SA 3.0 `https://commons.wikimedia.org/w/index.php?curid=22688163`

que les disques, la rapidité d'une tâche effectuée avec Apache Spark est souvent bien supérieure à Hadoop MapReduce.

La distribution des tâches avec Apache Spark est supportée par le concept de RDD (*Resilient Distributed Datasets*). La volatilité de la mémoire vive provoque la perte de toutes les données si le système est interrompu. Pour autant, ce n'est pas une faille à laquelle les RDD sont exposés.

En effet, les RDD peuvent être stockés en mémoire entre les requêtes sans nécessiter de réplication. Au lieu de cela, les données perdues sont reconstruites en utilisant le graphe de tâches nécessaire à sa création (Zaharia et al., 2012).

En fonction des évaluations de performance, la majorité des opérations sont entre 2 et 5 fois plus rapides avec Apache Spark (Shi et al., 2015).

Les données massives jouent un rôle essentiel en sciences et les architectures qui permettent de les traiter sont les supports primordiaux qui font avancer la recherche.

C'est le cas par exemple du CERN qui stocke et traite plus de 300 pétaoctets de données issues du LHC (Large Hardron Collider) à l'aide de technologies en code ouvert (Bell et al., 2015; Toor et al., 2012; Andrade et al., 2012).

1.2 Indexation et Recherche d'information

L'exploration de données massives impose l'utilisation d'infrastructures adaptées, que ce soit pour les logiciell comme le matériel.

Lorsque l'on s'intéresse à des données textuelles, l'un des meilleurs moyens de faire une exploration est de commencer par indexer ces textes à l'aide d'un moteur

d'indexation tel qu'Apache Lucene [11] (McCandless et al., 2010) en utilisant par exemple un moteur de recherche comme Apache Solr [12].

Un index est une représentation de données (dans notre cas des données textuelles) facilitant la récupération et la recherche d'informations. Cette représentation est souvent basée sur une table de hachage.

La représentation la plus simple d'un index de documents consiste à utiliser les mots comme les clés de la table de hachage, associés à une liste de documents qui contiennent ces mots.

Prenons par exemples les textes suivants :

"Le Labrador est la région continentale de la province canadienne de Terre-Neuve-et-Labrador."

"Le retriever du Labrador, plus communément appelé labrador retriever ou plus simplement labrador, est une race de chiens originaire du Royaume-Uni. C'est un chien de taille moyenne, à l'allure ronde et robuste, de couleur entièrement sable, chocolat ou noir. Issu du chien de Saint-John, la race a été importée puis développée au Royaume-Uni et au Canada. Le labrador est actuellement l'une des races les plus répandues dans le monde."

L'index de ces deux textes est représenté par le tableau 1.1

Pour arriver à cette structure [13], nous avons fait tout d'abord pré-traiter nos textes. Le protocole de pré-traitement est identique pour chaque texte.

---

11. `http://lucene.apache.org`

12. `http://lucene.apache.org/solr/`

13. code : `https://gist.github.com/antoine-briand/4c713892108fcecfe30b767eb3f61d05`

Tableau 1.1: Exemple d'index à partir des deux textes

Mot	**Documents**	...	**Mot**	**Documents**
labrador	Texte A, Texte B		originaire	Texte B
région	Texte A		royaume-uni	Texte B
continentale	Texte A		cest	Texte B
province	Texte A		taille	Texte B
canadienne	Texte A		moyenne	Texte B
terre-neuve-et-labrador	Texte A		lallure	Texte B
confondre	Texte A		ronde	Texte B
race	Texte A, Texte B		robuste	Texte B
chien	Texte A, Texte B		couleur	Texte B
retriever	Texte B		entièrement	Texte B
communément	Texte B		sable	Texte B
appelé	Texte B		chocolat	Texte B
simplement	Texte B		noir	Texte B
chiens	Texte B		issu	Texte B
saint-john	Texte B		importée	Texte B
développée	Texte B		canada	Texte B
actuellement	Texte B		lune	Texte B
races	Texte B		répandues	Texte B
			monde	Texte B

La ponctuation a été retirée, sauf pour le trait d'union. Cette étape est nécessaire pour construire un index afin que les mots Labrador et Labrador, (notez la présence de la virgule collée au mot) soient pris en compte comme le même mot. En effet, les systèmes d'indexation découpent les textes en mots en se basant sur un séparateur. Pour les langues comme le français, le séparateur le plus courant est

l'espace entre chaque mot.

Nous avons ensuite retiré les mots vides. Les mots vides sont les mots les plus fréquents dans une langue. Pour le français la liste se compose des "a", "et", "je", etc. Ces listes sont construites par des linguistes ou à l'aide d'algorithmes d'analyse de la distribution comme Zipfs Law (Manning et al., 1999). La suppression des mots vides est une technique reconnue pour améliorer les systèmes de recherche par mots-clés, car ils n'apportent pas de sens et peuvent même ajouter de la confusion (Silva et Ribeiro, 2003).

Enfin, nous avons transformé toutes les majuscules en minuscule, là aussi dans le but de ne pas traiter "Labrador" et "labrador" comme deux mots différents.

Ces étapes de prétraitement sont très communément adoptées, il en existe d'autres (Rajman et Besançon, 1998), comme la lemmatisation ou la racinisation. En recherche d'informations, la racinisation est souvent utilisée pour améliorer les systèmes de recherches.

Si l'on reprend l'exemple de notre index, les mots **canada** et **canadienne** font tous les deux référence au pays Canada. Différencier les deux comme nous l'avons fait peut réduire les performances d'un moteur de recherche. Si nous faisons une recherche avec l'une des deux formes, nous omettrions l'autre dans les résultats car ils seraient considérés comme deux mots différents. Avec des techniques de racinisation ces deux mots ne seraient plus qu'un sous la forme de **canad**.

Notre index une fois construit est donc consommé par un moteur de recherche, le plus souvent par mots-clés. Ces moteurs de recherches prennent une requête (une suite de mots-clés, une phrase) en entrée et l'utilisent pour retourner les documents pertinents.

Pour chercher des documents en rapport avec les mots clés "chien du Royaume-

Uni", à partir de notre index, on lui fait subir le même prétraitement (suppression des mots vides, de la ponctuation, tous les mots en minuscule). Le résultat de ce prétraitement est ensuite utilisé comme requête pour extraire les documents pertinents.

Une technique simple consiste à retourner tous les documents qui contiennent au moins un mot présent dans la requête. Le résultat ici serait tous les documents de l'index. Si nous souhaitons être plus précis, nous pouvons ajouter des critères à notre recherche.

Un élément important de la recherche d'informations est le classement des résultats. En effet, lorsque le système retourne des résultats, nous attendons les plus pertinents en premiers. Ce sont ces systèmes de classement qui permettent à Google, par exemple, de retourner le résultat recherché dans les cinq premiers de la première page. Il existe plusieurs techniques de classement, nous allons parler ici de deux très populaires, TF-IDF(*term frequency-inverse document frequency*) (Salton et McGill, 1986; Blei et al., 2003; Robertson, 2004) et Okapi BM25 (Büttcher et al., 2006).

Le principe de TF-IDF est d'utiliser une approche statistique pour évaluer la pertinence d'un terme (d'une requête) par rapport à la collection (l'index). Plus le terme est pertinent, plus il va peser dans le classement des documents. Les documents seront donc classés par ordre d'importance selon la pertinence des termes.

TF-IDF est composé de deux parties TF et IDF, qui pour le calcul d'un terme donné, peuvent être définis comme suit :

$$TF(t) = \frac{Nombre\ de\ fois\ que\ le\ terme\ t\ apparait\ dans\ un\ document}{Compte\ total\ de\ mots\ dans\ le\ document} \quad (1.1)$$

$$IDF(t) = \log(\frac{Nombre\ de\ documents}{Nombre\ de\ documents\ contenant\ le\ terme\ t}) \tag{1.2}$$

Ainsi TF-IDF peut être défini comme le produit de TF par IDF :

$$TFIDF(t) = TF(t) \cdot IDF(T) \tag{1.3}$$

La seconde fonction de classement, Okapi BM25 consiste à ordonner les documents en fonction de la fréquence des termes qui apparaissent dans chaque document, indépendamment des relations pouvant exister entre ces termes ou de leurs proximités relatives au sein du document. À partir d'une requête Q qui contient un certain nombre de mots clés $q_1...q_n$, le score BM25 du document D est :

$$score(D,Q) = \sum_{i=1}^{n} IDF \cdot \frac{f(q_i,D) \cdot (k_1+1)}{f(q_i,D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{avgdl})} \tag{1.4}$$

Avec $f(q_i, D)$ le TF dans le document D, $|D|$ la longueur du document D en mots et $avgdl$ la longueur moyenne des documents dans la collection de recherche. k_1 et b sont des paramètres libres utilisés en l'absence de techniques d'optimisation.

Ces systèmes d'indexation, de prétraitement et de classement sont tous disponibles dans la solution Apache Lucene, un moteur d'indexation en code ouvert. Apache Lucene est utilisé par de nombreux moteurs de recherche dont Apache Solr, un moteur de recherche complet et prêt pour la production. Un moteur d'indexation se concentre sur les tâches d'indexation et de recherche du texte tandis qu'un moteur de recherche s'appuie sur ce moteur d'indexation en plus de fournir une interface (graphique, API) et des systèmes de configuration pour permettre à l'utilisateur final de rechercher dans un ensemble de documents et d'utiliser d'autres fonctions comme l'expansion de requêtes.

Le processus de construction d'un moteur de recherche peut être réduit à 4 étapes : la construction d'un schéma d'indexation (basé sur les données à indexer), la sélection d'étapes de prétraitement des documents de l'index et des requêtes, l'indexation du texte, le test du moteur de recherche à l'aide de corpus de référence [14].

Le domaine de la recherche d'information supporte de nombreuses tâches dans différents domaines. La recherche de littérature en bio-informatique (Almeida et al., 2018)(Huang et Lu, 2015) par exemple, permet aux chercheurs de trouver des informations cruciales à leur travail au sein de larges corpus de données scientifiques.

Les techniques de recherche d'information sont également largement utilisées pour découvrir des connaissances dans le domaine de la santé mentale. Dans (Hammond et al., 2013), les auteurs présentent une étude pour soutenir le maintien de la santé mentale des soldats de l'armée américaine. L'objectif est d'aider les professionnels de la santé à effectuer un suivi efficace des soldats, le taux de suicide étant élevé dans cette population. L'approche a fait appel à la ressource VINCI (Veterans Informatics and Computing Infrastructure) afin d'extraire les informations de santé des vétérans. Ces informations sont non structurées sous la forme de notes cliniques. Les auteurs ont construit un moteur de recherche indexant ces données textuelles pour prédire le risque de tentative de suicide chez les soldats.

1.3 Annotation de textes

Les textes sont en grande majorité des données non structurées. Leur analyse nécessite donc de créer une structure minimale permettant d'extraire leurs com-

14. Les moteurs de recherches comme d'autres systèmes doivent être évalués. Pour ce faire il est courant d'utiliser la métrique *BLEU Score* (Papineni et al., 2002)

posantes intéressantes.

Une façon de faire est d'annoter les éléments d'intérêts d'un texte. L'annotation consiste à mettre en évidence un ensemble de caractères dans le but d'être exploité automatiquement par la suite.

L'annotation d'un texte peut être réalisée par un linguiste. Cette tâche étant longue et complexe, il est aussi possible d'annoter automatiquement des textes à l'aide de système à base d'apprentissage machine. La figure 1.7 [15] présente un exemple d'annotation de texte avec la solution libre Brat (Stenetorp et al., 2012).

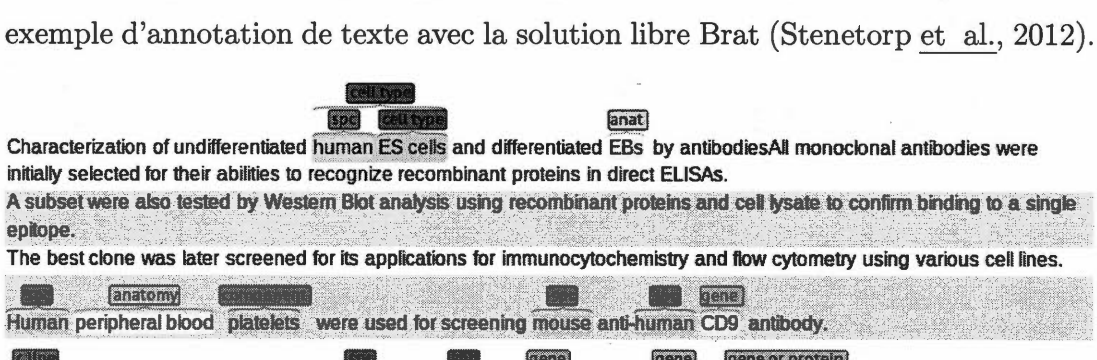

Figure 1.7: Exemple d'annotation de texte avec Brat

Il existe de nombreux systèmes d'annotation de texte mis à disposition de la communauté, en code source ouvert.

1.3.1 Stanford CoreNLP

Stanford NER (Finkel et al., 2005) est un système pour la découverte et l'annotation d'entités nommées qui utilise les modèles statistiques CRF (Lafferty et al., 2001) et une approche d'appariement de modèles. Les CRF sont une classe de

15. source : `http://brat.nlplab.org/examples.html`

méthodes statistiques utilisées dans l'apprentissage statistique, souvent utilisées dans les tâches de traitement de la langue naturelle, dont l'annotation de textes.

Cet outil fait partie de la boîte à outils Stanford CoreNLP (Manning et al., 2014), contenant de nombreuses fonctions et systèmes d'analyse et traitement de la langue naturelle.

Des modèles [16] préentrainés sont disponibles en plusieurs langues.

1.3.2 NeuroNER

NeuroNER (Dernoncourt et al., 2016; Dernoncourt et al., 2017) est un autre système d'annotation d'entités basé sur des techniques d'apprentissage profond, à l'aide de réseaux de neurones récurrents (RNN - *Recurrent neural network*). Ce système est lui aussi en code source ouvert [17] et propose un certain nombre de modèles préentrainés sur plusieurs corpus.

Bien que les résultats publiés par les auteurs soient très prometteurs, nous n'avons pu les reproduire pour tous les corpus. Par exemple le corpus MIMIC V3 n'est pas disponible dans la même version qu'utilisée par les auteurs [18].

---

16. `https://nlp.stanford.edu/software/CRF-NER.shtml#Models`

17. sources `https://github.com/Franck-Dernoncourt/NeuroNER`

18. Une version déidentifiée est disponible, mais n'est pas celle fournie au public. Après contact avec l'auteur il s'avère que le corpus est la propriété du MIT et qu'il ne peut être transféré actuellement.

CHAPITRE II

PROPOSITION DE RECHERCHE

2.1 Problématique

Comme évoqué dans l'introduction de ce mémoire, l'usage des données massives, bien que fort intéressant économiquement et stratégiquement, représente d'importants défis pour une organisation.

L'objectif de ce mémoire est de proposer un ensemble de méthodes et processus offrant un support pour créer un système d'exploration et d'exploitation de données massives, dans un cadre libre (à code source ouvert), avec un environnement maîtrisé de bout en bout et dont l'accessibilité et la maintenance sont simplifiées et évolutives.

La question de recherche principale se résume comme suit :

> Étant donné un ensemble de données massives, de nature textuelle, quelles sont les étapes, technologies et architectures les mieux adaptées afin de procéder à leur exploration et exploitation ?

Une expression visuelle de cette problématique comme illustrée par la figure 2.1 implique de dévoiler et détailler toutes les étapes entre la récupération de données brutes et l'extraction de valeur qui en découle.

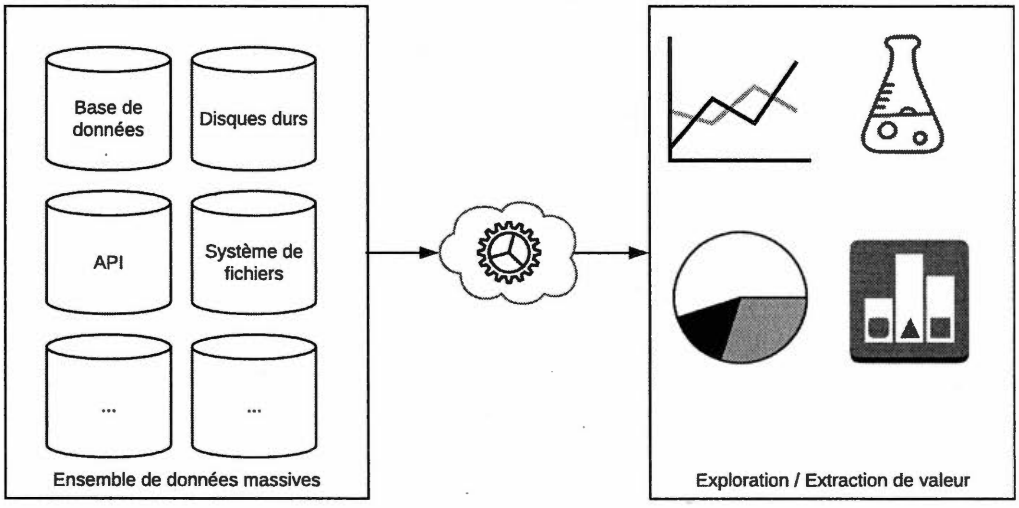

Figure 2.1: Illustration de la problématique

Nous répondrons à cette question en deux temps.

Premièrement, nous allons établir une infrastructure matérielle et logicielle permettant de supporter des tâches d'exploration et d'analyse de données massives. La variété de données étant très importante, notre contexte se limitera prioritairement au traitement de données textuelles, toutefois l'approche se veut extensible à tous types de données.

Cette infrastructure devra apporter des solutions aux problèmes posés par :

— La diversité des sources de données
— L'hétérogénéité des données
— Le volume et la vélocité variable en fonction des sources de données
— Les spécificités des données massives textuelles
— Les spécificités de certaines sources de données

Nous allons aussi tenter de répondre aux impératifs techniques induits, à savoir :

— La maintenance nécessaire à l'infrastructure
— La consistance de la configuration des services mis en place
— Le passage à l'échelle supérieure ou inférieure en fonction des besoins

L'infrastructure finale est illustrée par la figure 2.2. Celle-ci comporte une interface unifiée pour l'accès aux différentes sources de données, un environnement de calcul complet basé sur un système d'infrastructure en tant que service et différents services/systèmes. Tous les organes logiciels seront déployés et maintenus à l'aide d'infrastructure en tant que code, une technique consistant à décrire l'infrastructure logicielle avec un langage de description dont la syntaxe unique facilite la gestion des versions et les modifications à l'échelle de la grappe de calcul. Le tout devra être supporté par le matériel et les interconnexions adéquats.

Cette infrastructure sera ensuite testée et exploitée dans deux cas d'application de données massives textuelles.

2.2 Détection d'entités d'intérêts au sein de corpus de textes

Le premier exemple d'application s'intéresse à la protection de la vie privée et au respect des législations (*compliance*) comme PIPEDA, RGPD, HIPAA ou encore Privacy Act. Cette application a été réalisée en partenariat industriel, dans le cadre d'un stage Mitacs avec la société Netmail.

Cette application a été réalisée en lien avec l'activité de notre partenaire industriel qui se charge de l'archivage de données massives pour ses clients. Le but est d'aider la création d'un produit de détection de fuites de données a posteriori, au sein des archives de leurs clients.

Le stockage de masse et l'utilisation de nombreux services connectés (messageries, réseaux sociaux, communication en direct, ERP(*enterprise resource planning*),

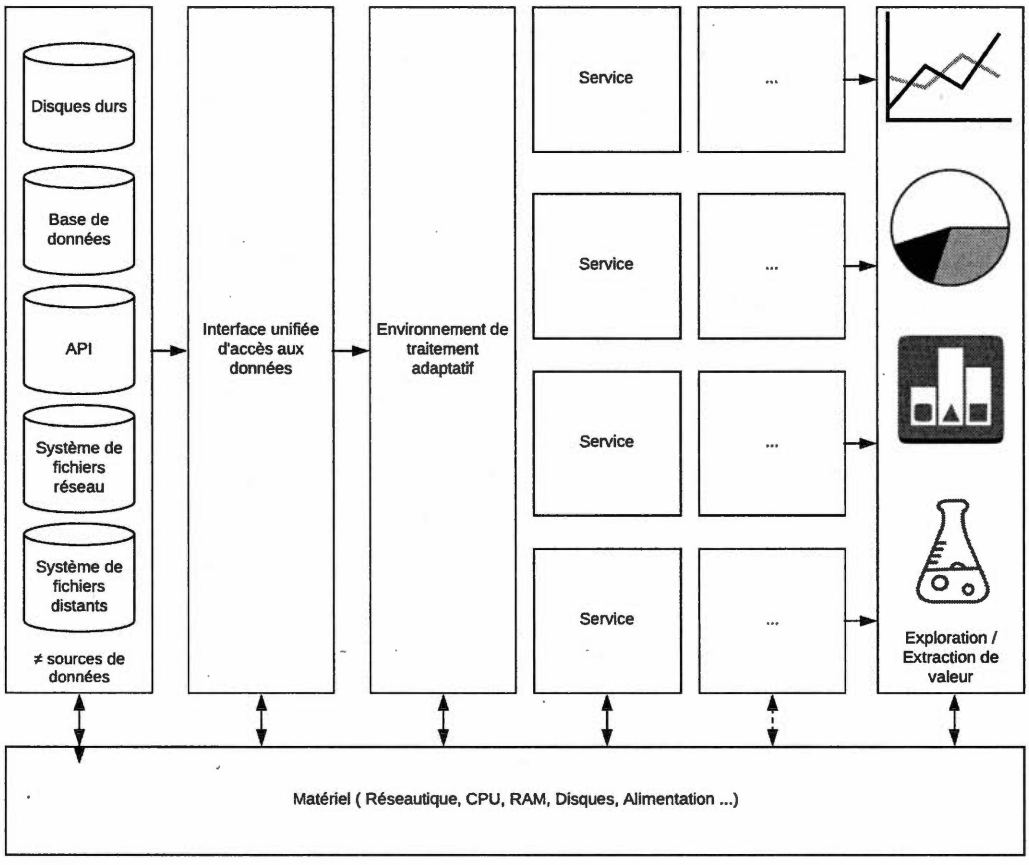

Figure 2.2: Illustration infrastructure complète

stockage infonuagique (*cloud*) ...) créent de nouveaux défis pour la compagnie en ce qui concerne la gestion de la conformité et plus particulièrement en termes de fuite d'informations sensibles.

La définition de l'information sensible diffère selon le domaine d'activité concerné.

Dans le cas d'une entreprise, les données telles que le courriel, les adresses, les numéros de carte bancaire ou les informations sur les différents types de montants (financiers, transactionnels, etc.) sont considérées comme sensibles.

En ce qui concerne les données médicales, il est nécessaire d'identifier des entités spécifiques, telles que le nom de la maladie, le diagnostic, le numéro de médecin ou le numéro d'assurance-maladie. La reconnaissance d'entités nommées est une tâche du domaine du Traitement Automatique de la Langue Naturelle, dans laquelle on identifie des objets du monde réel qui sont désignées par un nom (propre) ou par un terme appartenant à un champ lexical spécifique (Nadeau et Sekine, 2007).

Au sein d'une organisation, les questions qui se posent sont donc les suivantes :

> Comment pouvons-nous savoir si un échange non autorisé survient au sein de notre organisation ? Quand allons-nous le savoir ? Combien de temps peut s'écouler avant la détection ?

Dans notre cas, pour répondre à ces questions, nous souhaitons tout d'abord identifier les données sensibles à l'aide d'outil d'annotation de documents, dans un contexte de données massives.

Il faut différencier deux cas : d'un côté, les données statiques, ou à vélocité faible (archives, données qui arrivent par lot, etc.) et de l'autre les flux de données en temps réel (courriels, discussions, documents sur un espace de partage, etc.).

Dans le cas des données d'archives, les mesures qui peuvent être prises sont correctives. En effet, les archives de données sont créées au bout d'un laps de temps (semaines, mois, années) défini par l'organisme qui gère les données. Aussi, la détection des fuites de données au sein d'archives ne peut arriver en temps réel, mais au plus tôt au moment de la création de l'archive.

Toutefois cette tâche reste cruciale, car la détection de fuite de données a posteriori est très importante pour le respect des réglementations. Il faut être capable d'apporter les éléments de preuve de la fuite.

Dans le cas des données sur un système en ligne (flux de données), le système de détection de fuites devrait être bien plus réactif. Par exemple, si l'on envoie un courriel à une personne avec des coordonnées bancaires et que ce destinataire n'est pas censé le recevoir, ou si le canal utilisé n'est pas sécurisé, il faut que le système soit capable d'intervenir rapidement.

Certains systèmes d'entreprise proposent des fonctionnalités de prévention, basées sur des règles simples.

Par exemple, le système Google GSuite, qui propose entre autre Gmail pour l'entreprise, demande une confirmation à l'utilisateur si celui-ci essaie de joindre une personne dont l'adresse courriel ne fait pas partie de son organisation [1].

Lorsque l'on cherche à classer des informations avec un système automatique, il est important d'évaluer son système. La décision de classification, par exemple "donnée sensible" ou "donnée non sensible" peut être évaluée en termes de véracité : vrai positif, faux positif, vrai négatif, faux négatif.

Si l'on prend l'exemple d'un classement binaire de documents qui seraient a risque ou non, avec les classes "risque" et "non-risque", les évaluations de classification peuvent être définies ainsi :

— Vrais positifs (*True positives*, TP) : le nombre de documents correctement classés "risque".
— Faux positifs (*False positives*, FP) : le nombre de documents incorrectement classés "risque".
— Vrais négatifs (*True negatives*, TN) : le nombre de documents correctement classés comme "non-risque"

---

1. `https://gsuiteupdates.googleblog.com/2017/05/gmail-unintended-external-reply-warnings.html`

— Faux négatifs (*False negatives*, FN) : le nombre de documents incorrectement classés comme "non-risque"

Les règles basées sur des bases de connaissance (est-ce que mon destinataire fait partie de mon organisation ?) sont simples à produire, mais créent de nombreux faux positifs et ne couvrent qu'une infime partie de la problématique.

L'analyse du contenu du courriel est plus compliquée. Elle est pourtant indispensable si l'on souhaite développer des méthodes pour éviter d'envoyer un message contenant des informations sensibles à un tiers, s'assurer que le partage de fichier sur le nuage (*cloud*) est autorisé, et prévenir l'utilisateur le plus rapidement possible en cas de problème.

Le système mis en place pour répondre à ces questions s'appuie sur l'infrastructure définie précédemment. Nous verrons comment celle-ci impacte grandement l'exécution du système de détection et la vélocité avec laquelle les détections peuvent être opérées avec un système au coût maîtrisé et à la reproductibilité totale.

L'entreprise partenaire archive les données de ses clients dans de larges entrepôts de données. Elle aimerait être capable de fournir des services basés sur ces archives et son premier objectif est de détecter des entités d'intérêt (informations sensibles, de santé, etc.).

Afin d'atteindre cet objectif, nous avons réalisé les étapes suivantes :

1. Recherche de corpus de travail adéquat
2. Création d'un ensemble d'annotateurs d'entités d'intérêt
3. Distribution de l'architecture d'annotation
4. Indexation des textes dans un index distribué à l'aide d'Apache Solr Cloud.
5. Création d'un processus d'annotation distribué avec Apache Spark.

2.3 Détection de la dépression dans les médias sociaux

La seconde application que nous présenterons est un système de détection précoce de la dépression des utilisateurs de réseaux sociaux.

La fréquentation et la publication sur ces réseaux sont de plus en plus importantes et les usagers de ces plateformes confient beaucoup de leurs problèmes, angoisses, etc.

Grâce à cela, la quantité d'information disponible publiquement est phénoménale, donnant la possibilité à des communautés spécialisées de se créer et aux utilisateurs d'obtenir du soutien directement sur les médias sociaux comme reddit.

Toutefois, le nombre de messages postés sur ces médias étant en constante croissance, il devient de plus en plus compliqué pour les modérateurs (médecins, assistants) de répondre à tout le monde et d'accorder l'attention nécessaire à chaque utilisateur, en fonction de sa situation.

En effet, les usagers, en fonction de leurs publications, n'auront pas tous les mêmes besoins. Par exemple, les utilisateurs qui publient des messages dans lesquels ils expliquent qu'ils pensent à attenter à leurs jours devraient être traités en priorité par rapport à des utilisateurs qui demandent conseil sur le plus bel endroit pour aller faire de la randonnée.

Aussi l'intérêt de classer, le plus rapidement possible, un usager en danger de dépression ou non est une application critique des travaux de traitement du langage naturel. Des tâches internationales comme eRisk[2] (CLEF) ou CLPsych[3]

---

2. `http://early.irlab.org/`

3. `http://clpsych.org/`

regroupent ainsi les chercheurs de la planète à la recherche d'une solution la plus efficace à ce problème.

L'enjeu final est différent de la première application présentée, mais les défis en termes de données massives restent les mêmes. Aussi nous présenterons notre approche qui mêle apprentissage machine et recherche d'information, le tout supporté par l'infrastructure établie au préalable.

Tout au long de ces travaux, nous avons publié sept articles dans différentes conférences. Nous reviendrons sur ces publications lorsque nous aborderons les cas d'applications en rapport.

CHAPITRE III

INFRASTRUCTURE DE CALCUL ET STOCKAGE DE MASSE

3.1 Matériel

L'exploration de données massives implique de disposer d'une infrastructure répondant aux exigences de ce type de données. Cette infrastructure est composée de deux éléments : du matériel et des logiciels.

Les composants matériels nécessaires à toute infrastructure de calcul de données massives sont :

- Le stockage (disques, bandes)
- Le réseau (commutateurs, routeurs, câbles)
- La capacité de calcul (processeurs/mémoire vive)
- L'alimentation électrique

Afin de supporter nos travaux, nous avons construit une infrastructure de calcul à base de récupération et de neuf. Les critères sont les suivants :

- Capacité de stockage suffisante pour accueillir nos données
- Capacité réseau pour effectuer les expériences et gérer la grappe de calcul
- Nombre de processeurs et quantité de mémoire vive permettant de paralléliser les tâches

— Plusieurs nœuds de calcul pour distribuer la charge de travail

— Au moins deux nœuds de stockage pour assurer la redondance des données

La grappe de calcul ainsi créée est présentée en figure 3.1.

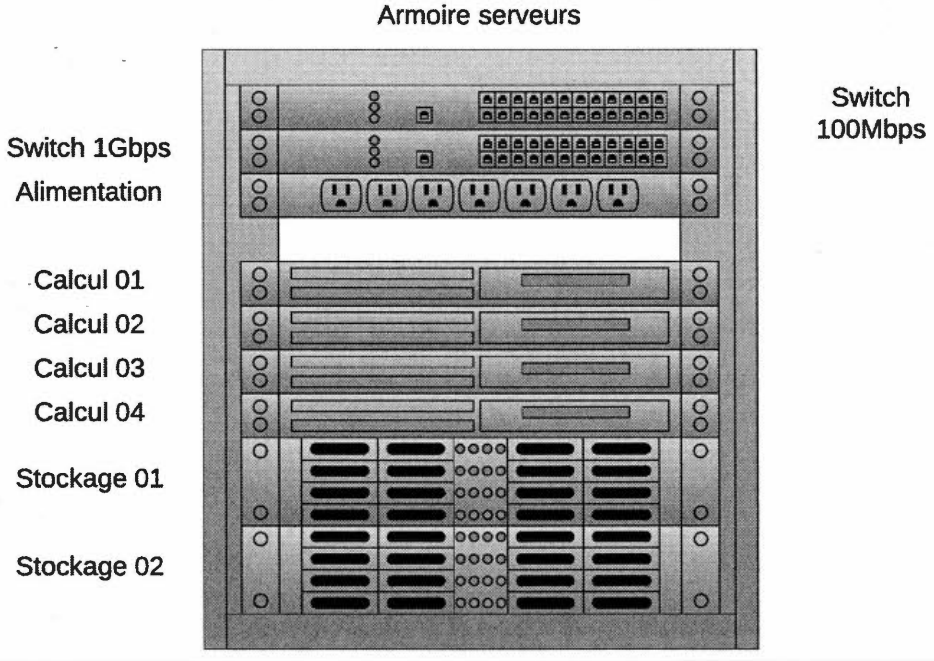

Figure 3.1: Architecture de traitement

Cette grappe est composée de deux réseaux, quatre unités de calcul et deux unités de stockage.

Le tableau 3.1 présente les ressources disponibles dans notre architecture de test.

Nous avons fait le choix de deux capacités de réseaux. La première a 100 mégas par seconde et l'autre a 1 giga (1000 mégas) par seconde.

L'intérêt de ces deux réseaux est de différencier la charge de calcul de la charge utile (accès à internet, mise à jour du *cluster*, etc).

Tableau 3.1: Architecture de traitement - Ressources disponibles

Caractéristique	**Ressource disponible**	**Total**
Réseau	16 * 1000 Mbps (1 Gbps) 24 * 100 Mbps	
Stockage de masse	16 * 415 GB	~7 TB
Processeurs	4 * 24	96
RAM	4 * 48	192 GB
Stockage local	4* 140 GB (RAID 1)	560 GB

Au total, le réseau dédié aux tâches de calcul et de stockage est de 10 fois 1Gbps symétrique (1Gb/s montant et 1Gb/s descendant). La répartition de la capacité du réseau est présentée dans le tableau 3.2.

Tableau 3.2: Répartition de la capacité réseau

Machine	**Capacité réseau (calcul)**	**Capacité réseau (stockage)**
calcul-01	1 Gbps symétrique	1 Gbps symétrique
calcul-02	1 Gbps symétrique	1 Gbps symétrique
calcul-03	1 Gbps symétrique	1 Gbps symétrique
calcul-04	1 Gbps symétrique	1 Gbps symétrique
stockage-01	N.A	1 Gbps symétrique
stockage-02	N.A	1 Gbps symétrique
Total	**4 Gb/s symétrique**	**6 Gb/s symétrique**

Différencier les réseaux de calcul et de stockage nous permet d'avoir une plus grande capacité pour les tâches les plus coûteuses.

Il existe plusieurs techniques pour augmenter la capacité d'un réseau en exploitant plusieurs ports, l'agrégation de liens. Le tableau 3.3 présente chaque méthode.

Tableau 3.3: Différents types d'agrégation

Mode	**Description**
balance-rr	Équilibrage de charge et tolérance aux pannes avec la répartition du trafic effectué par l'algorithme RoundRobin. Le principe consiste à envoyer les paquets par le premier port disponible.
active-backup	Ce mode est destiné à éviter les pannes. Le trafic ne passe que par un port. Si celui-ci ne fonctionne plus, le second port prend le relais.
balance-xor	Le trafic est équilibré en fonction du récepteur à l'autre extrémité.
broadcast	Tous les paquets sont transmis par tous les ports.
802.3ad	Créé des groupes d'agrégation de ports ayant la même vitesse. Comparable à balance-xor.
balance-tlb	Le trafic sortant est réparti en fonction de la charge de chaque port. Si le trafic entrant échoue, le port de réception défaillant est remplacé par un autre port.

Le réseau public, permettant aux machines d'accéder à internet, est de 100 Mbps partagé entre toutes les machines. Ceci est une contrainte imposée par le réseau de l'université. Ne servant qu'à la maintenance de la grappe de calcul, l'impact de cette limitation est minimal. Les tâches de maintenance se limitent à la mise à jour hebdomadaire des nœuds de la grappe, et les accès distants pour les utilisateurs. L'impact de cette limitation de débit sera plus important si un utilisateur souhaite rapatrier beaucoup de données au travers du réseau public. Dans ce cas, il faudrait augmenter la capacité du réseau public, soit en utilisant une agrégation de liens (plusieurs liens à 100Mb/s utilisés en parallèle), soit en disposant d'un lien plus rapide.

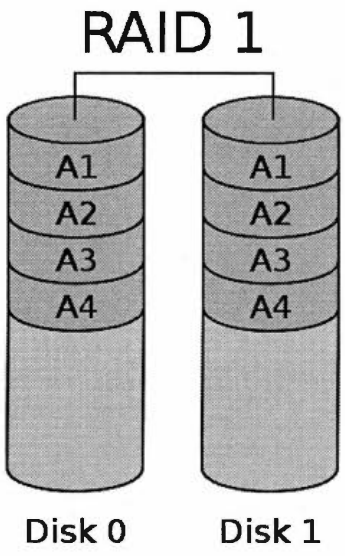

Figure 3.2: Illustration de la technologie RAID niveau 1

Nous avons aussi deux types de stockage, le stockage de masse et le stockage local.

Le stockage de masse est distribué, nous permettant d'emmagasiner plusieurs téraoctets de données. Le stockage local, lui, sert au système d'exploitation des serveurs et sera aussi utilisé comme cache local pour accélérer les opérations sur les données massives.

Le stockage local utilise la technologie RAID (*Redundant Array of Independent Disks*) de niveau 1. Les données sont stockées sur deux disques en parallèle.

Ainsi, si un disque venait à tomber en panne, le second continuerait de fonctionner seul. Une fois le disque défectueux remplacé, le contenu est de nouveau copié. Cette technologie renforce la tolérance aux pannes de notre infrastructure.

L'illustration 3.2 présente la technologie RAID 1 [1].

1. source `https://fr.m.wikipedia.org/wiki/Fichier:RAID_1.svg`

En termes de calcul, cette infrastructure dispose de 96 cœurs et 192 Gb de mémoire vive, répartis sur quatre nœuds, nous permettant de paralléliser et distribuer toutes les opérations.

Chaque serveur est installé avec Ubuntu Server 16.04 LTS, ce choix est fait par rapport aux prérequis du système OpenStack, détaillé en partie 3.2.

Tableau 3.4: Architecture de traitement - Coût

Matériel	**État**	**Marque / Modèle**	**Prix unitaire**
Serveurs de calcul	Occasion	HP ProLiant DL360 G6	480 $CAD
Serveurs de stockage	Occasion	HP ProLiant SE1220	660 $CAD
Commutateur (*Switch*) 1GBps	Neuf	TP-Link TL-SG1016D	80 $CAD
Commutateur 100 Mbps	Neuf	TP-Link TL-SG1024	130 $CAD

Le tableau 3.4 présente une estimation du coût de cette infrastructure.

À noter que les serveurs listés ne sont plus vendus par le constructeur et que ces tarifs sont ceux que l'on trouve sur eBay pour le même modèle ou équivalent.

À cela, il faut ajouter du matériel outil et des câbles Ethernet. Le coût total du matériel pour ce type d'infrastructure est de 3500$ CAD.

Le matériel représente les fondations de notre infrastructure. Afin d'exploiter tous ces composants comme une seule grappe de calcul, nous allons avoir besoin de gérer le réseau, les processeurs, la mémoire vive, le stockage, la surveillance et la réparation automatique, etc.

Afin de ne pas noyer l'utilisateur avec toutes ces tâches, nous allons nous appuyer sur les systèmes d'infrastructure en tant que service. Ceux-ci permettent la gestion et le passage à l'échelle de grappes de calcul tout en automatisant la plupart des tâches courantes.

3.2 IaaS - Infrastructure en tant que service

Notre infrastructure se veut évolutive et facile à maintenir. Dans cette optique, nous souhaitons tirer partie de l'infrastructure en tant que service IaaS (*Infrastructure as a Service*).

L'intuition est la suivante : Si l'on devait effectuer nos travaux directement sur le système d'exploitation des machines physiques, on se retrouverait à installer de nombreux services avec des configurations et une utilisation de ressources différentes. Très vite, ce genre de pratique nous amènerait à sans cesse effacer et réinstaller les serveurs devenus instables. Ce serait une perte de temps considérable. De plus, cela ne nous permettrait pas de gérer les ressources de calcul et de stockage finement, comme on pourrait le faire avec des machines virtuelles.

L'infrastructure en tant que service, au contraire, repose sur un ensemble de services qui gèrent et réservent un ensemble de ressources sur demande, via une interface de gestion ou une interface de programmation.

Les ressources qui sont habituellement gérées sont :

- Les machines virtuelles
- Les images systèmes
- Les disques
- Les réseaux virtuels (sous-réseaux, parefeu, routeurs, IP flottantes [2])
- Les utilisateurs
- Les quotas
- Le stockage de masse (ou d'objets)

---

2. Une adresse IP, souvent publique, assignée dynamiquement à une machine et pouvant être facilement déplacée

— Les types de stockage

Ces systèmes IaaS sont souvent sous la forme de multiples-projets qui sont individuellement capables de gérer un type de ressources à la fois, et qui sont coordonnés par une API commune.

Il existe plusieurs projets en code ouvert qui permettent de créer une infrastructure en tant que service, les plus populaires sont :

— Proxmox VE [3]
— oVirt [4]
— Apache CloudStack [5]
— OpenStack [6]

Bien que les possibilités offertes par ces quatre projets soient comparables, **Proxmox VE** et **oVirt** sont plus adaptés à de petites infrastructures ou un laboratoire local. Pour Proxmox VE, cette limitation est principalement due à son architecture très monolithique, empêchant un passage à l'échelle efficace. Pour oVirt et Proxmox, bien qu'actifs, ces projets ne sont pas autant supportés par la communauté que leurs rivaux. Enfin, l'absence de support de l'industrie a fait que ces projets sont moins populaires qu'OpenStack ou CloudStack.

OpenStack et **Apache CloudStack**, au contraire, de par leur architecture orientée services, offrent une flexibilité et une capacité de passage à l'échelle qui s'étend du petit laboratoire, à la distribution planétaire de plusieurs centres de données.

---

3. `https://pve.proxmox.com/wiki/Main_Page`

4. `https://ovirt.org/`

5. `https://cloudstack.apache.org/`

6. `https://www.openstack.org/`

RackSpace, un fournisseur majeur dans le domaine du *cloud computing* opère ses centres de données via OpenStack - `https://www.rackspace.com/openstack` - C'est aussi le cas d'OVH principal hébergeur Européen `https://www.ovh.com/fr/public-cloud/instances/technologies/`.

Notre choix s'est porté sur OpenStack. Fondé par la NASA et RackSpace, et maintenant maintenu par de nombreuses sociétés dont RedHat et IBM, OpenStack a en effet l'avantage d'avoir une communauté et une activité très nettement supérieure à CloudStack (voir le graphique de comparaison des intérêts de recherche 3.3). De plus, le support de CloudStack par les outils d'infrastructure en tant que code (dont nous parlerons en partie 3.3) est bien souvent inexistant.

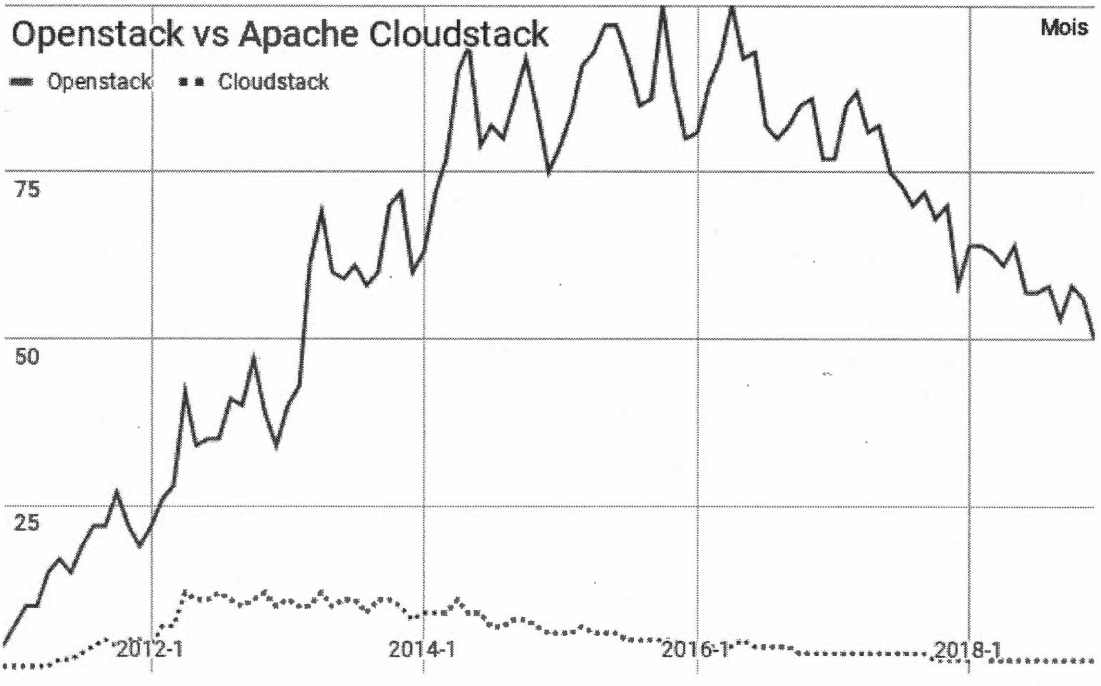

Figure 3.3: Occurences de recherche de OpenStack et de Apache Cloud Stack pour la période 2012 à 2018

La figure 3.3 représente les intérêts de recherches comparant le terme "OpenStack" au terme "CloudStack" dans le monde entier entre le 01/11/2010 et le

01/11/2018[7].

En raison de sa popularité, OpenStack propose beaucoup de services. La figure 3.4 présente une grande partie des services officiels[8].

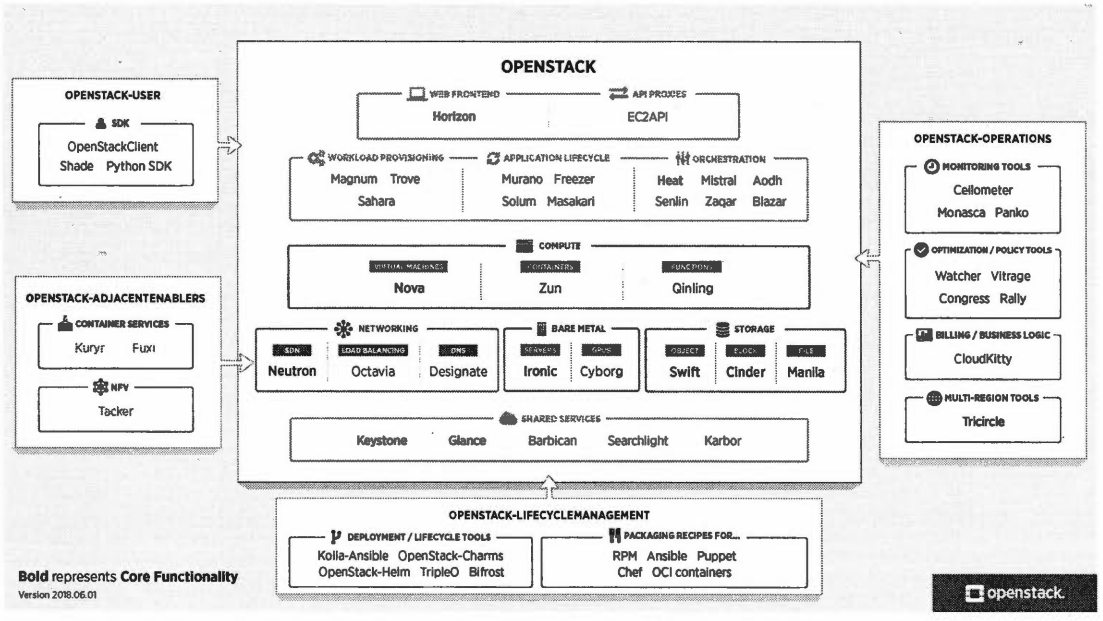

Figure 3.4: Cartographie des services OpenStack

L'architecture orientée services employée permet d'ajouter de nouvelles fonctionnalités en fonction des besoins, évitant d'installer la totalité des services dès le départ. Dans notre cas nous allons nous concentrer uniquement sur les services obligatoires (*core services*). Le tableau 3.5 fait état des services déployés pour notre infrastructure.

Tous ces services sont des programmes écrits en Python et qui communiquent via

---

7. `https://trends.google.fr/trends/explore?date=2010-01-11%202018-01-11&q=OpenStack,CloudStack`

8. source `https://www.openstack.org/assets/software/projectmap/openstack-map.pdf`

Tableau 3.5: Service OpenStack déployés

Service	**Rôle**
Nova	Gestion et planification des machines virtuelles
Neutron	Résautique
Cinder	Stockage d'objets
Glance	Stockage des images système
Keystone	Authentification
Horizon	Interface web

le protocole HTTP, sous la forme d'API REST. Il existe un certain nombre de pré-requis qui sont communs à l'ensemble des services et certains qui sont spécifiques à certains services.

Nous avons fait le choix de ne pas utiliser Swift comme système de stockage.

Swift est le système de stockage d'objets d'OpenStack. Le stockage d'objets est indiqué pour stocker des données sur une grappe de stockage distribué. A la manière de HDFS, les données sont stockées en ensembles de taille fixe sur plusieurs nœuds (redondance).

Nous avons fait le choix d'utiliser Ceph comme support de stockage. Entièrement compatible avec OpenStack, Ceph est, de par son architecture, indiqué pour les grappes de calcul monosites qui nécessitent de stocker des données consistantes (machines virtuelles, bases de données). Avec l'utilisation de CRUSH, les données stockées dans un cluster Ceph sont de suite disponibles sur l'ensemble de la grappe. L'accès aux données est lui aussi distribué à l'aide du concept des OSD. Les OSD (Object Storage Device) sont les disques qui servent à stocker les données. Chaque support de stockage est un OSD pour Ceph et chaque OSD est géré par un démon autonome qui est chargé de stocker les données, de les répliquer ou de

les redistribuer en cas de défaillance d'un équipement. Ce concept offre à Ceph des capacités de tolérance aux pannes bien meilleur en comparaison de Swift qui utilise un accès unique aux données.

Pour des infrastructures de plus grandes tailles, il est intéressant d'utiliser Swift. La consistance des données est moins évidente qu'avec Ceph, mais les performances sur de très importants volumes de données sont meilleures. Dans un environnement de production à large échelle, les deux systèmes devraient coexister pour servir deux cas d'utilisations différents. Ceph pour les machines virtuelles et tout support nécessitant de la consistance et une tolérance aux pannes accrue et Swift pour le stockage de fichiers de très grandes tailles.

Les prérequis sont : une base de données (PostgreSQL ou MySQL), un gestionnaire de file d'attente de messages (RabbitMQ), un serveur de cache (Memcached) et un service de base de données clé/valeur (etcd), python et un ensemble de librairies python.

Ces prérequis peuvent être installés sur les nœuds de calcul, ou des nœuds dédiés (pour de plus grands déploiements). Dans notre cas, nous avons décidé d'installer tous ces services sur les nœuds **calcul-01** et **calcul-02** afin de supporter une éventuelle panne.

Chaque serveur de calcul qui va exécuter des machines virtuelles doit avoir un hyperviseur installé. Un hyperviseur est un système qui permet de faire cohabiter plusieurs systèmes d'exploitation sur une même machine. Il en existe plusieurs, mais le choix le plus courant et le mieux supporté par OpenStack est **KVM**[9].

Tous les serveurs doivent aussi être synchronisés sur le même serveur de temps, et les noms d'hôtes de chaque serveur doivent être résolus par DNS sur l'ensemble

9. `https://www.linux-kvm.org/page/Main_Page`

du réseau.

Une fois tous ces prérequis en place, il est possible d'installer les services. Sous Ubuntu ou RedHat/CentOS cette installation peut se faire à l'aide du gestionnaire de paquets du système. Il est aussi possible de procéder à l'installation depuis les sources. Toute la procédure d'installation est très détaillée. Pour autant, l'installation et la maintenance de ce genre de système multiservice, multibases de données, multiréseau ... peuvent vite devenir un cauchemar.

Chaque machine, chaque service doit exécuter la même configuration. La moindre modification nécessite de redémarrer ou recharger un ensemble de services. Au final l'utilisation d'OpenStack, comme ceci, est presque plus contraignante que d'utiliser les systèmes "à même le métal".

C'est le reproche principal que l'on peut faire à OpenStack. C'est une plateforme idéale pour quiconque veut mettre en place une infrastructure en tant que service, compatible et évolutive, mais son déploiement et sa maintenance restent compliqués.

À cause de cette faiblesse, des projets ont émergé afin de faciliter le déploiement et la maintenance OpenStack.

Le premier, DevStack [10] est un script qui va installer tout le cluster sur une seule machine, automatiquement. Il suffit de télécharger le script, de changer quelques paramètres au besoin et au bout de plusieurs minutes (ou heures, en fonction de la machine), on se retrouve avec un système OpenStack parfaitement fonctionnel. Cette solution, comme son nom l'indique, est faite pour les développeurs et les personnes qui souhaiteraient évaluer OpenStack sur un environnement restreint. Il est fortement déconseillé d'utiliser DevStack en production.

---

10. `https://docs.openstack.org/devstack/latest/`

Bien qu'il existe beaucoup d'autres projets d'automatisation du déploiement d'OpenStack, nous allons nous concentrer sur deux des plus aboutis : OpenStack Ansible [11] et OpenStack Kolla [12]. Ces projets permettent de tirer parti du principe d'infrastructure en tant que code afin d'effectuer toutes les tâches nécessaires à l'installation des prérequis et des services qui composent OpenStack, de manière organique et automatique.

Dans cette partie, nous avons vu ce que sont les infrastructures en tant que service ainsi qu'un ensemble d'approches envisageables pour notre projet. OpenStack est la technologie retenue pour sa très grande communauté et sa grande compatibilité matérielle. Le déploiement et la maintenance manuelle de cette technologie sont fastidieux et peu recommandés pour les systèmes de production. Le besoin de synchroniser plusieurs nœuds en fonctions de leurs rôles et de leurs configurations implique une gestion centralisée et un autre modèle de déploiement. C'est à ces problèmes que répond le concept d'infrastructure en tant que code.

3.3 Iac - Infrastructure en tant que code

L'infrastructure en tant que code (*IaC - Infrastructure as Code*) consiste à déployer de l'infrastructure à partir d'un langage de description. Il est possible de faire cela à l'aide de simples scripts bash ou python, toutefois le concept IaC propose aujourd'hui des outils plus robustes et consistants. Nous allons nous intéresser à trois familles d'outils :

1. Les systèmes de configuration
2. Les systèmes de *provisioning*

---

11. `https://docs.openstack.org/openstack-ansible/latest/`

12. `https://wiki.openstack.org/wiki/Kolla`

3. Les systèmes à base de conteneurs

Les projets de systèmes configuration et de *provisioning* sont souvent confondus. Cette confusion est causée par les outils qui sont souvent capables de faire les deux tâches.

Les systèmes de *provisioning* sont responsables de la création des ressources à partir d'OpenStack (ou toute autre infrastructure en tant que service). C'est à partir de ces systèmes que l'on décrit les ressources réseaux, disques, processeurs, mémoire vive ... dont nous allons avoir besoin.

Le projet le plus populaire en termes de systèmes de *provisioning* est Terraform [13]. C'est un outil libre, développé par la société HashiCorp [14]. Il existe une version entreprise qui fournit quelques options supplémentaires.

Le langage de description utilisé est lui aussi développé par HashiCorp et s'appelle HCL (*HashiCorp Configuration Language*). Cette société développe exclusivement des produits pour la création d'infrastructure en tant que code, à grande échelle, et a ainsi développé son propre langage de configuration (libre) afin de l'intégrer à l'ensemble de leurs projets.

Voici un exemple de code pour provisionner une instance de machine virtuelle via OpenStack, depuis Terraform :

```
provider "openstack" {
  user_name   = "admin"
  tenant_name = "admin"
  password    = "pwd"
```

13. https://www.terraform.io/

14. https://www.hashicorp.com/

```
  auth_url     = "http://ikbig.info.uqam.ca:5000/v2.0"
  region       = "PresidentKennedy"
}

resource "openstack_compute_instance_v2" "serveur1" {
  name            = "serveur1"
  image_id        = "ad091b52-742f-469e-8f3c-
     fd81cadf0743"
  flavor_id       = "3"
  key_pair        = "my_key_pair_name"
  security_groups = ["default"]

  network {
    name = "defaukt"
  }
}
```

Listing 3.1: Création d'une instance OpenStack avec Terraform

Tout d'abord, il faut configurer ce que Terraform appelle un *provider*. C'est tout simplement le fournisseur de la ressource que nous souhaitons créer. Dans notre cas, nous utilisons OpenStack. Les informations *user_name, tenant_name, password, auth_url* et *region* permettent d'expliquer à Terraform comment se connecter aux API de notre grappe OpenStack. La seconde ressource *openstack_compute_instance_v2* est la manière de décrire une instance de machine virtuelle depuis Terraform pour OpenStack.

En français, nous demandons la création d'une machine virtuelle dont le nom (*name*) est "serveur1", basé sur l'image système (*image_id*) ad091b52-742f-469e-

8f3c-fd81cadf0743, de taille (*flavor_id*) 3, avec la clé SSH (*key_pair*) "my_key_pair_name". Nous précisions aussi que cette machine utilise le groupe de sécurité (*security_groups*) et le réseau (*network*) par défaut.

Tous les paramètres et toutes les ressources qui peuvent être gérés par Terraform sont décrits dans la documentation du *provider*[15].

Il existe d'autres projets de systèmes de *provisioning*, par exemple CloudFormation ou OpenStack Heat. Nous avons préféré Terraform pour sa flexibilité et sa compatibilité avec un ensemble d'infrastructures en tant que service, là où Heat n'est compatible qu'avec OpenStack et CloudFormation uniquement avec Amazon AWS. Il est donc possible avec Terraform de décrire une infrastructure multifournisseur avec la même syntaxe et la même logique.

Les systèmes de configuration sont utilisés pour configurer les ressources qui ont été provisionnées par les systèmes de *provisioning*. La configuration ne concerne souvent que les ressources de machines virtuelles, et donc la configuration du système : installation de dépendances, mises à jour, librairies, modification des configurations, etc.

Là encore, toutes les configurations sont définies dans un fichier (ou plusieurs) de configuration qui sont en suite utilisées pour configurer une ou un million de machines, de manière consistante.

Dans le domaine des outils de configuration, il existe plus d'alternatives que pour le provisionnement, les outils plus connus étant :

— Puppet[16]

---

15. `https://www.terraform.io/docs/providers/openstack/`

16. `https://puppet.com/`

— Chef[17]

— Saltstack[18]

— Ansible[19]

Ces outils proposent chacun leur langage de configuration et ont leur propre mode de fonctionnement. Tous, excepté Ansible, nécessitent par défaut la mise en place d'un serveur central en charge de déployer les configurations sur les nœuds cible. Ansible propose tout simplement un mode de connexion directe aux nœuds avec comme nécessité de maintenir un simple fichier inventaire, qui répertorie les cibles des scripts. Étant donné qu'il est plus simple de maintenir un fichier (que l'on pourra par exemple versionner à l'aide de Git) et pour ne pas ajouter la nécessité de maintenir un énième serveur, nous avons choisi Ansible. Maintenu par la société Red Hat, Ansible est un produit libre qui s'appuie sur le format yaml.

Ansible fonctionne avec le principe de *roles*, *playbooks* et *inventory*.

L'inventaire ou *inventory* est un fichier utilisé par Ansible pour stocker les adresses des ressources (machines virtuelles) à configurer. L'inventaire utilise la syntaxe toml. Par exemple, une grappe composée de 4 serveurs qui ont pour rôles *compute*, *network*, *monitoring* et *storage* aurait pour inventaire le fichier suivant :

```
[compute]
192.168.0.1
192.168.0.2
192.168.0.3
```

17. `https://www.chef.io/chef/`

18. `https://www.saltstack.com/`

19. `https://www.ansible.com/`

```
[network]
192.168.0.1
192.168.0.2
192.168.0.3

[monitoring]
192.168.0.1

[storage]
192.168.0.10
```

Cet inventaire définit quatre types de ressources : *compute, network, monitoring et storage.* Il est ainsi possible d'utiliser cet inventaire pour tout un déploiement, en utilisant les types de ressources pour filtrer les tâches à faire.

Ces inventaires sont consommés par des *playbooks.* Les *playbooks* sont définis en yaml et sont les fichiers dans lesquels les instructions de configuration sont décrites, pour un déploiement particulier. Prenons ce *playbook* pour exemple [20] :

---

20. Extrait de `https://github.com/openstack/kolla-ansible/blob/master/ansible/detect-release.yml`

```
---
- name: Detect openstack_release variable
  hosts: "{{ detect_release_hosts }}"
  gather_facts: false
  tasks:
    - name: Get current kolla-ansible version
      number
      local_action: command python -c "..."
      register: kolla_ansible_version
      changed_when: false
      when: openstack_release == "auto"

    - name: Set openstack_release variable
      set_fact:
        openstack_release: "{{
          kolla_ansible_version.stdout }}"
      when: openstack_release == "auto"
```

Listing 3.2: Exemple de *playbook*

Ce *playbook* exécute deux tâches que sont "*current kolla-ansible version number*" et "*Set openstack_release variable*". Ces tâches seront exécutées sur tous les hôtes correspondants à la variable *detect_release_hosts*.

Cette variable peut être lue depuis l'inventaire ou donnée en argument lors de l'exécution du *playbook*. Au final ce *playbook* récupère la version de Kolla utilisée, dans le but de l'utiliser comme argument d'autres *playbooks*.

Le dernier élément principal d'Ansible est le système de rôles. Les rôles sont des recettes de configuration préétablies qui peuvent être utilisées par les *playbooks*.

Ce sont des *playbooks* qui ont été généralisés suffisamment pour être utilisés et paramétrés afin être exécutés dans d'autres *playbook* pour d'autre types de déploiements.

Un rôle est composé d'un ensemble de dossiers et fichiers. Voici par exemple la structure du rôle mariadb dans le projet kolla-ansible :

```
|-- defaults
|.. -- main.yml
|-- handlers
|..  -- main.yml
|-- meta
|..  -- main.yml
|-- tasks
|.. |-- backup.yml
|.. |-- bootstrap.yml
|.. |-- bootstrap_cluster.yml
|.. |-- check.yml
|.. |-- ...
|-- templates
    |-- backup.my.cnf.j2
    |-- galera.cnf.j2
    |-- mariadb.json.j2
    |-- wsrep-notify.sh.j2
```

Figure 3.5: Arborescence d'un rôle Ansible

Chaque fichier .yml est structuré comme un *playbook*. Les tâches sont définies

dans le dossier *tasks*. Le dossier *templates* contient tous les fichiers types qui seront utilisés pour le déploiement du service concerné, ici mariadb. Ces fichiers types utilisent une syntaxe Jinja2, un moteur de *template* permettant de les personnaliser à l'exécution du rôle, en fonction des spécificités de l'infrastructure qui est ciblée.

Ces systèmes de provisionnement et de configuration permettent donc d'écrire les règles de mise en place d'une infrastructure de A à Z. Cela permet de faire une abstraction quasi totale du matériel qui fait fonctionner tout cela. Ces recettes peuvent être testées, versionnées et donc traitées comme une production logicielle. Le résultat étant prédictible, l'infrastructure déployée l'est aussi, il devient donc très simple de gérer un ensemble d'infrastructures complexes, à l'échelle.

La figure 3.6 illustre la différence entre les outils de configuration et les outils de provisionnement.

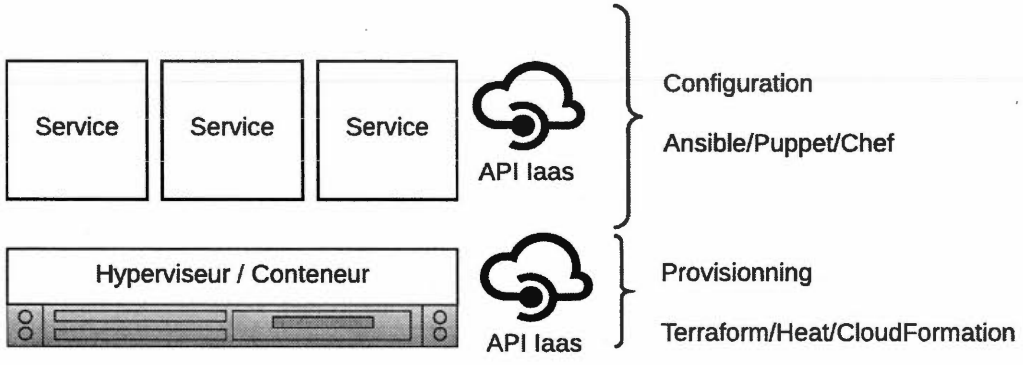

Figure 3.6: Configuration vs *Provisionnement*

Le projet **OpenStack Ansible** que nous avons évoqué plus tôt utilise ces principes et s'appuie sur Ansible pour déployer une grappe de calcul OpenStack complète, de manière prédictible et reproductible. Cela facilite grandement l'installa-

tion et la maintenance de cluster de toutes tailles.

Il reste un problème lorsque l'on installe OpenStack avec ce système. En effet, OpenStack consiste en l'installation de nombreuses dépendances, services, outils de supports, sur une ou plusieurs machines. Ansible facilite donc le déploiement, mais les risques que deux rôles aient besoin de faire des tâches contradictoires sur la même machine distante existent, risquant de compromettre le système entier. De plus, altérer un système d'exploitation qui est installé physiquement n'est pas aussi simple à gérer que si c'était une machine virtuelle qui était la cible.

Nous avons besoin que les cibles des tâches Ansible soient isolées du système d'exploitation du serveur, mais que ces cibles ne soient pas des machines virtuelles étant donné que ce sont les services que l'on installe qui vont gérer ce type de ressources.

C'est à ce genre de problématiques que les technologies à base de conteneurs répondent. Sur un système Linux, les conteneurs sont une technologie d'isolation de processus qui reposent directement sur le noyau du système hôte. Contrairement aux machines virtuelles qui font appel à l'utilisation d'un système complet, les conteneurs sont très légers et ne nécessitent que peu de configuration pour être opérationnels. La figure 3.7 présente une comparaison visuelle entre les conteneurs et les machines virtuelles.

Il existe plusieurs projets qui permettent l'exécution de conteneurs :

— CoreOS RKT [21]
— Canonical LXD (Linux Containers) [22]

---

21. `https://coreos.com/rkt/`

22. `https://linuxcontainers.org/lxd/introduction/`

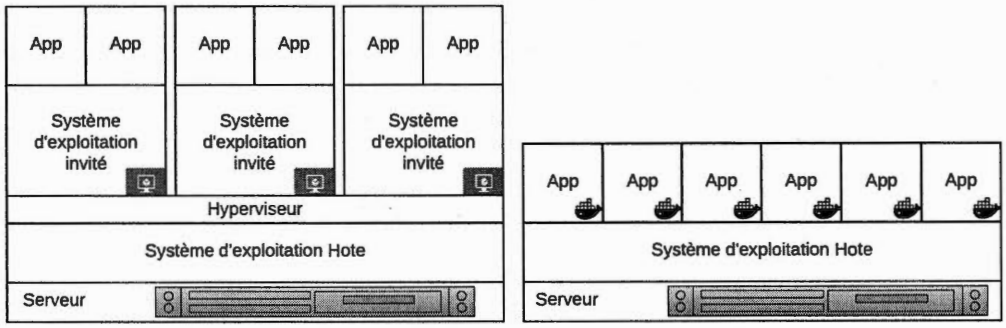

Figure 3.7: Machines virtuelles vs Conteneurs

— Docker [23]

Dans notre cas nous allons parler de Docker. Dans le monde des conteneurs, les autres technologies du marché sont quasiment anecdotiques tant Docker est la technologie privilégiée par la majeure partie de la communauté. La figure 3.8 présente les intérêts de recherche entre les mots-clés docker, lxc et rkt depuis 2013.

Le projet Docker est très mature et utilisé par de plus en plus d'industries, à très grande échelle. La communauté autour du projet est active et est aussi à l'origine de standards autour des conteneurs. La communauté Docker est à l'origine du projet Moby [24] dont le but est de fournir un cadre pour créer des systèmes de conteneurs pour des besoins spécifiques, sans avoir à partir du noyau Linux. Docker est lui-même construit sur ce cadre applicatif.

Docker n'est pas juste un simple système de gestion de conteneurs. C'est un écosystème complet qui permet :

---

23. `https://docker.com`

24. `https://mobyproject.org/`

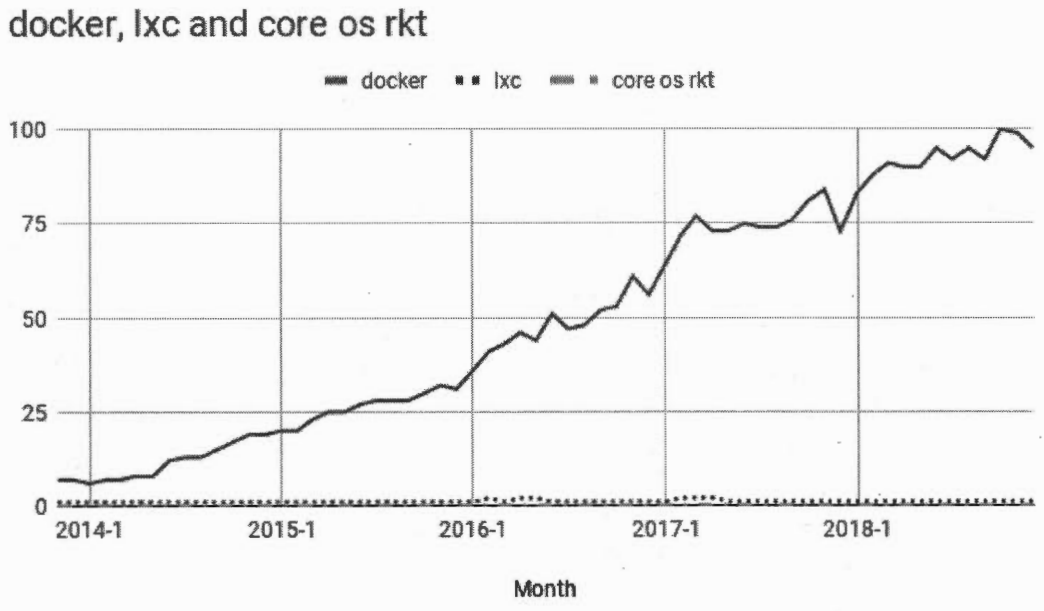

Figure 3.8: Docker vs Linux LXC vs CoreOS RKT - 2013 2018 (Google Trends)

- la gestion de conteneurs
- la définition de conteneurs à l'aide d'un langage déclaratif (Dockerfile)
- la définition de services basés sur des conteneurs (docker-compose)
- la gestion de réseau et résolution de noms entre les conteneurs
- la gestion de stockage attaché au conteneur
- la surveillance des conteneurs et la prise d'action en cas de faille
- le partage d'applications en conteneurs (Docker Registry, hub.docker.com)
- la gestion de conteneurs au sein d'une grappe de calcul (Docker Swarm), avec tolérances aux pannes, équilibrage de charges.

La base d'une application en conteneur Docker est décrite par un fichier : Dockerfile. Le format est relativement simple, chaque ligne est une instruction, les instructions peuvent être de type "système" (ex : utilisation d'aptitude dans un conteneur) ou des mots clés réservés par Docker permettant de faire des opéra-

tions de conteneurs. Voici l'exemple d'un conteneur qui exécute une application Java :

,

```
FROM openjdk:8-jdk-alpine

RUN apk add --update --no-cache \
          bash \
          supervisor

WORKDIR /stanford-corenlp-full-2017-06-09
ADD stanford/stanford-corenlp-full-2017-06-09 ./
ADD stanford/properties/* ./
ADD start-corenlp.sh ./
ADD stanford/ner-models/* ./
ADD features-true-striker/features-true-striker.sh ./

RUN export CLASSPATH="`find . -name '*.jar'`"

ENV PORT 9000

EXPOSE \$PORT
COPY supervisord.conf /etc/supervisor/conf.d/supervisord
   .conf
CMD ["/usr/bin/supervisord", "-c", "/etc/supervisor/conf
   .d/supervisord.conf"]
```

Listing 3.3: Exemple de Dockerfile pour Stanford Core NLP

Ce Dockerfile décrit l'image Stanford CoreNLP que nous utilisons dans le chapitre 4. Une image est une application qui est installée dans un conteneur prête à être lancée.

La première instruction *FROM* explique à Docker à partir de quelle image construire cette nouvelle image. En effet, chaque nouvelle image, même une image de très bas niveau dans laquelle on voudrait interagir avec le système, nécessite d'utiliser une image de base, l'image la plus bas niveau étant *FROM scratch*.

Ici, nous voulons créer une image d'une application Java, nous utilisons donc l'image *openjdk* avec pour version "8-jdk-alpine", ce qui signifie que nous voulons que cette image utilise la version 8 du *Java Development Kit*, basé sur Alpine Linux.

Alpine Linux est une distribution Linux au même titre qu'Ubuntu ou Red Hat. Comme précisé précédemment, les conteneurs n'embarquent pas un système contrairement à une machine virtuelle. L'utilisation d'alpine est un abus de langage, l'image alpine ne contenant que les outils et applications propres à la distribution Alpine Linux.

Par ailleurs les images systèmes utilisent *FROM scratch* comme image de base[25].

Dans notre exemple, la seconde instruction utilise le mot-clé *RUN* suivi d'une commande *apk*. Il s'agit ici d'exécuter une commande au sein du conteneur, au moment de la création de l'image.

Les mots-clés *WORKDIR*, *ADD* ou *COPY* qui suivent servent à ajouter des fichiers depuis l'hôte vers le conteneur. Le mot-clé *CMD* à la différence de *RUN*

---

25. source docker-alpinehttps://github.com/gliderlabs/docker-alpine/blob/master/versions/library-edge/aarch64/Dockerfile

va s'exécuter quand l'image sera démarrée.

En effet, ce Dockerfile est ensuite construit à l'aide de la commande *docker build* et devient une image. Cette image est comparable à un exécutable ou un Jar, c'est un paquet prêt à être exécuté sur n'importe quel système Linux et qui se comportera toujours de la même manière.

Dans notre cas, l'image est construite avec comme instruction "EXPOSE 9000" qui signifie que l'image une fois démarrée va exposer le port 9000 sur l'hôte. Ainsi l'exécution de cette image va lancer un serveur Stanford Core NLP au sein d'un conteneur et mettre à disposition de l'hôte le port 9000 pour accéder à l'interface web.

La figure 3.9 résume le processus de construction d'une image et l'utilisation d'un conteneur.

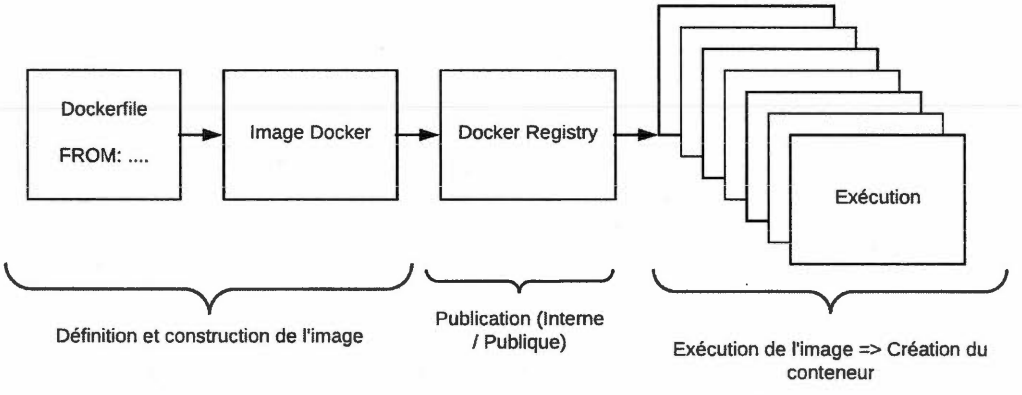

Figure 3.9: Flux de construction, publication et exécution d'un conteneur Docker

L'image donnée en exemple ici étant publiée sur le service Docker Hub, il est

possible de l'essayer sur votre propre machine en lançant la commande :

```
docker run -p 9000:9000 antoinebriand/corenlp-server
```

Si vous n'avez pas Docker installé sur votre machine, vous pouvez l'essayer en ligne en utilisant le service gratuit *Play With Docker*[26].

Déployer les services et prérequis OpenStack avec Ansible et Docker est la motivation du projet Kolla Ansible. La seule modification nécessaire sur les hôtes est l'installation de Docker.

Le projet fait officiellement partie des projets de la communauté OpenStack. Pour utiliser Kolla, pour installer, mettre à jour ou modifier une grappe OpenStack, il suffit d'avoir configuré les interfaces réseau de tous les serveurs (calcul, stockage), de les reporter dans l'inventaire Ansible du projet et de lancer les *playbooks*.

Ainsi, tous les composants OpenStack sont dans des conteneurs, et les machines virtuelles sont gérées par ces services, directement sur les hôtes. Il devient alors très simple d'effectuer des mises à jour, de modifier un conteneur ou même de passer à l'échelle. Le comportement des images étant constant et prévisible. La figure 3.10 schématise la répartition logique des services OpenStack et des futures machines virtuelles sur un nœud de calcul.

La configuration Kolla pour notre type de déploiement est disponible sur notre répertoire GitLab[27].

Notre infrastructure en tant que service, OpenStack, est maintenant fonctionnelle et nous permet de gérer nos ressources à partir d'une interface web comme OpenS-

---

26. `https://labs.play-with-docker.com`

27. `https://gitlab.ikb.info.uqam.ca/it/kollaikbig`

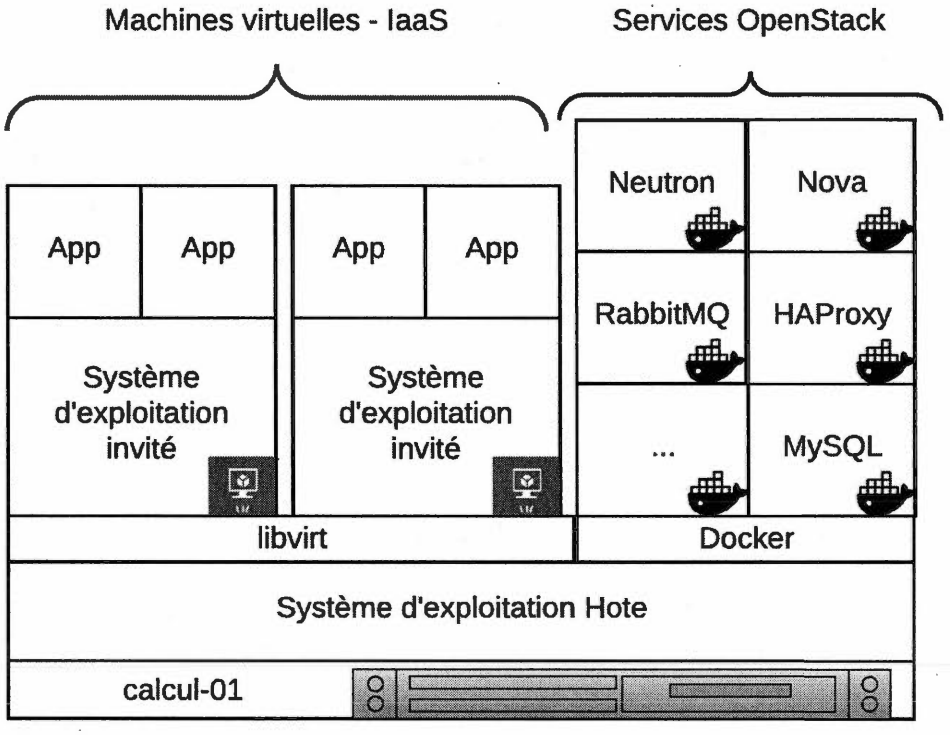

Figure 3.10: Répartition logique des conteneurs OpenStack et des machines virtuelles

tack Horizon, comme illustré par la figure 3.11, ou via des outils d'infrastructure en tant que code, Terraform ou Ansible. Le système de stockage Ceph permettant de stocker des disques durs virtuels, des fichiers, ou des objets, à l'échelle et répliqué, nous pouvons maintenant utiliser ces services dans le cadre de projets orientés données massives.

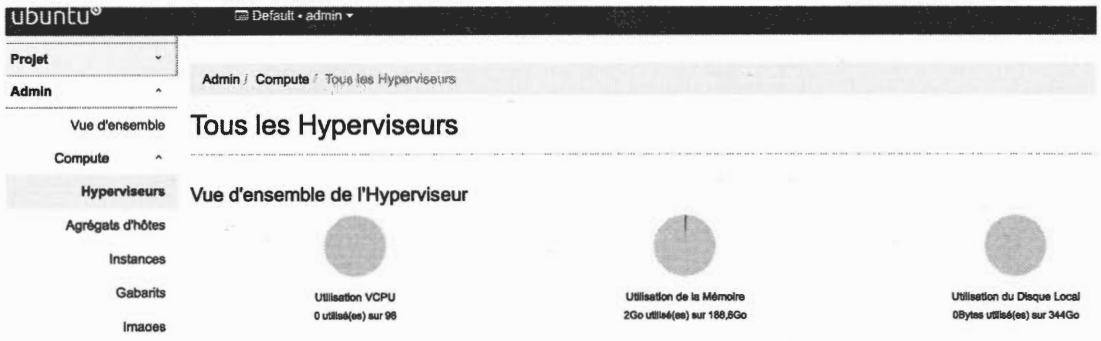

Figure 3.11: Interface de gestion OpenStack Horizon

Dans ce chapitre, nous avons vu comment mettre en place une infrastructure robuste, flexible, élastique, capable de traiter des données massives. Nous disposons d'un environnement qui va nous permettre de stocker une quantité importante de données et de faire des calculs en parallèle et distribués sur plusieurs nœuds. Enfin, l'utilisation exclusive de technologies libres et massivement supportées par leur communauté respectives nous laisse totalement libres de notre choix de configuration et de matériel, rendant ainsi possible l'extension de cette grappe à un coût maîtrisé.

Notre infrastructure de calcul et stockage étant maintenant en ligne, nous allons présenter un premier type d'application traitant des données massives textuelles, dans un cadre industriel.

CHAPITRE IV

RECHERCHE D'ENTITÉS D'INTÉRÊTS AU SEIN DE LARGES CORPUS

Les travaux présentés dans ce chapitre ont été publiés dans l'article :
"Identification of Sensitive Content in Data Repositories to Support Personal Information Protection",
Antoine Briand, Sara Zacharie, Ludovic Jean-Louis, Marie-Jean Meurs,
International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, p898–910, 2018 (Briand et al., 2018b).

4.1 Données massives et conformité

L'archivage des données en grande quantité est une activité pratiquée par de plus en plus de fournisseurs de service en informatique.

Toutefois, ceux-ci ne font que du stockage, bien souvent les données sont inertes ou très peu explorées.

La multiplication récente de la révélation des fuites de données donne lieu à un durcissement des politiques qui encadrent la détention de données privées et crée le besoin d'outils pour le respect de la conformité (*compliance*).

Une société doit maintenant être capable de tracer des données, de retrouver des

échanges, de s'assurer que le flux de données n'a pas été altéré (fuites), etc.

Plus récemment dans le cadre du RGPD (Règlement Général sur la Protection des Données), les données personnelles ne peuvent plus être simplement stockées, il faut être capable d'extraire celles d'un individu sous toutes leurs formes et à sa demande.

Autant de problématiques pour lesquelles les services d'archives ne sont pas prêts.

À l'aide de l'infrastructure décrite dans le chapitre précédent, nous souhaitons mettre en place un système de détection d'entités d'intérêt pour la découverte de données potentiellement sensibles au sein de larges archives.

Les entités d'intérêt sont des éléments de texte (mots, groupes de mots, etc.) qui ont une valeur (politique, stratégique, confidentielle), un intérêt à être découvertes dans l'amas de texte qui compose les entrepôts de données.

4.2 Corpus de travail

Afin de réaliser ces travaux, nous allons opérer par étapes. Tout d'abord nous allons définir un ensemble de données de travail, proches de la réalité industrielle de notre partenaire. Ces données doivent être suffisamment qualitatives et quantitatives pour supporter notre travail sur la partie exploitation de données massives, mais aussi sur la partie conformité règlementaire (*compliance*).

Notre cadre d'application est le suivant : des entreprises et des administrations de toutes tailles et toutes origines déposent des données sous forme d'archive dans des entrepôts. Ces données sont principalement des courriels, mais il est aussi possible de trouver des textes aux formats et domaines variés.

Comme nous travaillons dans un but de soutien de la conformité (*compliance*),

nos corpus doivent être les plus proches de la réalité industrielle tout en contenant suffisamment d'entités d'intérêt et potentiellement sensibles pour pouvoir faire des évaluations de cette partie du système par la suite.

En prenant cela en considération, nous avons exploré les corpus par domaines :

— Santé
— Corporatif
— Réseau social
— Nouvelles

Les données de santé et du secteur industriel contiennent indéniablement des données sensibles qui sont de fait couvertes par la majorité des législations en place. Grâce aux diverses compétitions scientifiques, il est possible d'accéder à des données non anonymisées ou partiellement.

Les données issues des réseaux sociaux sont plus compliquées à traiter. L'intérêt de ce domaine de données est de travailler sur des grands volumes de données au format et à la syntaxe très informelle, contrairement aux deux premiers.

Les nouvelles sont les corpus de référence pour l'entraînement de systèmes d'annotations d'entités d'intérêt. Ces corpus nous serviront de référence pour le travail parallèle à celui-ci et qui consiste en la création d'annotateurs d'entités.

La table 4.1 présente les corpus retenus pour notre travail.

Les corpus I2B2 et MIMIC v3 sont annotés par des experts. Ils sont donc une base solide pour le travail sur les entités d'intérêt dans le domaine médical.

Mimic v3 (Johnson et al., 2016) est un corpus ouvert, élaboré par le laboratoire de physiologie computationnelle du Massachusetts Institute of Technology

Tableau 4.1: Ensemble de corpus retenu

Corpus	# documents	Domaine	Type
I2B2	1304	Médical	Notes cliniques
MIMIC v3	2 083 180	Médical	Notes cliniques
Administration américaine	3 571 215	Corporatif	Courriels et pièces jointes
Reddit 2005 - 2017	> 2 millards	Varié	Réseau social

(MIT), qui comprend des données sur la santé qui ont été anonymisées et qui sont associées à environ 40 000 patients en soins intensifs. Il comprend les données démographiques, les signes vitaux, les tests de laboratoire, les médicaments, etc.

Nous avons utilisé la version MIMIC-III, qui comprend plus de 58 000 hospitalisations pour 38 645 adultes et 7 875 nouveau-nés. Les données couvrent la période allant de juin 2001 à octobre 2012. La base de données, bien que dépersonnalisée, contient encore des informations détaillées sur les soins cliniques des patients.

Plus précisément, nous nous sommes appuyés sur la table "NOTEEVENTS" qui contient 2,083,180 d'enregistrements. Ces enregistrements sont des notes cliniques liées aux patients hospitalisés.

Ce corpus est fourni sous forme de base de données relationnelle. Dans notre cas, la table "NOTEEVENTS" peut être schématisée comme dans la table 4.2.

I2B2

I2B2 (Informatics for Integrating Biology & the Bedside) [1] est une fondation permettant la collaboration autour du domaine médical en facilitant le partage, l'intégration, la normalisation et l'analyse de données hétérogènes provenant des soins

1. `https://www.i2b2.org/`

Tableau 4.2: Schéma de la table NOTEEVENTS - Corpus MIMIC-III

Colonne	**Parent**	**Description**
row_id		Identifiant unique de la ligne
subject_id	*patients*	Clé étrangère pour identifier le patient
hadm_id	*admissions*	Clé étrangère pour identifier l'hôpital d'accueil
chartdate		Date à laquelle la note a été consignée au dossier
charttime		Date et heure auxquelles la note a été enregistrée
storetime		Heure à laquelle la note à été enregistrée
category		Catégorie de la note, p. ex. décharge de l'hôpital
description		Une catégorie plus détaillée de la note, souvent saisie en texte libre
cgid	*caregivers*	Clé étrangère. Identifie le personnel soignant
iserror		Booléen pour signaler une erreur avec la note
text		Contenu de la note

de santé et de la recherche. Les jeux de données sont mis à disposition lors de tâches internationales. La dernière en date est 2014 *De-identification and Heart Disease Risk Factors Challenge.*

Dans cette dernière tâche, I2B2 met à disposition le corpus *NLP Data Set #7a : De-identification Challenge Data Set* (Stubbs et Uzuner, 2015; Stubbs et al., 2015). Les PHI (*Protected Health Information*) de ce corpus ont été dé-identifiées à l'aide du guide HIPAA et est composé de notes cliniques qui sont stockées au format XML, la note d'un côté et les PHI de l'autre.

Données confidentielles - Institution américaine

Ce corpus est directement fourni par notre partenaire industriel, avec l'accord

de l'institution propriétaire (une très grande ville américaine). C'est le jeu de données le plus proche de la réalité industrielle. Il contient exclusivement des échanges par courriels d'employés de l'administration ainsi que les pièces jointes associées. Aucun travail de dé-identification n'a été fait sur ce corpus et il nous est strictement impossible de le divulguer.

Ce corpus contient près de 3.6 millions de documents (courriels et pièces jointes).

Reddit

Reddit est un très grand réseau social où les membres peuvent échanger sur tous les sujets possibles et imaginables. Le réseau social est organisé comme un gigantesque forum avec des groupes/communautés autour de sujets communs que l'on appelle des subreddits. Ces données sont publiques et disponibles depuis le site Academic Torrents [2] ou via l'API Rest du service.

Ce corpus disponible présente l'avantage d'être très massif et en constante évolution (vélocité > nulle). On dénombre plus de 2 milliards de messages disponibles.

Nous avons envisagé d'autres corpus qui n'auront pas été retenus pour la suite du travail. Parmi ces corpus nous pouvons citer :

— ENRON [3]
— Panama Papers [4]
— documents issus de Wikileaks [5]

---

2. `http://academictorrents.com/browse.php?search=reddit`

3. `https://www.cs.cmu.edu/~./enron/`

4. `https://www.icij.org/investigations/panama-papers/`

5. `https://wikileaks.org/`

— courriels d'Hillary Clinton [6]

La raison principale est le manque d'annotations nous permettant d'évaluer le travail final. De plus, une grande partie des données sensibles ont été effacées avant la publication.

Ces corpus présentent l'avantage de provenir de sources hétérogènes, de posséder des formats qui leur sont propres et d'être de taille et vélocité différentes. La table 4.3 présente les sources, l'ordre de grandeur et la vélocité de chacun des corpus retenus.

Tableau 4.3: Sources, ordres de grandeur et vélocité des corpus

Corpus	**Source**	**Tb, Gb, Mb**	**Vélocité**
I2B2	Fichiers XML	Mb	Nulle
MIMIC v3	RBDD	Gb	Nulle
Organisation Américaine	Fichiers XML	Gb	Nulle
Reddit 2005 - 2017	Fichiers JSON API REST	Tb	2.8 millions par jour

4.3 Gestion des sources de données hétérogènes

Nos corpus proviennent de multiples sources. Des fichiers (de différents formats), des bases de données ou des sources de données en ligne (API Rest). L'objectif étant d'être capable de faire des recherches au sein de ces corpus, nous allons utiliser des interfaces d'accès uniques aux données.

Toutes les données brutes sont stockées dans CEPH, y compris les fichiers de

6. `https://wikileaks.org/clinton-emails/`

description de base de données de Mimic, ainsi nous pouvons y accéder partout sur le cluster via l'API de stockage d'objets.

La figure 4.1 représente cette étape de centralisation des données au sein d'un *bucket* Ceph. Un *bucket* est un compartiment de stockage qui contient plusieurs objets. Cela permet d'accéder aux objets du cluster avec la même logique que le parcours de dossiers/fichiers, tout en s'appuyant sur une structure distribuée.

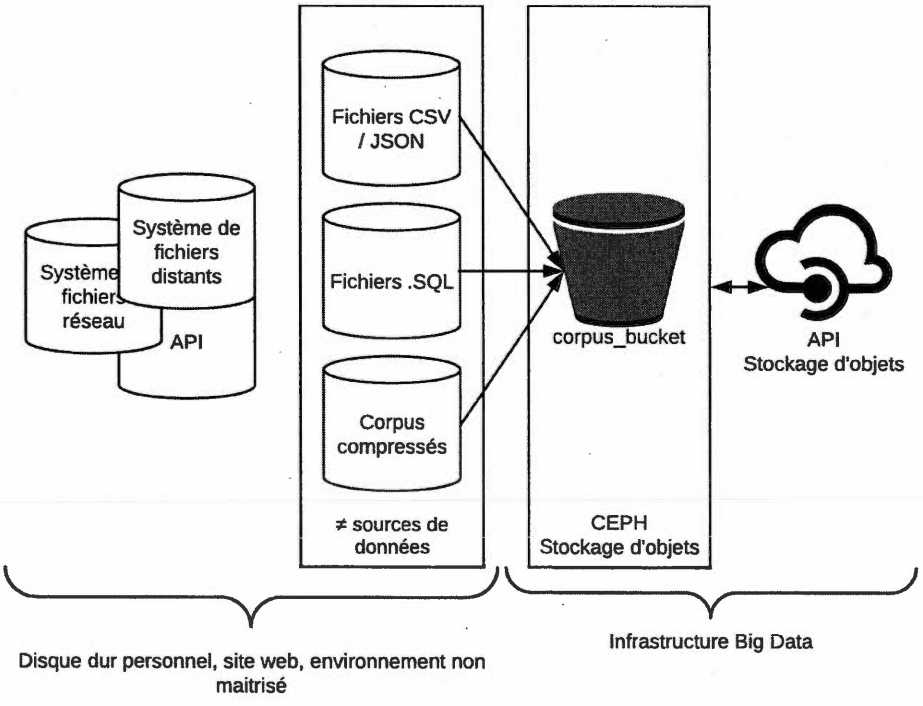

Figure 4.1: Centralisation des fichiers et données brutes dans un *bucket* Ceph

Sur la figure 4.1, nous avons mis de côté les données qui seraient fournies par un système de fichier réseau (NAS, NFS, Samba, ..), un système de fichiers distant

(Amazon S3, Google Drive, Nextcloud, ..) ou une API (réseau social, slack ..). Il est possible de rapatrier ces données localement pour les traiter directement sur notre réseau, toutefois nous pouvons, au moins pour les opérations de lecture, utiliser directement les sources de données. Nous évitons ainsi de surcharger notre *cluster* de stockage.

Ces données et celles fournies par Ceph peuvent être accédées depuis un même point d'accès : un système de stockage virtuel distribué, comme Alluxio. Un système de stockage virtuel (VSS) est une interface unique qui permet de se connecter à de multiples sources de stockage, locales ou distantes, et de les accéder de la même manière. Ces systèmes rendent l'exploitation de multiples sources de données simple. De plus, ils sont installés au plus proche des nœuds de calcul, voire dans de plus petites infrastructures, directement sur les nœuds de calcul. Comme ce ne sont que des interfaces, les données restent stockées sur le média de base, appelés *under storage system*, et grâce à des mécanismes de cache, une partie des données seulement est stockée, généralement en mémoire vive, permettant d'effectuer des tâches intensives et répétitives sans consommer beaucoup de réseau.

Cette interface d'accès aux données unique, implémentée avec Alluxio est représentée par la figure 4.2.

Alluxio est le premier service que nous mettons en place sur notre infrastructure en tant que service (IaaS). Dans un souci de reproductibilité, et pour faciliter le déploiement et le passage à l'échelle, nous utilisons nos outils d'infrastructure en tant que code (IaC) pour allouer et configurer les machines virtuelles qui seront responsables de la grappe Alluxio. Comme nous avons quatre nœuds de calcul, nous allons déployer au moins quatre instances d'Alluxio, une par nœud.

L'objectif est de se retrouver avec une grappe Alluxio, connectée à toutes nos sources de données et fournissant un point d'accès unique et distribué à nos don-

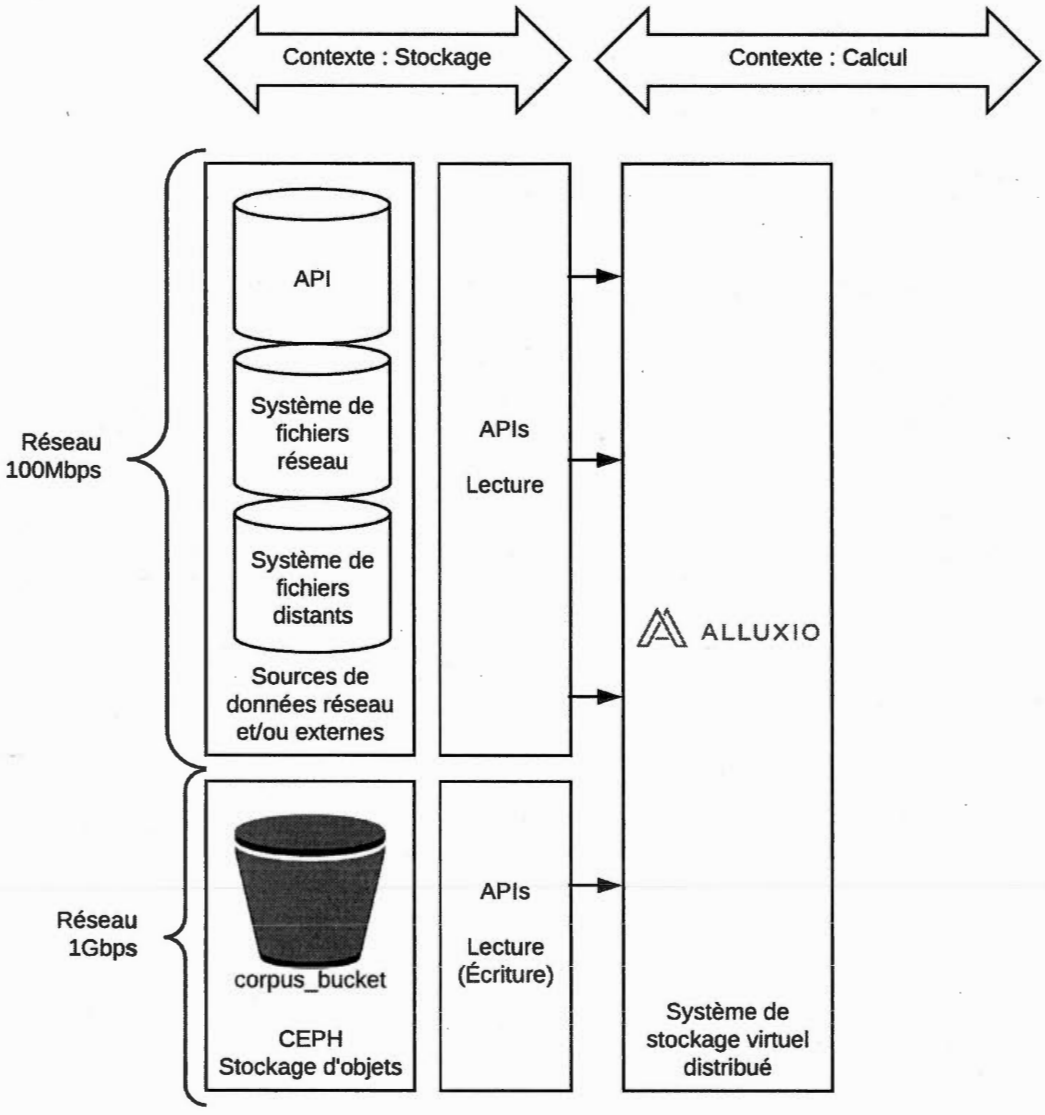

Figure 4.2: Interface d'accès unique aux différentes sources de données

nées, tout en accélérant nos travaux à l'aide du cache opéré sur le réseau de calcul. La figure 4.3 présente l'infrastructure avec le service Alluxio déployé.

Pour parvenir à ce résultat, nous allouons les ressources suivantes :

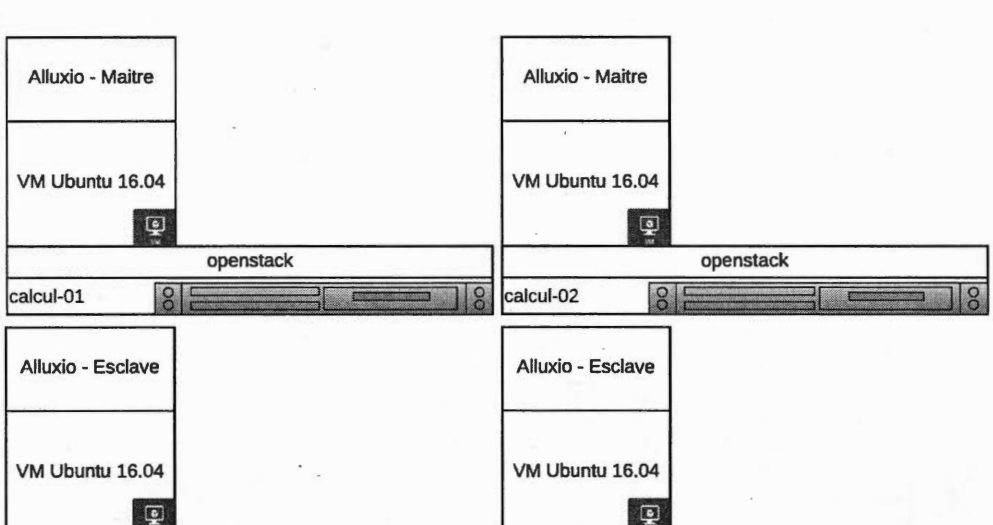

Figure 4.3: Grappe Alluxio sur OpenStack

— quatre machines virtuelles
— un réseau privé
— la connexion au réseau haute vitesse
— quatre IP flottantes pour accéder au cluster à l'extérieur du réseau
— un groupe de sécurité alluxio

Comme nous l'avons décrit dans le chapitre 3, Terraform est l'outil que nous utilisons pour créer cette infrastructure, via le langage de description d'infrastructure HCL. Grâce à cette technologie, nous pouvons définir toutes les ressources une seule fois, puis demander à l'outil plusieurs fois la même ressource. Les sources de l'infrastructure Alluxio sont disponibles dans notre répertoire Gitlab[7].

La topologie réseau de la grappe Alluxio une fois déployée est représentée par la

7. `https://gitlab.ikb.info.uqam.ca/antoine/netmail-annotation-infrastructure`

figure 4.4.

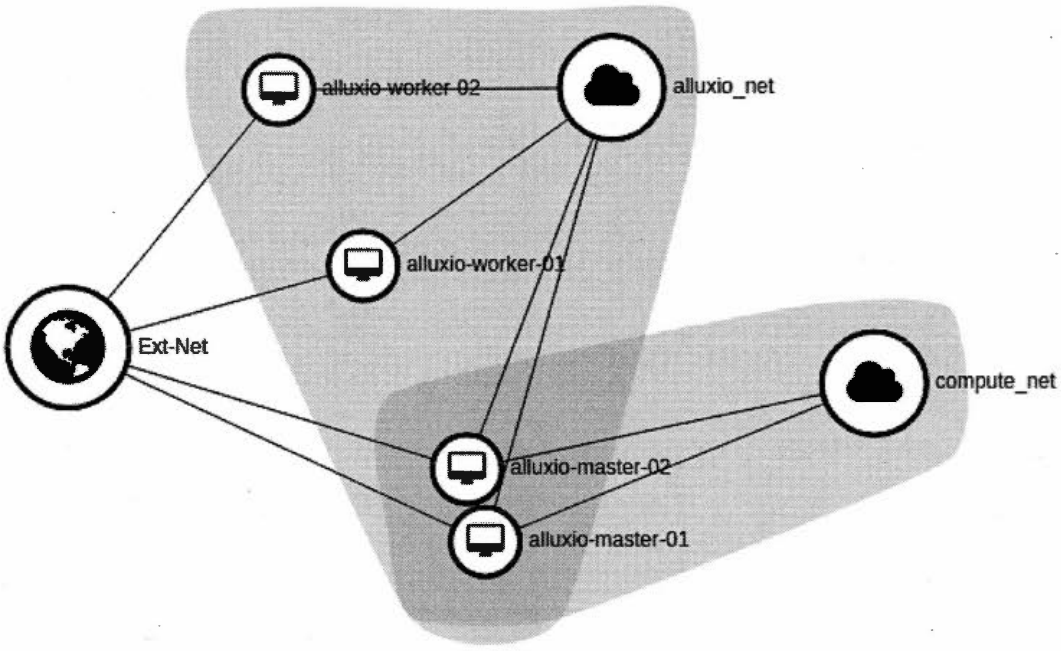

Figure 4.4: Grappe Alluxio - topologie réseau

Le réseau *compute_ net* correspond au réseau du *cluster* de calcul, à 1 Gb/s, tandis que le réseau Ext-Net correspond au réseau public, permettant d'accéder à Alluxio à distance. Un réseau virtuel *alluxio_ net* est défini pour assurer les communications du *cluster* Alluxio uniquement, les accès au système de stockage virtuel se passent par *compute_ net.* Comme nous avons défini un groupe de sécurité pour ce groupe de ressources, nous pouvons contrôler les flux réseau à l'échelle du *cluster.*

En effet, Alluxio permettant d'accéder à toutes nos données, il est important de filtrer les accès, notamment au niveau du réseau public.

Maintenant que nous pouvons accéder à toutes nos données via une seule interface, nous pouvons commencer à expérimenter et rechercher de l'information.

Dans notre application, nous souhaitons être capables d'identifier des entités d'intérêt dans tous les corpus. Ces données étant toutes textuelles, nous allons tirer parti des techniques d'indexation pour créer un moteur de recherche nous permettant de détecter les documents sensibles.

4.4 Indexation des corpus

L'exploration de larges corpus de textes ne peut être réalisée à l'aide de techniques classiques. Pour un fichier dont la taille n'excède pas l'échelle des mégaoctets, il est possible de faire des recherches à l'aide d'outils comme *grep* (utilitaire de recherche de texte en ligne de commande) sur linux, ou la fonction de recherche d'un éditeur de texte. Comme nous l'avons vu à la section 1.2 de l'état de l'art, les techniques d'indexation de documents permettent la création de systèmes de recherche basés sur des données massives.
Notre travail étant basé sur l'exploration d'archives textuelles très volumineuses, nous avons fait le choix d'indexer l'ensemble des corpus au sein d'un même index afin d'effectuer des requêtes et d'utiliser les moteurs de découvertes de caractéristiques (ou facets) d'Apache Solr afin d'obtenir des informations sur nos données
Comme dans tous les travaux de construction d'index de documents, nous avons commencé par établir un schéma en fonction des données d'entrées. Celles-ci ayant un format variable, nous avons défini un schéma très large qui ne sera pas pleinement utilisé pour chaque type de documents.

Contrairement à un schéma de base de donnée classique, le fait de ne pas utiliser pleinement le schéma dans le cas d'un index construit avec Apache Lucene et Apache Solr n'est pas un problème étant donné que le système est inspiré des bases NoSQL et se rapproche des bases de données orientées documents dont le schéma est dynamique. Apache Solr dispose lui aussi d'un mode dynamique lui

permettant de mettre à jour le schéma automatiquement sans avoir à le redéfinir. Si un nouveau champ non prévu est présenté au système, celui-ci ajoute un champ de type *text_general*. Ce comportement très flexible est parfait pour le travail avec des données dont le format est hétérogène. Toutefois il est important de maîtriser les types de champs Solr pour faire confiance à ce système. En effet, le type *text_general* n'applique aucun traitement au texte qui est donc indexé tel quel (pas de suppression de mots vides, pas de mise en majuscule, conservation de la ponctuation, etc.). C'est un comportement à évaluer au cas par cas, car il peut fortement altérer les performances de recherche sur l'index par la suite [8]. Le schéma de données est disponible sur notre répertoire GitLab [9].

Indexer les documents d'un ensemble de corpus est une opération coûteuse en temps et en ressources. L'opération peut prendre plusieurs heures à plusieurs jours en fonction de la quantité de documents à indexer. Toutefois, grâce à notre infrastructure de stockage et de traitement, nous pouvons distribuer cette tâche, réduisant ainsi le temps nécessaire.
La distribution d'un index Apache Solr est faite avec le mode *cloud*. Ce mode consiste à scinder l'index sur plusieurs instances Apache Solr (plusieurs nœuds). Toutes les instances sont ensuite configurées et surveillées avec Apache Zookeeper, un gestionnaire de configuration et système de découverte de services, qui devient le point d'entrée de notre grappe Apache Solr.
Enfin, l'éco-système Apache étant très complémentaire, il est possible d'utiliser le cadre applicatif Apache Spark pour créer un système d'indexation distribué de

---

8. La documentation des *Field Type Definitions and Properties* de Solr est d'une grande aide pour prendre la bonne décision : `https://lucene.apache.org/solr/guide/7_3/field-type-definitions-and-properties.html`

9. `https://gitlab.ikb.info.uqam.ca/antoine/mitacs-solr`

bout en bout. Couplé à Spark-Solr [10], Apache Spark est capable de se connecter à toutes les instances de Solr Cloud et distribuer toute la charge de calcul sur le cluster, tout en tirant partie de son système de RDD et donc de la mémoire vive disponible.

Tout ce système nous permet d'indexer plusieurs centaines de millions de documents en quelques heures seulement. L'index une fois créé nous permet de faire des recherches sur l'ensemble des corpus.

Nous pouvons donc utiliser le moteur de recherche par mots clés pour extraire des informations en quelques millisecondes, mais aussi effectuer des requêtes plus complexes à base de *facets* [11]. Le mécanisme de *facet* nous permet d'extraire des données statistiques dynamiquement sur la totalité du corpus, tout en tirant partie de l'index.

Ce moteur de recherche à disposition avec cet index de documents, nous pouvons l'utiliser comme source de données NoSQL, consommé par nos annotateurs distribués dont nous parlons dans la partie suivante.

4.5 Détection d'entités d'intérêt par traitement par lots

Afin de détecter les entités d'intérêts, potentiellement sensibles, qui se trouveraient au sein de notre corpus, nous utilisons un ensemble de systèmes de reconnaissance d'entités nommées. Ces systèmes sont listés dans le tableau 4.4.

Le système à base de CRF dans le domaine de la santé a été réalisé dans un travail parallèle (Briand et al., 2018b).

---

10. Un connecteur Solr pour Spark, distribué sur Github en code ouvert par Lucidworks `https://github.com/lucidworks/spark-solr`

11. `https://lucene.apache.org/solr/guide/6_6/faceting.html`

Tableau 4.4: Systèmes de reconnaissance d'entités nommées utilisés

Type	Base	Modèle	Domaine
I.A (CRF)	Stanford NER	CoNLL	Actualités
I.A (CRF)	Stanford NER	Personnalisé	Santé
Expressions régulières (Règles)	N.A	N.A	N.A

CRF (*Conditional Random Fields*) (Lafferty et al., 2001) est une famille de modèles statistiques largement utilisée en reconnaissance de motifs (*pattern*) au sein de textes.

Les annotateurs à base de règles sont basés sur des dictionnaires et (ou) des expressions régulières (*regular expression, regex*) qui reflètent une donnée à retrouver. Par exemple, la détection de numéros de cartes de crédit peut être effectuée à l'aide d'expressions régulières, en utilisant une série de quatre chiffres répétés quatre fois.

L'expression ci-dessous représente le numéro d'une carte de crédit VISA.

```
^(?:4[0-9]{12}(?:[0-9]{3})?|5[1-5][0-9]{14})\$
```

Les expressions régulières sont un moyen efficace de détecter des entités à partir de leur forme de surface, toutefois elles apportent aussi un grand nombre de faux positifs. Dans l'exemple de notre carte de crédit, une suite de 16 chiffres ne sera pas forcément un numéro de carte de crédit. Un moyen efficace est de coupler ces expressions régulières à un dictionnaire qui pourrait valider si la suite de chiffre correspond à une carte de crédit ou non. Toutefois, seuls les propriétaires de numéros des cartes de crédit peuvent accéder à ce type de bases de données.

Avec nos corpus et les ressources dont nous disposons, il faut en moyenne 1 à

3 secondes pour annoter un document avec un annotateur à base d'intelligence artificielle. L'annotation à base d'expressions régulières nécessite moins d'une seconde.

En fonction de la taille de notre corpus, annoter chaque document avec chacun de ces annotateurs, de manière séquentielle prendrait (par extrapolation du temps moyen d'annotation d'un document) entre 7200 et 17143200 de secondes (soit entre 2 et 4762 heures).

Que ce soit pour notre travail ou pour une application d'entreprise, ces ordres de grandeur ne sont pas acceptables. Dans le cas du corpus de Reddit qui contient près de 2 milliards de documents, il faudrait plus de 300 ans juste pour l'annotation [12].

Afin d'améliorer le processus, nous allons tirer parti de la distribution de l'annotateur. Ainsi, nous pouvons avoir plusieurs documents annotés en parallèle. Dans notre cas, pour faire fonctionner les 3 annotateurs à cette cadence, il nous faut 2 processeurs et 2 Gb de RAM.

Avec le matériel dont nous disposons, nous pouvons donc avoir 24 instances d'annotation qui fonctionnent en parallèle, réduisant ainsi les tâches sur nos corpus à quelques jours. Le tableau 4.5 présente une estimation du temps d'annotation nécessaire pour chaque corpus, avec D le nombre de documents à annoter, ΔT_{seq} le temps nécessaire à une annotation séquentielle et ΔT_{24} le temps nécessaire à une annotation faite par 24 annotateurs en parallèle.

La figure 4.5 met en évidence la différence de temps nécessaire entre les deux configurations (l'échelle du graphique a pour maximum 300 heures).

---

12. Ces chiffres sont approximatifs et dérivés du temps nécessaire pour l'annotation de 100 documents

Tableau 4.5: Tableau des temps d'annotation séquentiel contre distribué (ici 24 par 24)

Corpus	D	ΔT_{seq} **(Heures)**	ΔT_{24} **(Heures)**
I2B2	1304	2	0.08
MIMIC v3	58000	77	3.21
Organisation Américaine	3571215	4762	198.42
reddit 2007 2017	$\approx$ 2 000 000 000	2666666.67	111111.11

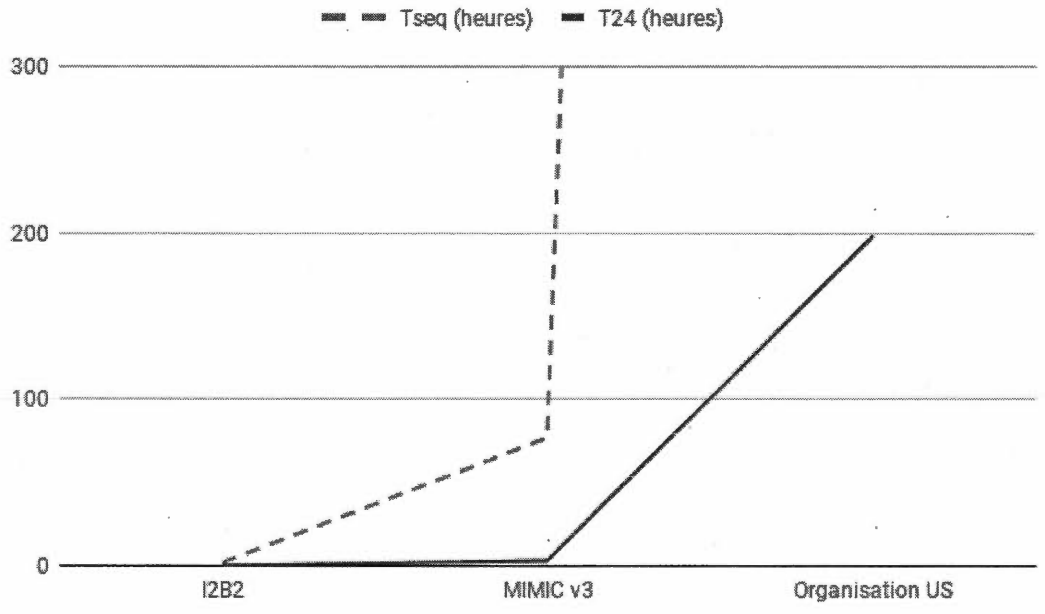

Figure 4.5: Temps d'annotation séquentiel contre distribué 24 fois

Dans notre cas, l'annotation de la totalité du corpus Reddit prendrait théoriquement une dizaine d'années. De ce fait, nous avons donc décidé de n'en traiter qu'un sous-ensemble.

Pour réaliser cette tâche, nous avons construit une image Docker qui exécute Stan-

ford Core NLP Server. Les sources de l'images sont disponible sur notre répertoire GitLab [13].

Docker Swarm fonctionne sur un principe de maîtres/esclaves.

Les nœuds maîtres étant responsables de l'orchestration du cluster, ils doivent s'assurer que les services sont toujours en ligne, gérer le réseau entre les conteneurs, etc.

Les nœuds esclaves ne font qu'exécuter les conteneurs que les maîtres leur demandent.

Dans notre cas, le déploiement du service d'annotation se passe ainsi. Sur le nœud maître de notre grappe, nous déclarons les deux services d'annotation. Docker Swarm se charge de télécharger l'image depuis le Docker Hub et lance le service pour nous. La figure 4.6 présente une visualisation du service démarré sur un *cluster* Docker Swarm avec 1 comme facteur de réplication.

On se retrouve ainsi avec un annotateur CoNLL et notre annotateur personnalisé. Pour augmenter la capacité d'annotation (Nombre de documents annotés par seconde), il suffit de dire à Docker Swarm d'augmenter le facteur de réplication des services. Le système va s'occuper de créer autant de conteneur que le facteur de réplication et les rendre disponibles à l'utilisateur. La figure 4.7 présente une visualisation du passage à un facteur de réplication six.

Tous les services (ici *corenlp* et *corenlp2*) sont accessibles avec une adresse unique, celle d'un des maîtres. Ainsi, lorsque l'on soumet un document à annoter, il est dirigé vers l'annotateur disponible. Cette répartition de charge utilise l'algorithme Round Robin par défaut.

---

13. `https://gitlab.ikb.info.uqam.ca/netmail/corenlp-service`

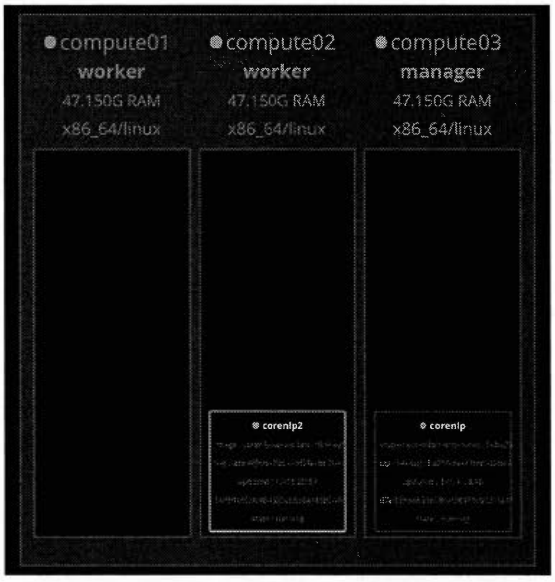

Figure 4.6: Services d'annotation dans Swarm

Figure 4.7: Services d'annotations - Facteur de réplication à 6

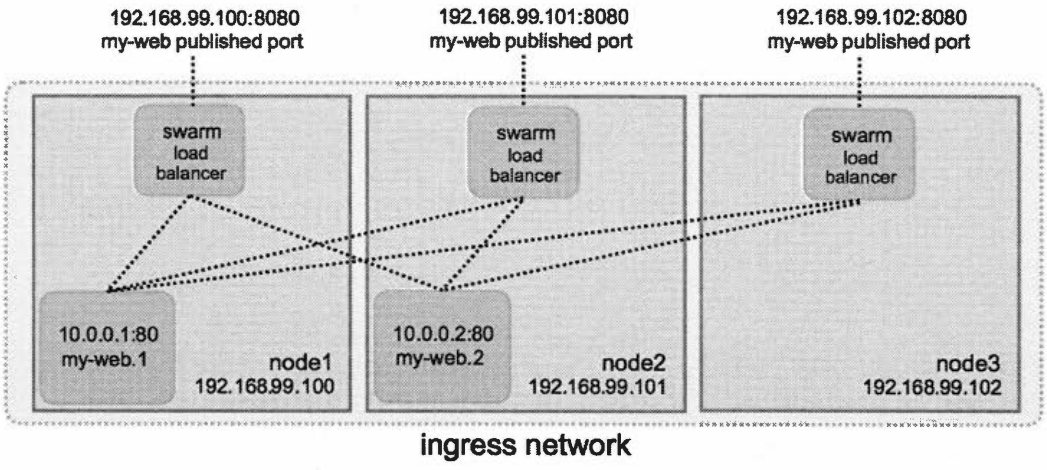

Figure 4.8: Docker Swarm - Equilibrage des charges

Grâce à notre infrastructure OpenStack, nous pouvons créer notre grappe Docker Swarm automatiquement à l'aide d'outils d'infrastructure en tant que code comme Terraform [14] et Ansible [15].

Le principe de l'infrastructure en tant que code consiste à définir un ensemble de ressources informatiques (machines virtuelles, réseaux, disques, règles de sécurité, groupes de stockage, etc) à l'aide d'un langage de configuration. Ces ressources peuvent ensuite être créées, déployées et supprimées dans une grappe de calcul de manière automatique et identique.

Dans notre projet, nous avons défini une infrastructure complète qui permet de déployer toute les ressources nécessaires à notre travail. Ce code est disponible sur notre répertoire GitLab [16].

---

14. `https://www.terraform.io/`

15. `https://www.ansible.com/`

16. `https://gitlab.ikb.info.uqam.ca/antoine/netmail-annotation-infrastructure`

Les ressources définies sont les suivantes :

- un réseau privé pour les communications des applications de calcul à l'échelle du *cluster*
- un ensemble de machines virtuelles pour les nœuds maîtres de Spark
- un ensemble de machines virtuelles pour les nœuds esclaves de Spark
- un réseau privé pour les communications du *cluster* Spark, ainsi que les règles de sécurité associées
- un ensemble de machines virtuelles pour les nœuds maîtres de Docker Swarm
- un ensemble de machines virtuelles pour les nœuds esclaves de Docker Swarm
- un réseau privé pour les communications du *cluster* Docker Swarm, ainsi que les règles de sécurité associées
- un ensemble de machines virtuelles pour les nœuds Solr de Solr Cloud
- un ensemble de machines virtuelles pour les nœuds Zookeeper de Solr Cloud
- un réseau privé pour les communications du *cluster* Solr Cloud, ainsi que les règles de sécurité associées

Une fois ces ressources en place, nous effectuons l'approvisionnement (installation des logiciels et dépendances) des machines virtuelles dans le même esprit à l'aide de l'outil Ansible.

A titre comparatif, la mise en place de notre infrastructure, en effectuant les tâches à la main, prend plus de deux jours. A l'aide de notre travail, et des outils libres que nous utilisons, la mise en place ne prend pas plus de 5 minutes.

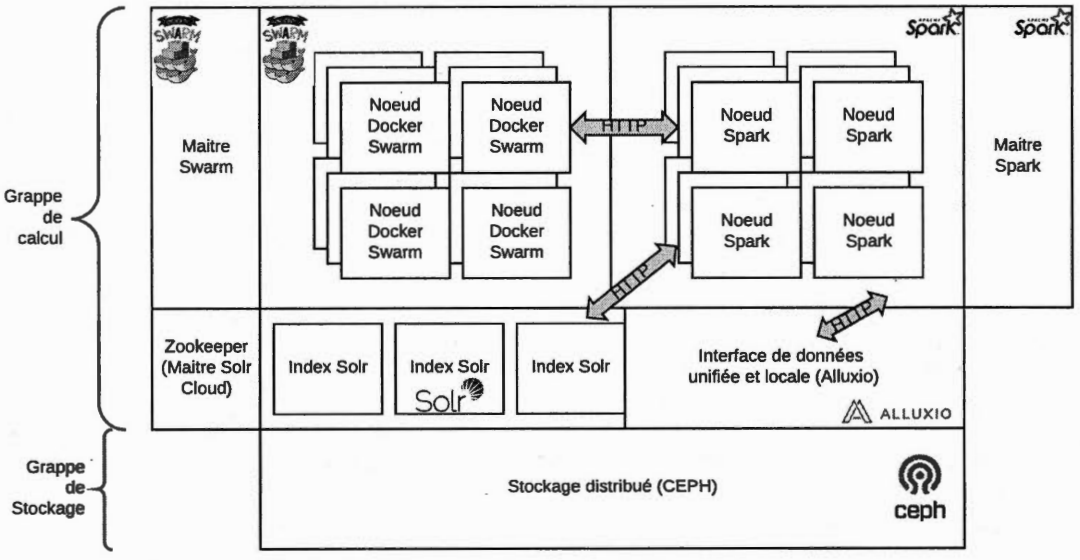

Figure 4.9: Architecture finale

4.6 Sélection de documents à annoter

A cette étape, notre infrastructure matérielle et logicielle est capable de supporter nos tâches basées sur nos corpus de données massives. Nous avons la capacité de détecter très rapidement des entités d'intérêt depuis une diversité de sources et avec un ensemble évolutif d'annotateurs.

Le travail principal est de supporter les traitements par lots effectués sur les entrepôts de données. Nous pouvons aussi avoir un besoin interactif pour le traitement de certains ensembles ou sous-ensembles des clients.

Grâce au travail précédent, il devient très simple de passer d'un mode de traitements par lots à un travail exploratoire et interactif sur un ou plusieurs des ensembles de données.

Afin de supporter le travail de création d'un annotateur basé sur un ensemble de données, nous avons eu besoin de procéder à l'annotation d'un sous-ensemble du

corpus de l'administration américaine. N'ayant pas la main d'œuvre pour annoter 3 millions de documents, nous avons procédé à la sélection d'un sous-ensemble de 1400 documents qui seront annotés par la suite.

L'objectif est d'obtenir un ensemble de données qui contiennent une distribution représentative des différents types d'entités présents dans le corpus complet.

Pour faire cela, nous avons procédé comme suit :

1. Annotation de la totalité du corpus, à l'aide des annotateurs publics et de notre annotateur personnalisé.
2. Exploration des annotations trouvées, suppression des documents non représentatifs
3. Sélection aléatoire d'un sous-ensemble
4. Annotation et évaluation du système

Dans notre cas, nous nous attarderons sur les trois premières étapes, la dernière ayant été traitée dans un travail parallèle à ce mémoire.

Afin d'explorer le corpus, nous allons nous appuyer sur la plateforme scientifique Jupyter Notebook[17]. C'est une plateforme d'analyse de données interactive qui s'exécute directement dans le navigateur.

Il est possible d'écrire du code en Python 2 et 3 ainsi que de prendre des notes en *Markdown* tout au long de la procédure. Il est aussi possible de supporter d'autre langages en ajoutant des *kernels*[18]. La figure 4.10 présente l'interface de Jupyter Notebook.

---

17. `https://jupyter.org/`

18. `https://github.com/jupyter/jupyter/wiki/Jupyter-kernels`

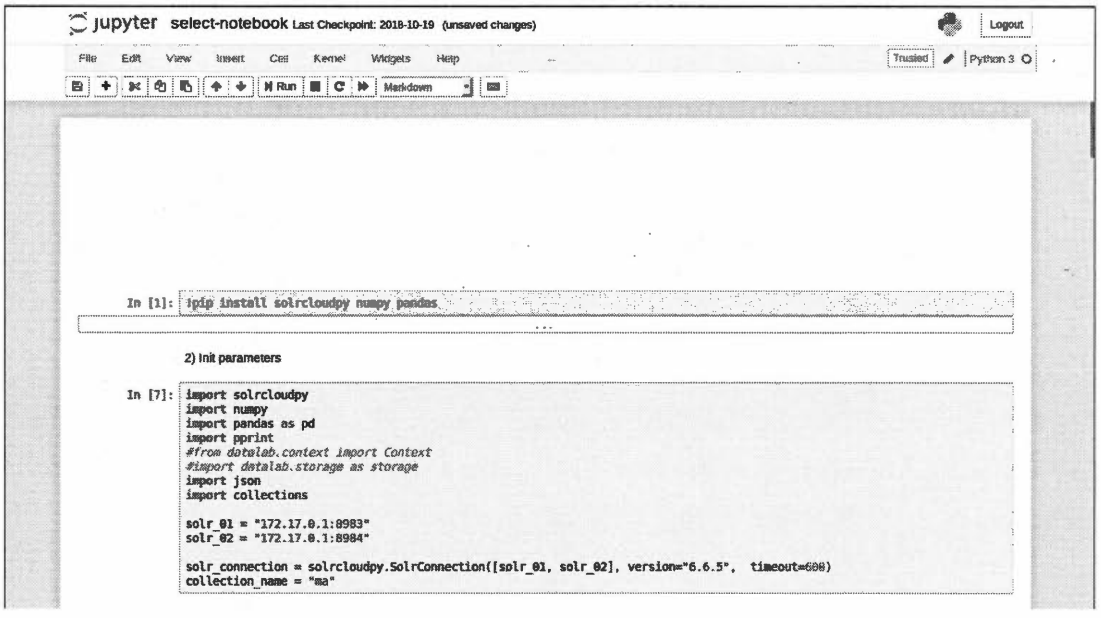

Figure 4.10: Interface d'exploration Jupyter Notebook

L'intérêt majeur de Jupyter Notebook c'est l'utilisation de *notebooks*. Ces *notebooks* fonctionnent avec des blocs logiques que l'on ajoute via l'interface. Chacun de ces blocs partagent un contexte commun, par exemple une librairie chargée dans un bloc est accessible dans un autre, mais l'exécution de ces blocs est faite séparément des autres.

Il est ainsi facile de faire varier des paramètres, de re-générer des graphes, des tables, sans avoir à relancer la totalité du programme. De plus ces notebooks supportent *Markdown*, ce qui permet de prendre des notes, que l'on peut formater avec une syntaxe simple, entre les blocs de code afin de documenter, expliquer, etc.

Enfin, Jupyter fonctionnant sur un environnement web, il est simple de le placer directement au cœur du *cluster* de calcul et donc d'exécuter le code au plus près des données.

Dans notre cas nous avons déployé cette plateforme à l'aide de Docker, sur le *cluster* de calcul, afin d'explorer notre index Solr Cloud, plus précisément le corpus annoté fourni par l'administration américaine.

Avant d'effectuer une sélection aléatoire, nous nous sommes intéressés au corpus annoté automatiquement afin de comprendre la distribution moyenne des entités et s'assurer que la sélection est faite sur un corpus suffisamment équilibré. Sur les 3571215 documents du corpus (messages et pièces jointes), nous avons pu extraire 25295571 annotations. Les statistiques du nombre d'annotations en fonction du type d'entités sont présentées dans le tableau 4.6.

Comme on peut le voir, la majeure partie des types d'entités retrouvées sont du type *NUMBER*. Si l'on s'intéresse à la distribution du nombre d'entités par document, on distingue nettement qu'une grande partie de ceux-ci contient moins de 2000 entités. Une infime partie de ces courriels détient un grand nombre d'annotations, dont une pièce-jointe avec plus de 175000 entités.

Le graphique 4.11 présente la distribution d'annotations par document.

En s'intéressant au document contenant 175000 entités, on se rend compte qu'il s'agit d'un fichier contenant les coins de rues de la ville. Cependant ces données géographiques (type *LOCATION*) ne sont pas utiles dans notre cas, car elles ne sont pas sensibles. Afin d'obtenir un ensemble de données que l'on peut exploiter pour l'annotation nous avons transformé ce jeu de données pour qu'il soit plus consistant. Ce raffinement est simplement effectué à l'aide de l'écart-type de la série. Nous retirons tous les documents avec un compte d'entités supérieur à la moyenne plus 3 fois l'écart-type de la série des comptes d'entités. Le résultat de ce raffinement est illustré par le graphique 4.12.

L'obtention d'un corpus annoté homogène nous a permis par la suite de sélection-

Tableau 4.6: Nombre d'entités par type au sein du corpus

Type d'entité	Total	Type d'entité	Total
AGE	1977	ORGANIZATION	2.26596e+06
CITY	12194	PATIENT	13445
COUNTRY	809	PERCENT	369979
CREDIT_CARD	82583	PERSON	2.77888e+06
DATE	3.98163e+06	PHONE	1.95602e+06
DOCTOR	3983	POSTAL_CODE_CA	2427
DURATION	773405	POSTAL_CODE_US	204812
EMAIL	1.08582e+06	PROFESSION	316
FAX	3	SET	166092
HOSPITAL	7843	SSN_CA	224215
IDNUM	591	SSN_FR	20142
LOCATION	795083	SSN_USA	5921
MEDICALRECORD	21522	STATE	64614
MISC	389520	TIME	2.33494e+06
MONEY	1.4133e+06	URL	103770
NUMBER	5.9167e+06	ZIP	61694
ORDINAL	235389		

ner un sous-ensemble à annoter à la main. L'objectif de cette tâche est de posséder un corpus qui nous permettra d'entraîner notre système d'annotation d'entités d'intérêt tout en possédant une base d'évaluation. Ce travail est décrit dans le mémoire (S.Zacharie, 2019). Dans cette partie nous venons de voir comment partir d'un ensemble de données massives afin de trouver des entités d'intérêts. Cet ensemble de données inerte nécessitait un temps de traitement très important, tellement qu'il n'était pas envisageable pour l'industrie d'exploiter ces données. À

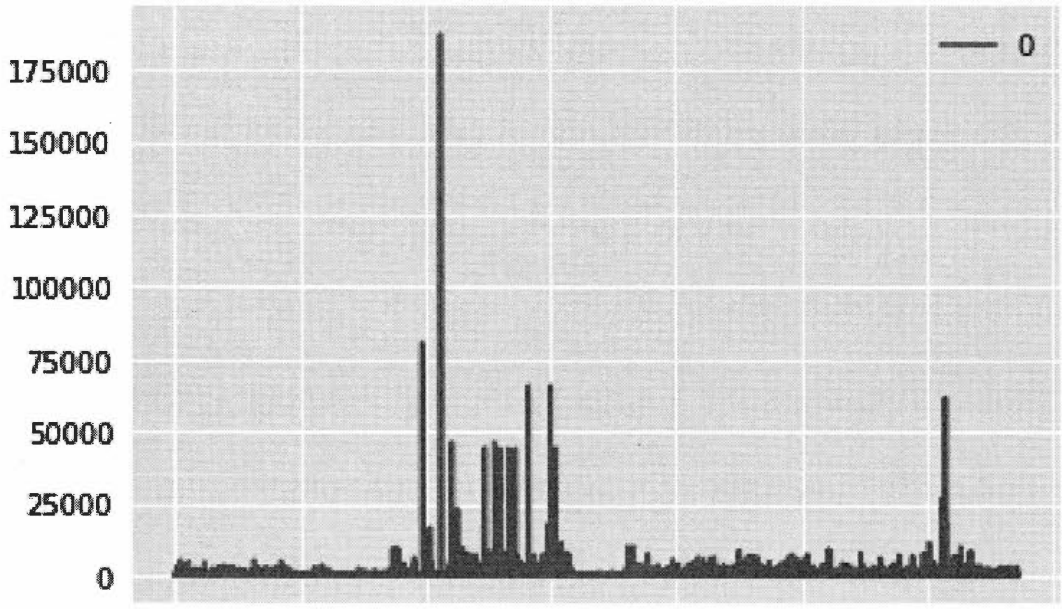

Figure 4.11: Nombre d'annotations par document - brute

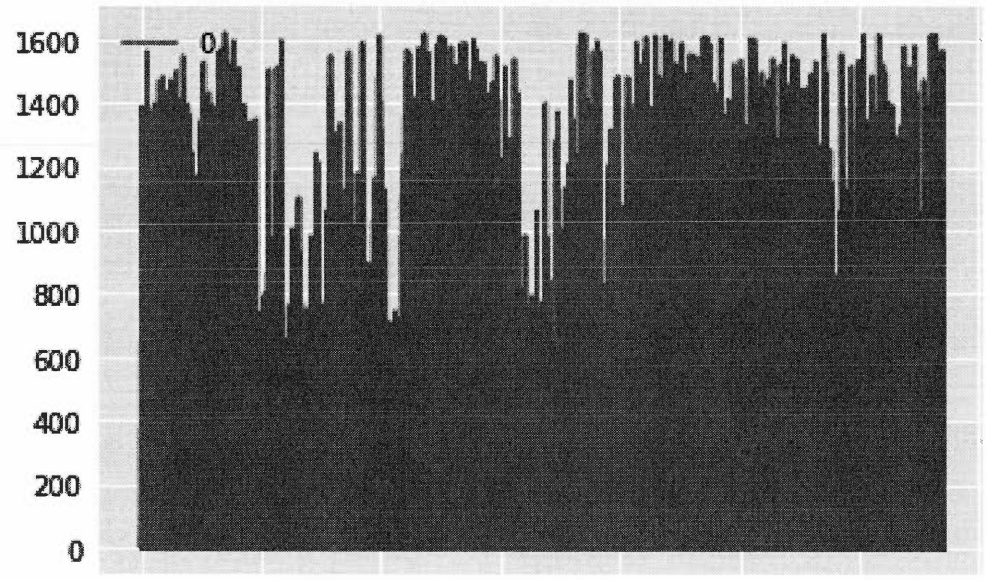

Figure 4.12: Nombre d'annotations par document - raffiné

Tableau 4.7: Statistique annotations par document (avant et après raffinement

	Brute	**Raffiné**
min	1	1
moyenne	75.25	64.62
écart type	515.54	116.28
max	187359	1620

l'aide de notre infrastructure de calcul et de stockage et des techniques d'indexation, de conteneurs et de distribution, nous avons réussi à obtenir des résultats en quelques heures. Le choix d'architecture ici fait a permis d'explorer finement les corpus sélectionnés afin d'en extraire une partie intéressante pour établir un système de classification de données sensibles. Ces travaux ont été créés pour être reproduits sur tous types d'ensembles de données textuelles et permettront par la suite d'explorer des données issues de l'industrie et ainsi d'aider notre partenaire industriel à assurer la conformité réglementaire de leurs clients.

Par la suite, nous allons voir une autre application des moteurs de recherche et d'infrastructure distribuées pour résoudre un problème de psychologie à l'aide d'intelligence artificielle et de données issues de média sociaux.

CHAPITRE V

APPLICATION DE LA RECHERCHE D'INFORMATIONS POUR LA CLASSIFICATION DE TEXTES

Les travaux présentés dans ce chapitre ont fait l'objet de plusieurs publications (Almeida et al., 2017b) (Almeida et al., 2017a) (Almeida et al., 2017c) (Briand et al., 2018a).

5.1 Introduction

Le contenu généré par les utilisateurs dans les médias sociaux a été largement exploré. Que ce soit pour la publicité, la création de profils ou l'analyse de sentiments. Les données sur ces médias ont également été utilisées pour élaborer des approches qui soutiennent les soins de santé mentale, en s'appuyant sur l'analyse de sentiments et les modèles statistiques (Rice et al., 2014). L'analyse du contenu généré par les utilisateurs peut fournir des renseignements précieux sur la santé mentale des utilisateurs. Elle peut aider à identifier les risques, et permettre de soutenir les utilisateurs déprimés.

Lors de nos travaux, nous avons été amenés à participer à la tâche eRisk : early Risk Prediction On The Internet[1]. La problématique est la suivante : sur un

1. `http://early.irlab.org/2017/index.html`

ensemble de productions écrites par des utilisateurs sur le réseau social reddit, comment déterminer le plus tôt possible le risque de dépression de ces utilisateurs ?

Afin de répondre à cette problématique, la conférence fournit un jeu de données d'entraînement à tous les participants. Le tableau 5.1 présente le corpus de données fourni pour la tâche.

Tableau 5.1: Statistiques du corpus de la tâche eRisk 2017

	Corpus d'entrainement		Corpus de test	
	non-risque	**risque**	**non-risque**	**risque**
# utilisateur	403	83	349	52
# documents	264,172	30,851	217,665	18,706
# moyen de documents par utilisateur	655.5	371.7	623.7	359.7
# moyen de mots par documents	21.3	27.6	22.5	26.9

Ce corpus est issu de la production d'utilisateurs sur le réseau social reddit. Il a été annoté par des experts en psychologie afin d'être utilisé comme données d'entraînement de systèmes de classification automatique d'utilisateurs en dépression.

Dans l'optique de classer le plus rapidement les utilisateurs, en fonction de leur risque de dépression, le jeu de test est découpé en 10, chaque morceau du jeu de test est donné de semaine en semaine et les équipes doivent rendre leurs prédictions de la semaine pour chaque utilisateur : risque, non-risque ou absence de décision.

Dans le cas où une décision est prise (risque ou non-risque), il n'est plus possible de revenir en arrière pour cet utilisateur dans la semaine qui suit. Ce mode d'évaluation est utilisé afin de calculer le score ERDE, qui prend en compte la rapidité

avec laquelle une décision a été prise.

5.2 Méthodologie

Notre système est basé sur une approche multisystème avec un mélange de technologies de l'apprentissage automatique et de la recherche d'information. La production des utilisateurs du réseau social est analysée par chaque système indépendamment et la prédiction de chaque sous-système permet de produire une décision finale. Le résultat est de la forme risque/non-risque (de dépression).

Le sous-système à base d'apprentissage machine s'appuie sur trois algorithmes de classification, et quatre types de représentations. Le tableau 5.2 présente les représentations utilisées.

Tableau 5.2: Nombre de caractéristiques uniques dans l'ensemble de données de formation eRisk 2017

	# Éléments
BOW (bag of words)	105,161
Bi-grams	1,544,714
Tri-grams	3,397,459
POS	118,139
Dictionnaire de sentiments	205
Dictionnaire de médicaments	30
Dictionnaire de drogues	57
Dictionnaire de maladies	43

Dans la table 5.2 nous présentons les représentations que nous utilisons dans

nos travaux. *Bag of words* (ou sac de mots) est une représentation simplifiée d'un document qui sera représenté par un sac des mots qu'il contient et leurs multiplicités, sans conserver la grammaire ou l'ordre des mots. Prenons la phrase d'exemple suivante :

Justin Pierre James Trudeau est le fils de Pierre Elliott Trudeau, 15e Premier ministre du Canada et de Margaret Trudeau.

La représentation en sac de mots de cette phrase serait :

```
{
    "Justin": 1,
    "Pierre": 2,
    "James": 1,
    "Trudeau": 2,
    "est": 1,
    "le": 1,
    "fils": 1,
    "de": 2,
    "Elliot": 1,
    "15e": 1,
    "Premier": 1,
    "ministre": 1,
    "du": 1,
    "Canada": 1,
    "et": 1,
    "Margaret": 1
}
```

Bi-grams et Tri-grams sont des représentations n-grams qui conservent l'ordre

d'apparition des mots d'un documents en utilisant une fenêtre de taille n. Si l'on prend le même exemple que plus haut la représentation Bi-grams ($n = 2$) serait :

```
{
    "Justin":   1,
    "Trudeau 15e",      1
    "Margaret Trudeau": 1,
    "de Margaret"       1,
    "et de":     1,
    "Canada et": 1,
    "du Canada": 1,
    "ministre du":      1,
    "Premier ministre": 1,
    "15e Premier"       1,
    "Elliott Trudeau":  1,
    "Justin Pierre":     1,
    "Pierre Elliott":   1,
    "de Pierre":        1,
    "fils de":  1,
    "le fils":  1,
    "est le":   1,
    "Trudeau est":      1,
    "James Trudeau":    1,
    "Pierre James":     1,
    "Trudeau": 1
}
```

Enfin POS pour *POS tagging* (*part-of-speech tagging*) ou Étiquetage morpho-syntaxique consiste à ajouter au document toutes les informations grammaticales

correspondantes. *Stanford Log-linear Part-Of-Speech Tagger*[2] est un outil qui permet d'étiqueter un document avec un large choix de langues. Toujours avec notre exemple de départ, l'étiquetage de la phrase serait :

Justin/NPP Pierre/NPP James/NPP Trudeau/NPP est/V le/DET fils/NC de/P Pierre/NPP Elliott/NPP Trudeau/NPP ,/PUNC 15e/DET Premier/ADJ ministre/NC du/P Canada/NPP et/CC de/P Margaret/NPP Trudeau/NPP ./PUNC

Avec **NPP** les noms propres, **V** les verbes, **DET** les déterminants, **NC** les noms communs, **P** les pronoms, **PUNC** les signes de ponctuation, **ADJ** les adjectifs et **CC** les conjonctions de coordinations.

Les trois modèles d'apprentissage supervisé ont été construits sur la base des classificateurs LMT (Logistic Model Tree) (Landwehr et al., 2005), un ensemble de classificateurs SMO (Sequential Minimal Optimization) (Platt, 1998) (ens_SMO), et un ensemble de classificateurs aléatoires de forêts (ens_RF). Les classificateurs SVM, qui sont similaires aux SMO, et Random Forest sont couramment utilisés dans des tâches comparables (Milne et al., 2016) montrant de bonnes performances. Les LMTs s'avèrent également performantes dans ces tâches, en particulier lors de la manipulation d'ensembles de données asymétriques (Almeida et al., 2016).

Le sous-système basé sur la recherche d'information est basé sur l'intuition suivante : si le message d'un utilisateur, dont la classe est inconnue, a une sémantique proche d'un ensemble de messages connus dont la classe est "risque", alors le message considéré devrait être classé à risque.

Ce sous-système, basé sur un moteur de recherche, évalue chaque texte à classer comme une requête qui est exécutée sur la base de l'index des documents

2. `https://nlp.stanford.edu/software/tagger.shtml`

manuellement annotés.

L'approche est basée sur l'outil Apache Solr[3] et Okapi BM25 (Jones et al., 2000). Okapi BM25 est une fonction de classement utilisée par les moteurs de recherche pour classer des documents en fonction de leur pertinence par rapport à une requête de requête donnée et s'appuie sur la représentation *Bag-of-Words*.

La décision de classification est prise comme suit : tous les messages d d'un nouvel utilisateur sont envoyés au moteur de recherche sous forme de requête.

Ensuite, les classes des n premiers documents retrouvés sont utilisées pour calculer le score $S_{IR}(d)$ comme indiqué dans l'équation 5.1. $S_{IR}(d)$ reflète la probabilité que d ait été produit par un utilisateur déprimé.

$$S_{IR}(d) = \frac{1}{n} \sum_{i=1}^{n} \delta(d_i) \tag{5.1}$$

avec d_i le document retrouvé par la requête d à la position i, et

$$\delta(d_i) = \begin{cases} 1, & \text{si } d_i \text{ possède la classe risque} \\ 0, & \text{sinon} \end{cases}$$

Au final nous avons présenté cinq systèmes différents à cette tâche :

— Système A : basé sur un algorithme de décision
— Système B : basé sur le sous-système de recherche d'informations seul
— Système C : LMT + BOW + dictionnaires
— Système D : ens_RF + BOW + dictionnaires
— Système E : ens_SMO + bigrams + dictionnaires

3. `http://lucene.apache.org/solr/`

Tableau 5.3: Étapes de pré-traitement

	Pré-traitement
Index	Segmentation (*Tokenization*)
	Tout en minuscule
	Racinisation (*Stemming*)
	Suppression des mots vides (*Stopwords*)
	Suppression de la ponctuation

Tableau 5.4: Champs indexés

#	**Champs indexés**
1	Titre
2	Contenu
3	Classe (risque/non-risque)
4	Text (combinaison de 1 + 2)

5.2.1 Métriques d'évaluation

L'évaluation de nos systèmes utilise les métriques suivantes :

Précision (*precision*)

La précision P_C en classification binaire est le rapport du nombre de documents correctement classés sur le nombre de documents attribués à la classe.

$$P_C = \frac{TP}{TP + FP}$$

Cette métrique correspond à la question : combien des éléments classés l'ont été correctement ?

Rappel (*recall*)

Le rappel R_C en classification binaire est le rapport du nombre de documents correctement classés sur le nombre total de documents de cette classe.

$$R_C = \frac{TP}{TP + FN}$$

Cette métrique correspond à la question : sur le nombre total d'élément à trouver, combien ont été découverts ?

F_1 Score

Le F_1 Score est la moyenne harmonique de la précision P_C et du rappel R_C.

$$F_1 = (\frac{R_C^{-1} + P_C^{-1}}{2}) = 2\frac{P_C \cdot R_C}{P_C + R_C}$$

ERDE (*Early Risk Detection Error*)

Cette métrique est définie dans (Losada et Crestani, 2016) comme une métrique pour une décision *dec* prise par un système donné avec retard k comme suit :

$$ERDE_o(dec, k) = \begin{cases} c_{fp} & \text{si } dec = \text{positif ET vérité} = \text{négatif (FP)} \\ c_{fn} & \text{si } dec = \text{négatif ET vérité} = \text{positif (FN)} \\ lc_o(k).c_{tp} & \text{si } dec = \text{positif ET vérité} = \text{positif (TP)} \\ 0 & \text{si } dec = \text{négatif ET vérité} = \text{négatif (TN)} \end{cases}$$

avec c_{fp} et c_{fn} les côuts des faux positifs et faux négatifs, $lc_o(k) \in [0, 1]$ correspond au coût qui augmente par rapport à la valeur de k, et calcule un coût pour une décision tardive de vrais positifs. Ce coût peut être considéré comme une pénalité qui augmente plus rapidement après qu'un certain nombre de documents de l'utilisateur o aient été observées. $lc_o(k)$ est défini comme suit :

$$lc_o(k) = 1 - \frac{1}{1 + e^{(k-o)}}$$

Selon la valeur de o, prendre une décision tardive est plus ou moins coûteux. Pour la tâche eRisk, les valeurs de $k = 5$ et $k = 50$ ont été imposées par les organisateurs.

Plus la valeur de ERDE est faible, plus une bonne décision a été prise rapidement.

5.3 Résultats

Le système A est basé sur les sous-systèmes à base d'apprentissage machine et de recherche d'information. La décision de classer un utilisateur en risque ou non est prise à l'aide de la formule suivante :

$$\Delta(d) = \mathbb{1}_{IR}(d) + \mathbb{1}_{SL}(d) + \mathbb{1}_{SLf}(d)$$

où d représente les nouveaux messages de l'utilisateur, et $\mathbb{1}_{IR}$, $\mathbb{1}_{SL}$, $\mathbb{1}_{SLf}$ sont des fonctions indicatrices de la classe de risque associée respectivement aux candidats

Tableau 5.5: Évaluation de nos systèmes présentés à eRisk 2017

Système	**ERDE 5**	**ERDE 50**	**F1**	**P**	**R**
A	14.03%	12.29%	**0.53**	0.48	**0.60**
B	13.78%	12.78%	0.48	0.49	0.46
C	13.58%	12.83%	0.42	0.50	0.37
D	**13.23%**	**11.98%**	0.38	**0.64**	0.27
E	13.68%	12.68%	0.39	0.45	0.35
Moy. 30 systèmes	14.68%	12.76%	0.40	0.37	0.51

IR, aux candidats SL et aux candidats SL de premier rang. Les résultats de ces fonctions indicatrices est égal à 1 si la décision est de classer un utilisateur comme étant à risque (de dépression), ou à 0 si l'utilisateur doit être classé comme étant sans risque.
Si $\Delta(d) \geq 2$, l'utilisateur qui a produit le document d est classé à risque.

Le tableau 5.5 présente les résultats de chacun de nos systèmes tandis que le tableau 5.6 présente les résultats des meilleurs systèmes de la tâche en fonction de la métrique utilisée.

Dans le cadre de cette tâche, notre meilleur système est le système A. En effet celui-ci atteint les meilleurs scores en termes de *F1-Score* et de Rappel, ce qui signifie que nous identifions un grand nombre des cas à risque. Certes, nous en identifions trop, toutefois dans ces applications, il est préférable d'avoir des patients (utilisateurs)

Tableau 5.6: Meilleurs systèmes par métrique, toutes équipes confondues

Équipe	**ERDE 5**	**ERDE 50**	**F1**	**P**	**R**
FHDOB	**12.70%**	10.39%	0.55	**0.69**	0.46
UNSLA	13.66%	**9.68%**	0.59	0.48	0.79
FHDOA	12.82%	9.69%	**0.64**	0.61	0.67
UArizonaC	17.93%	12.74%	0.34	0.21	**0.92**

classés à tort (FP) plutôt que de manquer ceux qui en ont vraiment besoin (FN).

Ce travail a été réalisé sur l'infrastructure de calcul décrite 3.

Bien que les jeux de données utilisés soient hors du champ des données massives, nous avons gagné beaucoup de temps pour l'exécution de nos nombreuses tentatives.

En effet, ne possédant pas assez de ressources pour exécuter nos différentes configurations en parallèle, il était facile de modifier notre infrastructure de test en quelques instants.

Le jeu de données étant stocké dans CEPH et rendu disponible au travers d'Alluxio, un seul point d'entrée pour les données nous a évité les problématiques de ne pas travailler sur les bons jeux de données ou avoir à se transférer le jeu de données constamment. Enfin, cette approche se basant sur reddit, travailler sur une infrastructure type nous a permis de construire des approches qui pourraient être utilisées sur le média social au complet (plus de 2 milliards de messages). Le seul changement que nous devrions faire est d'ajouter de la capacité de calcul afin de raccourcir les temps de calcul.

Le code de ce travail est public et accessible sur notre répertoire GitHub[4].

4. https://github.com/BigMiners/eRisk2017

CHAPITRE VI

CONCLUSION

À l'aide de matériel de récupération, ou d'un matériel non spécifique, et de systèmes dont le code est sous licence libre, nous avons réussi à créer une infrastructure capable de traiter des données massives à l'échelle.

Le couple d'infrastructure en tant que code et en tant que service nous a permis de facilement maintenir et modifier tout le système avec cohérence et reproductibilité.

L'adaptabilité de ce système nous a permis de mener à bien deux projets à base de données massives, que ce soit pour supporter la recherche en conformité réglementaire au sein d'archives et de données ou la recherche en traitement des langues naturelles alliée à la psychologie des usagers des médias sociaux.

Grâce à l'infrastructure de stockage et à son interface unifiée, compatible Hadoop et Spark, nous pouvons exploiter différentes données et différentes sources depuis une même application dans le but de construire des applications complexes, en lien avec les données massives.

Nous avons vu qu'il est possible de traiter des quantités de données importantes en un temps très largement réduit, grâce aux techniques d'indexation et l'utilisation de paradigmes de programmation comme MapReduce couplé au très performant système de RDD d'Apache Spark.

Étant donné que les technologies retenues reposent sur des communautés très actives, il est tout à fait envisageable d'utiliser ce travail dans un cadre de production, pour un laboratoire de recherche ou une entreprise qui aurait besoin d'exploiter les données dont elle dispose.

La flexibilité de l'infrastructure lui permet de passer à l'échelle horizontale pratiquement indéfiniment. En fonction de l'échelle, il faut modifier les configurations des systèmes afin de prendre en compte plusieurs niveaux de fonctionnement, plusieurs réseaux et sous-réseaux, etc.

Dans ce document, il n'est pas fait mention des notions de sauvegarde et restauration de la grappe. Un système de production devrait prévoir les pannes des nœuds responsables de la gestion des organes de la grappe (bases de données, queues, etc.).

Nos projets étant terminés, cette infrastructure est maintenant à disposition de notre équipe de recherche afin de lui permettre à elle aussi d'effectuer ses recherches.

Les articles suivant ont été publiés pendant ces travaux :

Antoine Briand, Sara Zacharie, Ludovic Jean-Louis, Marie-Jean Meurs "Identification of Sensitive Content in Data Repositories to Support Personal Information Protection", The 31st International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2018, Springer, pp.898-910 Montreal, QC, Canada, June 25 - 28, 2018

Antoine Briand, Hayda Almeida, Marie-Jean Meurs "Analysis of Social Media Posts for Early Detection of Mental Health Conditions", The 31st Canadian Conference on Artificial Intelligence, Springer, pp. 133-143. Toronto, ON, Canada, May 8 - 11, 2018

Hayda Almeida, Antoine Briand, Diego Maupomé, Marie-Jean Meurs "Prediction of Depression from User-generated Content in Social Media", Montreal IA Symposium 2017, Montreal, QC, Canada, September 29, 2017

Hayda Almeida, Antoine Briand, Marie-Jean Meurs "Detecting Early Risk of Depression from Social Media User-generated Content", eRisk - CLEF 2017, Dublin, Ireland, September 11 - 14, 2017

Hayda Almeida, Antoine Briand, Marie-Jean Meurs "Predicting Depression from User-generated Content in Social Media", WiNLP Workshop at ACL 2017, Vancouver, BC, Canada, July 30 - August 4, 2017

RÉFÉRENCES

Affaires Mondiales, Canada (2016). Les mégadonnées au Canada. Accédé le 23 Août 2017. Récupéré de `http://www.international.gc.ca/investors-investisseurs/assets/pdfs/download/Secteurs_de_Pointe-Megadonnees.pdf`

Almeida, H., Briand, A., Maupome, D. et Meurs, M.-J. (2017a). Prediction of depression from user-generated content in social media. *Montreal IA Symposium.*

Almeida, H., Briand, A. et Meurs, M.-J. (2017b). Detecting early risk of depression from social media user-generated content. *Working Notes of CLEF.*

Almeida, H., Briand, A. et Meurs, M.-J. (2017c). Predicting depression from user-generated content in social media. *WiNLP Workshop at ACL 2017.*

Almeida, H., Jean-Louis, L. et Meurs, M.-J. (2018). An open source and modular search engine for biomedical literature retrieval. *Computational Intelligence*, *34*(1), 200–218.

Almeida, H., Queudot, M. et Meurs, M.-J. (2016). Automatic Triage of Mental Health Online Forum Posts : CLPsych 2016 System Description. Dans *Proceedings of the Third Workshop on Computational Linguistics and Clinical Psychology*, 183–187.

Andrade, P., Bell, T., Van Eldik, J., McCance, G., Panzer-Steindel, B., dos Santos, M. C., Traylen, S. et Schwickerath, U. (2012). Review of CERN Data Centre Infrastructure. Dans *Journal of Physics : Conference Series*, volume 396, p. 042002. IOP Publishing.

Bell, T., Bompastor, B., Bukowiec, S., Leon, J. C., Denis, M., van Eldik, J., Lobo, M. F., Alvarez, L. F., Rodriguez, D. F., Marino, A. et al. (2015). Scaling the CERN OpenStack cloud. Dans *Journal of Physics : Conference Series*, volume 664, p. 022003. IOP Publishing.

Blei, D. M., Ng, A. Y. et Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, *3*(Jan), 993–1022.

Borthakur, D. (2007). The Hadoop Distributed File System : Architecture and Design. Récupéré de `https://svn.eu.apache.org/repos/asf/hadoop/common/tags/release-0.16.3/docs/hdfs_design.pdf`

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32.

Briand, A., Almeida, H. et Meurs, M.-J. (2018a). Analysis of social media posts for early detection of mental health conditions. Dans *Advances in Artificial Intelligence : 31st Canadian Conference on Artificial Intelligence, Canadian AI 2018, Toronto, ON, Canada, May 8–11, 2018, Proceedings 31*, 133–143. Springer.

Briand, A., Zacharie, S., Jean-Louis, L. et Meurs, M.-J. (2018b). Identification of sensitive content in data repositories to support personal information protection. Dans *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, 898–910. Springer.

Büttcher, S., Clarke, C. L. et Lushman, B. (2006). Term proximity scoring for ad-hoc retrieval on very large text collections. Dans *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, 621–622. ACM.

Chen, H., Chiang, R. H. et Storey, V. C. (2012). Business intelligence and analytics : from big data to big impact. *MIS quarterly*, 1165–1188.

Dean, J. et Ghemawat, S. (2008). MapReduce : Simplified Data Processing on Large Clusters. *Communications of the ACM*, *51*(1), 107–113.

Demchenko, Yuri et Grosso, P. e. D. L. C. e. M. P. (2013). Addressing big data issues in scientific data infrastructure. Dans *Collaboration Technologies and Systems (CTS), 2013 International Conference on*, 48–55. IEEE.

Department of Justice, US Government (1974). Privacy Act of 1974. Accédé le 17 juillet 2017. Récupéré de `https://www.justice.gov/opcl/privacy-act-1974`

Dernoncourt, F., Lee, J. Y. et Szolovits, P. (2017). Neuroner : an easy-to-use program for named-entity recognition based on neural networks. *arXiv preprint arXiv :1705.05487*.

Dernoncourt, F., Lee, J. Y., Uzuner, O. et Szolovits, P. (2016). De-identification of patient notes with recurrent neural networks. *Journal of the American Medical Informatics Association (JAMIA)*.

Donvito, G., Marzulli, G. et Diacono, D. (2014). Testing of several

distributed file-systems (hdfs, ceph and glusterfs) for supporting the hep experiments analysis. Dans *Journal of Physics : Conference Series*, volume 513, p. 042014. IOP Publishing.

Finkel, J. R., Grenager, T. et Manning, C. (2005). Incorporating non-local information into information extraction systems by gibbs sampling. Dans *Proceedings of the 43rd annual meeting on association for computational linguistics*, 363–370. Association for Computational Linguistics.

Gandomi, A. et Haider, M. (2015). Beyond the hype : Big data concepts, methods, and analytics. *International Journal of Information Management*, *35*(2), 137–144.

Gilbert, S. et Lynch, N. (2002). Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *Acm Sigact News*, *33*(2), 51–59.

Goel,Vindu et Perlroth, Nicole (2016). Yahoo Says 1 Billion User Accounts Were Hacked. Récupéré de `https://www.nytimes.com/2016/12/14/technology/yahoo-hack.html`

Goodrow, Cristos (2017). You know what's cool ? A billion hours. Accédé le 28 décembre 2018. Récupéré de `https://youtube.googleblog.com/2017/02/you-know-whats-cool-billion-hours.html`

Gouvernement du Canada (13 avril, 2000). The Personal Information Protection and Electronic Documents Act (PIPEDA). Accédé le 17 juillet 2017. Récupéré de `https://laws-lois.justice.gc.ca/fra/lois/P-8.6/index.html`

Hadoop, A. (2009). Hadoop.

Hammond, K. W., Laundry, R. J., OLeary, T. M. et Jones, W. P. (2013). Use of text search to effectively identify lifetime prevalence of suicide attempts among veterans. Dans *Proceedings of the 46th Hawaii International Conference on System Sciences (HICSS)*, 2676–2683. IEEE.

Huang, C.-C. et Lu, Z. (2015). Community challenges in biomedical text mining over 10 years : success, failure and the future. *Briefings in bioinformatics*, *17*(1), 132–144.

Johnson, A. E., Pollard, T. J., Shen, L., Li-wei, H. L., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Celi, L. A. et Mark, R. G. (2016). Mimic-iii, a freely accessible critical care database. *Scientific data*, *3*, 160035.

Jones, K. S., Walker, S. et Robertson, S. E. (2000). A probabilistic model of

information retrieval : development and comparative experiments : Part 2. *Information Processing & Management*, *36*(6), 809–840.

Kevin Wilfong, P. V. (2014). Scaling the Facebook Data Warehouse to 300 PB. Récupéré de `https://code.facebook.com/posts/229861827208629/scaling-the-facebook-data-warehouse-to-300-pb/`

Kwon, O., Lee, N. et Shin, B. (2014). Data quality management, data usage experience and acquisition intention of big data analytics. *International Journal of Information Management*, *34*(3), 387–394.

Lafferty, J., McCallum, A. et Pereira, F. C. (2001). Conditional random fields : Probabilistic models for segmenting and labeling sequence data.

Landwehr, N., Hall, M. et Frank, E. (2005). Logistic model trees. *Machine Learning*, *59*(1-2), 161–205.

Laney, D. (2001). 3d data management : controlling data volume, variety and velocity, meta group file 949.

Leung, S. A. W. A. W. et Maltzahn, S. A. B. C. Rados : A scalable, reliable storage service for petabyte-scale storage clusters.

Losada, D. E. et Crestani, F. (2016). A test collection for research on depression and language use. Dans *International Conference of the Cross-Language Evaluation Forum for European Languages*, 28–39. Springer.

Maltzahn, C., Molina-Estolano, E., Khurana, A., Nelson, A. J., Brandt, S. A. et Weil, S. (2010). Ceph as a scalable alternative to the hadoop distributed file system. *login : The USENIX Magazine*, *35*, 38–49.

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. et McClosky, D. (2014). The Stanford CoreNLP Natural Language Processing Toolkit. Dans *Proceedings of 52nd annual meeting of the association for computational linguistics : system demonstrations*, 55–60.

Manning, C. D., Manning, C. D. et Schütze, H. (1999). *Foundations of statistical natural language processing*. MIT press.

McCandless, M., Hatcher, E. et Gospodnetic, O. (2010). *Lucene in action : covers Apache Lucene 3.0*. Manning Publications Co.

Milne, D. N., Pink, G., Hachey, B. et Calvo, R. A. (2016). CLPsych 2016 Shared Task : Triaging content in online peer-support forums. Dans *CLPsych@ HLT-NAACL*, 118–127.

Mukherjee, A. *et al.* (2015). Benchmarking hadoop performance on different

distributed storage systems.

Nadeau, D. et Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, *30*(1), 3–26.

Papineni, K., Roukos, S., Ward, T. et Zhu, W.-J. (2002). Bleu : a method for automatic evaluation of machine translation. Dans *Proceedings of the 40th annual meeting on association for computational linguistics*, 311–318. Association for Computational Linguistics.

Parlement Européen (27 avril, 2016). Reglementation Générale sur la Protection des Données. Accédé le 17 juillet 2017. Récupéré de `https://eur-lex.europa.eu/legal-content/FR/TXT/PDF/?uri=CELEX:32016R0679&from=FR`

Platt, J. (1998). *Sequential minimal optimization : A fast algorithm for training support vector machines.* Rapport technique MSR-TR-98-14, Microsoft.

Radio Canada (30 novembre, 2018). Gigantesque vol de données chez Marriott International. Accédé le 04 décembre 2017. Récupéré de `https://ici.radio-canada.ca/nouvelle/1139006/gigantesque-vol-donnees-marriott-international-hotel-500-millions`

Rajman, M. et Besançon, R. (1998). Text mining : natural language techniques and text mining applications. In *Data mining and reverse engineering* 50–64. Springer.

Rice, S. M., Goodall, J., Hetrick, S. E., Parker, A. G., Gilbertson, T., Amminger, G. P., Davey, C. G., McGorry, P. D., Gleeson, J. et Alvarez-Jimenez, M. (2014). Online and social networking interventions for the treatment of depression in young people : a systematic review. *Journal of Medical Internet Research (JMIR)*, *16*(9), e206.

Robertson, S. (2004). Understanding inverse document frequency : on theoretical arguments for idf. *Journal of documentation*, *60*(5), 503–520.

Rosenberg, Matthew, Confessore, Nicholas et Carole Cadwalladr (2018). . Récupéré le 2018 de `https://www.nytimes.com/2018/03/17/us/politics/cambridge-analytica-trump-campaign.html`

Salton, G. et McGill, M. J. (1986). Introduction to modern information retrieval.

Schroeck, M., Shockley, R., Smart, J., Romero-Morales, D. et Tufano, P. (2012). Analytics : the real-world use of big data : How innovative enterprises

extract value from uncertain data, executive report. *IBM Institute for Business Value and Said Business School at the University of Oxford.*

Shi, J., Qiu, Y., Minhas, U. F., Jiao, L., Wang, C., Reinwald, B. et Özcan, F. (2015). Clash of the titans : Mapreduce vs. spark for large scale data analytics. *Proceedings of the VLDB Endowment*, *8*(13), 2110–2121.

Shvachko, K., Kuang, H., Radia, S. et Chansler, R. (2010). The hadoop distributed file system. Dans *Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium on*, 1–10. Ieee.

Silva, C. et Ribeiro, B. (2003). The importance of stop word removal on recall values in text categorization. Dans *Neural Networks, 2003. Proceedings of the International Joint Conference on*, volume 3, 1661–1666. IEEE.

Stenetorp, P., Pyysalo, S., Topić, G., Ohta, T., Ananiadou, S. et Tsujii, J. (2012). Brat : a web-based tool for nlp-assisted text annotation. Dans *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, 102–107. Association for Computational Linguistics.

Stubbs, A., Kotfila, C. et Uzuner, Ö. (2015). Automated systems for the de-identification of longitudinal clinical narratives : Overview of 2014 i2b2/uthealth shared task track 1. *Journal of biomedical informatics*, *58*, S11–S19.

Stubbs, A. et Uzuner, Ö. (2015). Annotating longitudinal clinical narratives for de-identification : The 2014 i2b2/uthealth corpus. *Journal of biomedical informatics*, *58*, S20–S29.

S.Zacharie (2019). *Identification Des Informations Sensibles Dans Des Sources De Données Hétérogènes.* (Mémoire de maîtrise).

Tang, G., Pei, J. et Luk, W.-S. (2014). Email mining : tasks, common techniques, and tools. *Knowledge and Information Systems*, *41*(1), 1–31.

Toor, S., Toebbicke, R., Resines, M. Z. et Holmgren, S. (2012). Investigating an open source cloud storage infrastructure for cern-specific data analysis. Dans *Networking, Architecture and Storage (NAS), 2012 IEEE 7th International Conference on*, 84–88. IEEE.

United States Government Accountability Office (2018). Récupéré de `https://www.gao.gov/assets/700/694158.pdf`

U.S. Department of Health & Human Services (1996). Health Insurance Portability and Accountability Act of 1996 (HIPAA). Accédé le 17 juillet

2017. Récupéré de https://www.hhs.gov/hipaa/for-professionals/security/laws-regulations/index.html

Veritas, Inc (2016). The databerg report. Accédé le 28 décembre 2018. Récupéré de https://www.veritas.com/content/dam/Veritas/docs/reports/veritas-strike-global-report_a4-sdc2.pdf

Weil, S. A., Brandt, S. A., Miller, E. L., Long, D. D. et Maltzahn, C. (2006a). Ceph : A scalable, high-performance distributed file system. Dans *Proceedings of the 7th symposium on Operating systems design and implementation*, 307–320. USENIX Association.

Weil, S. A., Brandt, S. A., Miller, E. L. et Maltzahn, C. (2006b). Crush : Controlled, scalable, decentralized placement of replicated data. Dans *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, p. 122. ACM.

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

Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M. J., Shenker, S. et Stoica, I. (2012). Resilient distributed datasets : A fault-tolerant abstraction for in-memory cluster computing. Dans *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, 2–2. USENIX Association.

Zhong, Raymond (04 décembre, 2018). Quora, the Q. and A. Site, Says Data Breach Affected 100 Million Users. Accédé le 04 décembre 2017. Récupéré de https://ici.radio-canada.ca/nouvelle/1139006/gigantesque-vol-donnees-marriott-international-hotel-500-millions