REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Ecole Nationale Polytechnique

Département Electronique

Mémoire de projet de fin d'études
pour l'obtention du diplôme d'ingénieur d'état en électronique

Conception et mise en œuvre de l'architecture d'une plateforme Cloud computing dédiée à IoT-IA

Mr Noureddine HENKA
Mr Sami Issam KHANFRI
Sous la direction de Mme. Nour-El-Houda BENALIA

Présenté et soutenu publiquement le (30/06/2019)

Composition du jury

Président	**Mr Mourad ADNANE**	**Dr**	**ENP**
Promoteur	**Mme Nour-El-Houda BENALIA**	**Dr**	**ENP**
Examinateur	**Mme Nesrine BOUADJENEK**	**Dr**	**ENP**
Co-promoteur	**Mme Widad KARTOUS**	**Dr**	**ENP**

ENP 2019

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Ecole Nationale Polytechnique

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

Département d'Electronique

Mémoire de projet de fin d'études
pour l'obtention du diplôme d'ingénieur d'état en électronique

Conception et mise en œuvre de l'architecture d'une plateforme Cloud computing dédiée à IoT-IA

Mr Noureddine HENKA
Mr Sami Issam KHANFRI
Sous la direction de Mme. Nour-El-Houda BENALIA

Présenté et soutenu publiquement le (30/06/2019)

Composition du jury

Président	**Mr Mourad ADNANE**	**Dr**	**ENP**
Promoteur	**Mme Nour-El-Houda BENALIA**	**Dr**	**ENP**
Examinateur	**Mme Nesrine BOUADJENEK**	**Dr**	**ENP**
Co-promoteur	**Mme Widad KARTOUS**	**Dr**	**ENP**

ENP 2019

ملخص عام

أنترنات الأشياء تهدف إلى وصل العديد من الأجهزة والأدوات بشبكة الأنترنات، ما يعطينا الكثير من التحكم بالعالم المادي، ويجعله أكثر ذكاء وموافقة لاحتياجاتنا. واحدة من أهم الوظائف لهذه التكنولوجيا هي جمع المعلومات في جميع المجالات، من أجل استغلالها، وتحليلها وخلق ذكاء اصطناعي قادر على اتخاذ قرارات بصفة مستقلة.

هذا المشروع يتضمن وضع معمارية برمجية تحمل منصة تحليل للبيانات بتقنيات الذكاء الاصطناعي من أجل تطبيقات في مجال انترنت الأشياء. هذه المنصة قائمة على تحليل البيانات الضخمة، ضمن معمارية سحابية، من أجل تحسين أدائها أمام البيانات الكبيرة والحسابات اللازمة. إنها تسمح بتطوير العديد من التطبيقات عالية الأداء التي بإمكانها حل مشاكل اقتصادية واجتماعية كحركة سير المرور، استهلاك المياه، الطاقة، ...

كلمات دالة: أنترنات الأشياء، الذكاء الاصطناعي، البيانات الضخمة، السحابة، منصة ذكاء اصطناعي.

Abstract

The Internet of Things aims to connect a lot of devices and tools to the Internet network, which will give us a lot of control over the physical world, and make it more smart and adaptable to our needs. One of the main functions of such a technology is to collect data on all areas, in order to exploit them, analyze them and create artificial intelligence to make decisions in an autonomous way.

This project involves the implementation of a software architecture supporting an AI analysis platform for IoT applications. This platform is based on Big Data analysis, on a Cloud architecture, in order to improve its performance in the face of massive data and the necessary computations. It allows to have several high performance applications that can solve socio-economic problems, such as road traffic, water consumption, energy...

Keywords: Internet of things, artificial intelligence, Big Data, Cloud, AI Platform.

Résumé

L'internet des objets a pour objectif de connecter pleins d'appareils et d'outils au réseau internet, ce qui va permettre d'avoir énormément de contrôle sur le monde physique, et le rendre plus intelligent et adaptable à nos besoins. L'une des principales fonctions d'une telle technologie et de récolter des données sur tous les domaines, afin de les exploiter, analyser et créer de l'intelligence artificielle pour prendre des décisions d'une manière autonome.

Ce projet porte sur la mise en œuvre d'une architecture logicielle portant une plateforme d'analyse AI pour des applications IoT. Cette plateforme est fondé avec de l'analyse des données en Big Data, sur une architecture Cloud, afin d'améliorer sa performance face à la massivité des données et du calcul nécessaire. Elle permet d'avoir plusieurs applications de haute performance qui peuvent résoudre des problèmes socio-économiques, comme le trafic routier, la consommation de l'eau, l'énergie,...

Mots clés : internet des objets, intelligence artificielle, données passives, nuage, plateforme IA.

Dédicace

Nous dédions ce modeste travail à nos très chers parents pour l'effort fournie et les immenses sacrifices consentis pour nos éducations et nos succès. Que dieu vous protège et vous garde pour nous.

nous dédions ce travail également :

À nos chers frères et chères sœurs, et leurs familles, à qui nous souhaitons beaucoup de réussites,

À nos grands-pères et nos grandes mères pour leur Douaa,

À tous les membres de nos familles : oncles, tantes, cousins et cousines,

À tous nos enseignants du primaire jusqu'à l'ENP, pour leur contribution à nos éducations,

À tous nos amis à l'école polytechnique,

À ceux qui nous ont aidé de près ou de loin pour réaliser ce travail.

Remerciement

***T**out d'abord, nous tenons à remercier « **Allah** », le clément et le miséricordieux de nous avoir donné la force et le courage de mener à bien ce modeste travail.*

Nous adressons nos remerciements à nos parents, qui nous ont apporté leurs aides, leurs sacrifices loyaux et leurs soutiens tout au long de nos études.

Nous tenons à remercier nos promoteurs pour leur accueil et leur professionnalisme. Ils ont su nous donner le gout du travail bien fait et un excellent aperçu du monde de travail.

Au-delà du mémoire, P.RABAH Sadoun a su nous écouter et nous accompagner par ses grandes qualités humaines. Il nous a transmis un bon bagage de connaissances qui nous seront n´nécessaires par la suite. Nous ne la remercierons jamais assez.

Un merci particulier à nos co-promoteurs Dr Widad Kartous et Dr Nour El Houda BENALIA pour leurs remarques et critiques constructives.

***N**os remerciements vont également à toutes les personnes qui ont, de près ou de loin, apporté aide et encouragement.*

Table des matières

Liste des tableaux

Liste des figures :

Liste des abréviations

AI: Artificial intelligence

IOT: internet of things

SaaS: Software as a service

PaaS: Platform as a service

IaaS: Infrastructure as a service

RDD: Resilient Distributed Dataset

JVM: Java Virtual Machine

GC: Garbage Collector

API: Application Programming Interface

DAG: Direct Acyclic Graph

MLlib: Machine learning library

JaaS: Juju as a service

MaaS: Metal as a service

PXE: Pre-boot eXecution Environment

OS: Operating System

VM: Virtual Machine

VCPU: Virtual Central Processing Unit

DB: DataBase

SGBD: Serveur de Gestion Base de données

SQL: Sturctured Query Language

NoSql: Not only SQL

DFS: Distributed file system

HDFS: Hadoop distributed file system

WebUI: Web User Interface

GUI: Graphical User interface

OBD: on board diagnostic

Introduction générale

La convergence des mondes physique et numérique est aujourd'hui une réalité industrielle et scientifique. La technologie devient de plus en plus un moteur essentiel du fonctionnement de notre société, redéfinissant ainsi le paysage dans lequel nous vivons et travaillons. Les organisations et les gouvernements doivent s'adapter pour tirer parti de ce paysage changeant dans l'intérêt de tous. Il ne fait aucun doute que l'intégration rapide du monde physique et numérique d'aujourd'hui commence à modifier fondamentalement de nombreux aspects de notre vie et de nos activités. En tant que société, nous voulons plus de durabilité et moins de déchets, moins d'impact climatique, des services plus mobiles et personnalisés et une plus grande inclusion. Telles sont quelques-unes des promesses des questions que nous pouvons aborder avec l'internet des objets (IoT). Le Forum économique mondial a qualifié cette situation de quatrième révolution industrielle. Cette opportunité est rendue possible par la convergence des mondes physique et numérique et est motivée par l'avènement de la connectivité omniprésente, des données massives, de l'analyse et du Cloud computing qui se combinent pour créer l'internet des objets.

Selon l'étude, environ 4,4 billions de Go de données seront générés d'ici 2020 grâce à l'Internet des objets. C'est sans doute difficile à comprendre facilement. Cependant, avec le nombre croissant d'appareils connectés, il n'est pas surprenant que d'ici 2020, plus de dix milliards de capteurs et d'appareils seront connectés à Internet. De plus, tous ces appareils recueilleront, analyseront, partageront et transmettront des données en temps réel. Par conséquent, sans les données, les dispositifs IoT n'auraient pas les fonctionnalités et les capacités qui leur ont permis d'attirer tant d'attention dans le monde.

Par ce fait, la capacité à connecter de plus en plus d'objets implique d'être en mesure de traiter les milliards de données captées et de les valoriser efficacement. Nous sommes face à une problématique de surproduction de la donnée qu'il faut pouvoir capter, analyser, agréger, transformer pour obtenir une information intéressante à utiliser. C'est l'une des raisons pour lesquelles on utilise de l'Intelligence Artificielle ou du cognitif et des techniques d'analyse sémantique de l'information, d'apprentissage (Machine Learning) ou de Deep Learning. Cela nous permet, dans ce flot considérable de données, de détecter l'information pertinente et de la traiter rapidement.

Ce projet vise à résoudre l'ensemble de ces problématiques, et de mettre en œuvre une solution performante, extensible et mise à l'échelle afin de pouvoir créer une intelligence industrielle dans le cadre des technologies IoT.

Chapitre 1 :

Plateforme AI : Technologies et besoins

1 Plateforme AI : Technologies et besoins

1.1 Introduction

Depuis de nombreuses années, nous avons voulu construire des ordinateurs de type cérébral qui peuvent imiter ou augmenter les fonctions humaines en matière de détection, de mémoire, de reconnaissance et de compréhension. Pour cette raison, la première partie s'engage à synthétiser les résultats de la recherche bibliographique effectuée qui gravite autour de l'intelligence artificielle, l'apprentissage automatique, les outils de traitement intelligent des données et le Cloud Computing.

Pour cela, nous commençons notre étude par une simple présentation du Intelligence Artificielle ensuite une briefe présentation de l'apprentissage automatique (Machine Learning), leur notions de base, les différents types et les familles d'algorithmes pour chaque type. Ensuite en entame écosystème Big Data dans le monde des IOT et le besoins du Machine Learning dans ce monde, la notion du cloud computing , leurs caractéristiques, architectures de base, et les modes de déploiements, et à la fin nous immergeons dans les outils d'analyse distribué pour le traitement massif des données.

1.2 Intelligence artificielle

L'intelligence artificielle (IA) est la simulation des processus d'intelligence humaine par des machines, en particulier des systèmes informatiques. Ces processus incluent l'apprentissage (acquisition d'informations et les règles d'utilisation de ces informations), le raisonnement (utiliser des règles pour tirer des conclusions approximatives ou définitives) et l'autocorrection. Les applications particulières de l'IA comprennent les systèmes experts, la reconnaissance vocale et la vision artificielle.

L'intelligence artificielle a fait son chemin dans un certain nombre de domaines comme la santé (Healthcare), Business, éducation, finance et la loi.

L'intelligence artificielle est intégrée à différents types de technologies comme automatisation, la vision par machine et apprentissage automatique. Dans cette partie, nous concentrons nos études sur l'apprentissage automatique, ses types et les différents algorithmes de calcul.

La figure ci-dessous montre quelques applications de l'intelligence artificielle, leurs types d'application.

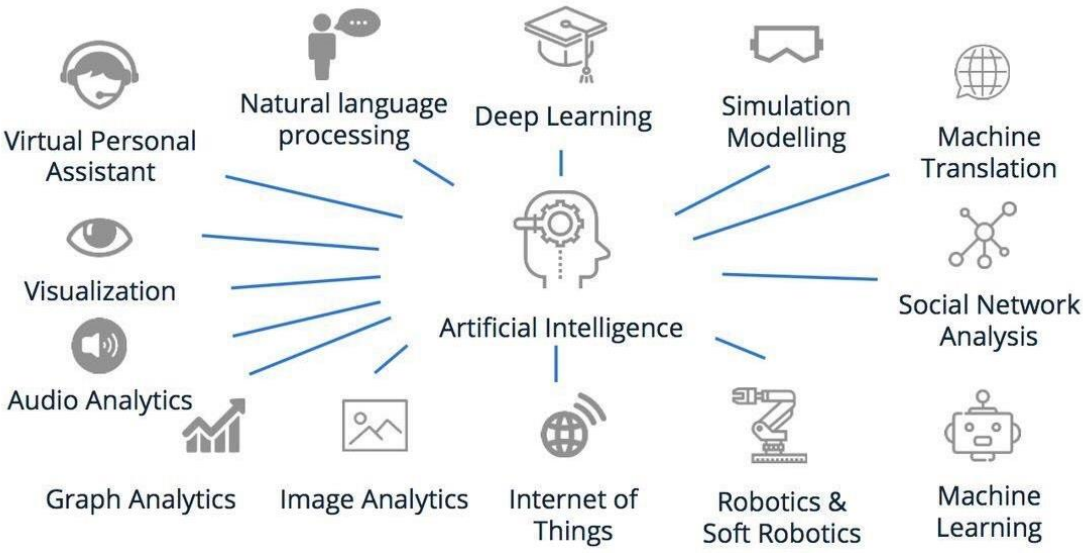


Figure 1.1 : Les différentes applications de l'intelligence artificielle

1.3 Généralités sur l'apprentissage automatique

L'apprentissage automatique est un domaine assez ancien qui a vécu sa croissance d'utilisation avec les progrès technologiques. Toutefois, son utilisation reste conditionnée par une bonne maîtrise de ses fondements théoriques dont cette section est consacrée.

1.3.1 Définition

« Machine Learning » ou Apprentissage automatique est, d'après la définition de (Mitchell 1997), « Un problème d'apprentissage se définit comme l'apprentissage automatique d'un programme informatique mis en œuvre sur une expérience **E**, par rapport à une certaine tâche **T**, mesurée par une mesure de performance **P**. Sa performance à résoudre **T** mesurée par **P**, s'améliore par l'expérience **E.** »[1]

Une solution d'apprentissage automatique repose donc sur deux étapes principales :

- **L'apprentissage** dans un premier temps : le programme apprend à résoudre le problème en question à travers un ensemble de données.
- **La prédiction** dans un deuxième temps : étant donné un nouvel ensemble de données, le programme résout le même problème avec l'expérience (paramétrage et fonctions) acquise lors de l'apprentissage.

Le programme essaye de trouver et de définir une fonction **f,** tel que pour le problème posé, et un ensemble de données d'entrée **X,** les solutions sont $\mathbf{Y = f(X)}$.

Il existe principalement deux types d'apprentissage automatique, **l'apprentissage supervisé** et **l'apprentissage non-supervisé.**

1.4 Types d'apprentissage automatique

Il existe plusieurs manières de faire apprendre un algorithme à une machine. Chaque approche est différente des autres et convient à des problématiques différentes selon les données à disposition et l'objectif fixé de l'apprentissage.[2]

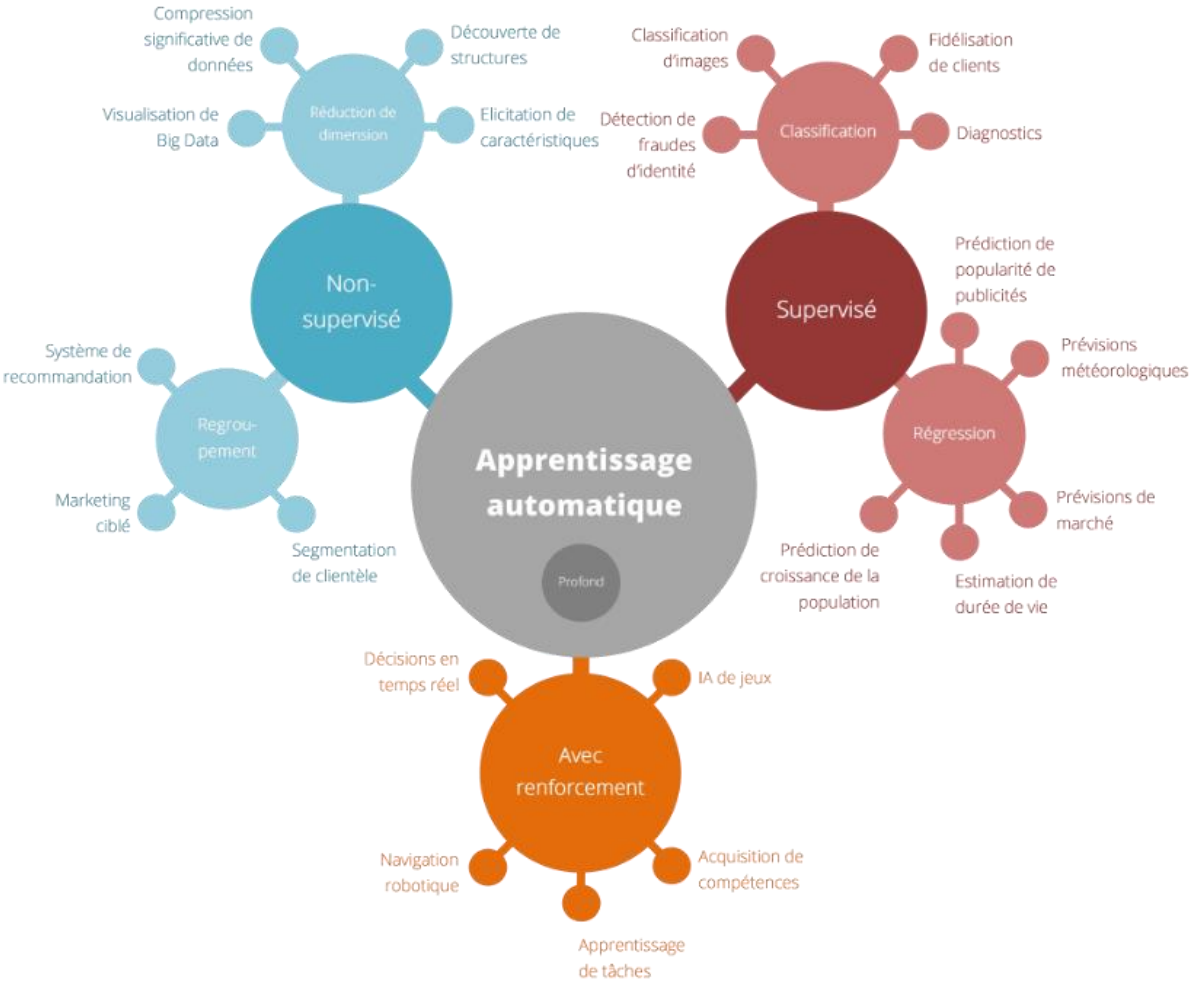


Figure 1.2 : Les différents types de l'apprentissage automatique et leurs applications

1.4.1 Apprentissage supervisé (supervised learning)

Dans ce cas, l'ensemble de données d'apprentissage (observations) **X** utilisé lors de cette phase, est labélisé par les solutions correctes **Y** au problème donné. Pour chaque individu dans cet ensemble, l'algorithme essaye de résoudre le problème posé, puis compare la solution trouvée **Y'** à la solution correcte **Y** (donnée). (Cornuéjols and Miclet 2010)

Il s'agit donc d'optimiser une fonction **f** tel que **Y'** se rapproche le plus possible de **Y**, et donc de minimiser l'erreur de prédiction. Cette fonction **f** est ensuite utilisée pour résoudre le problème posé avec n'importe quel autre ensemble de données d'entrée (observations) **X**.

L'optimisation de cette fonction peut se faire de différentes manières et par divers algorithmes. Le plus souvent, la fonction gradient est utilisée pour optimiser des probabilités. L'algorithme le plus connu est celui du « gradient descent » (Mason et al. 1999) qui utilise les dérivées partielles afin de minimiser l'erreur en fonction des paramètres de la fonction **f**.

Les problèmes d'apprentissage supervisé sont soit des problèmes de régression ou de classification.[3]

1.4.1.1 La régression

Dans la régression linéaire, l'espace des solutions **Y** est continu, c'est à dire que pour chaque individu de l'ensemble de données du problème, on prédit une solution continue (ne peut être dénombrable), par exemple, un prix, ou un coût représenté par un nombre réel (Siskos 1985).

La régression peut être formulée ainsi :

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_N x_N + \varepsilon \tag{1}$$

Ou :

- y *:* est la variable cible ;
- x_i : les variables prédictives ;
- ε **:** l'erreur à minimiser ;
- β_i : les poids à estimer.

Il s'agit donc d'ajuster les β_i de façon à obtenir une erreur ε la plus petite possible.

1.4.1.2 La classification

Dans cette catégorie, on cherche à classer un ensemble d'observations **X** selon des catégories Y discrètes, connues, et prédéfinies. Le programme apprend donc la fonction **f**, telle que :

si $f(x)=y$, (**x** dans X, **y** dans **Y**), alors l'observation x appartient à la catégorie y.

Parmi les algorithmes de classification les plus connus, nous pouvons citer : SVM (Hearst et al. 1998), les arbres de décision (Quinlan 1986) ou encore les forêts aléatoires (Breiman 2001).

1.4.2 Apprentissage non-supervisé (unsupervised learning)

Contrairement à l'apprentissage supervisé, on ne connaît pas la structure des solutions ou leurs formes durant la phase d'apprentissage, le programme essaye donc de découvrir une structure dans les observations **X.** Cette dernière représente est l'espace des solutions **Y**.

Les problèmes d'apprentissage non-supervisé sont regroupés en deux catégories principales : le clustering et l'association.(Cornuéjols and Miclet 2010)

1.4.2.1 Le clustering

Un problème est dit de « clustering » dans le cas où on essaye de découvrir des groupes ou des catégories (non prédéfinies et inconnues à l'avance) dans nos données, celles-ci sont regroupées selon leurs propriétés, de façon à ce que l'ensemble **X** soit partitionné en différents sous-ensembles homogènes (clusters) tel que : les individus d'un même cluster se ressemblent le plus possible et donc soient les plus proches possible et les individus appartenant à des clusters différents soient les plus différents (éloignés) possible.

Parmi les algorithmes de clustering, nous pouvons citer k-means (Hartigan 1975), Self Organizinf Map (SOM) (Kohonen 1990) ou encore density-based spatial clustering of applications with noise (DBSCAN) (Ester et al. 1996).

1.4.2.2 Association

Le but ici est de découvrir des règles qui régissent et qui décrivent l'ensemble des individus X, comme par exemple « l'ensemble des clients ayant acheté le produit a ont aussi acheté les produits b et c ».

Une règle d'association peut être définie formellement selon (Manku 2002) comme ceci :

Soit $I = \{i_1, i_2, \ldots . i_m\}$ un ensemble d'items. Soit $T = \{t_1, t_2, \ldots . t_m\}$ un ensemble de transactions, telles que t_i, soit un sous-ensemble de I. Une règle d'association s'exprime sous la forme :

$$X \rightarrow Y \; ou \; X \in T, Y \in T \; et \; X \cap Y = \emptyset \qquad (2)$$

De plus, X et Y sont choisis selon deux restrictions :

- Le compteur de support : le support de X est le nombre de transactions de T qui contiennent X. On le note : $Supp(X) = \frac{X.Count}{Card(T)}$

- La force d'une règle d'association est mesurée par son indice de support et son indice de confiance.

a. L'indice de support d'une règle $S(X \rightarrow Y)$ est défini par la proportion de transactions de T qui contiennent à la fois X et Y
b. L'indice de confiance d'une règle $(X \rightarrow Y)$, représente le pourcentage de fois où l'on obtient Y lorsqu'on a X, est défini par

$$Conf(X \rightarrow Y) = \frac{Supp(X \cap Y)}{Supp(X)} \quad (3)$$

Parmi les algorithmes d'extraction de règles d'association nous pouvons citer Apriori (R Agrawal et al 2005) ou encore fp-growth (Li and Deng 2007).

1.4.3 Apprentissage semi supervisé

Dans l'apprentissage semi-supervisé, l'idée de base est d'apprendre un modèle de classification à partir des données étiquetées et prédire les classes des données non étiquetées afin d'améliorer les performances de prédiction par rapport à la méthode non-supervisée et de diminuer les coûts d'étiquetage par rapport aux méthodes supervisées. Mais cela ne marche que si les données étiquetées et non-étiquetées ont la même distribution.

S'il y a une interaction explicite entre l'algorithme et l'humain on parle d'apprentissage actif.

L'idée est d'améliorer l'apprentissage semi-supervisé en autorisant des interactions entre l'algorithme d'apprentissage et l'opérateur qui effectue l'étiquetage des données afin d'enrichir la qualité des données utilisées pour l'apprentissage. Il est possible car le choix des instances à étiqueter pour l'apprentissage peut influencer considérablement la qualité du classifieur appris.

Contrairement à un apprentissage passif, l'apprentissage actif consiste ainsi à se demander quelles seraient les instances les plus pertinentes à présenter à l'opérateur, afin de les utiliser comme données étiquetées pour l'apprentissage (Bouguelia 2015).

1.4.4 Apprentissage par renforcement

L'apprentissage par renforcement selon (Sutton and Barto 1998) est une approche permettant de comprendre et d'automatiser l'apprentissage et la prise de décision ciblés. Il se distingue en mettant l'accent sur l'apprentissage par l'individu de l'interaction directe avec son environnement, sans dépendre d'une supervision exemplaire.

Au-delà de l'agent et de l'environnement, on peut identifier quatre sous-éléments principaux du système d'apprentissage par renforcement : une politique, une fonction de récompense, une fonction de valeur, et, optionnellement, un modèle de l'environnement.

1.4.5 Apprentissage profond (Deep Learning)

L'apprentissage profond est une technologie plus récente de l'apprentissage automatique. Elle est particulièrement efficace pour résoudre des problèmes difficilement programmables et qu'un être humain peut résoudre facilement comme reconnaître et comprendre du texte, une image, voitures autonomes, messagerie, classification de documents ou e-mails…etc (Ian Goodfellow, et al 2016)

L'apprentissage profond est une technique d'intelligence artificielle et un nouveau paradigme de programmation, basée sur les réseaux de neurones artificiels.[4]

1.4.5.1 Réseaux de neurones

Les réseaux de neurones sont inspirés du cerveau de l'être humain. Un réseau de neurones est composé de couches de neurones reliées entre elles et où chaque neurone et optimise une partie du problème, en appliquant des calculs sur les données qu'il reçoit.

1.4.5.1.1 Neurone

Un neurone reçoit en entrée un ensemble de valeurs en représentation vectorielle (la codification de nos observations en entrée, ou bien le signal d'autres neurones composant la couche précédente). Il multiplie chaque valeur par un poids (multiplication matricielle), ajoute un biais, puis applique une fonction d'activation sur la valeur obtenue. La fonction d'activation doit être une fonction non-linéaire et différentiable. Ces conditions sont nécessaires car elles permettent « l'apprentissage » ou l'ajustement des paramètres pour arriver à approximer les poids idéaux qui minimisent l'erreur lors de l'apprentissage.

1.4.5.1.2 Couche d'entrée (Input layer)

Cette couche est composée d'un ensemble de neurones d'entrée dont la fonction est de transmettre les données d'observations à la couche suivante. Ces neurones n'appliquent pas de fonction d'activation, les données d'entrées doivent cependant être normalisées avant d'être injectées.

1.4.5.1.3 Couche cachée (Hidden layer)

C'est la couche la plus importante car elle est responsable de la phase d'apprentissage, chaque neurone applique n poids à un résultat de la couche précédente puis une fonction d'activation. Ces poids sont ajustés à chaque itération lors de l'apprentissage, on appelle les itérations d'apprentissage « **epochs** ». La fonction d'activation peut être par exemple la fameuse fonction sigmoïde :

$$\sigma(Z) = \frac{1}{1 + e^{-Z}} \tag{4}$$

1.4.5.1.4 Couche de sortie (output layer)

Cette couche dépend de la structure et de la complexité des résultats que requiert la solution du problème, Elle est repréensté par un vecteur ou un scalaire de probabilités entre 0 et 1, en appliquant une fonction d'activation. Comme pour les neurones de la couche cachée.

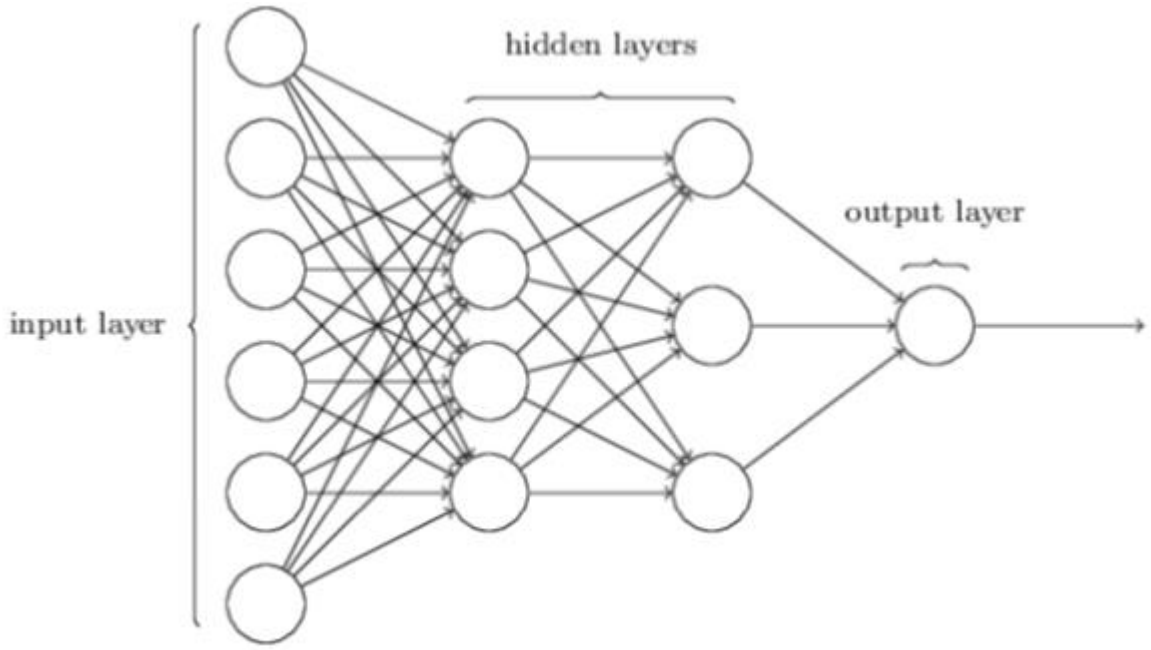


Figure 1.3 : Les différentes couches d'un réseau de neurones

1.4.5.1.5 Apprentissage et algorithme de propagation inverse (Backpropagation)

L'apprentissage consiste à minimiser une fonction d'erreur, ou une probabilité, en fonction des poids des neurones des couches successives. Pour cela, il faut définir le moyen d'agir sur la sortie en fonction de la variation des poids du réseau. C'est possible grâce à ce qu'on appelle l'algorithme de « backpropagation » et de minimisation du gradient.

Il est appelé propagation inverse, car comme la figure 2 tirée de (Nielsen 2015) l'indique, les changements en sortie du réseau se répercutent sur les poids entre les couches

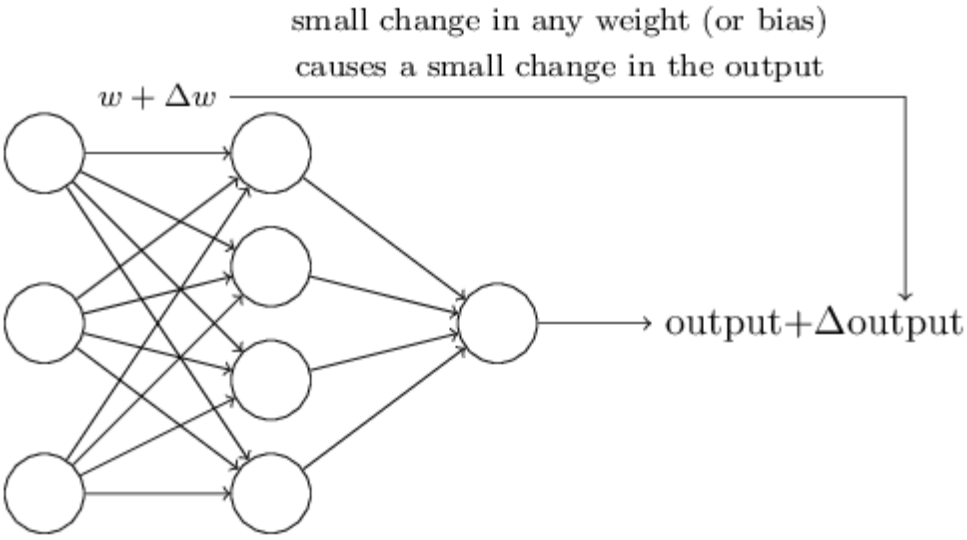


Figure 1.4 : Un réseau de neurones à propagation inverse

Cependant, afin de savoir dans quelle direction appliquer le changement sur chaque poids (diminuer ou augmenter le poids), il faut calculer la dérivée partielle de la fonction par rapport à ce poids, et diminuer le poids de la dérivée partielle multipliée par une constante qu'on appelle le « taux d'apprentissage » ou « Learning Rate », qui aura pour rôle d'ajuster la vitesse à laquelle on essaye d'atteindre le minimum de l'erreur (ou la fonction à minimiser).

1.5 Machine Learning pour un écosystème IOT-BIG Data

Actuellement, le monde des données est envahi par des masses de plus en plus volumineuses de données provenant de multiples sources essentiellement les IOT. L'exploitation

de ces données apporte une valeur ajoutée très important à l'entreprise et les aide à suivre, évaluer et améliorer ses performances.

Les solutions classiques de stockage et d'analyse de données n'arrivent plus à satisfaire les besoins en termes de performances.

Le Big Data, Cloud sont alors apparus pour pallier ces problèmes et remplacer les solutions classiques en permettant d'exploiter des masses importantes de données d'une maniéré plus forte qu'avant.

C'est pour cela que dans ce chapitre, nous présentons la notion de l'IOT, Big Data, Cloud Computing et le besoin du Machine Learning pour ce domaine d'application.[5]

1.5.1 Ecosystème IOT

L'Internet des objets, ou IoT, est un système d'appareils informatiques interdépendants, de machines, d'objets, d'animaux ou de personnes numériques dotés d'identifiants uniques (UID) et permettant de transférer des données sur un réseau sans aucune interaction .De plus, plusieurs entreprises de divers secteurs utilisent l'IOT pour fonctionner plus efficacement, mieux comprendre leurs clients, améliorer leur service client, améliorer leur processus décisionnel et accroître la valeur de leur activité.[6]

Un écosystème IoT se compose d'appareils intelligents activés sur le Web qui utilisent des processeurs, des capteurs et du matériel de communication intégrés pour collecter, envoyer et exploiter les données acquises dans leurs environnements. Les périphériques IoT partagent les données de capteurs qu'ils collectent en se connectant à une passerelle IoT ou à un autre périphérique de périphérie où les données sont envoyées dans un cloud pour être analysées ou analysées localement (edge computing). Parfois, ces appareils communiquent avec d'autres appareils apparentés et agissent en fonction des informations qu'ils obtiennent les uns des autres. Les appareils IOT effectuent la majeure partie du travail sans intervention humaine, bien que les utilisateurs puissent interagir avec eux pour les configurer, leur donner des instructions ou accéder aux données.

Les protocoles de connectivité, de mise en réseau et de communication utilisés avec ces périphériques Web dépendent largement des applications IoT spécifiques déployées.

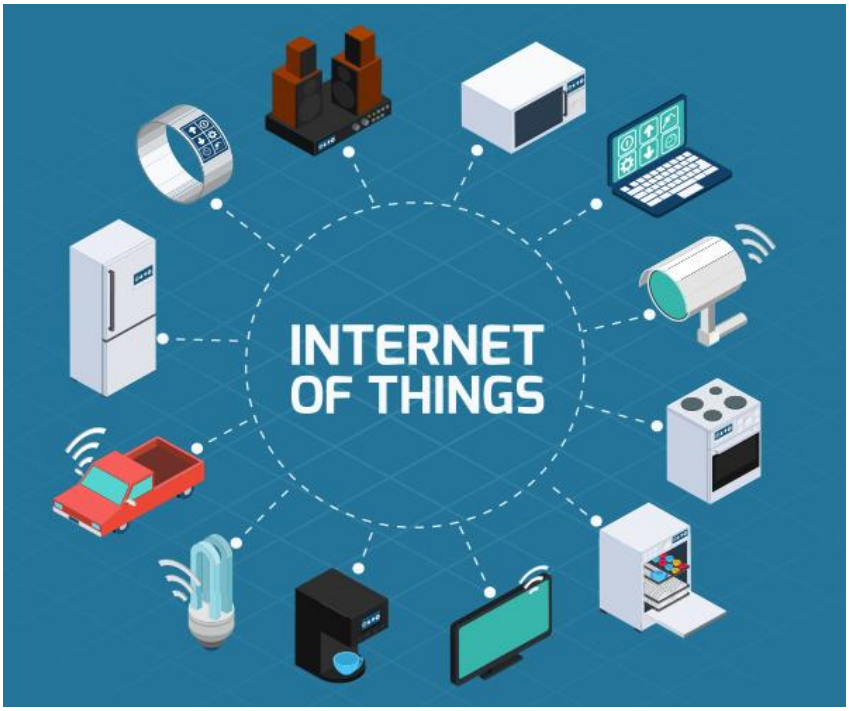


Figure 1.5 : Les différents objets d'un réseau IoT

1.5.2 Ecosystème Big Data

Le Big Data est l'ensemble des ressources d'information volumineuses, à haute vélocité et très variées qui exigent des formes innovantes et rentables de traitement de l'information pour améliorer la compréhension et la prise de décision.[7]

Les données dans les Big Data sont d'un volume important, aussi les données à analyser sont complexes et proviennent de plusieurs sources hétérogènes.

Le Big Data désigne des données souvent à caractéristiques diverses, cependant dans la littérature, il existe plusieurs approches pour synthétiser les particularités des Big Data. La manière la plus répandue, est celle des "3V".[8]

1- Volume : elle décrit la quantité de données grandissantes des Big Data causée notamment par l'avancée technologique, ce qui a fait que les traitements relationnels classiques ne sont plus suffisants, voire obsolètes.

2- Vélocité : Concerne la vitesse à laquelle les données sont générées, et le besoin de les traiter en temps-réel ou en batch.

3- Variété : Désigne la complexité des données composant un Big Data, elles peuvent être structurées, non structurées et semi-structurées.

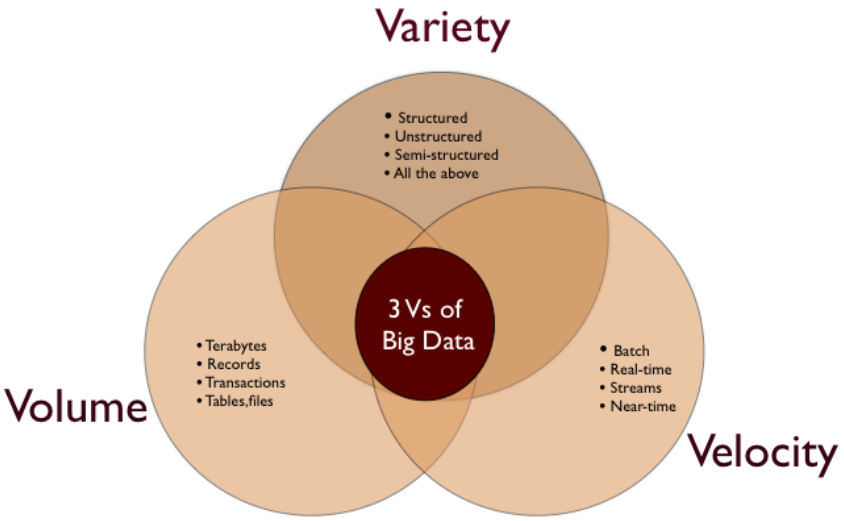


Figure 1.6 : Modèle des 3V d'une solution Big Data

L'analyse de données volumineuses est un travail extrêmement ardu sur l'infrastructure, car les données sont disponibles en grandes quantités, à des vitesses variables et avec des types d'infrastructures que les infrastructures traditionnelles ne peuvent généralement pas maintenir à jour. Comme le Cloud Computing fournit une infrastructure flexible, que nous pouvons adapter en fonction des besoins du moment, il est facile de gérer les charges de travail.[9]

1.5.3 Ecosystème Cloud Computing

Depuis quelques années, avec le développement rapide des technologies du traitement, du stockage et le succès de l'internet, le Cloud Computing est un terme de plus en plus présent. Le fait que les ressources informatiques soient moins chères et plus puissantes que jamais auparavant permet au Cloud Computing d'offrir et d'utiliser des infrastructures informatiques, plates-formes et applications de toutes natures sous la forme de services qui sont disponibles à la demande sur le Web.[10]

1.5.3.1 Définition

Le Cloud computing est un modèle qui permet un accès ubiquitaire au réseau, à la demande à un bassin partagé de ressources informatiques configurables (réseaux, serveurs, stockage, applications et services) qui peuvent être rapidement approvisionnée et libéré avec un effort minimal de gestion ou d'interaction du fournisseur de services. Ce modèle Cloud favorise la disponibilité et est composé de cinq caractéristiques essentielles, trois modèles de services et quatre modèles de déploiement.

1.5.3.2 Les caractéristiques de Cloud Computing

Selon la définition du Cloud, le terme «Cloud Computing » doit être utilisé uniquement dans les cas où une offre de Cloud Computing possède toutes les caractéristiques suivantes

- Des services à la demande autonomes « On-demand self-service »
- L'accès au réseau étendu « Broad network access »
- La mise en commun des ressources « Resource pooling »
- Elasticité rapide « Rapid elasticity »
- Service mesuré « Measured service »

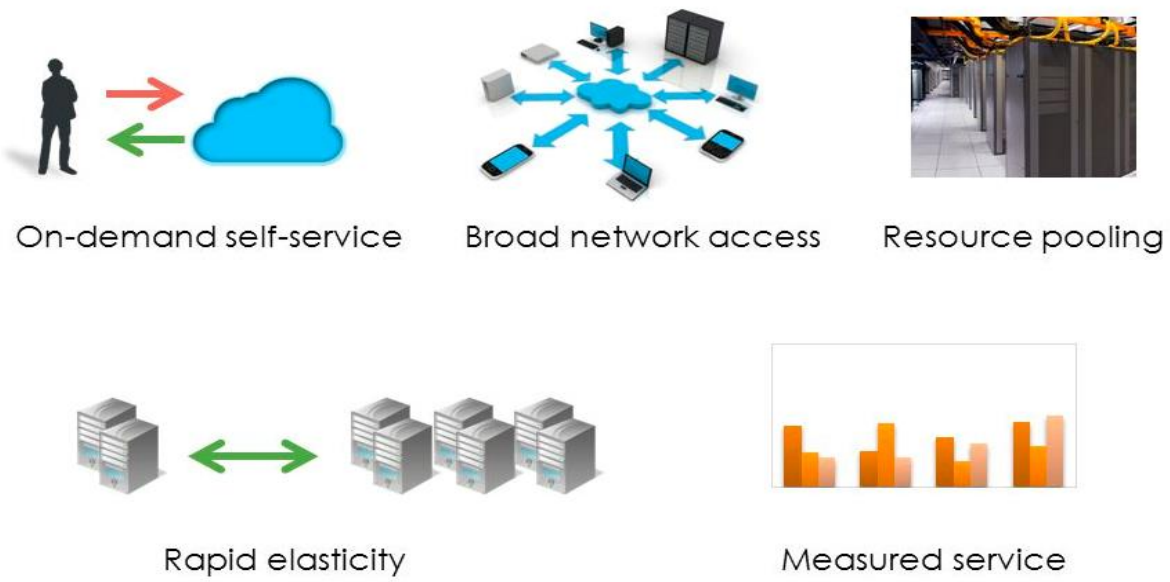


Figure 1.7 : Les avantages du Cloud Computing

1.5.3.3 Architecture de Could Computing

L'architecture décrit quel genre de service peut être obtenu à partir du Cloud (voir Figure :1.8). Trois grandes catégories de services sont distinguées : IaaS (Infrastructure as a Service), PaaS (plateforme as a service), et SaaS (Software as a Service). Selon le modèle choisi, le fournisseur offre des services différents. Ces services sont généralement classés selon le niveau de l'architecture IT sur laquelle ils résident.[11]

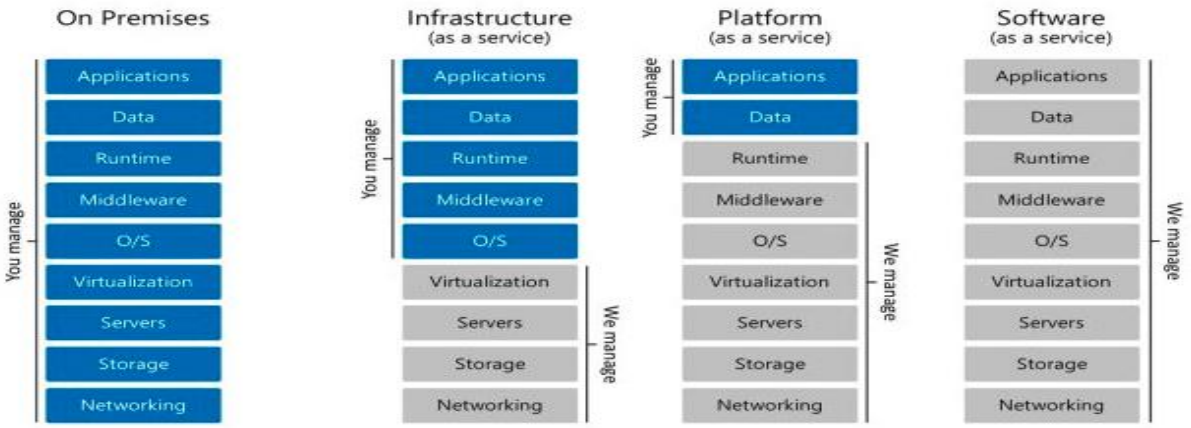


Figure 1.8 : Les types principaux des services sur Cloud

1.5.3.3.1Software as a service (SAAS)

SaaS fournit des solutions logicielles via Internet éliminant ainsi le besoin d'installer ou d'exécuter une application sur un ordinateur local. Lors de l'utilisation en mode SaaS, les données sont stockées en permanence sur les systèmes informatiques à distance, les applications sont livrées par exemple par le biais d'un navigateur et les données sont mises en cache temporairement sur le côté client.

1.5.3.3.2Platform as a service (PAAS)

Platform as a Service (PaaS) est une façon de louer du matériel, des systèmes d'exploitation, du stockage et la capacité du réseau sur Internet. Le modèle de prestation de service permet au client de louer des serveurs virtualisés et des services associés pour exécuter des applications existantes ou développer et en tester de nouvelles.

1.5.3.3.3Infrastructure as a service (IAAS)

La couche IaaS donne aux utilisateurs une vue abstraite du matériel (ordinateurs, les systèmes de stockage de masse, les réseaux,...). Ceci est réalisé en fournissant une interface utilisateur pour la gestion d'un certain nombre de ressources dans la sous-couche « Resource Set ». Les fonctions typiques disponibles à partir de l'interface utilisateur incluent la création ou la suppression des images du système d'exploitation, l'extension des capacités requises, ou la définition des topologies de réseau.

1.5.3.4 Les modèles de déploiement de Cloud Computing

Il est très important pour les entreprises de comprendre leurs besoins avant d'opter pour les modèles de déploiement du Cloud disponible. Il y a principalement quatre modèles de déploiement, Nous allons décrire dans ce qui suit ces modèles.[12]

1.5.3.4.1 Cloud publique

Un Cloud public (également appelé «Cloud externe») comprend tous les offres du Cloud où les fournisseurs et les utilisateurs potentiels n'appartiennent pas à la même unité organisationnelle. Les fournisseurs rendent leur Cloud accessible au public via un portail Web où les utilisateurs peuvent spécifier la portée des services souhaités. Les services sont facturés sur la base des ressources effectivement utilisées pour la période correspondante « pay-per-use ».Cela peut être comparé au système d'électricité, nous payons seulement pour le montant de ce que nous utilisons

Ce modèle est le mieux adapté pour les besoins d'une entreprise dans laquelle il est nécessaire de gérer des pics de charge, héberger des applications SaaS, utiliser des infrastructures provisoires pour développer.

1.5.3.4.2 Cloud privé

Contrairement au modèle précèdent, les fournisseurs et les utilisateurs d'un Cloud privé (également dénommé «Cloud interne» ou «IntraCloud ») appartiennent à la même unité organisationnelle. La raison principale pour laquelle un Cloud privé serait préférable à un Cloud public est habituellement la sécurité : Dans un Cloud privé, les données restent sous le contrôle des utilisateurs ou de leur organisation.

1.5.3.4.3 Cloud hybride

Ce modèle de déploiement aide les entreprises à tirer profit des applications sécurisées et de l'hébergement de données sensibles sur un Cloud privé, tout en profitant des avantages de coûts en conservant les données partagées et certaines applications sur le Cloud public.

Ce modèle est également utilisé pour le scénario où l'infrastructure de Cloud privé existant n'est pas en mesure de gérer les pics de charge et nécessite une solution de repli pour supporter la charge. Dans ce cas-là, la charge de travail se déplace entre un Cloud public et un privé, sans aucune gêne pour les utilisateurs.

1.5.3.4.4 Cloud communautaire

L'infrastructure du Cloud communautaire est partagée par plusieurs organisations et supporte une communauté spécifique qui partage les mêmes préoccupations. Cela contribue à réduire l'inconvénient du coût par rapport à un Cloud privé, car il est partagé par grand groupe.

L'infrastructure de Cloud pourrait être hébergée par un fournisseur tiers(third party) partie ou dans l'un des organismes de la communauté.

Par exemple, divers ministères au niveau des états doivent avoir accès aux mêmes données relatives à la population locale ou à des informations relatives à l'infrastructure, comme les hôpitaux, les routes, les stations électriques,...etc. Ils peuvent utiliser un Cloud communautaire afin de gérer les applications et les données

1.5.4 L'intérêt de l'usage de Machine Learning pour un écosystème Big Data-IOT-Cloud

De nos jours, la quantité de données explose à une vitesse très grandes et sans précédent en raison de l'évolution des technologies Web, des médias sociaux, la télécommunication, et des appareils mobiles et des IOT. Par exemple ABI Research estime qu'en 2020, il y aura plus de 30 milliards d'appareils connectés (IOT). Ces Big Data possèdent un potentiel considérable en termes de valeur commerciale dans divers domaines tels que les soins de santé, la biologie, les transports, la publicité en ligne, la gestion de l'énergie et les services financiers. [13]

L'analyse des données implique diverses approches, technologies et outils, Cependant, les approches traditionnelles ont du mal à faire de l'analyse et traitement complexe. Du fait, la concentration sur l'apprentissage automatique (ML) en tant que composante fondamentale de l'analyse de données est obligatoire. Le McKinsey Global Institute a déclaré que ML serait l'un des principaux moteurs de la révolution du Big Data. Aussi, les deux technologies sont fortement liées. Le Big Data est vu comme l'essence du Machine Learning. Le Machine Learning perdrait sa vraie puissance en l'absence des traitements des grands volumes de données. La relation entre ces deux technologies réside dans les points suivants (L. Bastien, 2018) :

- Le Machine Learning est considéré comme la technologie qui permet d'exploiter pleinement le potentiel des Big Data et d'extraire de la valeur à partir des données massives et variées sans avoir besoin de compter sur l'intelligence humaine ;
- Le Machine Learning est plus efficace que les méthodes classiques en termes de précision et de vitesse dans la découverte des patterns à partir des grands volumes de données.
- Le Machine Learning est dirigé par les données, et convient à la complexité d'immenses sources de données : plus le volume des données injectées à un système de Machine Learning est grand, plus le système peut apprendre mieux, et plus la qualité des résultats est améliorée.
- Le Big Data permet d'accélérer la courbe d'apprentissage des algorithmes de Machine Learning en fournissant des données massives d'une manière continue et avec une fréquence considérable.

Néanmoins l'utilisation des outils de traitement distribuer (Clustring Methods) est requis dans le cas où les data deviens de plus en plus massive et que l'évolution des ressources sont devient de plus en plus lente et chère, aussi avec l'avancement des technique de stockage et de calcul, le cloud devient l'un des outils nécessaire pour effectuer les calculs d'une maniére plus rapide et plus optimisé en terme de couts et vitesse. [14]

1.6 Le calcul distribué : les outils de calculs distribués dédié au Big Data

Afin de pouvoir assurer de hautes performances en termes de disponibilité et de temps de traitement, les solutions Big Data se basent sur la distribution des données et de leurs traitements. Il existe plusieurs outils de distribution qui sont détaillés dans ce qui suit.

1.6.1 La distribution et leur nécessité pour un écosystème Big Data

La distribution consiste à avoir une architecture composée de plusieurs clusters de machines, configurés selon la manière dont on veut distribuer les données et les traitements. En termes de Big Data, l'architecture adoptée est souvent distribuée et ce pour différentes causes :

- La distribution offre un meilleur degré de fiabilité
- Peut-être plus profitable en termes de coût.
- Par rapport à un système monolithique, un système distribué est plus flexible, son extensibilité est plus facile à faire en cas de besoin
- Minimiser le risque de perte des données
- Augmenter les performances en termes de traitement des données.

Pour cette raison, nous faisons dans la prochaine partie une comparaison entre les outils de calcul distribué les plus utilisé.

1.6.2 Les architectures du calcul distribué

Principalement, il y'a deux grande architectures, avec et sans maitre

1.6.2.1 La distribution avec maitre

La distribution avec maître s'appuie sur la présence d'une machine maîtresse qui conserve la configuration du système, reçoit les requêtes des clients et les redirige vers les machines qui contiennent les données désirées.

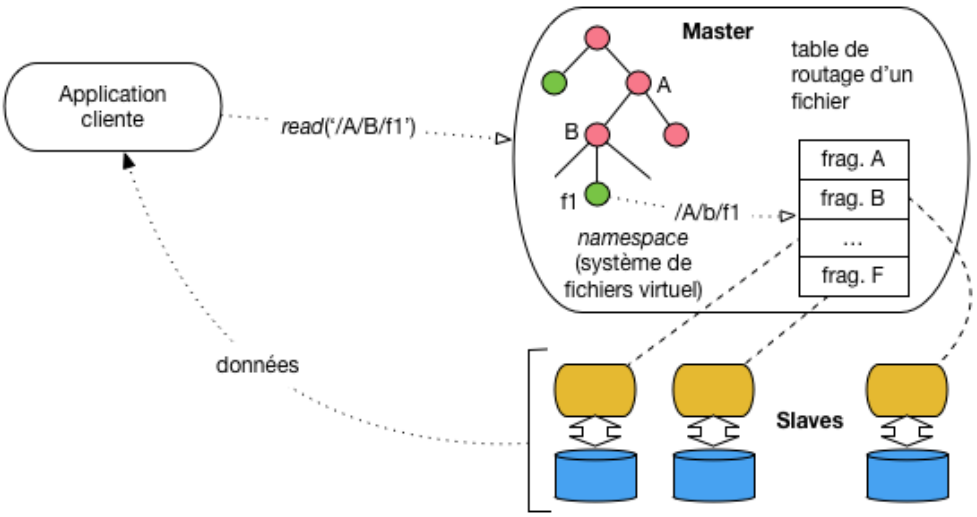


Figure 1.9 : Architecture du calcul distribué avec maitre

1.6.2.2 La distribution sans maitre

Dans un système distribué sans maître, toutes les machines ont la même importance et les mêmes capacités dans le cluster. L'objectif de ce type de distribution est de pouvoir assurer la disponibilité des services en se connectant à n'importe quelle machine, même si plusieurs membres du cluster tombent en panne.

1.6.3 Les outils de calculs distribués du Big Data

Pour faire du calcul distribué dédié au Big Data, il faut mettre en œuvre certaines solutions qui décrivent la topologie de la distribution, un système de communication spécialisé, une interface client pour commander le calcul, un système de gestion de calcul et d'optimisation, des Frameworks pour enrichir les fonctionnalités du système et soutenir le développement,... Dans ce qui suit, nous allons présenter des outils de calculs distribués, dédiés au Big Data, les plus utilisés et réputé par leurs performances et puissance de développement.

1.6.3.1 Paradigme MapReduce

MapReduce est un paradigme de programmation qui permet de faire du traitement distribué à grande échelle sur un ensemble de machines. MapReduce effectue les calculs à travers deux fonctions : Map et Reduce. Dans l'étape Map le nœud analyse le calcul à effectuer, le découpe en sous-tâches et délègue ces sous-tâches à d'autres nœuds. Ces nœuds-là peuvent à leur tour faire la même chose récursivement.

Ensuite pour l'étape Reduce, les nœuds les plus bas remontent leurs résultats au nœud parent qui leur avait affecté la tâche. Le nœud parent calcule le résultat intermédiaire et le fait remonter à son tour, jusqu'à arriver au nœud d'origine.

A la fin du processus le nœud d'origine compose le résultat final en combinant tous les résultats intermédiaires qui lui ont été communiqués. Les phases décrites sont illustrées dans le schéma suivant :

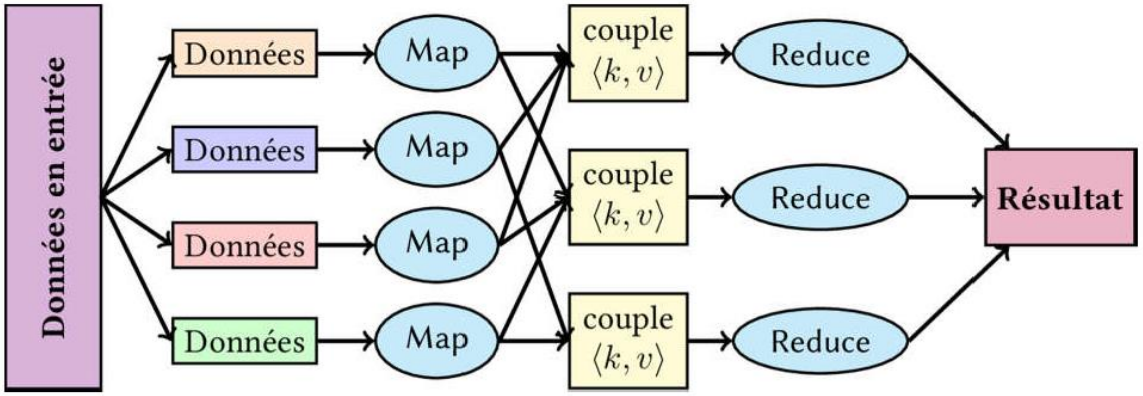


Figure 1.10 : Circulation des données dans une approche fonctionnelle Map-Reduce

Il existe plusieurs implémentations de MapReduce comme Dryad de Miscrosoft, Disco de Nokia, etc. L'implémentation la plus populaire reste Apache Hadoop et Apache Spark qui sont open source. Dans ce suit, on présente ces deux éléments à savoir les caractéristique de chacun.

1.6.3.2 Apach Hadoop

Apache Hadoop Software Library est un Framework open source qui permet le traitement distribué de grands ensembles de données à travers des clusters en utilisant des modèles de programmation Simples.[15]

Le projet Apache Hadoop inclut plusieurs modules, on cite les principaux :

- Hadoop Common : Utilitaires communs prenant en charge les autres modules Hadoop.
- Système de fichiers distribués Hadoop (HDFS) : système de fichiers distribué offrant un accès à haut débit aux données d'application.
- Hadoop YARN : un Framework pour la planification des tâches et la gestion des ressources de cluster.
- Hadoop MapReduce : Un système basé sur YARN pour le traitement parallèle de grands ensembles de données.

1.6.3.3 Apach Spark

Spark Software Library est un projet open source et gratuit à la fois comme le Hadoop et qui permet de réalise un traitement distribué a grand échelle .Il peut fonctionner en deux modes : Avec Hadoop en remplaçant la section Hadoop MapReduce, ou bien comme

un moteur de traitement autonome. En tant que processeur autonome, Spark ne dispose pas de sa propre couche de stockage distribué, mais peut utiliser le système de fichiers distribué (HDFS) de Hadoop ou une base de données (SQL ou No-SQL) pour le stockage.[16]

Il est constitué de plusieurs couches intégrées qui offrent un environnement de Big data complet. Son architecture est constituée des modules suivants :

- Spark Core
- Spark SQL
- Spark Streaming
- MLlib
- GraphX

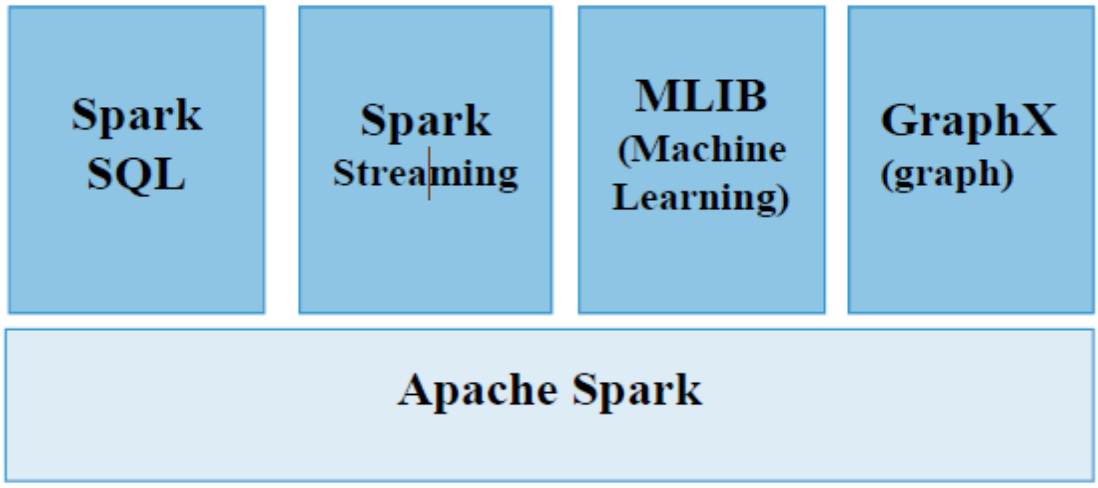


Figure 1.11 : Pile applicative du framework Apache Spark

Le tableau ci-dessous montre une brève comparaison entre le Hadoop et Spark :

Type de comparaison	Hadoop	Spark
Catégorie	moteur de traitement de données de base	moteur d'analyse de données
Usage	Traitement par lots avec un énorme volume de données	Traitement par lot et des données en temps réel, à partir d'événements tels que Twitter et Facebook
Latence	calcul à haute latence	calcul à faible latence
Donnée	Traite les données en mode batch	Peut traiter d'une manière interactive

Facilité d'usage	Le modèle MapReduce de Hadoop est complexe, il faut gérer les APIS de bas niveau	Spark est plus facile à utiliser, l'abstraction permet à un utilisateur de traiter des données à l'aide d'opérateurs de haut niveau
planificateur	Un planificateur de travaux externe est requis.	Calcul en mémoire, aucun ordonnanceur externe requis
Sécurité	Hautement sécurisé.	Moins sécurisé comparé à Hadoop
Coût	Moins coûteux puisque le modèle MApReduce fournit une stratégie moins coûteuse.	Plus coûteux que Hadoop car il a une solution de calcule en mémoire

Tableau 1 : Comparaison entre Apache Spark et Apache Hadoop

De cette comparaison, nous trouvons que le Spark a des points fort comme la simplicité, le traitement en temps réel et la bibliothèque du machine Learning, ce qui nous aide à adopter le Spark comme une solution de calcule pour notre problème.

Hadoop MapReduce et Apache Spark sont les deux outils les plus importants pour le traitement des données massives. L'avantage majeur de Hadoop MapReduce est qu'il est facile de mettre à l'échelle le traitement des données sur plusieurs nœuds de calcul tandis qu'Apache Spark offre une grande vitesse de calcul, une agilité et une facilité d'utilisation relative qui complètent parfaitement le Hadoop MapReduce. Hadoop MapReduce et Apache Spark ont une relation symbiotique. Hadoop offre des fonctionnalités que Spark ne possède pas, comme un système de fichiers distribué, et Spark fournit un traitement en temps réel et en mémoire pour les ensembles de données qui le nécessitent. Hadoop MapReduce est un Disk-Based Computing tandis qu'Apache Spark est un RAM-Based Computing. L'ensemble de ces résultats favorise Spark dans le choix de l'outil de calcul exploité le long de ce projet, principalement pour la rapidité du calcul et la solution machine learning intégrée.

1.7 Conclusion

Les conclusions tirées de ce chapitre peuvent être résumé dans les points suivants :

- Le Big Data est un domaine qui évolue à grande vitesse.

- Les systèmes IOT sont liés au monde du Big Data, du fait qu'ils soient une source génératrice de données massives.
- Les méthodes de Machine Learning permettent d'analyser n'importe quel domaine métier, de manière précise grâce aux différentes techniques qu'elles regroupent.
- L'étude bibliographique sur ces domaines, nous a permis d'avoir une vision globale sur ce qui est déjà fait afin de nous guider dans les étapes suivantes du projet, notamment dans le choix d'une solution.

Dans les prochains chapitres, on va aborder l'Openstack comme un service d'infrastructure monté sur le Cloud, et le Spark comme un service de calcul distribué utilisé principalement pour ses capacités dans l'apprentissage automatique.

Chapitre 2 :

Solution open source dédié au traitement intelligent du Big DATA: APACHE SPARK

2 Solution open source dédié au traitement intelligent du Big DATA: APACHE SPARK

2.1 Introduction

Les analystes de données, les spécialistes des sciences cognitives et les informaticiens doivent travailler ensemble pour résoudre des problèmes pratiques. Cet apprentissage collaboratif doit faire appel aux Cloud, aux appareils mobiles, les centres de données et les ressources IoT. Le but ultime est de découvrir de nouvelles connaissances, ou prendre des décisions importantes, intelligemment. Aujourd'hui, Google, IBM, Microsoft, l'Académie chinoise de Science, et Facebook explorent les technique d'IA dans les applications en Cloud et IoT, à travers des technologies d'analyse de donnée et des architectures de plus en plus optimale, pour un traitement puis une analyse intelligente des données massives qu'on peut récolter ces dernières années, à travers l'IoT et le trafic internet, ce qui nourrit nos plateforme AI des donnée et les rends encore plus performantes et efficaces.

2.2 Présentation du Apache Spark

Selon Apache, Spark est un moteur rapide et général pour le traitement de données massives (BigDATA), on peut le qualifier aussi comme framework open source de calcul pour le traitement rapide à travers une architecture distribué. C'est un moteur de traitement de données distribué conçu pour garantir la vitesse, la facilité d'utilisation et l'efficacité. La combinaison de ces trois propriétés rend Spark si populaire et largement adopté dans l'industrie. Il supporte le calcul en mémoire (In-memory processing), la tolérance aux pannes (fault-tolérant) à travers la notion des RDDs et le calcul en map reduce. Les contributeurs qui participent à son développement sont nombreux et issus d'environ 200 sociétés différentes, comme Intel, Facebook, IBM et Netflix). Ainsi depuis 2015 on recense plus de 1000 contributeurs et plus de 200 000 conférences sur Apache Spark.

La soumission de Databricks prétend que Spark a été en mesure de trier 100 To de données trois fois plus vite et en utilisant dix fois moins de ressources que le précédent record du monde établi par le Hadoop MapReduce. Le site Web d'Apache Spark prétend qu'il peut exécuter un certain travail de traitement de données jusqu'à 100 fois plus rapide que Hadoop MapReduce. Depuis le lancement du projet Spark, la facilité d'utilisation a été l'une des principales raisons pour lesquelles il a été lancé. Il offre plus de 80 services de haut niveau, les opérateurs de traitement de données les plus utilisés pour faciliter la tâche des développeurs, des spécialistes de la science des données et analyseurs de données,

pour créer toutes sortes d'applications de traitement de données intéressantes. En outre, ces opérateurs sont disponibles en plusieurs langues, notamment Scala, Java, Python et R. Les ingénieurs, les scientifiques et les analystes de données peuvent donc choisir leurs model de programmation en fonction de langage de programmation souhaité pour le développement, dans le but de résoudre des problèmes de traitement des données à grande échelle avec Spark.

Spark réalise une lecture des données au niveau du cluster (grappe de serveurs sur un réseau), effectue toutes les opérations d'analyse nécessaires, puis écrit les résultats à ce même niveau. Malgré le fait que les tâches s'écrivent avec les langages Scala, Java et Python, il utilise au mieux ses capacités avec son langage natif, Scala

En termes de flexibilité, Spark offre une pile de traitement de données unifiée qui peut être utilisé pour résoudre de multiples types de charges de travail liées au traitement des données, y compris le traitement par lots, les requêtes interactives, les traitements itératifs nécessaires aux algorithmes d'apprentissage automatique, et traitement en continu en temps réel pour extraire des informations exploitables en temps quasi réel. Avant l'existence de Spark, chacun de ces types de charge de travail exigeait une solution venant d'une technologie différente. Désormais, les entreprises peuvent simplement utiliser Spark pour la plupart de leurs besoins en traitements de données.[17]

L'utilisation d'une seule pile technologique aidera à réduire considérablement les émissions de coûts opérationnels et les ressources.

Cette pile est un ensemble de librairies qui forment, avec le moteur de calcul de Spark, l'architecture en Framework globale du Spark. Elle se forme par le moteur de calcul et ces cinq librairies principales :

- Noyau de Spark (Spark core) : Comme son nom l'indique, il s'agit de l'unité de commande centrale du framework Spark. Il s'occupe principalement de la planification et de la gestion des tâches. Celles-ci utilisent une abstraction Spark appelée ensemble de données réparties résilientes (RDDs). RDD est l'unité d'abstraction de données de base de Spark et sera examinée en détail dans les sections suivantes.
- MLlib : Ce module pour Spark est principalement une implémentation des algorithmes statistiques et des algorithmes d'apprentissage automatique les plus utilisés en science des données. C'est un module hautement évolutif, distribué et tolérant aux pannes, qui fournit des fonctions implicites des implémentations sur la classification, l'analyse des composants, la classification non supervisé (clustering), régression, et ainsi de suite.

- SparkSQL : Ce module de Spark est typiquement conçu pour les ingénieurs de données qui sont familiers avec l'utilisation de SQL sur des ensembles de données structurés d'Oracle, SQL Server, MySQL, etc. Il supporte le répertoire de stockage et on peut facilement interroger les données à l'aide de HiveQL - Langage de Requête HiveQL - qui est très similaire à SQL. Spark SQL s'interface également avec des connecteurs populaires comme JDBC et ODBC qui permettent où développeurs d'intègrent à des bases de données populaires et à d'autres datamarts, ainsi qu'à la BI et à l'informatique décisionnelle et les outils de visualisation.
- GraphX : GraphX était un projet indépendant au centre de recherche de Berkley, mais qui a été plus tard donné à Spark et est donc devenu partie intégré dans le Framework Apache Spark. Il s'agit essentiellement du module qui prend en charge le calcul et l'analyse graphique d'un grand volume de données. Il supporte Pregel API et une grande variété d'algorithmes graphiques.
- Spark Streaming : Il s'agit du module Spark qui prend en charge le traitement des données en temps réel/en temps quasi réel. Il s'intègre parfaitement aux pipelines d'ingestion tels que Kafka, RabbitMQ, Flume, etc. Ce module est conçu pour un traitement évolutif et tolérant aux pannes, à grande vitesse.
- Spark R : C'est l'un des modules d'étincelles ajoutés tardivement et c'était principalement conçu comme un outil pour les scientifiques des données. Les analystes de données et les Les scientifiques ont largement utilisé R Studio comme outil de conception de modèles de l'échantillon de données. Ces modèles nécessiteront plus tard beaucoup de retouches et en termes de logique et d'optimisation pour exécuter sur un ensemble plus large de données. Spark R est une tentative pour combler cet écart ; c'est un cadre léger qui exploite les capacités de Spark et permet aux spécialistes des données d'exécuter le modèle R dans le format mode distribué sur un ensemble plus large de données sur le noyau Spark.

Un écosystème de BigDATA est constitué de plusieurs éléments technologiques, dont un moteur de calcul (généralement distribué), un système de gestion de données distribuées de stockage appelé HDFS, un système de gestion de cluster pour gérer efficacement un cluster de machines, et différents formats de fichiers pour stocker efficacement une grande quantité de données en binaire et en colonnes. Une autre raison pour laquelle l'adoption de Spark s'est développée à un rythme très rapide est le faite qu'il s'intègre très bien dans le grand écosystème d'analyse de données, et aussi, parce qu'il appartient à la

catégorie des open source. Dans certains cas, ceci peut considérablement aider à la réduction du temps de débogage des problèmes.[18]

2.3 L'origine de Spark

Spark a débuté comme projet de recherche chez Berkeley AMPLab en 2009 lors d'un doctorat de son premier co-fondateur Matei Zaharia. A cette époque, les chercheurs de ce projet ont observé les inefficacités du framework Hadoop MapReduce dans le traitement des cas d'utilisation interactifs et itératifs du traitement des données, ils ont donc trouvé des moyens de surmonter ces inefficacités en introduisant des idées comme le calcul en mémoire et un moyen efficace de traiter la reprise après défaillance. Une fois que ce projet de recherche s'est avéré être une solution viable qui a surpassé MapReduce-Hadoop, il a été mis comme open source en 2010 sous la license BSD et est devenu le projet de premier niveau Apache en 2013. En 2014, Spark a gagné le « Daytona GraySort Contest » dont l'objectif est de trier 100 To de données le plus rapidement possible. Ce record était préalablement détenu par Hadoop. Pour ce faire, Spark a utilisé 206 machines obtenant un temps d'exécution final de 23 minutes alors que Hadoop avait lui utilisé 2100 machines pour un temps d'exécution final de 72 minutes. La puissance de Spark fut démontrée en étant 3 fois plus rapide et en utilisant approximativement 10 fois moins de machines. Un groupe de chercheurs travaillant sur ce projet de recherche s'est réuni et a fondé une entreprise appelée Databricks ; ils ont recueilli plus de 43 millions de dollars en 2013. Databricks est le principal responsable commercial derrière Spark. En 2015, IBM a annoncé un investissement majeur dans la construction d'un centre de technologie Spark pour faire progresser Apache Spark en travaillant en étroite collaboration avec la communauté open source et en intégrant Spark au cœur des plateformes analytiques et commerciales de son entreprise.

2.4 Applications de Spark

Spark est un moteur de traitement de données polyvalent, rapide et évolutif. Il a été conçu pour être un moteur général depuis le début et a prouvé qu'il peut être utilisé pour résoudre divers cas d'utilisation. De nombreuses entreprises de divers secteurs d'activité utilisent Spark pour résoudre des cas d'utilisation réels. Dans ce qui suit, une petite liste d'applications qui ont été développées avec Spark :

- Applications d'intelligence client.
- Solutions d'entrepôt de données.
- Solutions de streaming en temps réel.
- Recommandation moteurs.

- Traitement des billes.
- Services en contact direct avec les utilisateurs.
- Détection de fraude.

2.5 Concepts principaux et architectures de Spark

2.5.1 Architectures

2.5.1.1 Grappe de Spark et management des ressources de grappe

Spark est essentiellement un système distribué qui a été conçu pour traiter efficacement et rapidement un grand volume de données. Ce système distribué est généralement déployé sur un ensemble de machines, connu sous le nom de cluster Spark. La taille d'un cluster peut être aussi petite que quelques machines ou aussi grande que des milliers de machines. Le plus grand cluster Spark annoncé publiquement dans le monde compte plus de 8 000 machines ! Pour gérer efficacement et intelligemment une collection de machines, les entreprises s'appuient sur un système de gestion des ressources tel qu'Apache YARN ou Apache Mesos, ou le gestionnaire de cluster native de Spark Standlone. Les deux principales composantes d'un système typique de gestion des ressources sont le gestionnaire du cluster (cluster manager) et le travailleur (worker). Le gestionnaire de cluster sait où se trouvent les travailleurs, combien de mémoire ils ont, et le nombre de cœurs de CPU dont chacun dispose (ceci pourra aussi être configuré selon le cahier de charge et les besoins). L'une des principales taches du gestionnaire du cluster est d'orchestrer le travail (job) en l'assignant à chaque travailleur. Chaque travailleur offre des ressources (mémoire, CPU, etc.) au gestionnaire du cluster et effectue le travail qui lui est assigné. Un exemple de ce type de travail est le lancement d'un processus particulier et le suivi de son état (en vie, accompli ou fini avec un signal). Spark est conçu pour s'inter-opèrent facilement avec ces systèmes. La plupart des entreprises qui ont adopté de gros volumes de données ces dernières années, les technologies de l'information et de la communication disposent déjà d'un gestionnaire de cluster YARN pour exécuter des tâches MapReduce, ou d'autres framework de traitement de données telles que Apache Pig ou Apache Hive.

Les startups qui adoptent pleinement Spark peuvent simplement utiliser le gestionnaire de cluste Spark prêt à l'emploi pour gérer un ensemble de machines dédiées à l'exécution du traitement des données en utilisant Spark. La figure 2-1 illustre ce type de configuration :

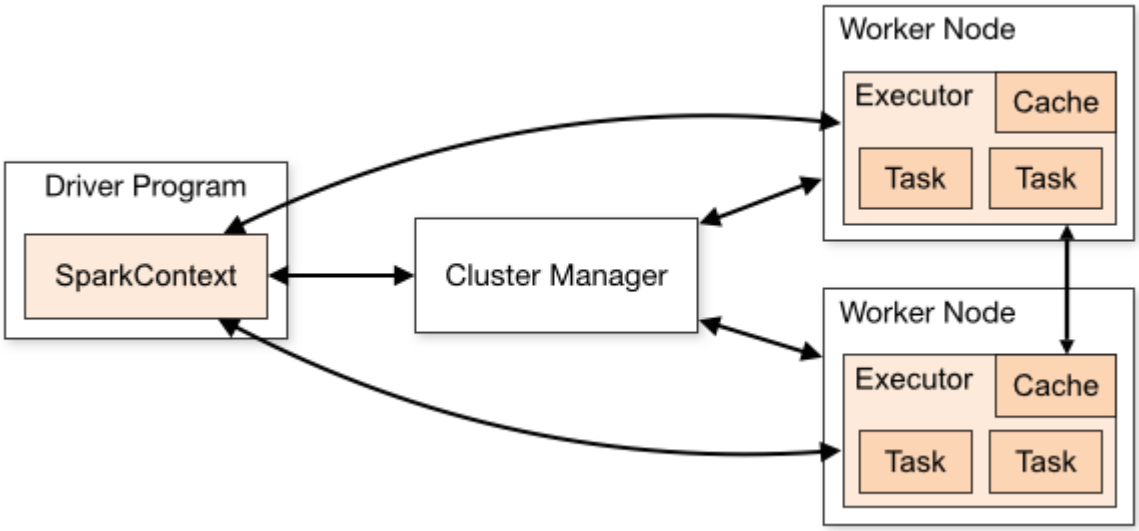


Figure 2.1 : Architecture du cluster sur Spark

En plus de fonctionner sur les gestionnaires de cluster Mesos ou YARN, Spark offre également un mode de déploiement Standlone sur un seul nœud (une seule machine avec les processus JVM nécessaire). Vous pouvez lancer un cluster en monoposte soit manuellement, en démarrant un maître et des esclaves à la main, soit en utilisant les scripts de lancement fournis avec la version compilé de Spark. Ceci est généralement intéressant pour les tests.

2.5.1.2 Pilote Spark et exécuteur Spark

Chaque exécuteur Spark est un processus JVM et est exclusivement affecté à une application Spark spécifique. C'était une décision de conception consciente d'éviter de partager un exécuteur Spark entre plusieurs applications Spark afin de les isoler les unes des autres pour qu'une application Spark qui se comporte mal n'affecte pas les autres applications Spark. La durée de vie d'un exécuteur Spark est la durée d'une application Spark, qui peut durer quelques minutes ou quelques jours. Étant donné que les applications Spark s'exécutent dans des exécuteurs Spark distincts, le partage des données entre eux nécessitera l'écriture des données dans un système de stockage externe comme HDFS (qu'on va aborder par la suite). Comme le montre la Figure 2-2, Spark utilise une architecture maître-esclave, où le pilote Spark est le maître et l'exécuteur Spark est l'esclave. Chacun de ces composants fonctionne comme un processus indépendant sur un cluster Spark. Une application Spark se compose d'un et d'un seul pilote Spark et d'un ou plusieurs exécuteurs Spark. Jouant le rôle d'esclave, chaque exécuteur Spark fait ce qu'on lui dit, c'est-à-dire exécuter la logique de traitement des données sous la forme de tâches. Chaque tâche est exécutée sur un noyau CPU séparé. C'est ainsi que Spark peut accélérer le traitement d'une grande quantité de données en les traitant en parallèle. En plus d'exécuter les tâches

assignées, chaque exécuteur Spark a la responsabilité de mettre en cache une partie des données en mémoire et/ou sur disque lorsque la logique de l'application lui enjoint de le faire.

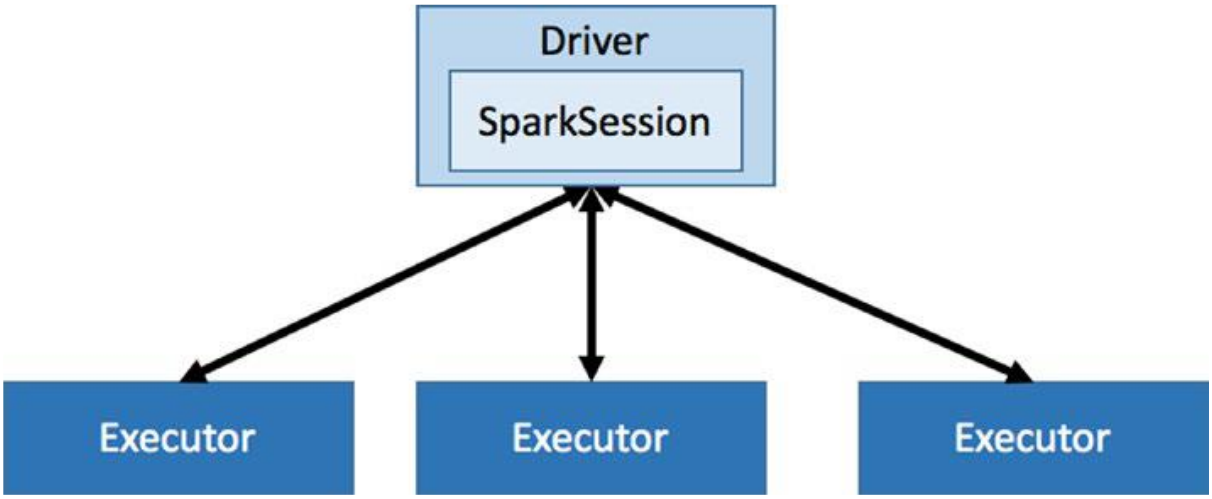


Figure 2.2 : Relation pilote-exécuteurs sur Spark

Lors du lancement d'une application Spark, vous pouvez demander combien d'exécuteurs Spark a cet application besoin et combien de mémoire et le nombre de cœurs CPU chaque exécuteur doit avoir. Pour déterminer le nombre approprié d'exécuteurs Spark, la quantité de mémoire et le nombre de CPU, il faut quantifier de données qui seront traitées, définir la complexité de la logique de traitement des données et la durée souhaitée pendant laquelle une application Spark doit terminer le calcul de traitement. Toutefois, cette tâche est relativement délicate et nécessite l'avis des spécialistes dans l'optimisation des architectures de calculs.

2.5.1.3 Application Spark

Une application Spark se compose de deux parties. La première est le traitement des données applicatives logiques exprimées à l'aide des API Spark, et l'autre est le pilote Spark. La logique de traitement des données de l'application peut être aussi simple que quelques lignes de code pour effectuer quelques traitements de données ou peut être aussi complexe que la formation d'un grand modèle d'apprentissage machine qui nécessite de nombreuses itérations et peut durer plusieurs heures. Le pilote Spark est le coordinateur central d'une application Spark, et il interagit avec un gestionnaire de cluster pour savoir sur quelles machines exécuter la logique de traitement des données. Pour chacun d'entre eux le pilote Spark demande au gestionnaire de cluster de lancer un processus appelé exécuteur Spark. Une autre tâche importante du pilote Spark est de gérer et de distribuer des tâches Spark sur chaque exécuteur au nom de l'application. Si le traitement des données nécessite l'affichage des résultats calculés à un utilisateur par le pilote Spark, alors il

va coordonner avec chaque exécuteur Spark pour recueillir le résultat calculé et le fusionner ensemble. Le point d'entrée dans une application Spark se fait via une classe appelée SparkSession, qui fournit des fonctions de configuration ainsi que des API pour l'expression des données logique de traitement (Dataframes et Datasets). Lorsqu'on travaille avec des RDDs, on est parfois emmené à utiliser une classe appelé SparkContext, qui est un point d'entrée vers les taches (job) Spark, il contient aussi un objet SparkConf() qui permet de générer l'ensemble de configurations que le gestionnaire du cluster utilisera pour gérer le cluster. Séparer les SparkContext entre les JVMs est une bonne pratique.

2.5.2 Concepts

2.5.2.1 RDDs, DataFrames et Datasets

La principale innovation apportée par Spark est le concept de ***Resilient Distributed Dataset***(RDD). Un RDD est une collection (pour en rester à notre vocabulaire) calculée à partir d'une source de données (par exemple une base de données Mongo, un flux de données, un autre RDD) et placée en mémoire RAM. Spark conserve l'historique des opérations qui a permis de constituer un RDD, et la reprise sur panne s'appuie essentiellement sur la préservation de cet historique afin de reconstituer le RDD en cas de panne. Pour le dire brièvement : Spark n'assure pas la préservation des données en ***extension*** mais en ***intention***. C'est l'idée principale pour la ***résilience*** des RDDs.

Par ailleurs, les RDDs représentent des collections partitionnées et distribuées. Chaque RDD est donc constitué de ce que nous avons appelé ***fragments***. Une panne affectant un fragment individuel peut donc être réparée (par reconstitution de l'historique) indépendamment des autres fragments, évitant d'avoir à tout recalculer.

Un RDD est une "boîte" destinée à contenir n'importe quel document, sans aucun préjugé sur la structure (ou l'absence de structure) de ce dernier. Cela rend le système très généraliste, mais empêche une manipulation fine des constituants des documents, comme par exemple le filtrage en fonction de la valeur d'un champ. C'est le programmeur de l'application qui doit fournir la fonction effectuant le filtre.

Spark propose (depuis la version 1.3, avec des améliorations en 1.6 puis 2.0) des RDD ***structurés*** dans lesquels les données sont sous forme tabulaire. Le schéma de ces données est connu. On peut alors assimiler ces RDD à des tables relationnelles. La connaissance du schéma - et éventuellement de leur type - permet à Spark de proposer des opérations plus fines, et des optimisations inspirées des techniques d'évaluation de requêtes dans les systèmes relationnels. En fait, on se ramène à une implantation distribuée du langage SQL. En interne, un avantage important de la connaissance du schéma est d'éviter de recourir à la sérialisation des objets Java (opération effectuée dans le cas des RDD pour écrire sur disque et échanger des données en réseau).

Ces RDDs structurés sont nommées *Dataset* quand le type des colonnes est connu, *Dataframe*sinon. Un *Dataframe* n'est rien d'autre qu'un *Dataset* contenant des lignes de type *Row* dont le schéma précis n'est pas connu.

2.5.2.2 Principe de MapReduce

MapReduce est un principe dérivé du paradigme de la programmation fonctionnelle. Le modèle de programmation MapReduce est conçu spécifiquement pour lire, traiter et écrire des volumes de données très importants. Les programmes adoptant ce modèle sont automatiquement parallélisés et exécutés sur des clusters (grappes). MapReduce consiste en deux fonctions **Map()** et **Reduce()**.

Map est le nom d'une fonction de haut niveau qui applique une fonction donnée à chacun des éléments d'une liste et retourne une liste d'objets. **Reduce** est le nom d'une fonction de haut niveau qui applique une fonction donnée à tous les éléments d'une liste et retourne un objet unique. Voici un exemple du principe dans la figure ci-dessous :

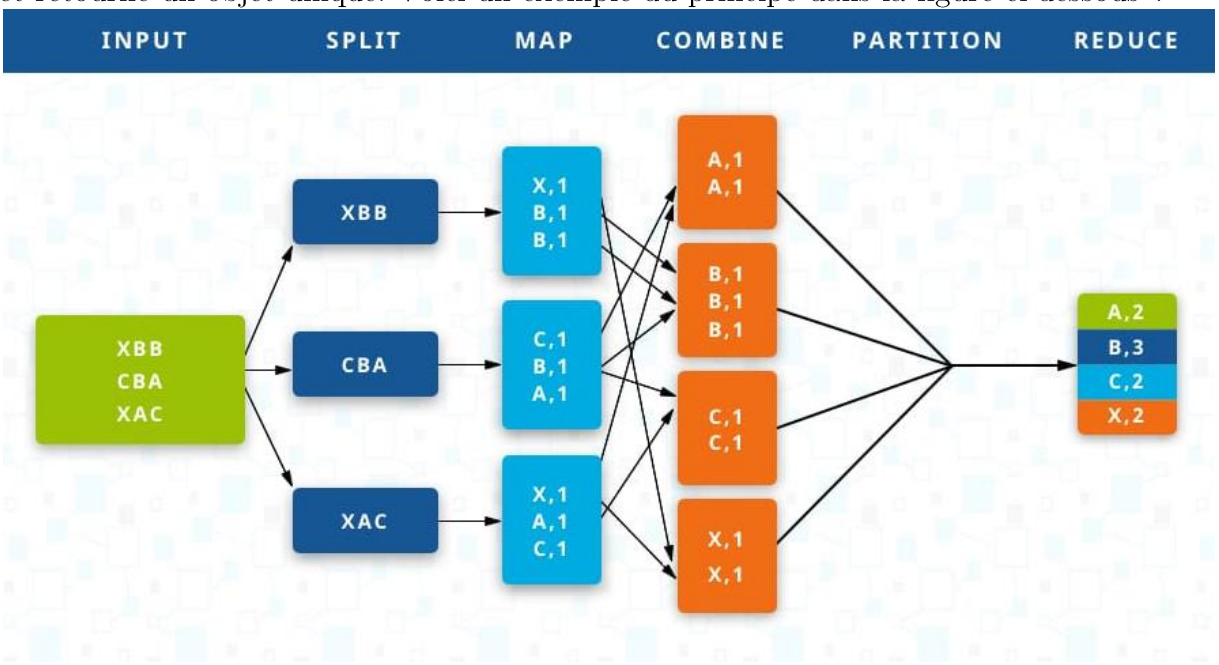


Figure 2.3 : Les étapes du calcul MapReduce à travers un exemple « comptage des lettres »

2.5.2.3 DAG Ordonnanceur

Pour comprendre l'ordonnancement sur Spark, il faut faire la différence entre deux types d'opération possible sur Spark , transformation et action. Les transformations sont des opérations qui transforment vos données RDDs d'un formulaire à un autre. Et lorsque vous appliquez cette opération sur n'importe quel RDD, vous obtenez une nouvelle RDD avec des données transformées (les RDD dans Spark sont immuables). Les opérations comme map, filter, flatMap sont des transformations. Les actions sont des opérations

RDDs qui donnent des valeurs non-RDDs. Les valeurs d'action sont stockées dans les pilotes ou dans le système de stockage externe. Il met en mouvement la paresse de la RDD. On prend comme exemple sur des actions reduce(), count(), first(), et foreach().

Pour l'ordonnanceur DAG (Direct Acyclic Graph) (DAG scheduler), Il s'agit d'une couche d'ordonnancement dans Spark qui implémente un ordonnancement orienté par étapes. Il convertit le plan d'exécution logique en un plan d'exécution physique. Lorsqu'une action est appelée, Spark va directement à l'ordonnanceur DAG. Il exécute les tâches qui sont soumises à l'ordonnanceur. Spark utilise des opérations de pipelines (lignage) pour optimiser son travail, ce processus combine les transformations en une seule étape. Le concept de base de DAG scheduler est de maintenir les emplois et les étapes. Il fait le suivi des registres et des compteurs internes.

Lorsque l'utilisateur exécute une action (opération comme "count"), le graphique (le schéma d'ordonnancement des taches) est soumis à un DAG Scheduler. Le planificateur DAG divise le graphique de l'opérateur en étapes (mapper et réduire). Une étape (stage) est composée de tâches (tasks) basées sur des partitions des données d'entrée, où DAG est appliqué pour connecter ces tâches. De plus, le planificateur DAG permet aux opérateurs de pipelines d'optimiser l'ensemble du graphe.

2.6 Pile Applicative de Spark

2.6.1 Présentation

Contrairement à ses prédécesseurs, Spark fournit un moteur de traitement de données unifié connu sous le nom de Pile de Spark (Spark Stack). Semblable à d'autres systèmes bien conçus, cette pile est construite sur une structure solide appelée noyau de Spark (Spark Core), qui fournit toutes les fonctionnalités nécessaires à la gestion et l'exécution des applications distribuées telles que la planification, la coordination et la tolérance aux pannes. De plus, il fournit une abstraction de programmation puissante et générique pour les données, appelés ensembles de données distribuées résilientes (RDD). En plus de cette base solide et d'ensemble des composants dont chacun est conçu pour le traitement de données spécifique comme le montre la figure 2-4. Spark SQL est fait pour les données par lots ainsi que les données interactives la transformation. Spark Streaming est destiné au traitement en temps réel des données de streaming. Spark GraphX à pour objective de traitement des graphiques. Spark MLlib est destiné à l'apprentissage automatique. Finalement, objective du Spark R est d'exécuter des tache d'apprentissage automatique à l'aide de l'interpréteur de commandes R.

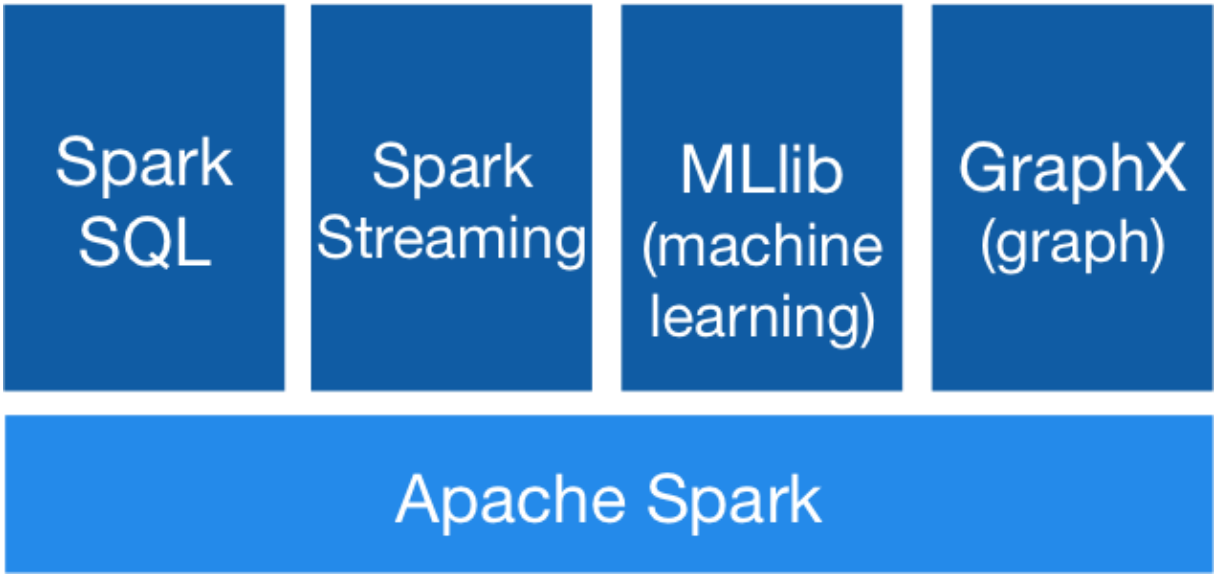


Figure 2.4 : La pile des librairies intégrées à Spark

2.6.2 Avantages du model unifié de la pile

Ce moteur unifié apporte plusieurs avantages importants à la tâche de construction d'applications évolutives et des applications intelligentes de données massives.

- Tout d'abord, les applications sont plus simples à développer et à utiliser et déployer parce qu'ils utilisent un ensemble unifié d'API et tournent sur un seul moteur.
- Deuxièmement, il est plus efficace pour combiner différents types de traitement de données (batch, streaming, etc.) parce que Spark peut exécuter ces différents ensembles d'APIs sur les mêmes données sans écrire les données intermédiaires vers un système de stockage.
- Enfin, l'avantage le plus excitant est que Spark permet de nouvelles applications qui n'étaient pas possibles auparavant en raison de sa facilité de la composition de différents ensembles de types de traitement de données dans une application Spark. Par exemple, il peut exécuter des requêtes interactives sur les résultats des prédictions d'apprentissage automatique en temps réel. Une analogie à laquelle tout le monde peut s'identifier est le smartphone, qui consiste d'un appareil photo puissant, d'un téléphone portable avec des modules de communication, un processeur relativement puissant, un OS et d'un ensemble de capteurs. En combinant les fonctions d'un smartphone, on a pu développer des applications

innovantes telles que Waze, un système de gestion du trafic et une application de navigation.

2.6.3 Composants de la pile unifiée de Spark

2.6.3.1 Noyau de Spark (Spark core)

Spark Core est la base du moteur de traitement de données distribué Spark. Il se compose de deux parties : l'infrastructure informatique distribuée et l'abstraction de la programmation en RDD.
L'infrastructure informatique distribuée est responsable de la distribution, de la coordination et de la planification des tâches informatiques sur de nombreuses machines du cluster. Cela permet d'effectuer le traitement parallèle d'un grand volume de données de manière efficace et rapide sur un grand nombre de machines. Deux autres responsabilités importantes de l'infrastructure informatique distribuée sont le traitement des défaillances des tâches informatiques et le transfert efficace des données d'une machine à l'autre, ce que l'on appelle le brassage des données (data shuffeling). Les utilisateurs avancés de Spark doivent avoir une connaissance approfondie de l'infrastructure informatique distribuée Spark pour être en mesure de concevoir des applications Spark hautement performantes.

L'abstraction de la programmation clé dans Spark s'appelle RDD. Le développeur de Spark doit avoir une certaine connaissance, en particulier de ses API et de ses principaux concepts. La définition technique d'un RDD est qu'il s'agit d'une collection d'objets immuable et tolérante aux pannes répartis sur un cluster qui peuvent être manipulés en parallèle. Essentiellement, il fournit un ensemble d'API permettant aux développeurs d'applications Spark d'exécuter facilement et efficacement les tâches de traitement de données à grande échelle sans se soucier de l'endroit où les données résident dans le cluster ou de l'endroit où elles sont traitées ou bien des pannes de machines. Par exemple, disons que vous avez un fichier journal de 1,5 To qui réside sur HDFS et vous devez connaître le nombre de lignes contenant le mot Exception. Vous pouvez créer une instance de RDD pour représenter toutes les instructions de journal dans ce fichier journal, et Spark peut les partitionner à travers les nœuds du cluster de manière à ce que la logique de filtrage et de comptage puissent être exécutée en parallèle pour accélérer la logique de recherche et de comptage.

Les API RDDs sont exposées dans de multiples langages de programmation (Scala, Java, et Python), et ils permettent aux utilisateurs de passer des fonctions locales à exécuter sur le cluster, qui est quelque chose de puissant et d'unique

Le reste des composants de la pile Spark sont conçus pour fonctionner sur Spark Noyau. Par conséquent, toute amélioration ou optimisation effectuée dans Spark Core entre de Spark seront automatiquement disponibles pour les autres composants.

2.6.3.2 Spark SQL

Spark SQL est un composant construit sur Spark Core, et il est conçu pour le traitement structuré des données à l'échelle. Sa popularité a monté en flèche depuis sa création parce qu'il apporte un nouveau niveau de flexibilité, de facilité d'utilisation et de performance.

Le langage SQL (Structured Query Language) a été la langue véhiculaire du traitement des données parce qu'il est assez facile pour les utilisateurs d'exprimer leur intention, et le besoins alors que le moteur d'exécution effectue les optimisations intelligentes nécessaires. Spark SQL apporte cela dans le monde du traitement des données au niveau du Pétaoctet (2^{50} octets). Les utilisateurs de Spark peuvent maintenant émettre des requêtes SQL pour exécuter le traitement des données ou l'utilisation de l'abstraction de haut niveau exposée à travers les DataFrames APIs. Une DataFrame est en fait une collection distribuée de données organisées en colonnes. Ce n'est pas une idée nouvelle ; en fait, cette idée a été inspirée par les trames de données de R et de Python. Une façon plus facile de penser à un DataFrame est qu'il est conceptuellement équivalent à une table dans une base de données relationnelle.

Dans les coulisses, Spark SQL exploite l'optimiseur Catalyst pour effectuer les types d'optimisations qui sont couramment effectuées dans de nombreux moteurs de bases de données analytiques.

Une autre caractéristique de Spark SQL est la possibilité de lire et d'écrire des données à partir de divers formats structurés et de systèmes de stockage, tels que JavaScript Object Notation (JSON), fichiers de valeurs séparées par des virgules (CSV), fichiers Parquet ou ORC, bases de données relationnelles, Hive, et autres. Cette fonctionnalité aide vraiment à élever le niveau de polyvalence de Spark car Spark SQL peut être utilisé comme un outil de conversion de données pour convertir facilement des données d'un format à un autre.

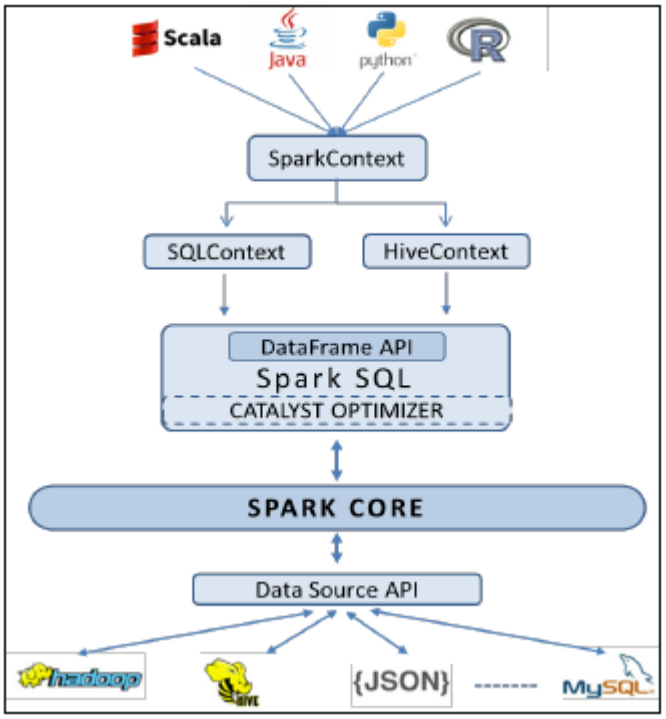


Figure 2.5 : L'architecture fonctionnelle du SparkSQL avec le coeur Spark

2.6.3.3 Spark Streaming

Il a été dit que "Les données en mouvement ont une valeur égale ou supérieure aux données historiques." La capacité de traiter les données au fur et à mesure qu'elles arrivent devient un avantage concurrentiel pour de nombreuses entreprises dans des secteurs hautement concurrentiels. Le module Spark Streaming permet de traiter les données de streaming en temps réel à partir de différentes sources de données à haut débit et de manière tolérante aux pannes. Les données peuvent provenir de sources telles que Kafka, RabbitMQ, Flume, Kinesis, Twitter, HDFS ou TCP.

La principale abstraction dans la première génération du traitement Spark Streaming est appelé flux discret (DStream), qui implémente un flux incrémentiel en divisant les données d'entrée en petits lots (en fonction d'une période de temps) qui permet de combiner régulièrement l'état de traitement actuel pour produire de nouveaux résultats. En d'autres termes, une fois que les données entrantes sont divisées en petits lots, chaque lot est traité en tant que RDD et répliqué sur le cluster pour qu'elles puissent être traitées en conséquence.

Le traitement d'un flux implique parfois de se joindre à des données au repos, et Spark le rend facile à faire. En d'autres termes, combiner les requêtes batch et interactives avec le traitement en continue (streaming) peut être facilement effectué dans Spark grâce à la pile Spark unifiée.

Un nouveau moteur de traitement de flux évolutif et tolérant aux pannes appelé « Structured Streaming » a été introduit dans Spark 2.1, et il a été construit sur le moteur SQL Spark. Ce moteur simplifie davantage la vie des développeurs d'applications de traitement en continu en traitant le calcul du streaming de la même manière qu'on exprimerait un lot calcul sur données statiques. Ce nouveau moteur exécutera automatiquement le streaming le traitement de la logique de façon incrémentale et continue au fur et à mesure que de nouvelles données de streaming continuent d'arriver. Une caractéristique nouvelle et importante que le streaming structuré offre est la possibilité de traiter les données de streaming entrantes en fonction de l'heure de l'événement, ce qui est nécessaire pour de nombreuses personnes des nouveaux cas d'utilisation du traitement en continu. Une autre caractéristique unique de l'outil Structured Streaming et que le moteur de streaming est la garantie de bout en bout, exactement une fois, ce qui fera une grande quantité de données. Ceci facilite la tâche de l'ingénieur beaucoup plus qu'avant en termes de sauvegarde des données dans un système de stockage tel qu'une base de données relationnelle ou une base de données NoSQL .

Au fur et à mesure que ce nouveau moteur mûrit, il permettra sans aucun doute une nouvelle classe de streaming le traitement d'applications faciles à développer et à maintenir. Selon Reynold Xin, l'architecte en chef de Databricks, la façon la plus simple d'exécuter des projets streaming analytics est de ne pas avoir à raisonner sur le streaming.

2.6.3.4 Spark MLlib

En plus de fournir plus de 50 algorithmes d'apprentissage, la bibliothèque Spark MLlib fournit des abstractions pour la gestion et la simplification de nombreuses tâches de construction de modèles d'apprentissage automatique, telles que la fonctionnalisation, le pipeline pour la construction, l'évaluation et le réglage du modèle et la persistance des modèles pour aider au passage du modèle du développement à la production.

A partir de Spark 2.0, les API MLlib seront basées sur des DataFrames à prendre en charge l'avantage de la facilité d'utilisation et des nombreuses optimisations fournies par le Catalyseur et composants en tungstène dans le moteur SQL Spark.

Les algorithmes d'apprentissage automatique sont de nature itérative, ce qui signifie qu'ils passent par plusieurs itérations jusqu'à ce qu'un objectif souhaité soit atteint. L'étincelle rend les choses extrêmement faciles pour implémenter ces algorithmes et les exécuter de manière évolutive à travers un cluster de machines. Les algorithmes d'apprentissage automatique couramment utilisés tels que la classification, la régression, la classification non-supervisé (clustering) et le filtrage collaboratif sont disponibles dès à présent pour les données des scientifiques et des ingénieurs à utiliser.

2.6.3.5 Spark Graphx

Le traitement graphique fonctionne sur des structures de données composées de sommets et de bords qui les relient. Une structure de données graphique est souvent utilisée pour représenter des réseaux réels, des entités interconnectées, y compris les réseaux sociaux professionnels sur LinkedIn, les réseaux de pages Web connectées sur Internet, et ainsi de suite. Spark GraphX est un composant qui permet des calculs parallèles au graphique en fournissant une abstraction d'une valeur de multigraphe avec des propriétés attachées à chaque sommet et bord. Le composant GraphX comprend une collection d'algorithmes communs de traitement des graphiques, y compris les classements de pages, les composants connectés, les chemins les plus courts et autres.

2.6.3.6 Spark R

SparkR est un paquet R qui fournit un frontal léger pour utiliser Apache Spark. R est un langage de programmation statistique populaire qui prend en charge le traitement des données et les tâches d'apprentissage automatique. Cependant, R n'a pas été conçu pour traiter de grandes bases de données qui ne peuvent pas tenir sur une seule machine. SparkR exploite le moteur de calcul distribué de Spark pour permettre l'analyse de données à grande échelle en utilisant l'interpréteur de commandes R et les API populaires que de nombreux scientifiques ont l'habitude d'utiliser.

2.7 Spark pour l'apprentissage automatique

2.7.1 MLlib

MLlib est la bibliothèque d'apprentissage machine d'Apache Spark. Il est évolutive et se compose de nombreux algorithmes d'apprentissage machine couramment utilisés[19]. MLlib intègre des algorithmes pour :

- Traitement des types de données sous forme de vecteurs et de matrices
- Calculer des statistiques de base comme des statistiques sommaires et des corrélations, ainsi que produire des échantillons aléatoires et stratifiés simples, et effectuer des tests d'hypothèses simples
- Réalisation de la classification et de régression
- Filtrage collaboratif
- Regroupement
- Réaliser la réduction de la dimensionnalité
- Extraction et transformation d'entités conductrices
- Extraction fréquente de gabarits

- Développer l'optimisation
- Exportation de modèles PMML

Le MLlib Spark est toujours en cours de développement et de nouveaux algorithmes sont attendus à chaque nouvelle version. Conformément à la philosophie informatique d'Apache Spark, le MLlib est conçu pour une utilisation et un déploiement facile, avec des performances élevées. MLlib utilise le paquet d'algèbre linéaire Breeze, qui dépend de netlib-java, et jblas. Les paquets netlib-java et jblas dépendent également des routines Fortran natif. Les utilisateurs doivent installer la bibliothèque d'exécution gfortran si elle n'est pas déjà installée présents sur leurs nœuds (contenu dans la version compilé de Spark). MLlib lancera une erreur de liaison s'il ne peut pas les détecter automatiquement.

2.7.2 Autres librairies compatibles avec Spark

Comme nous l'avons vu dans la partie précédente, MLlib a rendu disponibles de nombreux algorithmes fréquemment utilisés comme la régression et la classification. Mais ces bases ne sont pas suffisantes pour un apprentissage machine compliqué.

Si nous attendons que l'équipe d'Apache Spark ajoute tous les algorithmes ML nécessaires, il se peut que ça prenne du temps. De nombreuses tierces parties ont contribué au projet ML à Apache Spark.

IBM a apporté sa bibliothèque d'apprentissage automatique, « SystemML », à Apache Spark.

En plus de ce que MLlib fournit, SystemML offre beaucoup plus d'algorithmes ML supplémentaires comme celles sur l'imputation des données manquantes, SVM, GLM, ARIMA et l'imputation non linéaire, quelques algorithmes de modélisation graphique et de factorisation matricielle.

Développé par le groupe IBM Almaden Research, SystemML d'IBM est un moteur pour l'apprentissage automatique distribué et peut s'adapter à de grandes tailles de données arbitraires. Il offre les avantages suivants :

- Unifie les environnements d'apprentissage machine fragmentés
- Donne à l'écosystème de base de Spark un ensemble complet de DML
- Permet à un spécialiste des données de se concentrer sur l'algorithme et non sur l'implémentation.
- Augmente la valeur du temps pour les équipes de recherche en sciences des données
- Établit une norme de facto pour les routines d'apprentissage automatique réutilisables.

SystemML est modélisé d'après la syntaxe et la sémantique R, et offre la possibilité de créer de nouveaux algorithmes via son propre langage.

Grâce à une bonne intégration avec R by SparkR, les utilisateurs d'Apache Spark ont également la possibilité d'utiliser des milliers de paquets R pour des algorithmes d'apprentissage automatique, si nécessaire.

2.8 Conclusion

- Apache Spark a devenu l'un des frameworks pour l'analyse Big Data les plus reconnus depuis sa création. Cela a créé beaucoup d'enthousiasme et de nombreuses possibilités dans le monde des données massives. Le plus important encore, est qu'il vous permet de créer de nombreuses nouvelles et innovantes applications, d'analyse sur des données massives pour résoudre un grand nombre de problèmes divers selon les cas d'utilisation.
- Les trois propriétés importantes de Spark à noter sont la facilité d'utilisation, la rapidité et la flexibilité.
- L'infrastructure informatique distribuée Spark utilise l'architecture maître-esclave. Chaque application Spark est constituée d'un pilote qui joue le rôle principal, et un ou plusieurs exécuteurs, qui sont les esclaves, pour traiter les données en parallèle.
- Spark fournit un moteur de traitement de données unifié, évolutif et distribué qui peut être utilisé pour le traitement par lots, traitement en temps réel, interactif et traitement exploratoire des données, formation les modèles d'apprentissage automatique et l'exécution de prédictions, et les graphiques la transformation.
- Les applications Spark peuvent être écrites en programmation multiple y compris Scala, Java, Pythone

Chapitre 3 :

Solution Cloud Open Source: OpenStack

3 Solution Cloud Open Source: OpenStack

3.1 Introduction

Lors de la conférence OpenStack de Vancouver en mai 2015, le géant américain du commerce de détail Walmart a annoncé qu'ils avaient déployé un cloud OpenStack avec 140 000 cœurs de calcul prenant en charge 1,5 milliard de pages vues. Le CERN, ancien utilisateur d'OpenStack, annonce que leur cloud privé OpenStack avait atteint 100 000 cœurs fonctionnant en calcul, ce calcul est chargé sur deux pétaoctets de disque en production. Aussi, 250 autres entreprises et organisations à travers 9 secteurs de l'industrie ont annoncé qu'elles avaient adopté OpenStack dans leurs centres de données.

Par son importance, OpenStack avait complètement redessiné le paysage du cloud privé au cours des cinq courtes années de son existence. Dans cette partie, nous voyons une brève explication d'Openstack, son utilité, une architecture globale d'Openstack et ses composants, ainsi qu'une structure générale de chacun de ses composants.

3.2 Qu'est-ce que Openstack

OpenStack est mieux défini par ses cas d'utilisation, car les utilisateurs et les contributeurs abordent le logiciel avec de nombreux objectifs différents à l'esprit. Pour les hébergeurs tels que Rackspace, OpenStack fournit l'infrastructure pour une plate-forme de services partagés multi-locataires. Pour d'autres, il pourrait fournir un mécanisme de provisionnement des données et de calcul pour une application de Business Intelligente distribuée.[20]

3.3 Historique d'Openstack

Le système opérationnnel du Cloud OpenStack a été créé en juin 2010 en tant que projet permettant de relier les Système de développement de serveurs virtuels Nova crée par NASA et le système de stockage de données Swift du fournisseur d'hébergement américain « Rackspace » .La première version est sous le nom de code Austin, est sortie en octobre 2010.

Il est important de comprendre que OpenStack est en soi, un projet de développement. Le site web Openstack.org ne fournit aucune référence pour la distribution. Sinon, différents fournisseurs pourraient créer leurs propres distributions basées sur ce code de projet.

Actuellement, OpenStack est développé sous le contrôle de la « OpenStack Foundation » avec environ 18 000 membres individuels et plus de 500 membres corporatifs. Presque tous les leaders du marché informatique prennent en charge [21]

La figure ci-dessus présente les versions vivantes d'Openstack :

Series	Status	Initial Release Date	Next Phase	EOL Date
Train	Development	2019-10-16 *estimated* (schedule)	Maintained *estimated 2019-10-16*	
Stein	Maintained	2019-04-10	Extended Maintenance *estimated 2020-10-10*	
Rocky	Maintained	2018-08-30	Extended Maintenance *estimated 2020-02-24*	
Queens	Maintained	2018-02-28	Extended Maintenance *estimated 2019-08-25*	
Pike	Maintained	2017-08-30	Extended Maintenance *estimated 2019-03-03*	
Ocata	Extended Maintenance	2017-02-22	Unmaintained *estimated TBD*	

Figure 3.1 : Les différentes versions d'OpenStack

3.4 Architecture d'Openstack

Les composants d'Openstack se divisent en trois groupes :

- Contrôle
- Réseaux (Network)
- Calcul (Compute)

Le niveau de contrôle exécute les services d'interface de programmation d'application (API), de l'interface Web, de la base de données et de bus de messages.

Tandis que le niveau réseau (Network) exécute des agents de ce service pour la mise en œuvre du réseau. Alors que le nœud de calcul est l'hyperviseur de virtualisation qui fournis des services et des agents pour gérer les machines virtuelles.[22]

Tous les composants utilisent une base de données et / ou un bus de messages. La base de données peut être MySQL, MariaDB ou PostgreSQL. Les bus de messages est un outil de transmission et réceptions des messages entres les services, les moyens les plus utilisés et les plus populaires pour la fonction de la messagerie, sont RabbitMQ, Qpid et ActiveMQ. Pour plus petit déploiements, la base de données et les services de messagerie sont généralement exécutés sur le nœud de contrôle, mais ils pourraient avoir leurs propres nœuds si nécessaire.

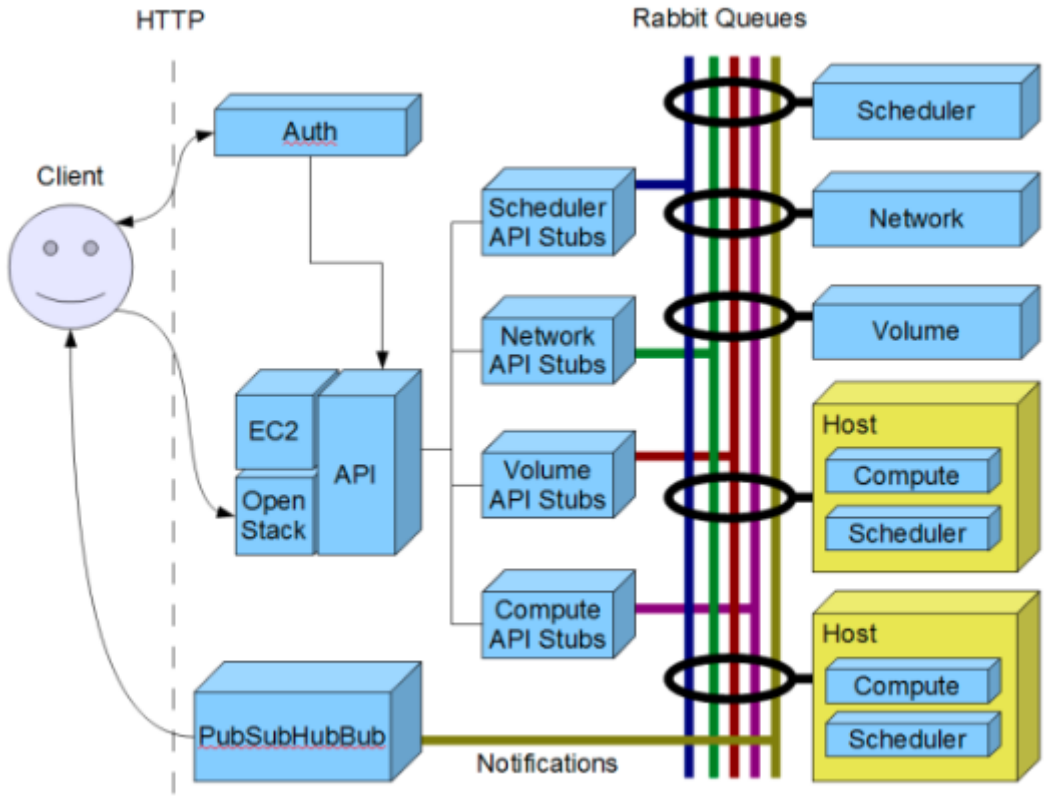


Figure 3.2 : Une architecture globale d'OpenStack

3.5 Composants d'Openstack

Maintenant qu'une architecture logique de base d'OpenStack est définie, voyons quels composants de cette architecture de base qui nous donne une structure générale du Cloud OpenStack. Pour ce faire, nous allons faire une description de chaque élément avec ses caractéristique, fonctionnement et son architecture.[23]

3.5.1 Service d'authentification : Keystone

Le service OpenStack Identity, connu sous le nom de Keystone, fournit des services d'authentification et de gestion des comptes d'utilisateurs et des informations de rôle pour notre environnement OpenStack en Cloud.

C'est un service crucial qui sous-tend l'authentification et la vérification entre tous nos services cloud OpenStack et c'est le premier service qui doit être installé dans un environnement Cloud (OpenStack). L'authentification avec le service OpenStack Identity renvoie un jeton d'autorisation (identification Token) qui est passé entre tous les services, une fois validé. Ce jeton est ensuite utilisé comme une authentification pour que puissiez utiliser les autres services, tel que le stockage et le calcule. En conséquence, la configuration du service identité d'OpenStack consiste à créer des rôles appropriés pour les utilisateurs et les services, les locataires, les comptes utilisateurs et les points finaux des API des services qui composent notre infrastructure cloud.

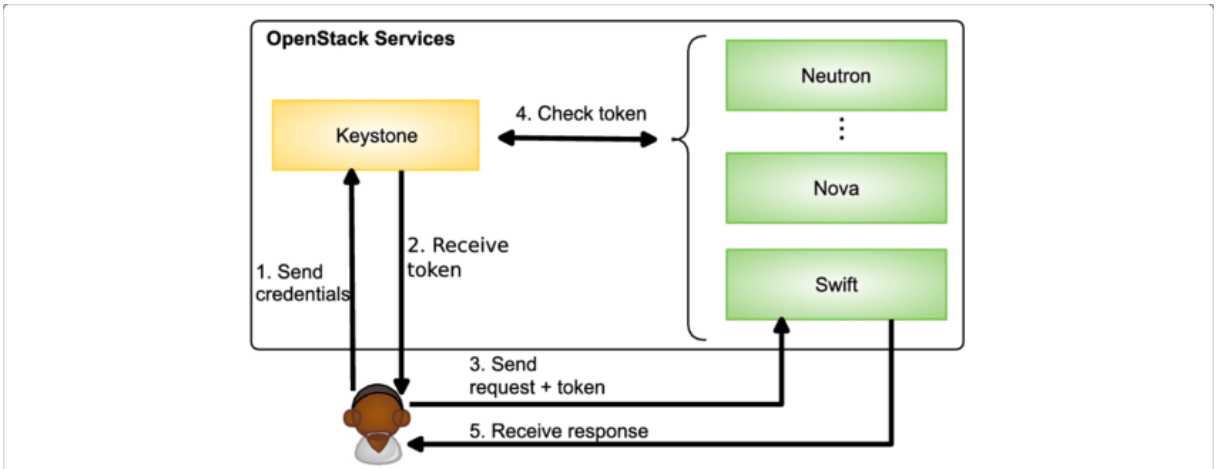


Figure 3.3 : Architecture Keystone

3.5.2 Service de gestion des images :Glance

OpenStack Image Service, connu sous le nom de Glance, est le service qui vous permet d'enregistrer, de découvrir et de récupérer des images de machines virtuelles pour une utilisation dans l'environnement OpenStack. Les images mises à disposition via OpenStack Image Service peuvent être stockées dans une variété d'emplacements back-end, du stockage local du système de fichiers aux systèmes de fichiers distribués tels que OpenStack Storage.

Ce disque d'image qui stocke toute le contenue du disque dure, ce classe selon les types de format, on site ici quelque types de ces images :

- **QCOW2 (QEMU Copy-On-Write):**

  Utilisé avec QEMU et QEMUKVM hyperviseurs et nommés d'après leur stratégies d'optimisation de stockage sur le disque qui retardent l'allocation de stockage jusqu'au besoin.

- **VMDK (Virtual Machine Disk):**

  Initialement conçu pour être utilisé avec la gamme d'hyperviseurs de VMware, mais désormais un format ouvert compatible avec d'autres hyperviseurs non VMware tels que VirtualBox.

- **VHD and VHDX (Virtual Hard Drive):**

  Créé à l'origine par le Connectix Corporation pour l'hyperviseur de type 2, Virtual PC acquise par Microsoft en 2003. Il est couramment utilisé avec les produits liés à Microsoft et services, tels que Hyper-V et Azure.

- **VDI (Virtual Disk Image):**

Utilisé par VirtualBox - Oracle.

- **ISO (International Organization for Standardization, ISO9660):**

Un fichier d'archive d'un disque optique. Il est composé du contenu des données de chaque secteur qu'écrit sur un disque optique, y compris le système de fichiers du disque optique.

- **RAW** :

Une image disque non compressée. Nova va décompresser tout image compressée en nuage et placez-la au format brut au moment du démarrage de l'instance.

La figure suivante donne une simple explication du fonctionnement générale de service Glance :

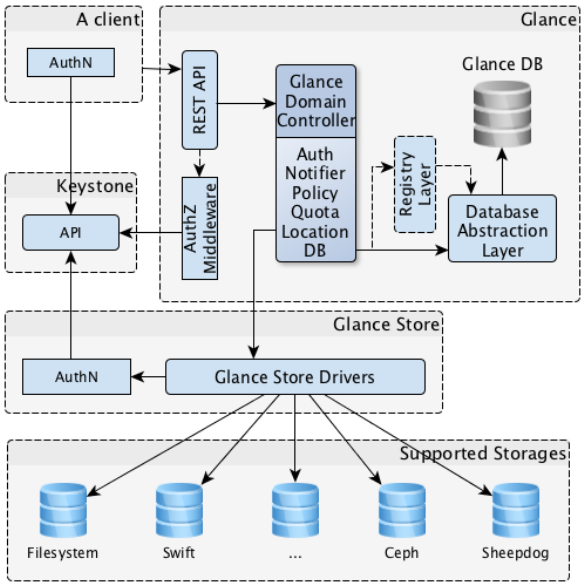


Figure 3.4 : Architecture Glance

3.5.3 Service de calcul : Nova

Service d'OpenStack de calcul (OpenStack Compute), aussi connu sous le nom de Nova, est le composant de calcul du système d'exploitation. C'est le composant qui vous permet d'exécuter plusieurs instances de machines virtuelles sur n'importe quel nombre d'hôtes exécutant le service OpenStack Compute, vous permettant ainsi de créer un environnement Cloud hautement évolutif et redondant. OpenStack Compute alimente certains des plus grands Cloud de calcul tels que le Rackspace Open Cloud.

La figure suivante montre l'architecture générale d'unité de calcule Nova

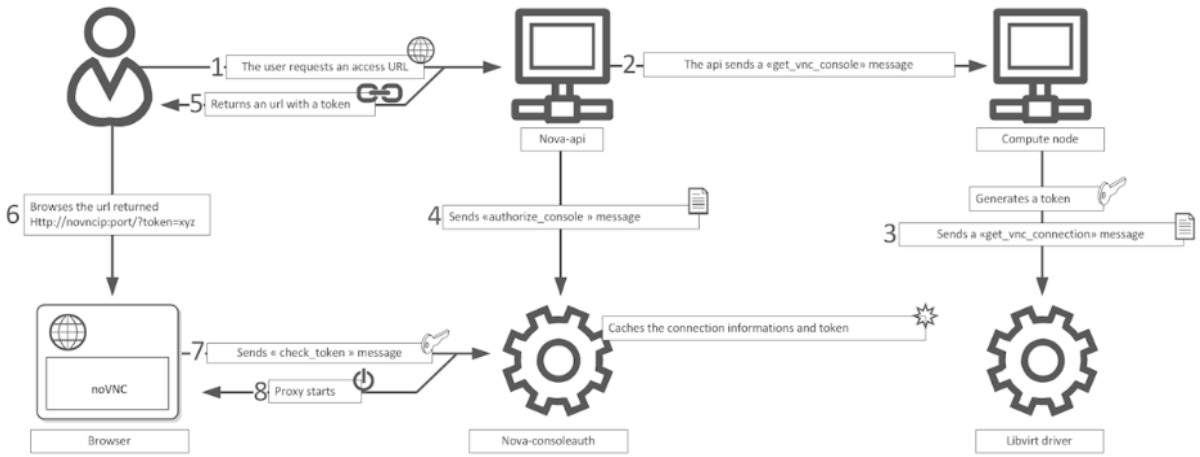


Figure 3.5 : Architecture Nova

3.5.4 Service de stockage des objets :Swift

OpenStack Object Storage, également connu sous le nom de Swift, est le service qui permet de stocker massive des objets. Ce service de stockage est évolutif et hautement redondant sur du matériel standard. Aussi il est analogue au service de stockage S3 d'Amazon qu'est géré de manière similaire sous OpenStack. Avec OpenStack Swift, nous pouvons stocker de nombreux objets de taille virtuellement illimitée, limités par la fonction matérielle disponible et faire évoluer notre environnement en fonction des besoins, afin d'accommoder notre espace de stockage. La nature hautement redondante d'OpenStack Object Storage est idéale pour l'archivage des données (comme les logs) ainsi qu'un système de stockage tel que OpenStack Compute peut etre utiliser pour les machine virtuels .la figure suivante montre les principaux du composants Swift

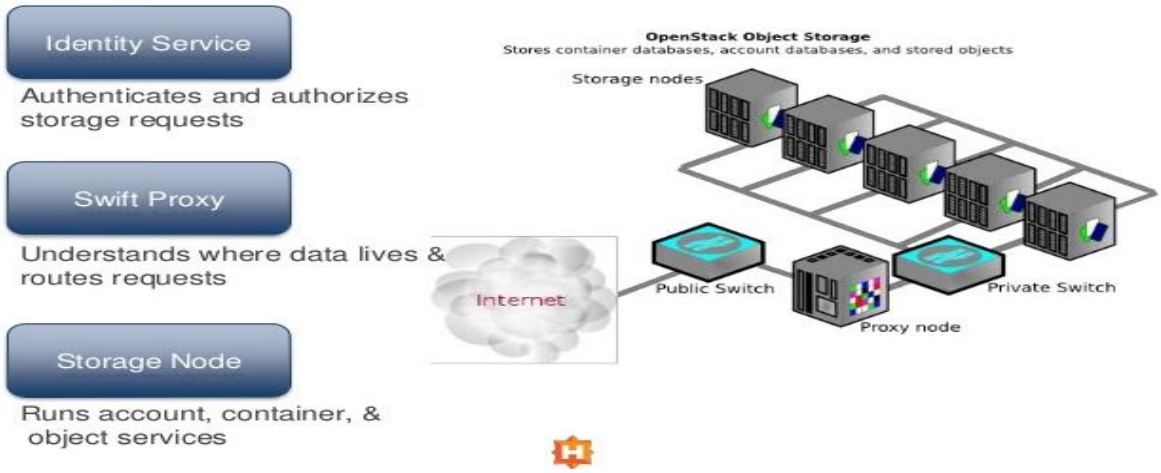


Figure 3.6 : Architecture Swift

3.5.5 Service des stockage en volume :Cinder

Les données écrites sur les instances en cours d'exécution sur les disques ne sont pas persistantes, ce qui signifie que lorsque vous mettez fin à de telles instances, toute écriture sur disque sera perdue. Les volumes sont des éléments de stockage persistant que vous pouvez attacher à vos instances OpenStack Compute en cours d'exécution.

Dans les versions antérieures d'OpenStack, les services de volume étaient fournis par nova-volume qui ont évolué au fil du temps en OpenStack Block Storage, alias Cinder. OpenStack Block Storage est très similaire à Elastic Block Storage d'Amazon EC2 - la différence réside dans la façon dont les volumes sont présentés aux instances en cours. Sous OpenStack Compute, les volumes peuvent être facilement gérés à l'aide d'un groupe de volumes LVM exposé iSCSI nommé cinder-volumes, donc cela doit être présent sur tout hôte exécutant le service Cinder volume.

La figure suivante montre le cheminement de lancement des instances pour le Cloud Openstack .

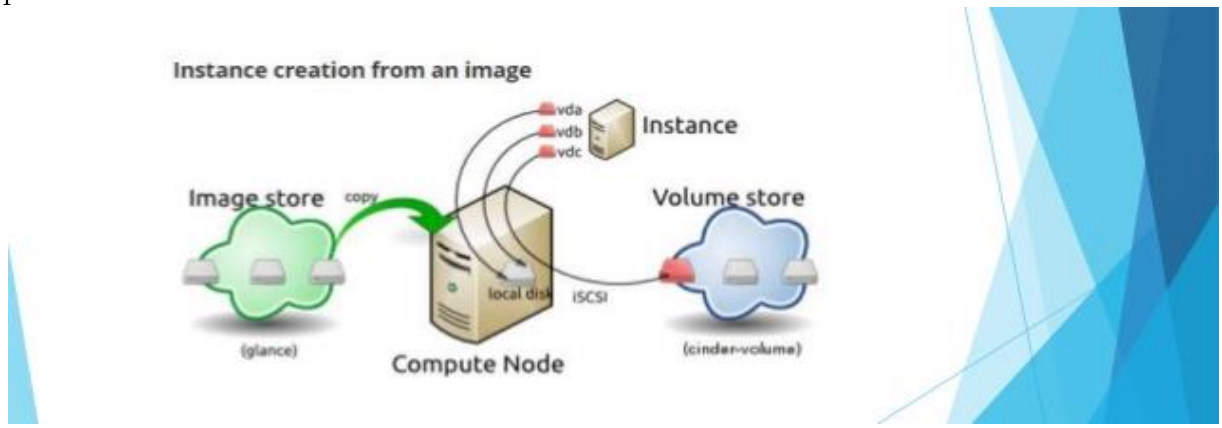


Figure 3.7 : Architecture Cinder

3.5.6 Service de gestion du réseau : Neutron

OpenStack supporte trois modes de mise en réseau dans la version actuelle de Grizzly. Il s'agit Flat Networking, VLAN Manager, et le tout dernier SDN (Software Defined Networking). Le SDN est une approche de réseautage dans laquelle les administrateurs de réseau et les opérateurs de cloud peuvent définir des services de réseau virtuel par programmation. Le composant réseau qui est défini par d'OpenStack Networking est appelé Neutron.

Avec le SDN, nous pouvons décrire des réseaux complexes dans un environnement sécurisé multi-locataires qui surmonte les problèmes souvent associés aux réseaux Flat et VLAN.

SDN dans OpenStack est également une architecture connectable, ce qui signifie que nous sommes en mesure de brancher et de contrôler divers commutateurs, pare-feu, répartiteurs de charge et de réaliser diverses fonctions comme le service du pare-feu - le tout défini dans le logiciel pour vous donner le contrôle fin du grain sur votre infrastructure cloud complète.

La figure suivante montre l'architecture globale de Neutron.

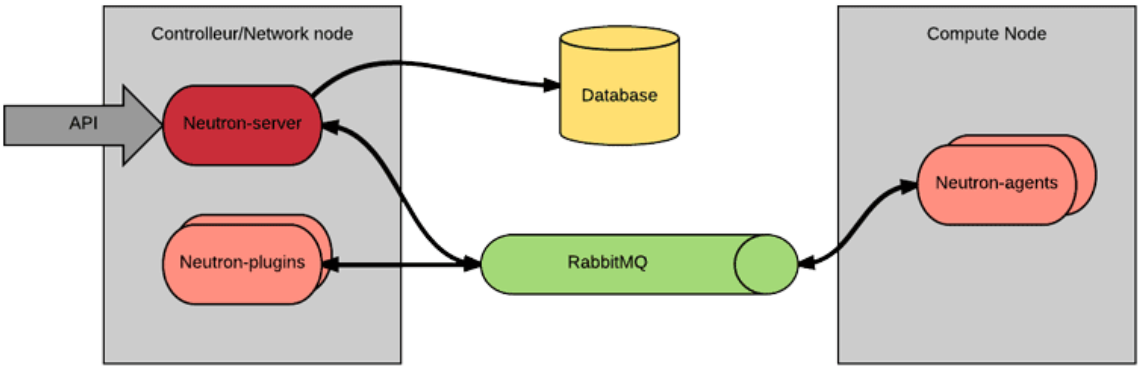


Figure 3.8 : Architecture Neutron

3.5.7 Tableau de bord : Horizon

La gestion de notre environnement OpenStack via une interface en ligne de commande nous permet un contrôle complet de notre environnement cloud, mais disposer d'une interface graphique que les opérateurs et administrateurs peuvent utiliser pour gérer leurs environnements et les instances, facilite ce processus. OpenStack Dashboard, connu sous le nom de « Horizon », fournit cette interface graphique.Horizon est un service Web qui s'exécute à partir d'une installation Apache, en utilisant l'interface WSGI (Web Service Gateway Interface) de Python et Django, qui est un framework Web en développement rapide.

Avec OpenStack Dashboard, nous pouvons gérer tous les composants principaux de notre environnement OpenStack.

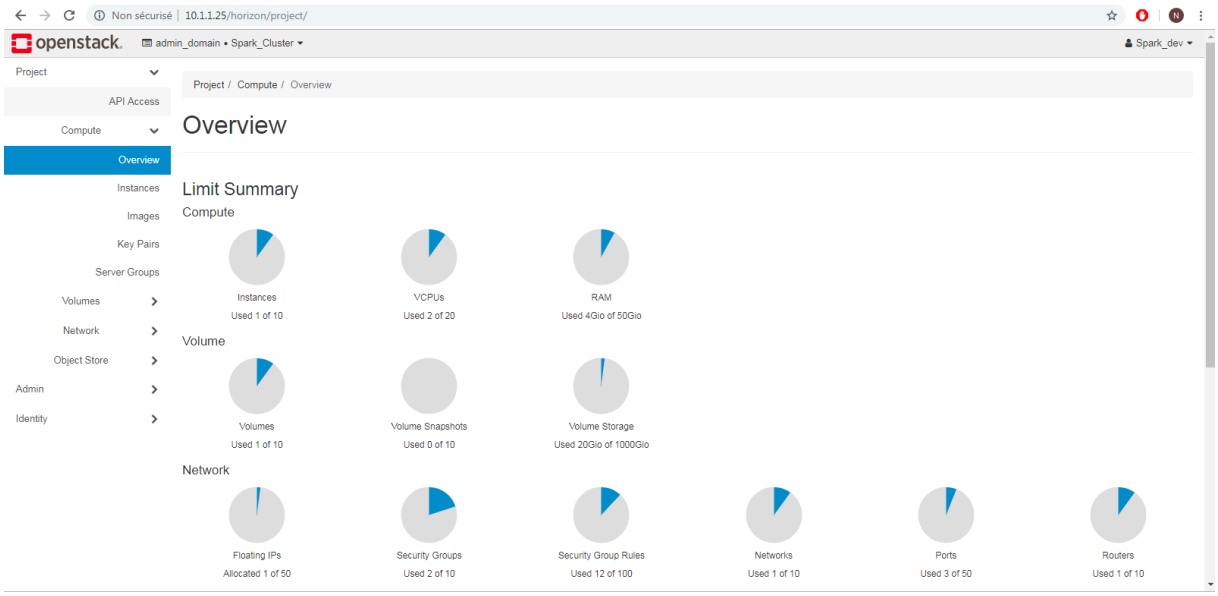


Figure 3.9 : Vue générale d'une session sur Horizon

3.5.8 Service d'orchestration : Heat

Heat est un service d'orchestration, il peut être utilisé pour créer des machines à la demande à partir de modèles, il permet à une application de se développer horizontalement sans avoir besoin de commandes directes des administrateurs.

Pour aider les administrateurs qui doivent gérer plusieurs infrastructures sur OpenStack et Amazon, ou qui migrent de l'infrastructure d'Amazon vers OpenStack, Heat supporte la syntaxe de template Amazon CloudFormation.

Le Heat peut être comparée à l'Amazon CloudFormation d'Amazon.

La figure suivante montre une architecture de fonctionnement de Heat :

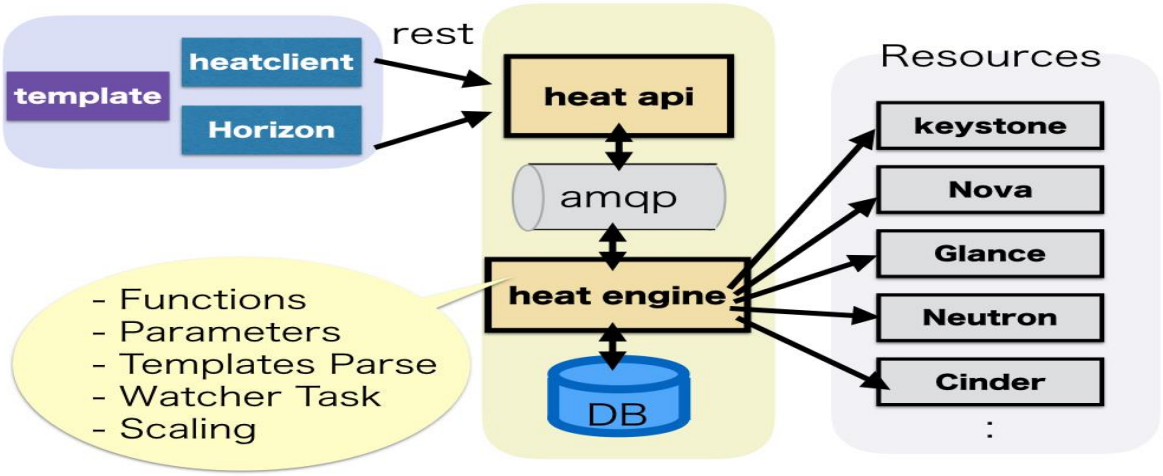


Figure 3.10 : Architecture Heat

3.5.9 Service de Telemetry : ceilometer

Ceilometer est un outil qui fournit des données sur l'utilisation des ressources par l'utilisateur, afin de pouvoir facturer les personnes en fonction des ressources réellement utilisées.

Ceilometer fournit un seul service qui centralise chaque compteur de services, ce qui permet d'exporter les données d'utilisation nécessaires au calcul de la facturation client. Toutes les données disponibles dans Ceilometer sont traçables et l'ensemble du processus peut être audité. Les données du Ceilometer peuvent également aider les entreprises qui utilisent OpenStack dans leur cloud privé à comprendre quels processus et quelles Business Units utilisent le plus de ressources.

La figure suivante explique l'architecture générale du service Ceilometer

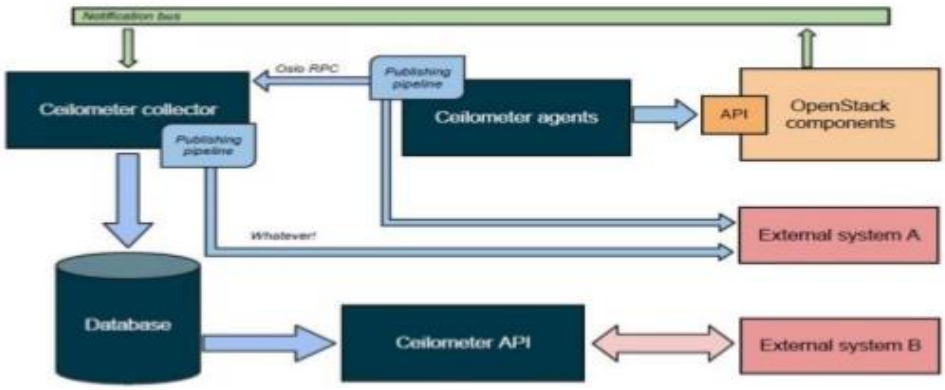


Figure 3.11 : Architecture Ceilometer

L'ensemble de tous ces composants forme une architecture de base d'Openstack, ces composants communiquent entre eux à travers un service de messagerie (comme nous l'avons déja mentionnée). Pour ce projet, le RabbitMQ est responsable de la communication entre toutes les services .

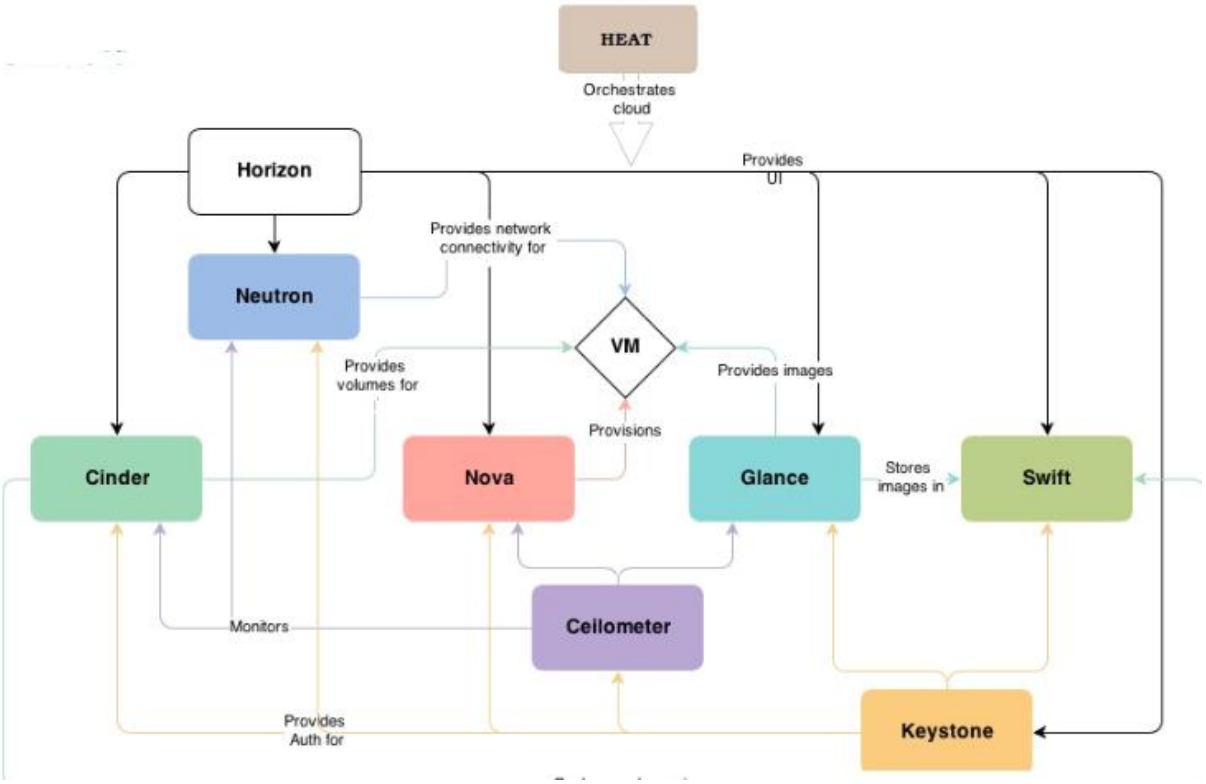


Figure 3.12 : Conclusion globale sur le fonctionnement des services OpenStack

3.6 Openstack, plate-forme dans un cloud

OpenStack est traditionnellement concentré sur la fourniture de capacités d'Infrastructure as a Service dans le style d'Amazon Web Services, de nouveaux projets ont été lancés récemment, qui commencent à fournir des capacités qui pourraient être davantage associées à Platform as a Service.Par conséquent, OpenStack a fourni plusieurs modules nécessaires à la construction d'une plate-forme de cloud automatisée comme Trove (gestion des base de données),Sahara (traitement de données) et magnum (orchestrateur des conteneurs) [24].

L'aspect le plus important d'OpenStack en ce qui concerne son utilisation en tant que plate-forme est le modèle du locataire (Tenant). Les services d'authentification et d'autorisation qui fournissent ce service sont implémentés dans le service Identité, Keystone. C'est ce qui le différencie du centre de données traditionnel (Data center) et les plates-formes de cloud Computing.

3.7 Conclusion

Pour conclure, les points principaux abordés dans ce chapitre sont :

- OpenStack est un projet open source, ceci est l'un de ses plus grands avantages
- OpenStack est une pile de composants software (microservices) dédié à fournir un service Cloud (principalement de l'IaaS) sur un ensemble de serveurs physiques.
- Ses composants sont interconnectés d'une manière à assurer le bon fonctionnement de l'ensemble, chacun d'eux fournis un service spécialisé aux clients (un service est aussi client d'un autre service)
- De nombreux services peuvent être intégrés à l'OpenStack de base pour fournir du PaaS
- OpenStack est généralement utilisé pour créer du cloud privé.

Chapitre 4 :

Implémentation de la solution

4 Implémentation de la solution

4.1 Introduction

Dans le monde du Big Data et du cloud computing, on trouve pleins de solutions et de technologies pouvant remplir certaines tâches. Pour faire face à un cahier de charge pour une application Big Data dans le contexte du Cloud Computing, il faut construire l'architecture fonctionnelle de la solution en partant du cœur vers la périphérie. Ensuite, il faut remplir chacune des fonctions par les technologies correspondantes qui satisfont un ensemble de caractéristiques fondamentales. Dans la gestion d'un projet informatique, le bon fonctionnement du système d'information, ainsi que sa performance sont les deux grands éléments qu'il faut tenir en compte, cela doit se faire à travers les caractéristiques des différentes parties de la solution. Ce projet utilise plusieurs technologies pour atteindre ses objectifs, de la génération et la récolte des données, vers les systèmes de stockage, de gestion de données et de fichiers, les systèmes de gestion de ressources et orchestration d'autres systèmes d'information, les systèmes de gestions de calculs et développement IA, les interfaces d'exploitation et de monitoring, comme elle le montre la figure suivante :

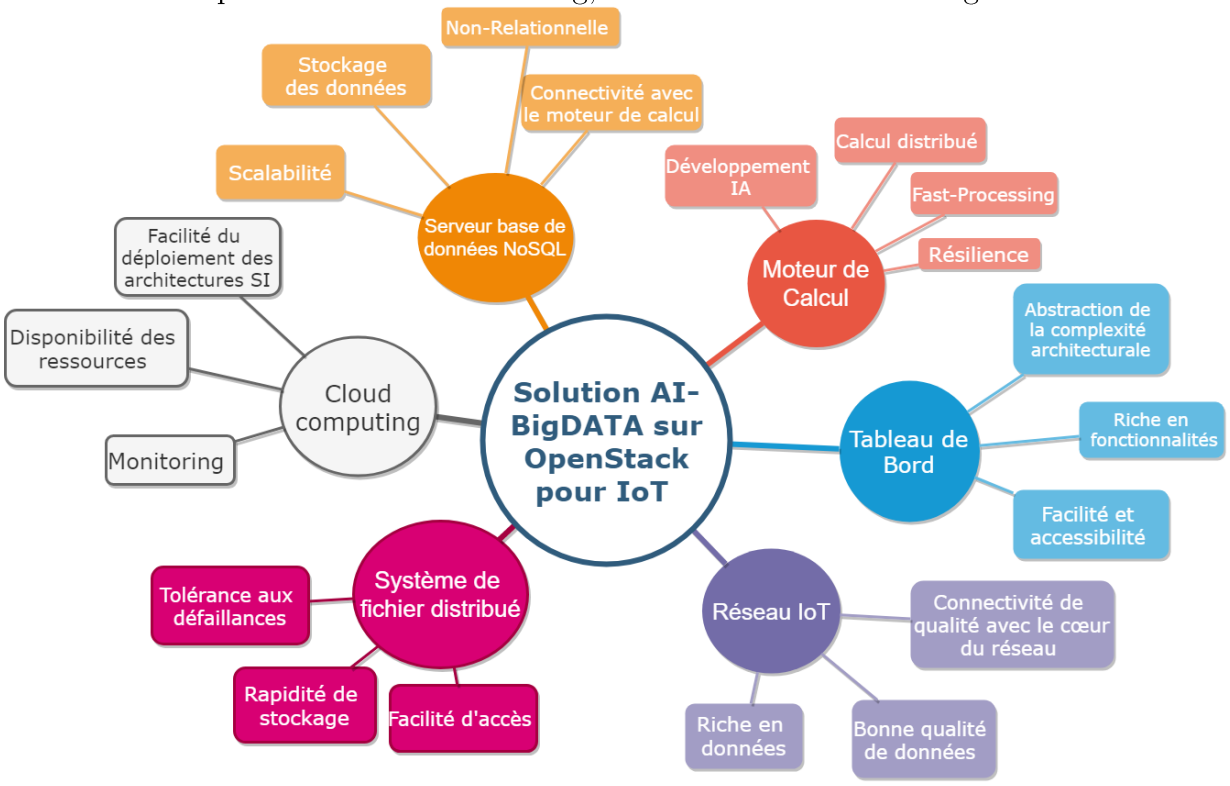


Figure 4.1 : Architecture globale de la solution développée dans le projet

Chacune représente une des briques de la solution. leurs caractéristiques et outils d'implémentation vont être traités dans ce chapitre, en expliquant leurs architectures, leurs caractéristiques, et comment sont-ils déployés dans ce projet.

4.2 Déploiement d'Openstack

Openstack repose sur plusieurs services, ceux-ci sont fournis par plusieurs composants (microservices), comme définit avant, pour installer Openstack dans son intégralité (avec les services nécessaires pour fournir un PaaS), il faut les installer un par un, avec les configurations nécessaires pour chaque composant, ainsi que sa connectivité avec les autres services.

Cette méthode prend beaucoup de temps et de concentration, et risque de ne pas pouvoir fonctionner intégralement, dû aux différentes erreurs de configurations et d'installation qu'un opérateur du déploiement cloud peut commettre, d'où vient la nécessité d'un outil qui fait toute cette partie d'une façon automatique et facile à gérer et à réappliquer.

Juju est un outil destiné à faire le travail de déploiement d'infrastructures Cloud, y compris Openstack, d'une manière simple, rapide et efficace. Il existe d'autres solutions de déploiement automatique et gestion de logiciels, on y cite les plus fameux : Terraform, Ansible, Puppet, Chef et SaltStack.

Figure 4.2 : Les outils du déploiement automatique des infrastructures logiciels les plus connus

4.2.1 Juju

4.2.1.1 Définition

Juju est un outil de modélisation d'applications open source. Avec lui, on peut déployer configurer, mettre à l'échelle et exploiter des logiciel sur des clouds publics et privés.

Généralement, Juju est utilisé à travers JAAS (Juju as a service). JAAS est une application web qu'a été construit pour l'utilisation du Juju. L'utilisation de JAAS n'exclut

pas l'utilisation du CLI pour gérer Juju. Tout ce qu'on peut faire dans JAAS est transparent pour le client Juju.[25]

4.2.1.2 Concepts et fonctionnement

Pour bien voire le fonctionnement général du JUJU, nous allons d'abord initialement définir quelques concepts propres à JUJU.

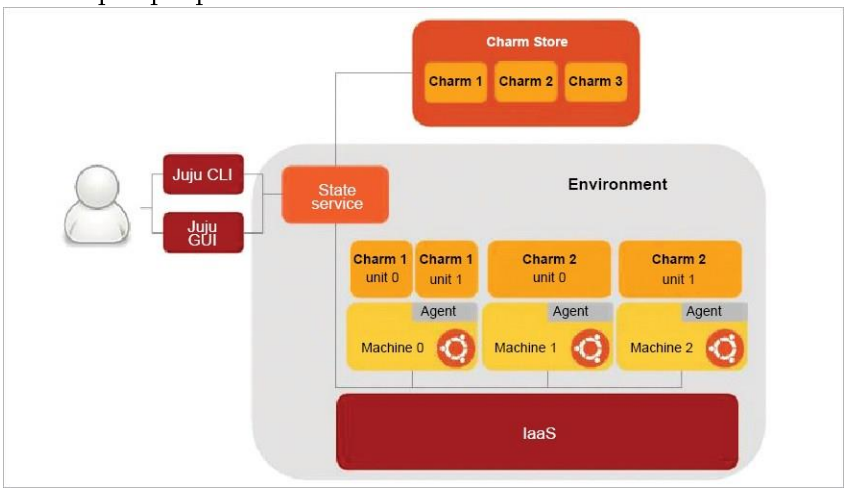


Figure 4.3 : Architecture Juju

4.2.1.3 Le contrôleur

Le contrôleur Juju est l'instance initiale crée pour que Juju puisse accéder au Cloud. Il est créé en demandant au client Juju de contacter l'API du cloud. Le contrôleur est un nœud de gestion central pour le Cloud choisi, prenant en charge toutes les opérations demandées par le client Juju.

4.2.1.4 Charme

Un charme Juju contient toutes les instructions nécessaires au déploiement et à la configuration des unités d'application. Les charmes sont accessibles au public dans le CharmStore en ligne et représentent les connaissances distillées des experts. Les charms facilitent le déploiement fiable et répété d'applications, puis la montée en charge (et la réduction) comme nous le souhaitons avec un minimum d'effort.

4.2.1.5 Les paquets (Bundle)

Un bundle de Juju est une collection de charmes qui ont été soigneusement combinées et configurées afin d'automatiser une solution à plusieurs charmes. Les opérations sont transparentes pour Juju et le déploiement peut donc continuer à être géré par Juju comme si tout était effectué manuellement.

4.2.1.6 Machine

Une machine Juju est le terme utilisé pour décrire une instance de Cloud qui a été demandée par Juju. Les machines abritent généralement une seule unité d'une application déployée, mais ce n'est pas toujours le cas. Si l'utilisateur le demande, une machine peut

héberger plusieurs unités, ou même aucune unité du tout : une machine peut être créée indépendamment des applications, même si c'est généralement avec l'intention de lancer une application sur elle !

4.2.1.7 Modèle

Un modèle est associé à un contrôleur unique et représente l'espace dans lequel les unités d'application sont déployées. Un contrôleur peut avoir un nombre indéfini de modèles et chaque modèle peut avoir un nombre indéfini de machines (et donc d'applications). Les modèles eux-mêmes peuvent être partagés entre les utilisateurs de Juju.

Pour Juju, un cloud est une ressource qui fournit des machines (instances), et éventuellement du stockage, afin de déployer des unités d'application sur elles qui sont décrite sur un charme on utilisant un Bundle qui décrit toutes les configurations voulues. Le but d'utiliser Juju est de déployer une architecture fonctionnelle d'Openstack sur des serveurs avec tous les composants nécessaires. Juju peut utiliser des environnements qui ne sont pas des Cloud en soi, mais que Juju peut néanmoins les traiter comme un Cloud. MAAS entre dans cette dernière catégorie.

4.2.2 MaaS

Metal as a Service, MAAS, nous permet de traiter les serveurs physiques comme des machines virtuelles dans le cloud. On peut Connecter, mettre en service (iLO cards) et déployer des machines physiques en un temps record à partir des images ISO selon nos choix (PXE boot), tout cela peut se faire seulement depuis le serveur MAAS, réaffecte dynamiquement les nœuds entre les services, maintienne-les à jour et Plutôt que d'avoir à gérer chaque serveur individuellement, MAAS transforme votre Bare-Metal en une ressource élastique de type Cloud[26].

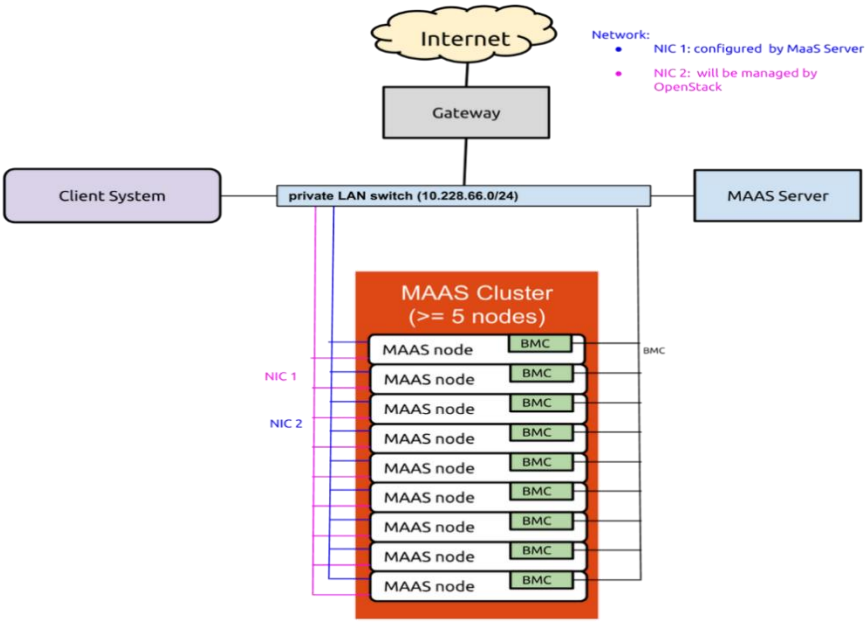


Figure 4.4 : Architecture d'un cluster MaaS

En conjonction avec Juju, MAAS nous permet de tirer le meilleur parti du matériel physique et de déployer dynamiquement et facilement tous les composants d'Openstack, le MAAS fournit à Juju les nœuds dont il a besoin pour alimenter ces services. Avec cette démarche, on a pu déployer un PaaS basé OpenStack sur nos quatre serveurs.

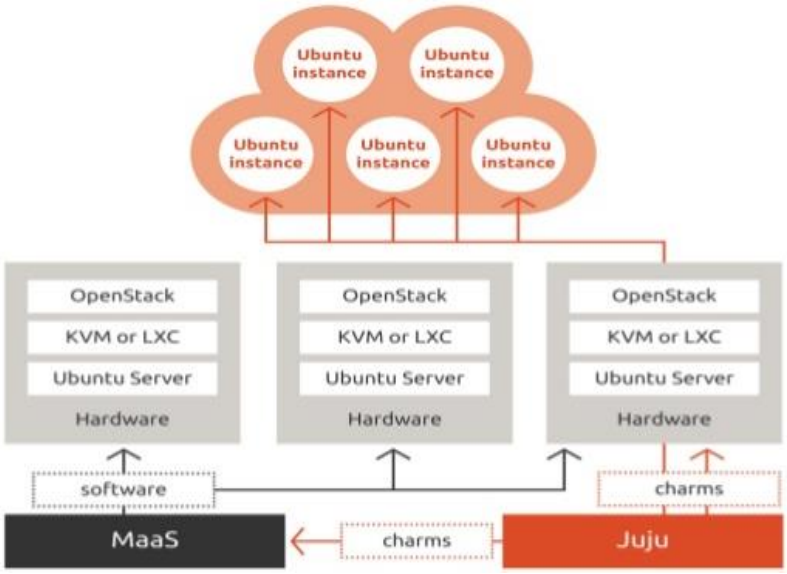


Figure 4.5 : Déploiement OpenStack avec Juju à travers MaaS

Pour le déploiement de la pile OpenStack, il est recommandé d'utiliser le bundle « openstack telemetry » pour le déploiement non seulement de base d'openstack, mais

aussi d'autres composants, comme par exemple Ceilometer pour le service de facturation, Gnocchi pour le stockage des données en séries temporelles. Voici l'architecture finale du cloud déployé sur 4 machines physique selon un bundle qui la décrive, avec JUJU en passant par MaaS :

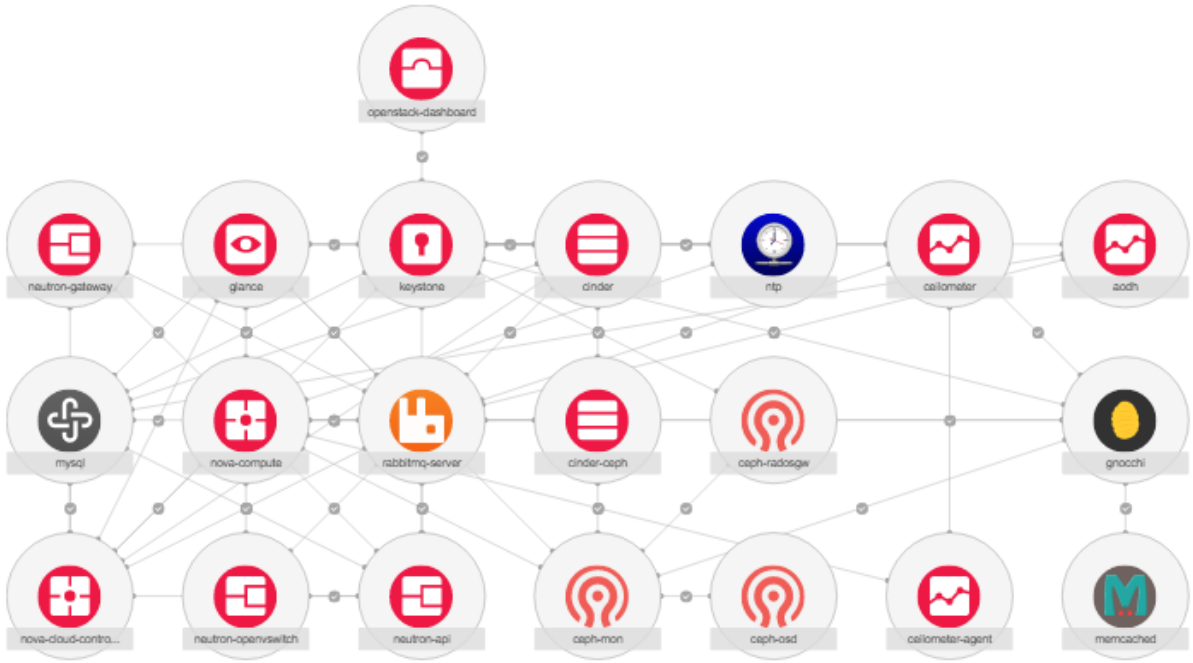


Figure 4.6 : L'architecture OpenStack déployé dans le projet

4.3 Base de données NoSQL orienté Big Data

Les problèmes fondamentaux que posent le Big Data (volume, vitesse et variété) rendent nécessaire l'utilisation de nouveaux moyens ou bien technologies pour faire persister ces grands volumes de données ainsi que pour les interroger et les traiter. Les bases de données NoSQL, c'est-à-dire « Not only SQL », sont une solution presque parfaite à ce problème

4.3.1 Définition

Le NoSQL, désignant « Not only SQL », représente une catégorie de bases de données apparue au cours de l'année 2009 qui se différencie du modèle relationnel que l'on retrouve dans les systèmes de bases de données connues (MS SQL Server, Oracle ou PostgreSQL). Cette catégorie de produits fait le compromis d'abandonner certaines fonctionnalités classiques des SGBD relationnels au profit de la simplicité, la performance et une

forte scalabilité. La scalabilité est la capacité d'un système à répondre à une demande toujours croissante de la part des utilisateurs en termes de requêtes.

Aujourd'hui, le terme NoSQL englobe tous les SGBD qui ne suivent pas la tendance de type relationnel. Cela signifie que le NoSQL n'est pas un seul produit ou même une technologie unique. En effet, il représente de nombreux produits ainsi que plusieurs concepts de stockage et de manipulation de données

4.3.2 Avantages des bases de données NoSQL dans la Big Data

4.3.2.1 Scalabilité élastique

Pendant des années, les administrateurs de bases de données se sont fiés à la mise à l'échelle - l'achat de serveurs plus gros à mesure que la charge de la base de données augmente - plutôt que la mise à l'échelle - la distribution de zone de stockage de données sur plusieurs hôtes à mesure que la charge augmente. Cependant, à mesure que les taux de transaction et les exigences de disponibilité augmentent, et que les bases de données évoluent dans le cloud ou dans des environnements virtualisés, les avantages économiques de la mise à l'échelle du matériel de base deviennent irrésistibles.

Les SGBDR peuvent ne pas s'étendre facilement sur les clusters de base de données, mais la nouvelle génération de bases de données NoSQL est conçue pour s'étendre de manière transparente afin de tirer parti des nouveaux nœuds, et elles sont généralement conçues en tenant compte du matériel de base à faible coût.

4.3.2.2 Big Data

Tout comme les taux de ressources des données sont passés de la reconnaissance au cours de la dernière décennie, les volumes de données stockées ont également augmenté massivement. O'Reilly à habilement appelé cela la "révolution industrielle des données". La capacité du SGBDR s'est accrue pour faire face à ces augmentations, mais comme pour les taux de transaction, les contraintes liées aux volumes de données qui peuvent pratiquement être gérés par un seul SGBDR deviennent intolérables pour certaines entreprises. Aujourd'hui, les volumes de " grandes données " qui peuvent être traitées par les systèmes NoSQL, tels que Hadoop, dépassent ceux qui peuvent être traités par les plus grands SGBDR.

Dans notre implémentation, nous utilisons MongoDB comme un SGBD pour notre base No-SQL, ce choix sera justifié par la suite à travers les multiples avantages caractérisant MongoDB.

4.3.3 MongoDB

MongoDB est un système de gestion de base de données orientée documents « documents store », répartissable sur un nombre quelconque d'ordinateurs et ne nécessitant pas de schéma prédéfini des données (comme dans le cas des SGBD relationnelle). Il est écrit en C++. Le serveur et les outils sont distribués sous licence SSPL, les pilotes sous licence Apache et la documentation sous licence Creative Commons. Il fait partie de la mouvance NoSQL, avec d'autres serveurs base de données comme CassandraDB, Redis, HBase, Oracle NoSQL, Neo4j, ...

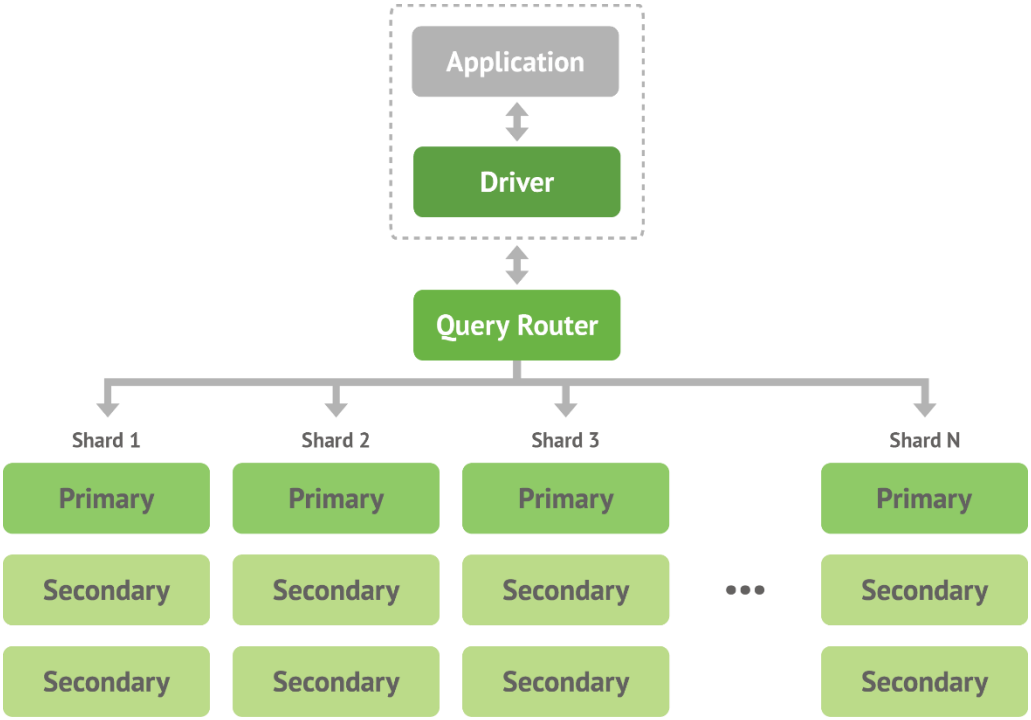


Figure 4.7 : L'architecture MongoDB

La nouveauté du MongoDB par rapport à ses antérieures relationnelles, c'est qu'il n'organise pas les données dans des tableaux à colonnes et les rangées. Au lieu de cela, les données sont stockées dans des "documents", une version plus élastique des lignes de donnée classique du relationnelle. Bien que MongoDB ne soit pas relationnel, il implémente de nombreuses fonctionnalités de bases de données relationnelles, telles que le tri, l'indexation, et les requêtes de portée. Les documents MongoDB sont sérialisés naturellement comme objets, Javascript Object Notation (JSON), et sont en fait stockés en interne en utilisant un encodage binaire de JSON appelé BSON. Voici l'exemple d'un document en MongoDB avec un format JSON :

```
{
        "_id": ObjectId("4efa8d2b7d284dad101e4bc7")
        "name": { "khanfri" , "henka" } ,
        "project" : "AI_Platform" ,
        "year": 2019
}
```

Figure 4.8 : Exemple d'un format JSON

L'ensemble des documents forment une collection (l'équivalent d'un tableau pour SQL), et un ensemble de collections forment une base de données.

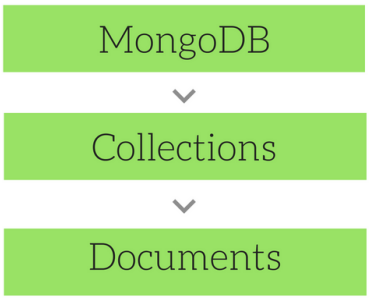


Figure 4.9 : La topologie du stockage des données dans MongoDB

MongoDB est un entrepôt de données puissant, flexible et évolutif. Il combine la capacité de mise à l'échelle avec plusieurs fonctions les plus utiles des bases de données relationnelles, telles que les index secondaires, les requêtes de plage et le tri. MongoDB est aussi incroyablement fonctionnel : il possède des tonnes de fonctionnalités utiles telles que la prise en charge intégrée de l'agrégation de style MapReduce et des index géospatiaux.

Voici ses principales caractéristiques :

- Installation facile .
- Base de données sans schéma
- Base de données évolutive horizontalement
- Fournit une performance élevée
- Auto-sharding
- Exécution sur plusieurs serveurs
- Prise en charge de la réplication maître-esclave
- Les données sont stockées sous forme de documents de style JSON
- Indexer n'importe quel champ d'un document
- Il a une configuration d'équilibrage de charge automatique en raison des données placées dans des tessons.

- Prise en charge des recherches d'expressions régulières
- Facile à administrer en cas d'échec

4.4 Système de Fichier distribué

4.4.1 Système de fichier

Un système de fichiers est un processus qui gère comment et où les données sont stockées sur un disque de stockage. Généralement un disque dur (HDD), les données sont stockées, accédées et gérées. C'est un composant de disque logique qui gère les opérations internes d'un disque en relation avec un ordinateur et qui est abstrait pour un utilisateur humain.

Un système de fichier peut être créé sous deux forme, local (comme dans le cas du système de fichier d'une seul machine portant un OS) ou distribué, mais la nécessité des systèmes de stockage de données distribué sur plus de ressources est devenu trivial dans le monde de Big Data, Cette exigence nous a amené à ce type de système de fichier qui est le système de fichier « distribué »[15].

4.4.1.1 Système de fichier distribué (DFS)

Un système de fichiers distribué (DFS) est un système de fichiers dont les données sont stockées sur plusieurs nœuds matériels. Les données sont accessibles et traitées comme si elles étaient stockées sur la machine client locale. Le DSF facilite le partage de l'information et des fichiers entre les utilisateurs d'un réseau d'une manière contrôlée et autorisée. Le serveur permet aux utilisateurs clients de partager des fichiers et de stocker des données tout comme ils stockent les informations localement. Cependant, les serveurs ont un contrôle total sur les données et donnent le contrôle d'accès aux clients.

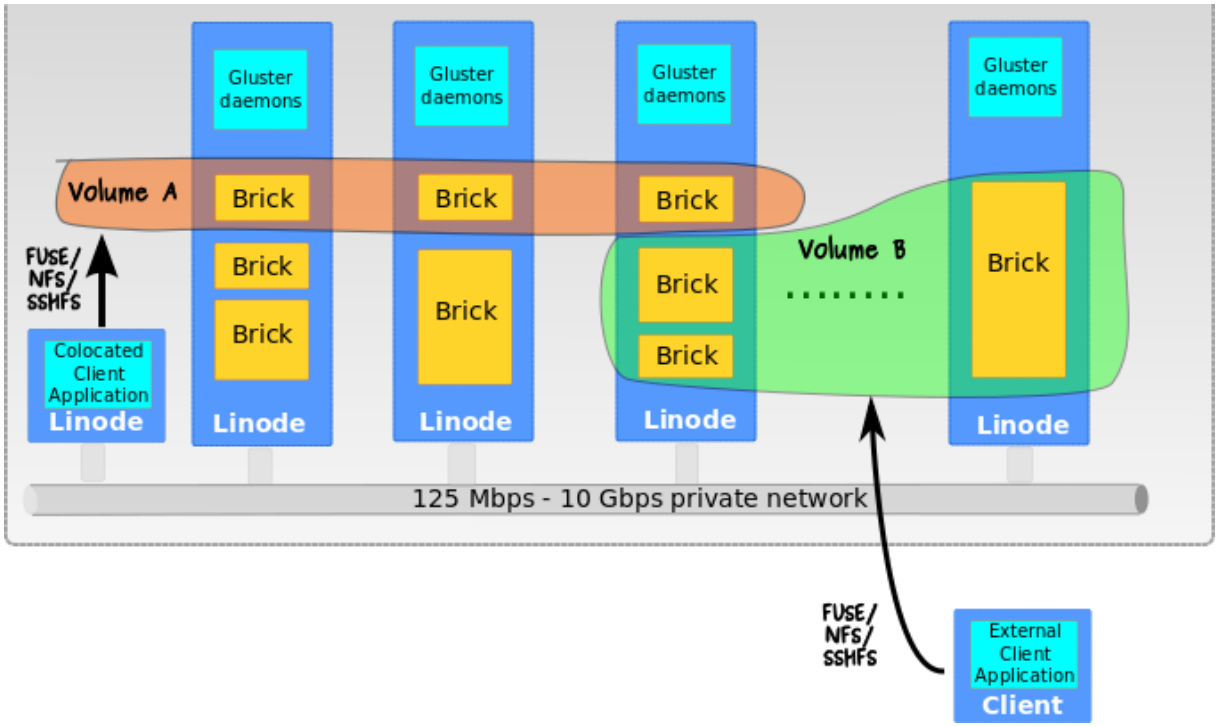


Figure 4.10 : Un exemple d'un système de fichiers distribué Linode

Plusieurs projets ont été développé dans ce domaine, le plus fameux dans le contexte de Spark est celui crée par Hadoop, HDFS (Hadoop Distributed File System).

4.4.1.2 HDFS

Le Hadoop Distributed File System (HDFS) est le principal système de stockage de données utilisé par les applications Hadoop. Il utilise une architecture NameNode et DataNode pour implémenter un système de fichiers distribué qui fournit un accès haut performance aux données à travers des clusters Hadoop hautement évolutifs.

Le HDFS est un élément clé des nombreuses technologies de l'écosystème Hadoop, car il fournit un moyen fiable de gérer des pools de données massives et de prendre en charge les applications connexes d'analyse de grandes données.

4.4.1.3 Le fonctionnement de HDFS

HDFS prend en charge le transfert rapide des données entre les nœuds de calcul. Au départ, il était étroitement associé à MapReduce, un cadre programmatique pour le traitement des données.

Lorsque le HDFS recueille des données, il les décompose en blocs séparés et les distribue à différents nœuds d'un cluster, ce qui permet un traitement parallèle très efficace.

De plus, le système de fichiers distribué Hadoop est spécialement conçu pour être hautement tolérant aux pannes. Le système de fichiers réplique, ou copie, chaque élément de données plusieurs fois et distribue les copies à des nœuds individuels, plaçant au moins une copie sur un rack serveur différent des autres. Par conséquent, les données sur les

nœuds qui plantent peuvent être trouvées ailleurs dans un cluster. Ceci garantit que le traitement peut continuer pendant que les données sont récupérées.

HDFS utilise aussi une architecture maître/esclave (comme le cas du Spark). Dans son incarnation initiale, chaque cluster Hadoop se composait d'un seul NameNode qui gérait les opérations du système de fichiers et prenait en charge les DataNodes qui géraient le stockage des données sur des nœuds de calcul individuels. Les éléments HDFS se combinent pour prendre en charge des applications avec de grands ensembles de données.la figure suivante montre l'architecture globale de HDFS.

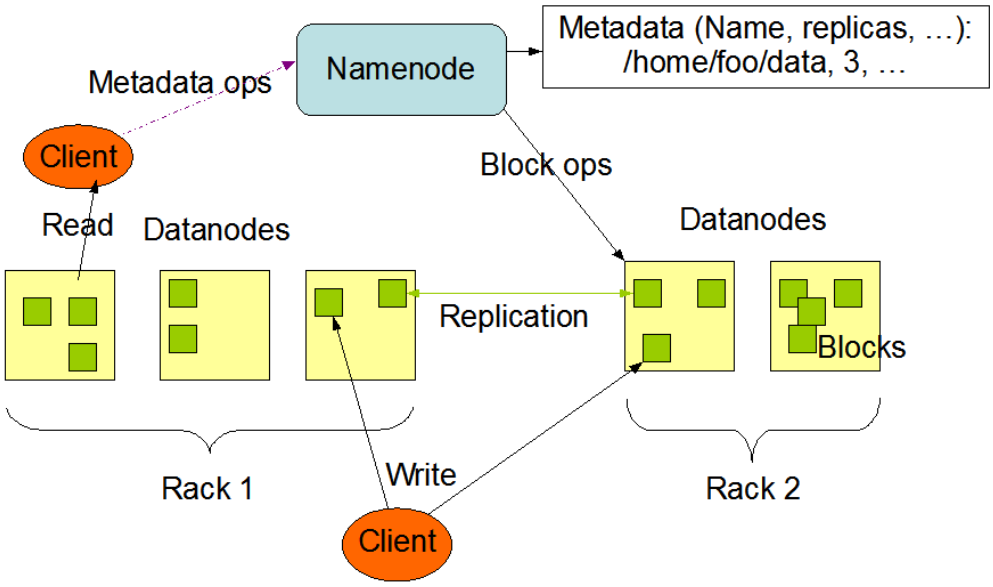


Figure 4.11 : L'architecture d'un système de fichiers Hadoop HDFS

4.4.1.4 L'utilité de HDFS

Le système de fichiers distribués Hadoop est largement utilisé par plusieurs entreprises du High-Tech comme Facebook, Twitter et Yahoo. Puisque ils sont retrouvé à jongler avec une variété d'applications auxquelles un nombre croissant d'utilisateurs accèdent et qui crée de plus en plus de données. Donc ils utilisent HDFS pour étayer l'analyse de données importantes afin de répondre à ces mêmes exigences.

Étant donné que le HDFS est généralement déployé dans le cadre d'implémentations à très grande échelle, la prise en charge du matériel de base à faible coût est particulièrement utile. De tels systèmes, qui exécutent des recherches sur le Web et des applications connexes, par exemple, peuvent atteindre des centaines de pétaoctets et des milliers de nœuds. Ils doivent être particulièrement résistants, car les pannes de serveurs sont fréquentes à une telle échelle.

L'HDFS fournit une interface web sur le port 50070, elle fournit plusieurs informations sur l'état du Namenode et ses logs, ainsi que les Datanodes, les rapports sur les défaillances :

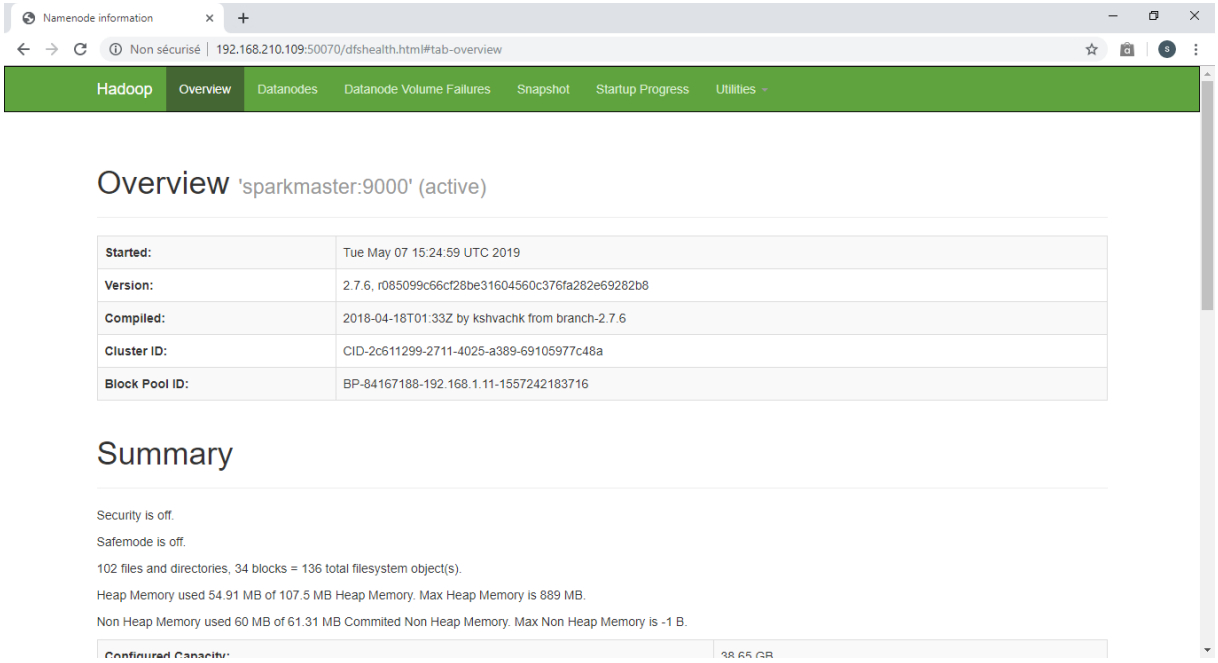


Figure 4.12 : Vue générale de l'HDFS

HDFS-WebUI donne aussi des statistiques sur l'utilisation du cluster HDFS :

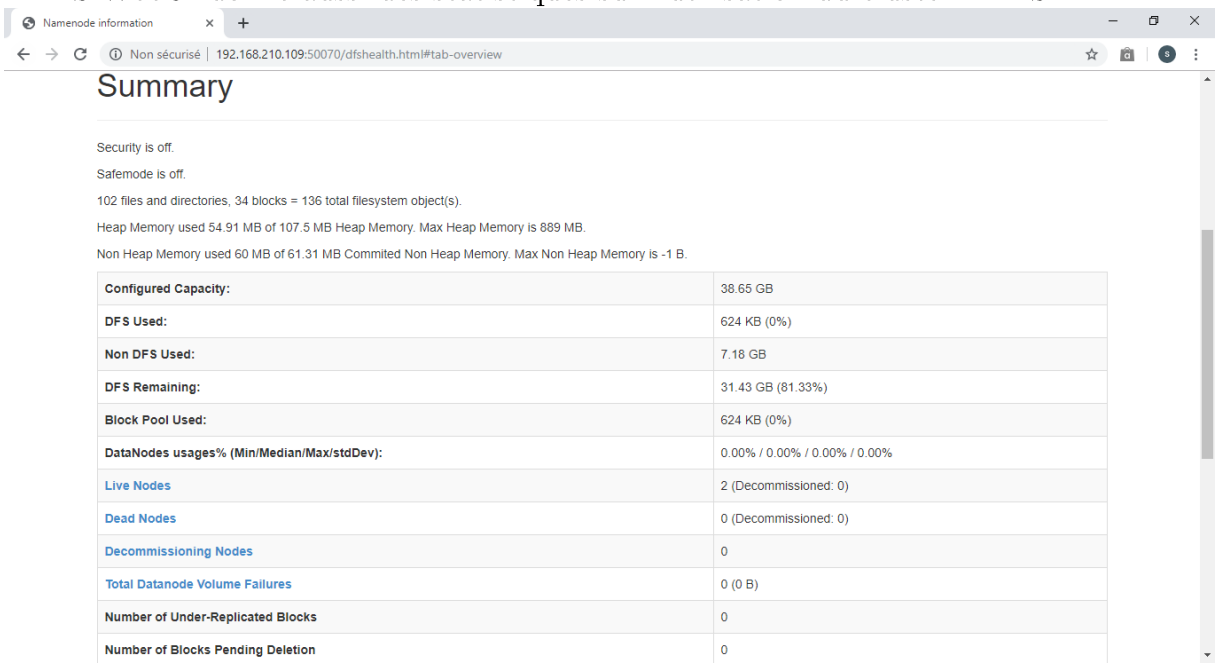


Figure 4.13 : Interface de monitoring du cluster HDFS

Aussi, la possibilité d'explorer notre système de fichiers distribué avec une interface graphique et abstraction totale de la distributivité du système :

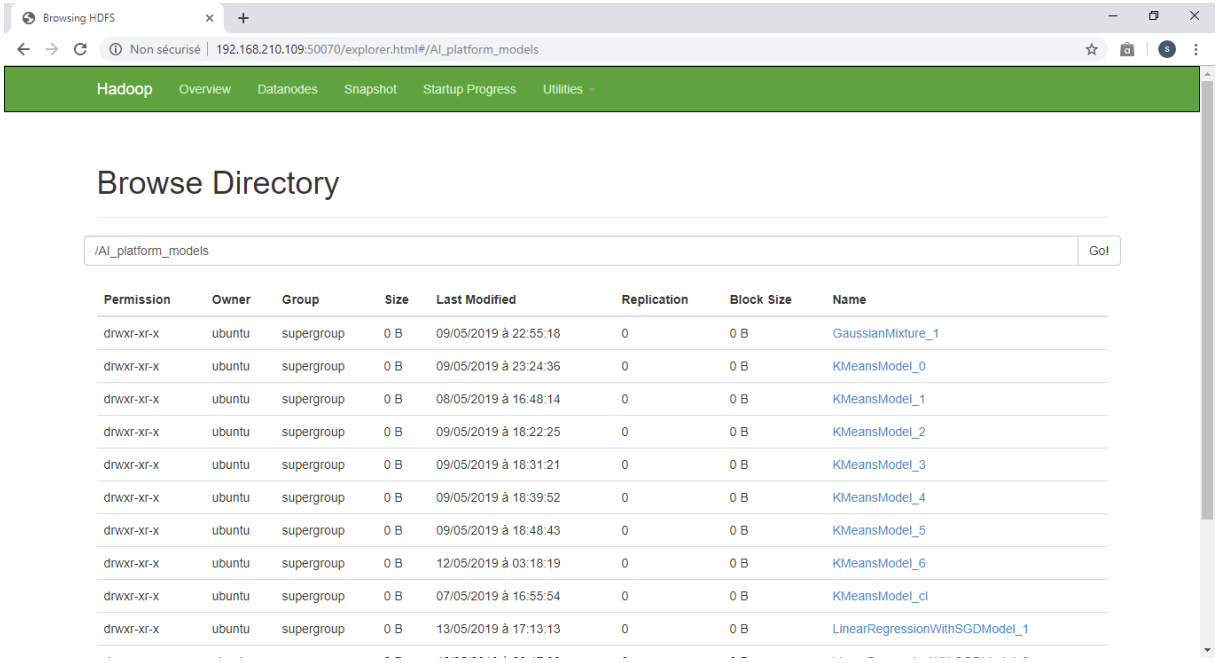


Figure 4.14 : Explorateur HDFS à travers la WebUI

4.5 Tableau de bord

La solution d'une plateforme AI pour l'analyse en Big Data a besoin d'être projetée sur une interface graphique, user-friendly et riche en fonctionnalités. Ceci pourra améliore énormément l'expérience des utilisateurs de la plateforme, et facilite l'interaction, aussi l'accès, vers notre base de données avec les données exploité et tous résultats fournis par nos algorithmes et modèles élaboré et stocker au sein de la plateforme. Le tableau de bord va donc principalement interagir avec le serveur mongod de notre base de données MongoDB.

AdminMongo est une application web écrite en JavaScript, CSS et HTML5 s'exécutant sur Node.js. C'est un projet open source publié sur GitHub et fournit des services d'interactivité avec des entités (base de données, collections et documents) de MongoDB à travers son serveur démon mongod, lancer des queries pour accéder à une partie voulu de la base de donnée, faire de la recherche selon les champs, faire du filtrage pour l'affichage, ajouter, supprimer ou modifier un document d'une collection de la base, avoir des statistiques sur la taille de données. On peut aussi créer et gérer des utilisateurs et spécifier leurs privilèges en termes d'accès et de modification dans une base donnée.

Cet outil de gestion et d'administration de MongoDB, permet de faire pleins d'ajustement et de reconfiguration du service de gestion selon le cas d'exploitation (modifier le port ou l'adresse du service web, ajouter un mot de passe de sécurité, la langue utilisé dans l'interface, activer/désactiver le monitoring, modifier la GUI...), aussi de faire du monitoring sur l'utilisation en mémoire du service en temps réel. L'accès aux différentes

bases de données et collections se fait par des connexions, un URL à configurer dans l'application adminMongo pour spécifier l'adresse et le port des nœuds portant le serveur MongoDB, la base de données et la collection souhaitées.

Voici une vue générale sur notre version modifié de l'interface :

1. On connecte après avoir lancé l'application, sur le port 1234 qu'on a configuré :

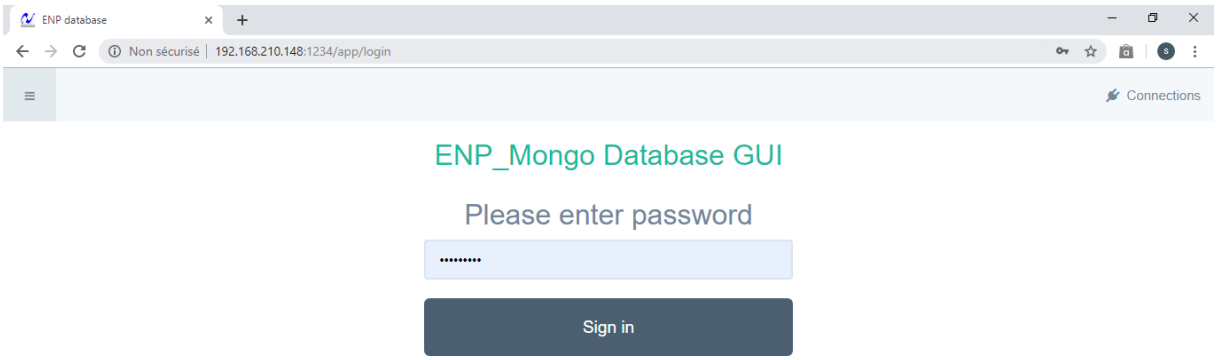


Figure 4.15 : Authentification d'accès au tableau de bord du projet

2. Après avoir introduit le mot de passe, on accède à une page qui vous propose les connections existantes, et vous donne la main d'en créer d'autres connections (port 27017 est le port par défaut utilisé par les services mongod et mongos du MongoDB) :

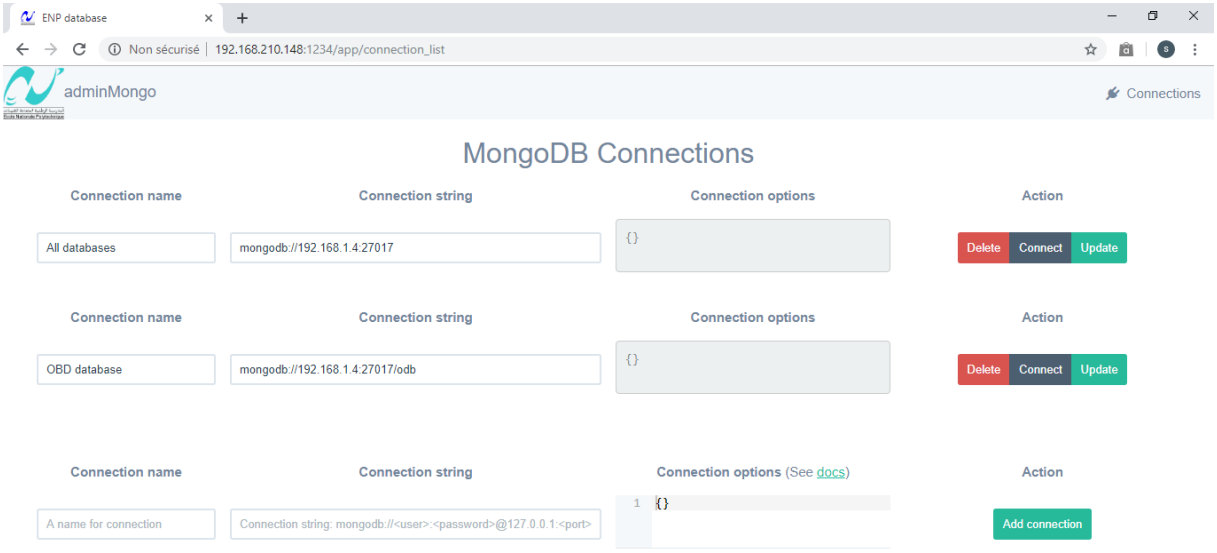


Figure 4.16 : Liste des connections vers la base de données

3. En connectant à la première ligne de connexion destiné à voir toutes les bases de données, une page se lancera et affiche des données concernant toutes les bases de données existant :

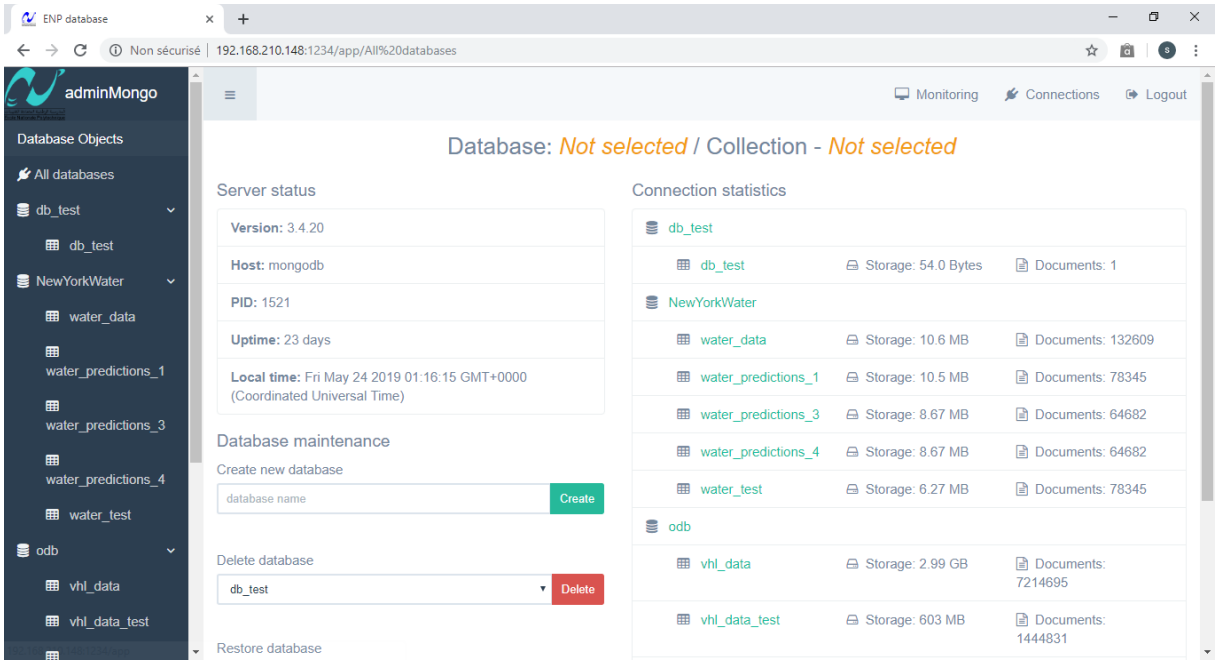


Figure 4.17 : Vue sur toutes les bases de données créées le long du projet

4. En cliquant sur une de ces collections (affiché dans la barre à gauche en deuxième ordre), on peut accéder aux documents dans la collection ciblé, on peut faire de la recherche et de la statistique en queries, supprimer des documents, ou même les modifier :

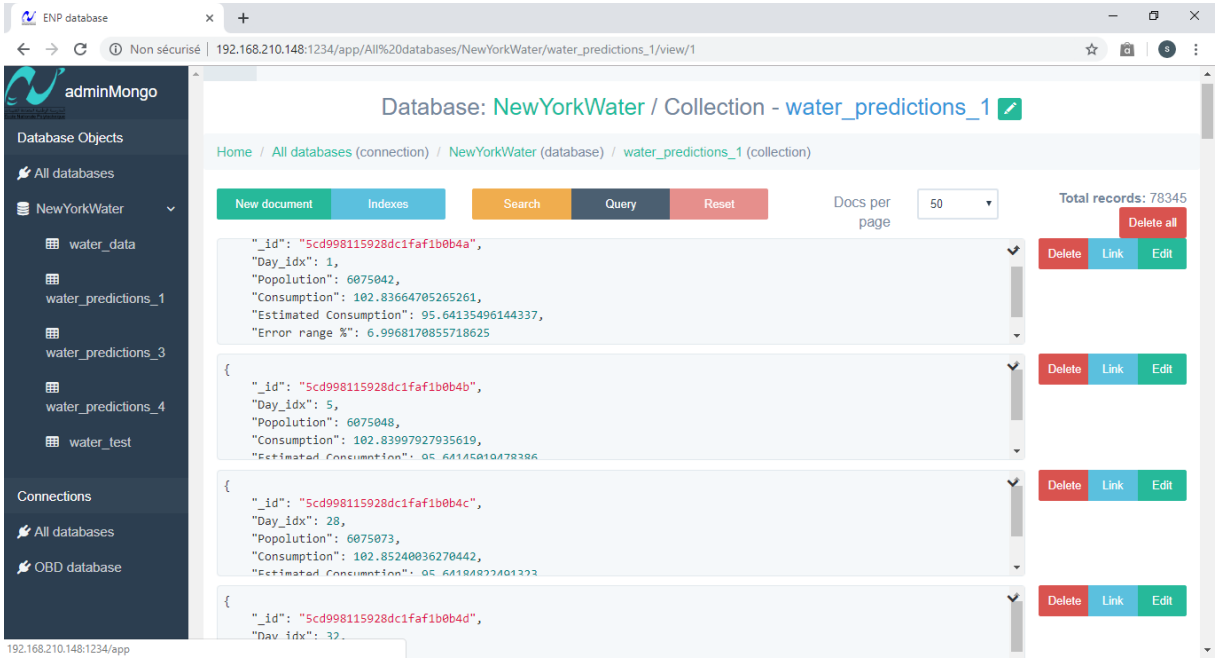
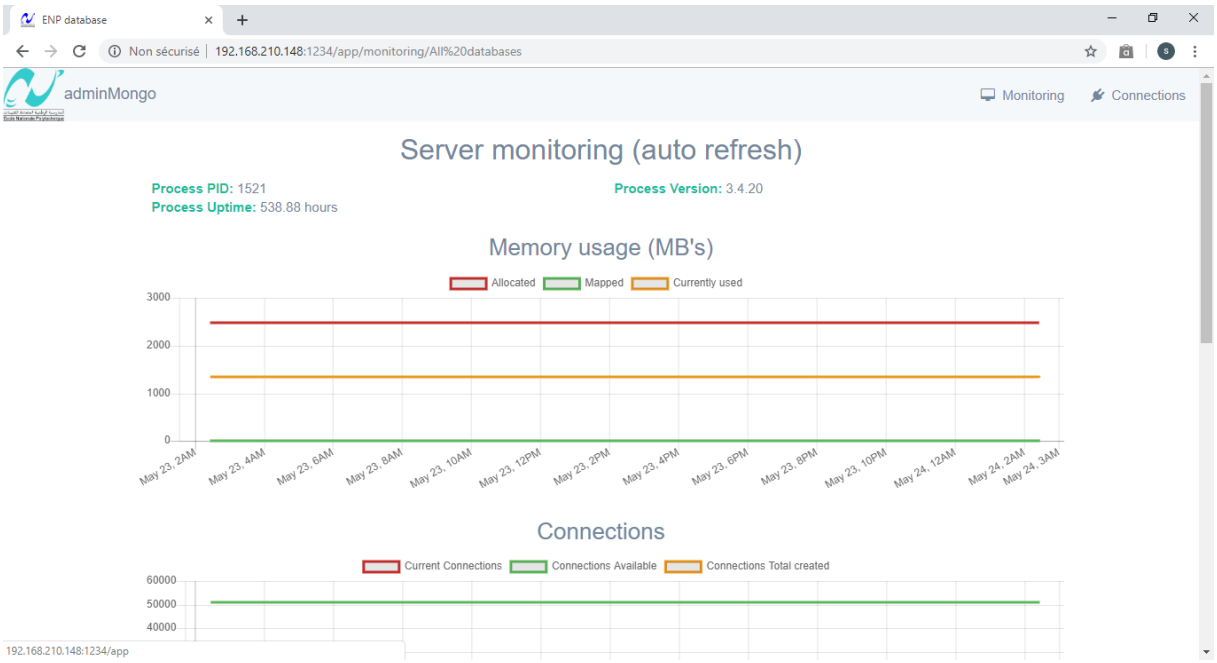


Figure 4.18 : Accès aux résultas obtenus lors d'une analyse prédictive sur la consommation de l'eau dans la ville de New York

5. En cliquant sur le bouton monitoring, on accède à la partie de gestion du service mongos et mongod, un affichage va porter des graphes sur les statistiques concernant la consommation en mémoire des services MongoDB, les connections disponible, le taux de comptage de documents..., le mode auto refresh assure l'affichage en temps réel des graphes :



4.6 Création d'un cluster du Spark sur Openstack

Afin de mettre tous ces outils en place, nous voulons créer une grappe (cluster) d'instances sur lequel nos calculs générés par nos applications Spark à travers son noyau vont être portés et exécutés. Ceci a comme but d'allouer plus de ressources physiques et logiques (cœurs CPU et espaces en mémoire) pour notre application, et de mettre en œuvre la notion de scalabilité pour notre architecture de calcul porté par Openstack. Il faut noter qu'une application spark porté par un seul nœud est faisable, mais pour un calcul intense comme l'algorithme Kmeans sur de grandes bases de données, on a constaté que le temps d'exécution est considérablement long. Le nombre immense de jobs crée par notre application spark avec leurs différentes complexités vont causer des saturations en mémoire pour ce nœud, ce qui fait que le processus va être suspendu par le système d'exploitation, et notre application se terminera sans générer son ouput. On dit qu'on est dans un contexte de big data, lorsque nos charges de données dépassera les capacités de traitement et de calcul d'une seule machine avec la manière qu'on cherche, d'où vient la motivation à s'intéresser au cluster computing dans ce qui suit.

Openstack propose plusieurs méthodes pour la création d'une grappe Spark, ou d'une grappe en générale, selon deux facteurs principaux :

- l'outil et l'architecture d'orchestration.
- le type des machines qui le forment (VM ou conteneur).

Dans ce suit, on va aborder les méthodes les plus utilisées et adaptées à OpenStack. Il faut tenir compte que ces méthodes sont souvent dépendantes l'une de l'autre, on tient compte de l'architecture globale et les services offerts par leurs serveurs API dans la classification.

4.6.1 Selon l'environnement d'exécution

4.6.1.1 Machine virtuelle

Une machine virtuelle (VM) est une émulation d'un système informatique. Les machines virtuelles sont basées sur des architectures informatiques et fournissent les fonctionnalités d'un ordinateur physique. Leur mise en œuvre peut nécessiter l'utilisation de matériel, de logiciels ou d'une combinaison des deux. Les machines virtuelles système (également appelées machines virtuelles intégralement virtualisées) remplacent une machine réelle. Ils fournissent les fonctionnalités nécessaires à l'exécution de systèmes d'exploitation complets. Un hyperviseur utilise une exécution native pour partager et gérer le matériel, ce qui permet de créer plusieurs environnements isolés les uns des autres, mais qui existent sur la même machine physique. Les hyperviseurs modernes utilisent la virtualisation assistée

par le matériel, du matériel spécifique à la virtualisation, provenant principalement des CPU hôtes. Les plus utilisés sont VMware, VirtualBox et Hyper-V.

Certaines machines virtuelles, comme QEMU, sont conçues pour émuler différentes architectures et permettre l'exécution d'applications logicielles et de systèmes d'exploitation écrits pour une autre CPU ou architecture. La virtualisation au niveau du système d'exploitation permet de partitionner les ressources d'un ordinateur via le noyau. Ce qui est important à retenir c'est que chaque machine virtuelle nécessite une copie complète du système d'exploitation, de l'application en cours d'exécution et de toute bibliothèque de support.

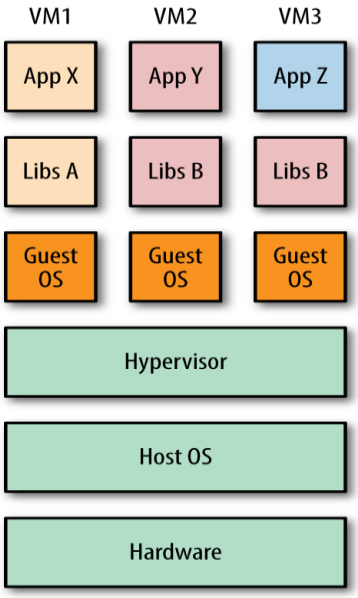


Figure 4.19 : Modèle en couche de la virtualisation des machines

4.6.1.2 Conteneurs et avantages sur les VMs

Les conteneurs sont une encapsulation d'une application avec ses dépendances. A première vue, ils semblent n'être qu'une forme légère de machines virtuelles (VM) - comme une VM, un conteneur contient une instance isolée d'un système d'exploitation (OS), que nous pouvons utiliser pour exécuter des applications.

Cependant, les conteneurs présentent plusieurs avantages qui permettent des cas d'utilisation difficiles, voire impossibles, avec les VMs traditionnelles :

- Les conteneurs partagent les ressources avec l'OS hôte, ce qui les rend d'un ordre de grandeur plus efficace. Les conteneurs peuvent être démarrés et arrêtés en une fraction de seconde. Les applications s'exécutant dans des conteneurs n'entraînent que peu ou pas de frais généraux par rapport aux applications s'exécutant nativement sur l'OS hôte.

- Les machines virtuelles consomment beaucoup de ressources système. Chaque VM exécute non seulement une copie complète d'un système d'exploitation, mais aussi une copie virtuelle de tout le matériel dont le système d'exploitation a besoin pour fonctionner. Cela s'additionne rapidement à beaucoup de RAM et de cycles CPU. En revanche, tout ce dont un conteneur a besoin, c'est d'un système d'exploitation, de programmes de support, de bibliothèques et de ressources système suffisants pour exécuter un programme spécifique.
- La portabilité des conteneurs a le potentiel d'éliminer toute une classe de bugs causés par des changements subtils dans l'environnement de fonctionnement - elle pourrait même mettre fin au refrain séculaire du développeur "mais ça marche sur ma machine" !
- La légèreté des conteneurs permet aux développeurs d'utiliser des douzaines de conteneurs en même temps, ce qui permet d'émuler un système distribué prêt à la production. Les ingénieurs d'exploitation peuvent utiliser beaucoup plus de conteneurs sur une seule machine hôte qu'avec des machines virtuelles seules.
- Les conteneurs présentent également des avantages pour les utilisateurs finaux et les développeurs en dehors du déploiement dans le cloud. Les utilisateurs peuvent télécharger et exécuter des applications complexes sans avoir à passer des heures sur des problèmes de configuration et d'installation ou à s'inquiéter des modifications à apporter à leur système. En retour, les développeurs de telles applications peuvent éviter de s'inquiéter des différences entre les environnements utilisateurs et de la disponibilité des dépendances.

Le plus important encore, les objectifs fondamentaux des machines virtuelles et des conteneurs sont différents : l'objectif d'une machine virtuelle est d'émuler complètement un environnement étranger, tandis que l'objectif d'un conteneur est de rendre les applications portables et autonomes.

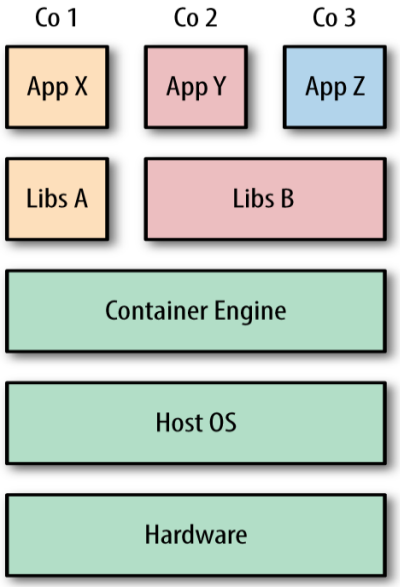


Figure 4.20 : Modèle de la conteneurisation

Les conteneurs changent fondamentalement la façon dont nous développons, distribuons et utilisons les logiciels. Les développeurs peuvent créer un logiciel localement, sachant qu'il fonctionnera de la même manière quel que soit l'environnement hôte - qu'il s'agisse d'un rack dans le département informatique, d'un ordinateur portable d'un utilisateur ou d'un cluster dans le cloud. Les ingénieurs d'exploitation peuvent se concentrer sur les réseaux, les ressources et le temps de disponibilité et passer moins de temps à configurer les environnements et à combattre les dépendances du système.

4.6.1.3 Modèle hybride

Les conteneurs offrent plusieurs avantages en plus de ceux offerts par la virtualisation. Cependant, Un certain nombre de faiblesses peuvent exister à l'intérieur même de la technologie des conteneurs. Celle-ci vont d'un noyau Linux non corrigé qui est vulnérable à l'élévation des privilèges résultant en un compromis de tous les conteneurs sur un seul nœud à des services exposés écoutant sur l'hôte local en tant que root qui sont vulnérables et peuvent être attaqués depuis un autre conteneur. On trouve sur internet, de nombreux tutoriels et scénarios possibles pouvant causer l'effondrement de toute la couche de sécurité d'un ensemble de conteneurs en exploitant une faille de sécurité applicative dans un seule conteneur de l'ensemble, on parle d'un problème d'isolation. Les machines virtuelles ont un degré d'isolation supplémentaire grâce à l'hyperviseur, c'est une technologie fiable et mature. Les conteneurs sont relativement nouveaux et de nombreuses organisations hésitent à faire entièrement confiance aux caractéristiques d'isolation des conteneurs avant d'avoir fait leurs preuves. Pour cette raison, il est courant de trouver des systèmes hybrides avec des conteneurs fonctionnant à l'intérieur des machines virtuelles afin de tirer parti

des deux technologies. La figure ci-dessous fait la comparaison entre les trois modèles machine :

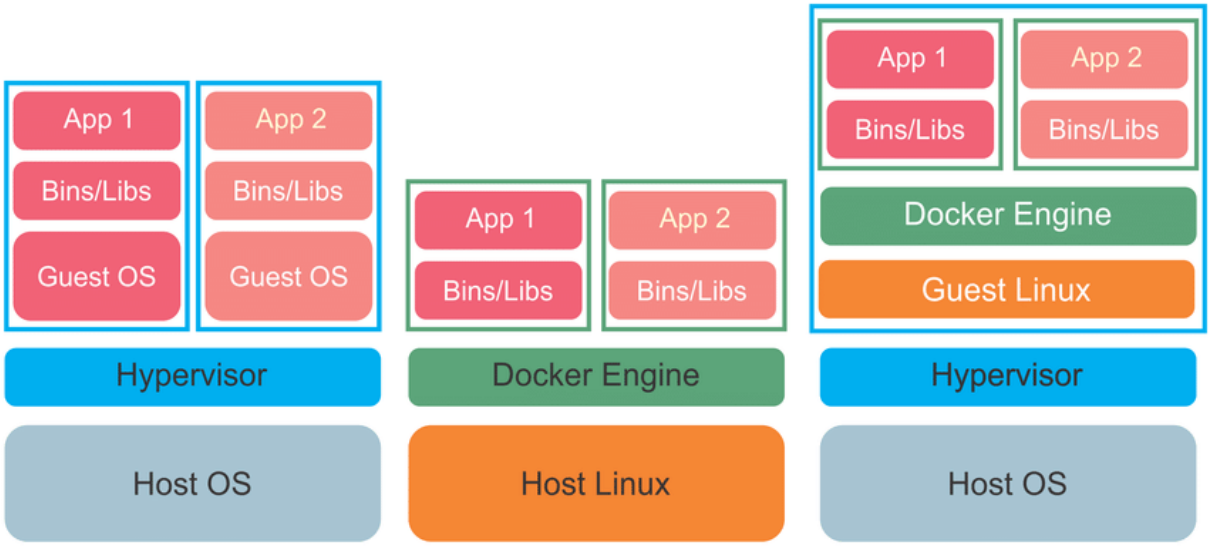


Figure 4.21 : Comparaison des 3 modèles principaux selon l'environnement d'exécution : Virtualisation - Conteneurisation - Modèle hybride

4.6.2 Selon l'architecture d'orchestration

4.6.2.1 Cluster basé sur Docker-Swarm

4.6.2.1.1 Docker

Les conteneurs sont un concept ancien. Depuis des décennies, les systèmes UNIX ont la commande chroot qui fournit une forme simple d'isolation du système de fichiers. Depuis 1998, FreeBSD dispose de l'utilitaire jail, qui étend le sandboxing chroot aux processus. Les zones Solaris offraient une technologie de conteneurisation relativement complète vers 2001, mais limitée à l'OS Solaris. Toujours en 2001, Parrallels Inc. (alors SWsoft) a lancé la technologie commerciale de conteneur Virtuozzo pour Linux, puis OpenVZ en 2005.3 Ensuite, Google a commencé le développement de CGroups pour le noyau Linux et a commencé à déplacer son infrastructure vers les conteneurs. Le projet Linux Containers (LXC) a débuté en 2008 et a rassemblé CGroups, les espaces de noms du noyau et la technologie chroot (entre autres) pour fournir une solution complète de conteneurisation. Enfin, en 2013, Docker a apporté les dernières pièces du puzzle de la conteneurisation, et la technologie a commencé à entrer dans le courant dominant.[27]

Docker est une solution complète pour la création et la distribution des conteneurs. Il s'est servi de la technologie des conteneurs Linux existante pour l'emballer et l'étendre de diverses façons principalement par le biais d'images portables et d'une interface conviviale. Docker est une plate-forme de virtualisation de conteneurs open source pour construire,

empaqueter et exécuter des applications distribuées dans des conteneurs qui sont des instantanés légers de l'OS sous-jacent. La plate-forme Docker comprend deux composants distincts : le moteur Docker (Docker Engine), qui est responsable de la création et de l'exploitation des conteneurs, et le Docker Hub, un service en Cloud pour la distribution des conteneurs.

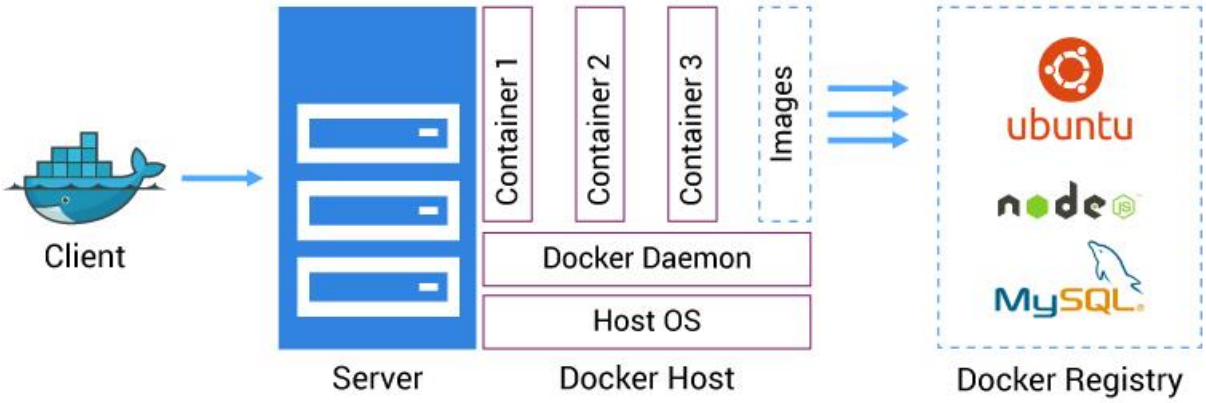


Figure 4.22 : L'architecture Docker

Le moteur docker offre une interface rapide et pratique pour le fonctionnement des conteneurs. Auparavant, l'utilisation d'un conteneur à l'aide d'une technologie telle que LXC nécessitait des connaissances spécialisées et un travail manuel important. Les conteneurs Docker sont isolés les uns des autres et disposent de leur propre réseau et système de fichiers et fournissent un service de conteneur sous forme de service (Containers as a Service CaaS). Le Docker Hub fournit un nombre énorme d'images de conteneurs publics à télécharger, permettant aux utilisateurs de démarrer rapidement et d'éviter de dupliquer le travail déjà fait par d'autres. Parmi les autres outils développés par Docker figurent Swarm, un gestionnaire de clustering, Kitematic, une interface graphique pour travailler avec les conteneurs, et Machine, un utilitaire en ligne de commande pour provisionner les hôtes Docker.

4.6.2.1.2 Swarm

Swarm est l'outil de clustering natif de Docker. Swarm utilise l'API standard de Docker, c'est-à-dire que les conteneurs peuvent être lancés en utilisant les commandes docker normales et Swarm se chargera de sélectionner un hôte approprié pour exécuter le conteneur. Cela signifie également que d'autres outils qui utilisent l'API Docker - tels que Composer et les scripts personnalisés - peuvent utiliser Swarm sans aucun changement et tirer parti de l'exécution sur un cluster plutôt que sur un seul hôte[28].

L'architecture de base de Swarm est assez simple : chaque hôte exécute un agent Swarm et un hôte exécute un gestionnaire Swarm (sur de petits clusters de test, cet hôte peut également exécuter un agent). Le gestionnaire est responsable de l'orchestration et de

l'ordonnancement des conteneurs sur les hôtes. Voici les principales caractéristiques de Swarm :

- Swarm peut être exécuté en mode haute disponibilité où etcd, Consul ou ZooKeeper sont utilisé pour gérer le basculement vers un gestionnaire de sauvegarde.
- Scalabilité : Nous pouvons déclarer le nombre de tâches que nous voulons exécuter, pour chaque service. Le gestionnaire Swarm s'adapte automatiquement en ajoutant ou supprimant des tâches pour maintenir l'état désiré, chaque fois que nous augmentons ou diminuons l'échelle.
- Réseaux multi-hôtes : Nous pouvons également spécifier un réseau de superposition pour nos services. Lorsque le gestionnaire Swarm initialise ou met à jour l'application, il assigne automatiquement des adresses aux conteneurs sur le réseau de recouvrement.
- Découverte de services : Les nœuds du gestionnaire Swarm attribuent à chaque service de Swarm un nom DNS unique ainsi que les équilibres de charge des conteneurs courants. Et, grâce à un serveur DNS intégré dans Swarm, nous pouvons interroger chaque conteneur fonctionnant dans Swarm.
- Équilibrage de charge : Fondamentalement, nous pouvons exposer les ports pour les services à un répartiteur de charge externe. De plus, Swarm nous permet de spécifier comment distribuer les conteneurs de service entre les nœuds, en interne.

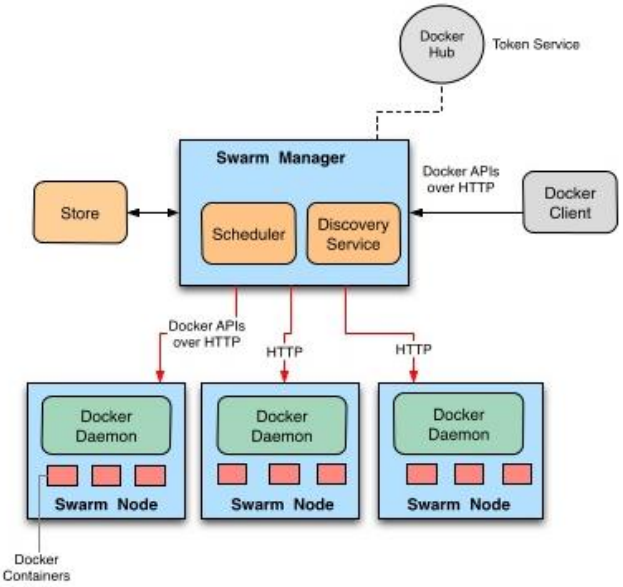


Figure 4.23 : Architecture d'orchestration sous Swarm

4.6.2.2 Cluster basé sur SAHARA

Sahara est un système Hadoop-as-a-service. C'est très nouveau, en fait, il a été ajouté dans la dixième version.

Sahara permet à l'utilisateur de créer des clusters Hadoop (Spark) rapidement et facilement. Il permet également à l'utilisateur d'avoir le contrôle total des clusters, en étant capable de définir un grand nombre de paramètres tels que comme la version du Spark, la topologie du cluster et les détails matériels du nœud. Après que l'utilisateur complète cette information, Sahara déploie le cluster en quelques minutes.[21]

Sahara permet également à l'utilisateur de lancer et de gérer les jobs MapReduce sur les clusters qui ont créé.la figure montre l'architecture de Sahara

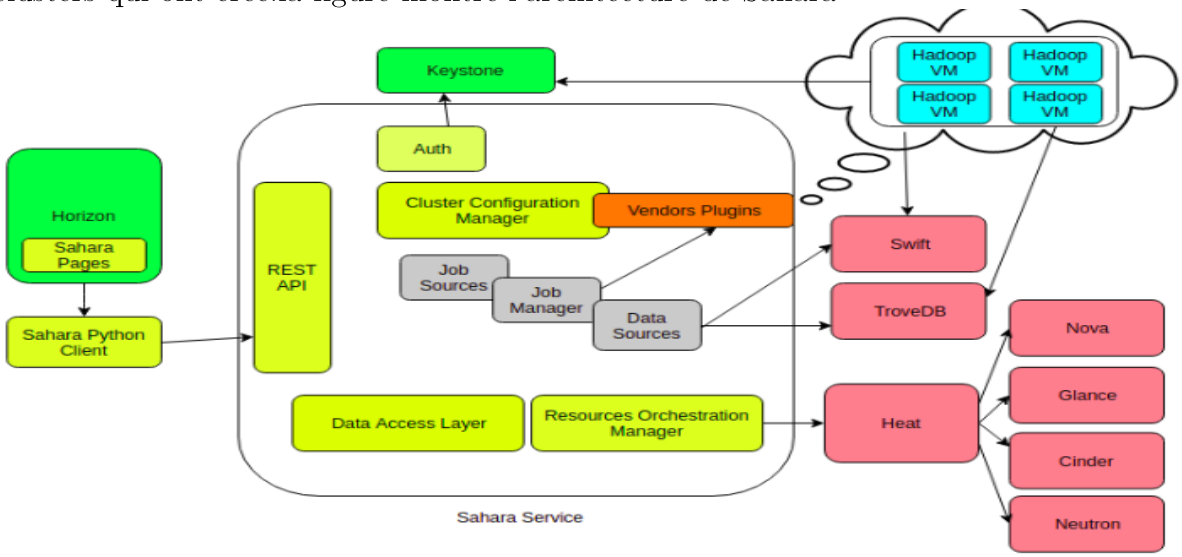


Figure 4.24 : Architecture du service SAHARA

4.6.2.3 Cluster basé sur Magnum

Magnum est un service OpenStack qui a été créé en 2014 par l'équipe des conteneurs OpenStack pour permettre à un moteur d'orchestration de conteneurs (COE) de déployer et de gérer des conteneurs comme ressources de premier ordre dans OpenStack.[21]

Actuellement, Magnum supporte Kubernetes, Apache Mesos et Docker Swarm COEs. Magnum utilise Heat pour faire l'orchestration de ces COEs sur des VMs ou des métaux fournis par OpenStack. Il utilise des images du système d'exploitation qui contiennent les outils nécessaires à l'exécution des conteneurs. Magnum propose des API compatibles KeyStone et une solution complète pour la gestion de vos COEs sur un cluster OpenStack.

Un cluster Magnum est un ensemble de diverses ressources fournies par différents services OpenStack. Il se compose d'un groupe de machines virtuelles approvisionnées par Nova, de réseaux reliant ces machines virtuelles créées par Neutron, de volumes attachés aux machines virtuelles créées par Cinder, et ainsi de suite. Un cluster Magnum peut également avoir des ressources externes en fonction des options fournies lors de la création

d'un cluster. Par exemple, nous pouvons créer un équilibreur de charge externe pour notre cluster en spécifiant l'option -master-lb-enabled dans le modèle de cluster. Certaines des caractéristiques saillantes de Magnum sont :

- Fournit une API standard pour la gestion complète du cycle de vie des centres d'excellence.
- Prise en charge de plusieurs centres d'excellence tels que Kubernetes, Swarm, Mesos et DC/OS.
- Prise en charge de la capacité d'extension d'un cluster vers le haut ou vers le bas
- Prise en charge de la multilocation pour les groupes de conteneurs
- Différents choix de modèles de déploiement de clusters de conteneurs : VM ou métal.
- Fournit une sécurité multi-locataires basée sur KeyStone et une gestion des droits d'accès.
- Contrôle et isolation d'un réseau multi-locataires à base de neutrons
- Supporte Cinder pour donner du volume aux conteneurs
- Intégrité avec OpenStack
- Accès sécurisé aux clusters de conteneurs (Transport Layer Security (TLS)) activé
- Prise en charge de l'infrastructure externe peut également être utilisée par le cluster, comme le DNS, le réseau public, le service de découverte publique, le registre Docker, le load-balancer, etc.
- Barbican fournit le stockage des secrets tels que les certificats utilisés pour TLS au sein du cluster.
- Mise en réseau basée sur Kuryr pour l'isolation au niveau des conteneurs

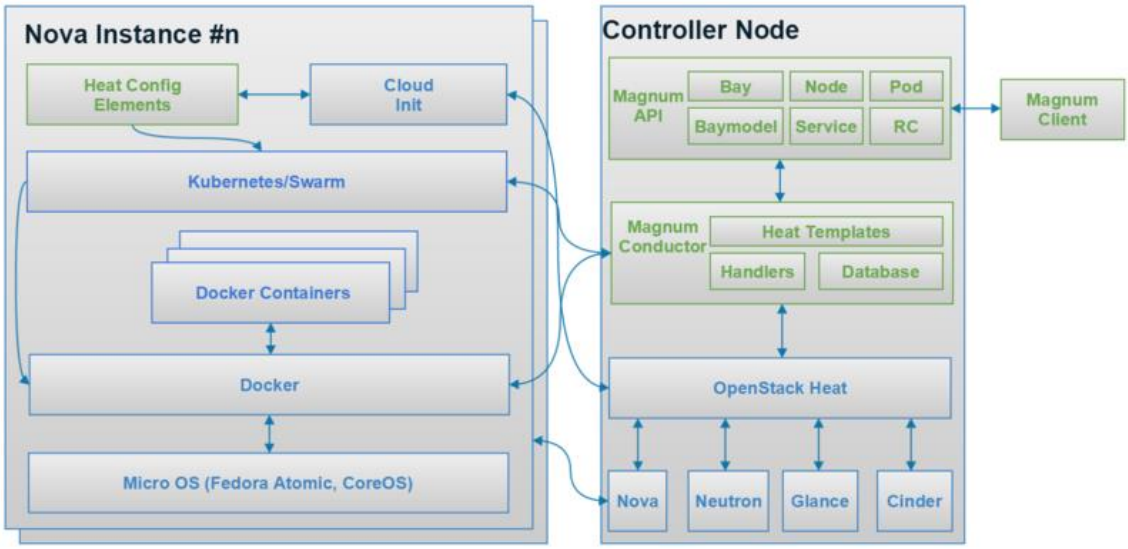


Figure 4.25 : Architecture Magnum

4.6.2.4 Cluster basé sur Kubernetes

Kubernetes est un gestionnaire de cluster pour les conteneurs Linux. Bien que Kubernetes supporte d'autres types de conteneurs tels que Rocket, et que le support pour d'autres types soit à ajouter, nous discuterons de Kubernetes dans le contexte des conteneurs Docker uniquement. Kubernetes (généralement abrévié k8s) est un système open-source d'orchestration de conteneurs pour automatiser le déploiement, la scalabilité et la gestion des applications. Il a été conçu à l'origine par Google, et est maintenant maintenu par la Cloud Native Computing Foundation. Il vise à fournir une "plate-forme pour automatiser le déploiement, la scalabilité et les opérations des conteneurs d'applications à travers des clusters de machines". De nombreux services cloud offrent une plate-forme ou une infrastructure basée sur Kubernetes en tant que service (PaaS ou IaaaS) sur laquelle Kubernetes peut être déployé en tant que service fournissant une plateforme [29]. De nombreux fournisseurs proposent également leurs propres distributions Kubernetes de marque. Kubernetes est largement utilisé comme un moyen d'héberger une implémentation basée en microservices, parce qu'il et son écosystème d'outils associés fournissent toutes les capacités nécessaires pour répondre aux principales préoccupations de toute architecture de microservices. Ses avantages principaux sont :

- Automatise divers processus manuels : par exemple, Kubernetes contrôlera pour vous quel serveur hébergera le conteneur, la maniéré dont-il comment il sera lancé, etc.
- Interagit avec plusieurs groupes de conteneurs : Kubernetes est capable de gérer plus de clusters en même temps

- Fournit des services supplémentaires : outre la gestion des conteneurs, Kubernetes offre des services de sécurité, de mise en réseau et de stockage.
- Autosurveillance : Kubernetes vérifie en permanence l'état de santé des nœuds et des conteneurs
- Scalabilité horizontale : Kubernetes vous permet d'échelonner les ressources non seulement verticalement mais aussi horizontalement, facilement et rapidement.
- Orchestration du stockage : Kubernetes monte et ajoute un système de stockage de votre choix pour exécuter des applications
- Automatise les déploiements et les rollbacks : si après une modification de votre application quelque chose ne va pas, Kubernetes l'annulera automatiquement.
- Équilibrage du conteneur : Kubernetes sait toujours où placer les conteneurs, en calculant le "meilleur emplacement" pour eux.
- Fonctionne partout : Kubernetes est un outil open source qui vous donne la liberté de tirer profit de l'infrastructure sur site, hybride ou en nuage public, vous permettant de déplacer les charges de travail où vous le souhaitez.

Voici l'architecture qui assure à Kubernetes ses caractéristiques :

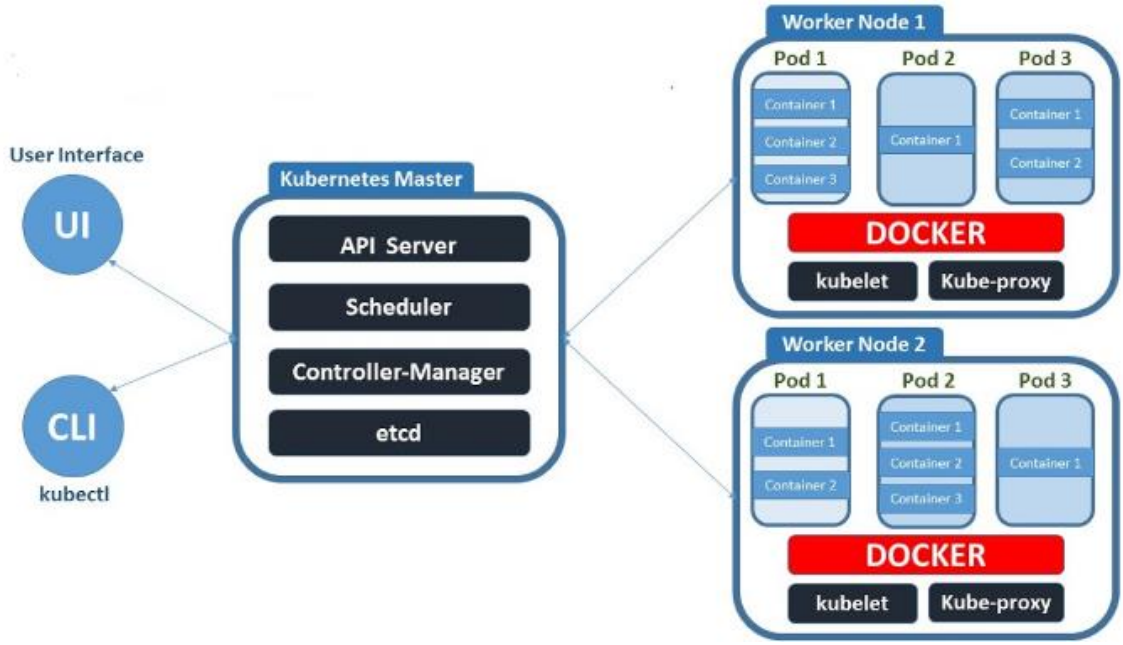


Figure 4.26 : Architecture d'orchestration sous Kubernetes avec Docker

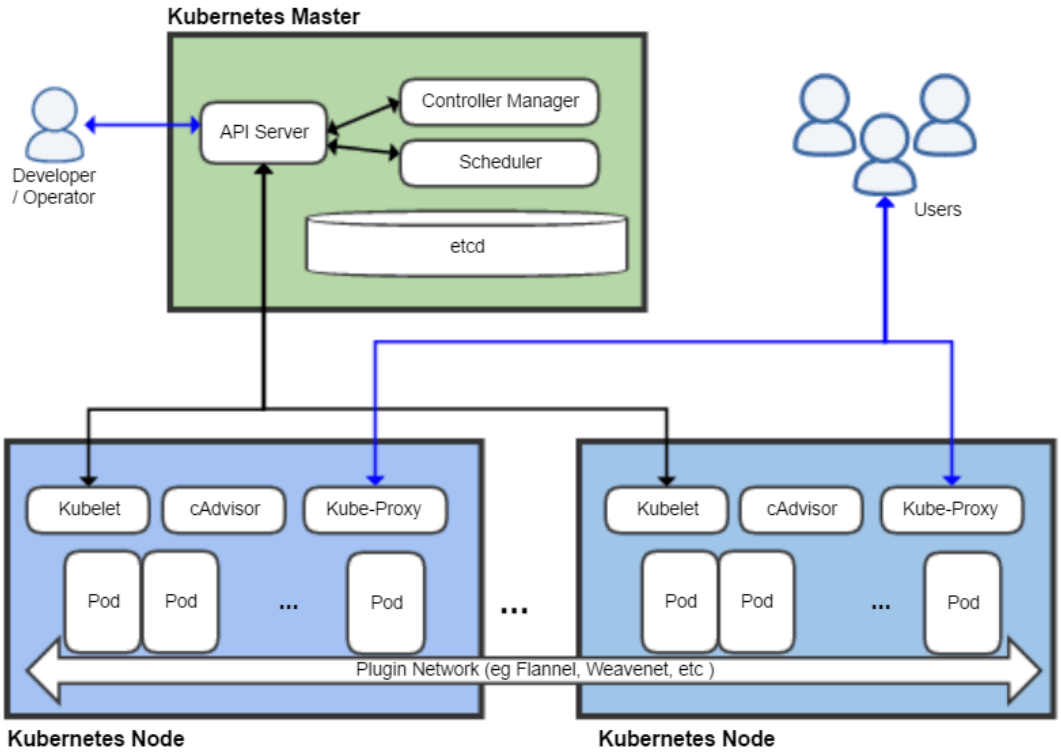


Figure 4.27 : Architecture d'orchestration sous Kubernetes vue du client

Kubernetes définit un ensemble d'éléments de base (" primitives "), qui fournissent collectivement des mécanismes de déploiement, de maintenance et de mise à l'échelle des applications basées sur le CPU, la mémoire ou des mesures personnalisées.Kubernetes est librement couplé et extensible pour répondre aux différentes charges de travail. Cette extensibilité est assurée en grande partie par l'API Kubernetes, qui est utilisée par les composants internes ainsi que par les extensions et les conteneurs fonctionnant sur Kubernetes. La plate-forme exerce son contrôle sur les ressources de calcul et de stockage en définissant les ressources comme des objets, qui peuvent ensuite être gérés comme tels. Les objets clés sont :

- Cluster : Un cluster est une collection de nœuds incluant d'autres ressources telles que le stockage pour exécuter des applications Kubernetes. Un cluster a un seul nœud maître Kubernetes et zéro ou plusieurs nœuds ouvriers. Un cluster hautement disponible se compose de plusieurs maîtres ou noeuds maîtres.
- Pod : Un Pod est une collection de conteneurs qui sont colocalisés et forment une unité atomique. Plusieurs applications peuvent être exécutées au sein d'un Pod et bien que les différents conteneurs d'un Pod puissent être destinés à la même application, les différents conteneurs sont généralement destinés à des applications différentes. Un Pod est une abstraction de niveau supérieur pour gérer un groupe de conteneurs avec des volumes et un espace de noms réseau partagés. Toutes les

applications (conteneurs) d'un Pod partagent le même système de fichiers et la même adresse IP, le port sur lequel chaque application est exposée étant différent. Les applications s'exécutant dans un Pod peuvent se connecter entre elles sur "localhost". L'ordonnancement et la réplication sont effectués au niveau du Pod plutôt qu'au niveau du conteneur individuel.

- Service : Un Service est l'interface externe d'un ou de plusieurs Pods fournissant le ou les points finaux (endpoints) auxquels la ou les applications représentées par le Service peuvent être invoquées. Un service est hébergé sur une seule adresse IP, mais ne fournit aucun point d'accès ou plus, selon l'application ou les applications interfacées par le service. Les services sont connectés aux Pods à l'aide de sélecteurs d'étiquettes. Les Pods ont des étiquettes sur eux et un Service avec une expression de sélecteur identique à une étiquette Pod représente le Pod pour un client externe.
- Etiquette (Label) : Une étiquette est une paire clé-valeur identifiant une ressource telle qu'un contrôleur de pod, de service ou de réplication : le plus souvent un pod. Les étiquettes servent à identifier un groupe ou un sous-ensemble de ressources pour des tâches telles que leur affectation à un service. Les services utilisent des sélecteurs d'étiquettes pour sélectionner les Pods qu'ils gèrent.
- Nom : Un nom identifie une ressource. Un nom n'est pas la même chose qu'une étiquette. Pour faire correspondre les ressources à un service, on utilise une étiquette et non un nom.
- Espace de nommage (namespace) : Un espace de nommage est un niveau au-dessus du nom pour délimiter un groupe de ressources pour un projet ou une équipe afin d'éviter les collisions de noms. Les ressources dans des espaces de noms différents peuvent avoir le même nom, mais les ressources dans un espace de noms ont des noms différents.
- Volume : Un volume est un répertoire dans le système de fichiers d'un conteneur. Un volume peut être utilisé pour stocker les données. Les volumes de Kubernetes évoluent à partir des volumes de Docker.
- Kubelet : Kubelet est responsable de l'état de fonctionnement de chaque nœud, s'assurant que tous les conteneurs sur le nœud sont sains. Il s'occupe du démarrage, de l'arrêt et de l'entretien des conteneurs d'application organisés en pods selon les instructions du plan de contrôle. Kubelet surveille l'état d'un pod, et s'il n'est pas dans l'état souhaité, le pod se redéploie sur le même nœud. L'état du nœud est relayé toutes les quelques secondes par l'intermédiaire de messages de battements de cœur (heartbeats) envoyés au nœud primaire. Une fois que le primaire détecte

une défaillance de noeud, le contrôleur de réplication observe ce changement d'état et lance des pods sur d'autres nœuds sains.

- Kube-proxy : Le Kube-proxy est une implémentation d'un proxy réseau et d'un load-balancer, et il supporte l'abstraction de service ainsi que d'autres opérations réseau. Il est responsable de l'acheminement du trafic vers le conteneur approprié en fonction de l'IP et du numéro de port de la requête entrante.

4.6.2.5 Cluster basé sur Nova-Openstack

L'un des buts de ce projet est de mettre en évidence l'avantage d'une architecture Spark à conteneurs par rapport à la même architecture en machines virtuelles, cela nécessite de mettre en œuvre un cluster en VM portant une architecture de Spark avec les outils d'évaluation nécessaire afin de pouvoir quantifier sa performance et l'utiliser comme référence pour d'autre architecture de calcul. Dans le contexte du cloud computing sur OpenStack, Nova est pratiquement le seul composant pouvant fournir la base d'un service IaaS :La virtualisation et la gestion des ressources physiques. Comme s'est mentionné dans le chapitre 2, Nova va assurer la création et la gestion en runtime de machines virtuelles portantes Spark. Les principales fonctionnalités supportées par nova :

- Création d'un serveur et la suppression un serveur
- Instantané d'un serveur
- Mise en marche du serveur
- Recréation du serveur
- Redimensionner le serveur
- Opérations de volume
- Configurations de disque personnalisées au démarrage
- La mise en pause d'un serveur
- Suspendre un serveur
- Sortie console serveur
- Sauvetage du serveur
- Lecteur de configuration de serveur
- Configuration d'une clé SSL pour les connections du serveur
- Changement de mot de passe du serveur (expérimental)
- Étagères et rayonnages de serveur

Ces fonctionnalités sont largement suffisantes pour le déploiement d'une solution Spark avec les critères qu'on a abordé ci-dessus.

4.6.3 Choix des solutions et procédure d'installation du Apache Spark sur un cluster :

4.6.3.1 Cluster Spark en machines virtuelles

Pour mettre en œuvre une architecture Spark en VM, il faut d'abord créer ces machines virtuelles qui vont former notre cluster. Ceci doit être fait après la création d'un projet Spark dans un nom de domaine Spark, un utilisateur Spark avec des privilèges administratifs. Dans l'OpenStack, on va utiliser :

- Nova pour la création et la gestion des machines, ainsi que les gabarits caractérisant les ressources physiques allouées à nos machines (minimum 2GB de RAM et 1 cœur CPU pour Spark)
- Glance pour stocker l'image qui va booter nos machines (il est recommandé que ça soit une image Ubuntu Xenial 16.04 commune pour toutes les machines du cluster, avec un format QCOW2 et non pas ISO).
- Cinder pour la gestion des volumes des différentes machines.
- Neutron pour création du sous-réseau contenant nos machines, le réseau publique provider-net pour l'accès à l'internet, les routeurs qui vont faire le NAT privé-publique en créant les IPs-flottants.
- Keystone pour générer la clé de communication SSH qu'on va s'en servir pour accéder à nos machines à partir d'un terminal en local (Putty par example).
- Toutes ces services sont disponibles à travers le tableau de bord Horizon.
- Il est aussi possible de faire toutes les configurations ci-dessus avec un seul fichier .yaml et le faire passer au serveur API du Heat. Ce dernier va créer les instances en communicants avec les composants nécessaires à travers des API qu'on appelle des ressources. l'intérêt d'une telle solution sera évidemment la facilité et la portabilité de la solution du déploiement.

Une fois les configurations nécessaires sont faites, notre cluster de machines sera prêt pour être créé. Après la création du cluster, il est recommandé de faire des tests sur la communication interne du cluster, l'accès au réseau public d'OpenStack et internet de chacune de nos machines, l'accès en SSH à partir d'un desktop du réseau public. Ceci garantit la bonne démarche de l'installation du reste de la solution. Voici une vue générale du cluster créé avec 3 machines, un maître et 2 esclaves :

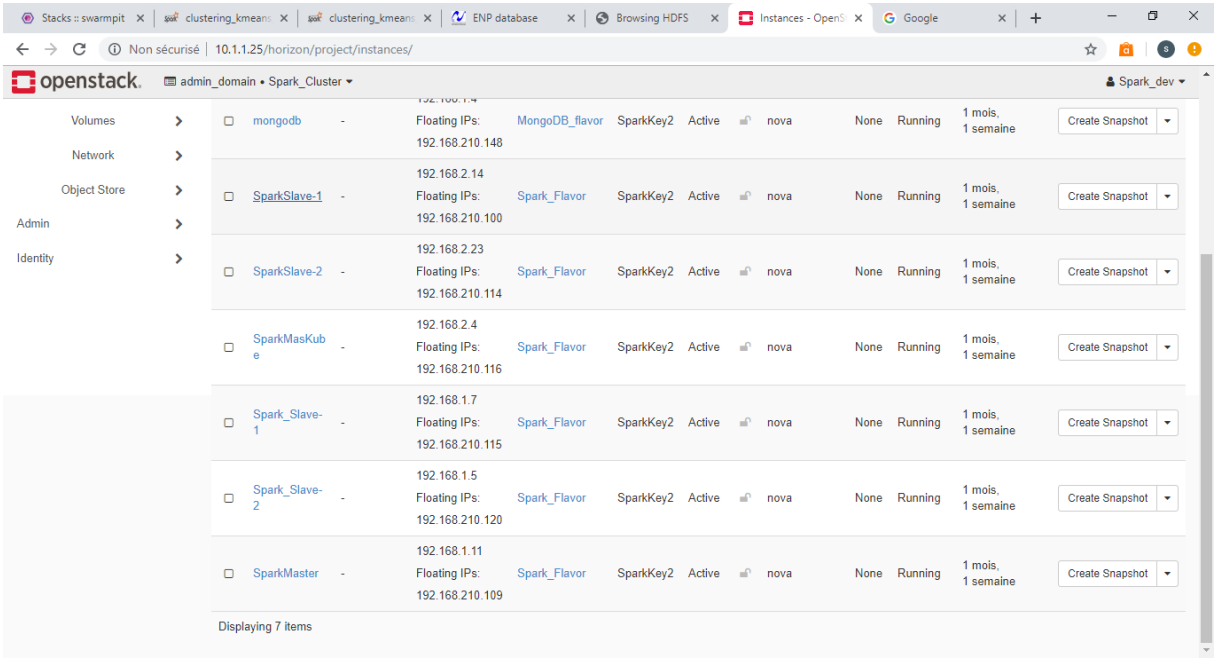


Figure 4.28 : L'ensemble des machines virtuelles formants le cluster Spark crée sur OpenStack vue Nova

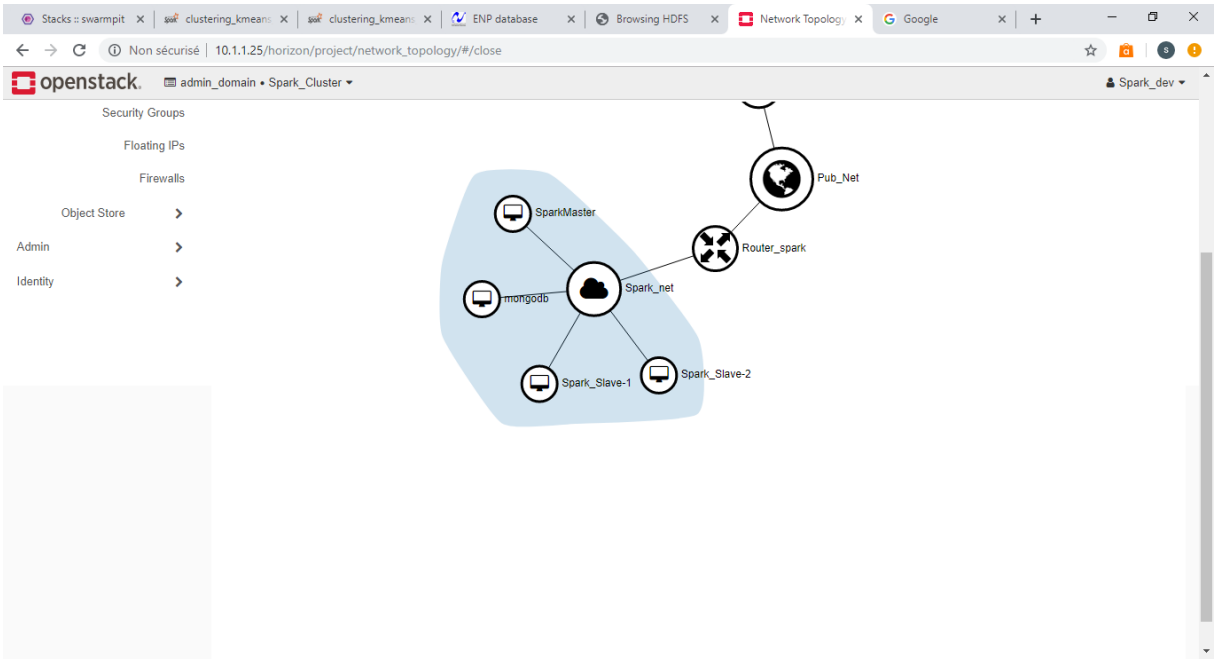


Figure 4.29 : La topolgie réseau du cluster Spark des VMs vue Neutron

Le cluster est créé, mais ce n'ai qu'une infrastructure pour le cluster de calcul Spark. Pour installer Spark, il faut d'abord installer Java et Scala sur toute les machines, puisque une partie de Spark est écrite en Scala, et tout le plan de compilation du code source ou d'exectuion des applications Spark repose sur JAVA, c'est la raison pour laquelle on parle que des JVMs dans le contexte de Spark. Après la validation de l'installation du Java ensuite Scala, On peut télécharger le binaire du Spark à partir

du site officiel Apache en précisant la version du spark souhaitée (une version est lancé chaque mois), et le type de paquetage. Si on choisit un type de paquetage code source, on peut créer le binaire à partir du code source télécharger, l'un des meilleurs gestionnaire de compilation automatique utilisé est Maven, mais il faut l'installer aussi. Une fois le binaire est téléchargé et décompressé, il est préférable de le déplacer vers un répertoire de privilèges root pour des raisons de sécurité. Les trois répertoires clés lors du développement des applications Spark sont : bin, sbin et conf. Bin contient les fichiers binaires pour les terminaux spark-shell en scala et pyspark en python, spark-sql pour exploiter les foncionnalité des bases de données SQL à travers un terminal de spark, et spark-submit pour lançer des applications Spark. Les terminaux sont très utiles pour le debuggage. Sbin est le dossier qui sert à déployer l'architecture Spark. Il contient des scripts qui exploitent des classes en JAVA pour déployer le maitre, l'esclave, le maitre et les esclaves sur des autres machines ou bien les esclaves seulement, le history-server qui va servir comme interfaces web pour accéder au logs des différentes applications Spark exécutés après leurs temps d'exécution, ceci est très utiles pour faire une analyse sur l'exécution des applications Spark, dans un contexte d'évaluation ou de débogage. Conf est le répertoire qui fournisse les configurations portées par l'objet SparkContext lors de lancement de l'application et que sera exploité par le gestionnaire du cluster, c'est le répertoire qui définit l'environnement de l'exécution de l'application. Les fichiers principaux à configurer sont : slaves, spark-defaults.conf et spark-env.sh. slaves pour définir les noms d'hôte de chacun des esclaves, la résolution en IP dans un système d'exploitation Debian (comme Ubuntu) se fait par l'intermédiaire du fichier /etc/hosts, donc les esclaves doivent être inscrit aussi dans ce fichier. Spark-env.sh est un script qui s'exécute à chaque exécution d'une application spark, il contient les variables d'environnement comme les chemins, les IPs du maitre ou esclaves. Spark-defaults.conf contient l'ensemble des configurations qui servent comme un levier de contrôle pour l'exécution d'une application Spark, du fait qu'ils peuvent paramétrer l'exécution à plusieurs niveaux, voici quelques configurations qu'on peut utiliser pour optimiser la performance de notre cluster Spark :

Paramètre	Valeur par défaut	Signification
`spark.driver.memory`	1 G	Quantité de mémoire à utiliser pour le processus du pilote
`spark.executor.memory`	1 G	Quantité de mémoire à utiliser pour le processus de l'exécuteur

spark.executor.cores	1	Nombre de cœurs CPU à utiliser pour le processus de l'exécuteur
spark.dynamicAllocation.enabled	false	L'utilisation de l'allocation dynamique des ressources, qui permet d'échelonner le nombre d'exécuteurs enregistrés auprès de cette application en fonction de la charge de travail.
spark.extraListeners	vide	Une liste de classes séparées par des virgules qui implémentent SparkListener ; lors de l'initialisation de SparkContext, les instances de ces classes seront créées et enregistrées dans le bus de l'auditeur de Spark.
spark.master	local	Le gestionnaire de cluster pour se connecter (local, local avec multi-process, cluster avec standalone, cluster avec mesos,...)
spark.submit.deployMode	vide	Le mode de déploiement du programme de pilote Spark, soit "client" ou "cluster", ce qui signifie lancer le programme de pilote localement ("client") ou à distance ("cluster") sur un des nœuds du cluster.

spark.driver.supervise	false	Si true, redémarre automatiquement le pilote s'il échoue avec un statut de sortie différent de zéro. N'a d'effet qu'en mode Spark autonome ou en mode de déploiement en cluster Mesos.
spark.eventLog.enabled	false	S'il faut journaliser les événements Spark, utile pour reconstruire l'interface utilisateur Web une fois l'application terminée.
spark.eventLog.dir	file:///tmp/spark-events	Répertoire de base dans lequel les événements Spark sont enregistrés, si spark.eventLog.enabled est vrai.
spark.jars	Jars native Spark	Liste de jars séparés par des virgules à inclure sur les chemins de classes du pilote et de l'exécuteur. Les chemins top-level sont autorisés.

Tableau 2 : Configurations nécessaires au bon fonctionnement du calcul sur Spark

Pour créer notre cluster Spark, on lance un processus maitre dans la machine master, en exécutant le script start-master.sh dans /spark-2.4.1-bin-hadoop2.7/sbin/. Le master fournit une interface web sur le port 8080. On tire l'URL du maître spark://spark-master-IP-or-hostname :7077, on accède aux machines esclaves pour déployer les travailleurs avec le script start-slave.sh et comme option l'URL du maitre. On peut également configurer les esclaves dans le fichier /conf/slaves et créer une paire de clé SSH-RSA dans le root@master, et diffuser la clé SSH publique dans les machines esclaves parmi les clés autorisées de l'utilisateur root@slave1/2, et lancer tout le cluster avec une seule commande à partir du maitre /sbin/start-all.sh. Si cela s'exécute sans erreur,

notre cluster Spark et intra-lié, on peut le consulter à travers l'interface web avec l'URL spark-master-IP:8080 comme suit :

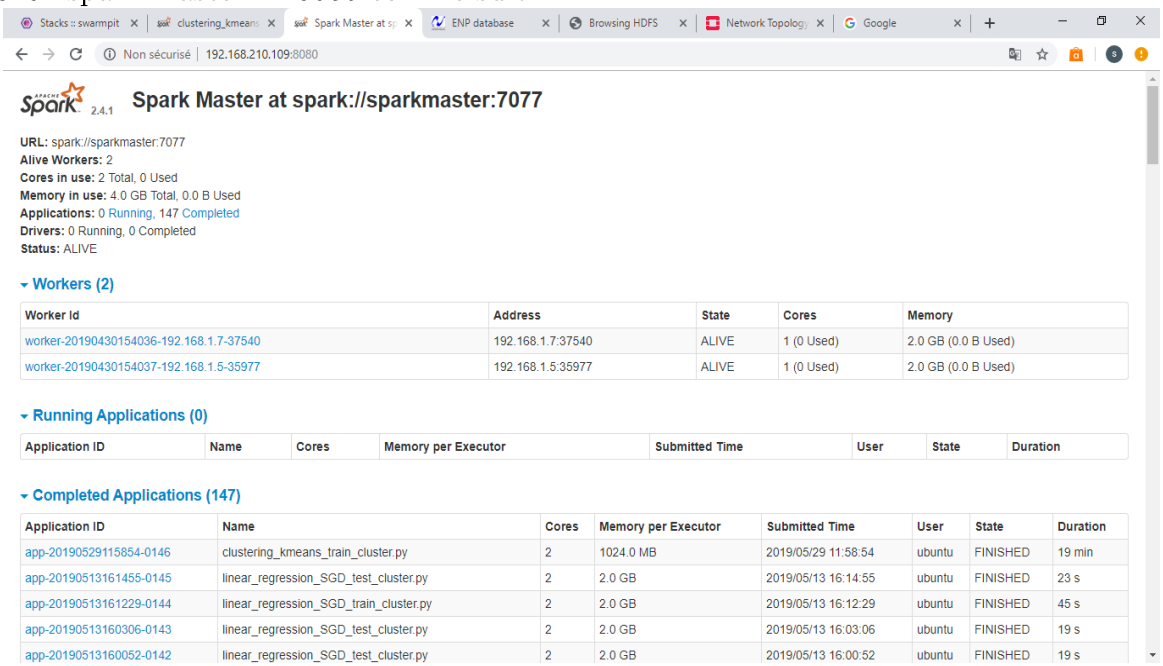


Figure 4.30 : WebUI du cluster Spark en VMs

On peut voir les différents Job et stages exécuté par une application Spark durant le temps d'exécution dans le port 4040 depuis le master, mais lorsque l'application est terminée, on perdra l'accès à ces information à travers une page web, donc les détails sur la performance et l'exécution seront invisible, sauf si on utilise un serveur de l'historique. Pour utiliser le serveur de l'historique, il faut utiliser les configurations Spark pour activer l'option d'enregistrement des logs et créer leur chemin et le configurer aussi. Le serveur de l'historique diffuse une interface web dans le port 18080, la voici :

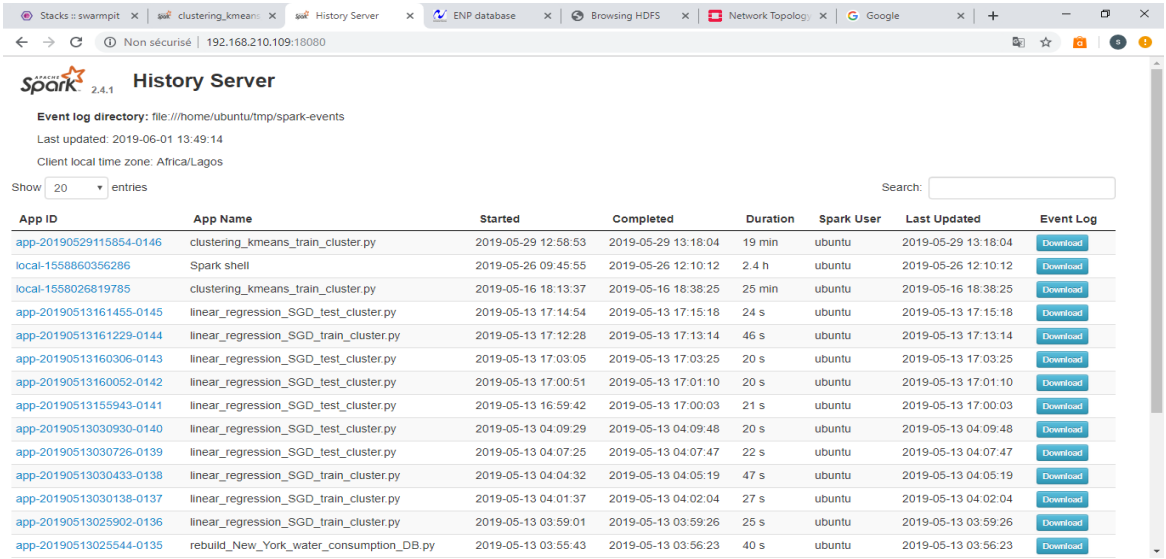


Figure 4.31 : WebUI du serveur de l'historique

Pour le fonctionnement de l'HDFS, il faut installer Hadoop, dans chacune des machines du cluster et les configurer avant de pouvoir lancer le service dfs dans le cluster.

En configurant `spark.master`=spark://spark-master-IP-or-hostname :7077, et après la procédure expliqué ci-dessus, lors du lancement d'un processus Spark depuis le master, l'application en jeu sera prise ne charge pour tout le cluster (les travailleurs concrètement).

4.6.3.2 Cluster Spark en Conteneurs

La démarche suivis lors la création du cluster sur des conteneurs est similaire à celle qui a été faite sur des conteneurs dans la partie connexion maitre-esclave, même si celles-ci peuvent être automatisées avec le déploiement lui-même, mais c'est le déploiement des conteneurs Spark qui fera la différence. Pour le déploiement, il faut avoir un autre cluster de machines (à ne pas confondre avec le cluster en Spark, le cluster en machine ne fait que l'infrastructure pour les conteneurs dans ce cas), avec Docker et Kubernetes installé dans chacune d'eux. Voici l'architecture des machines portantes Docker et Kubernetes :

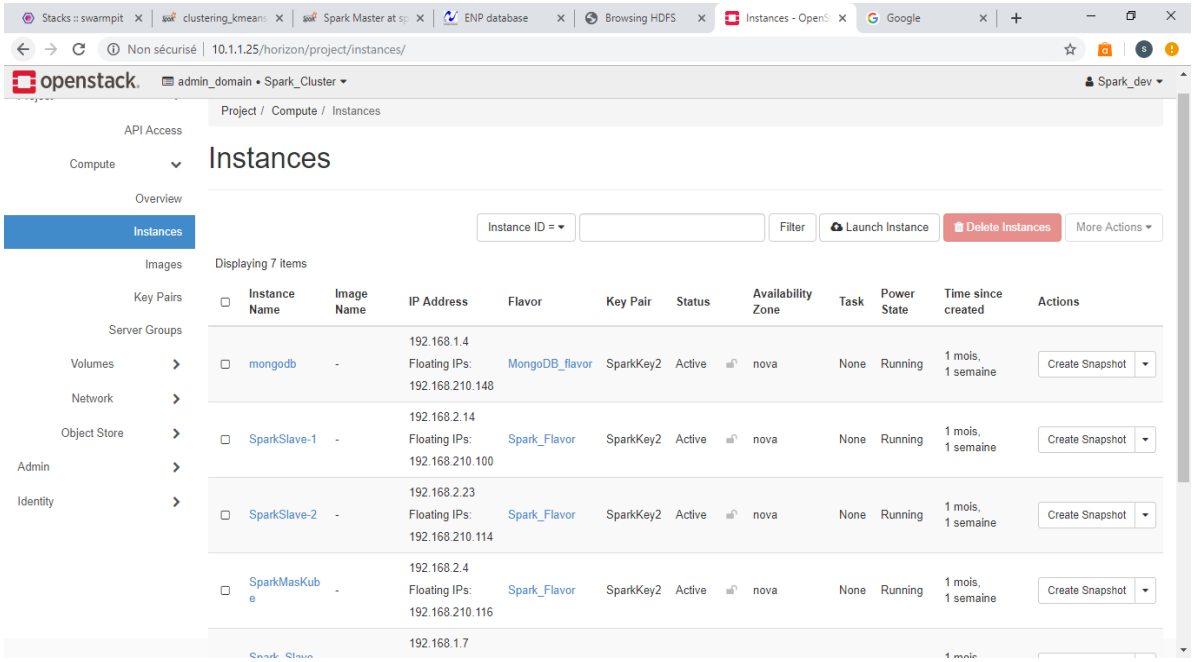


Figure 4.32 : L'ensemble des machines virtuelles hébergeants le cluster Spark en conteneurs

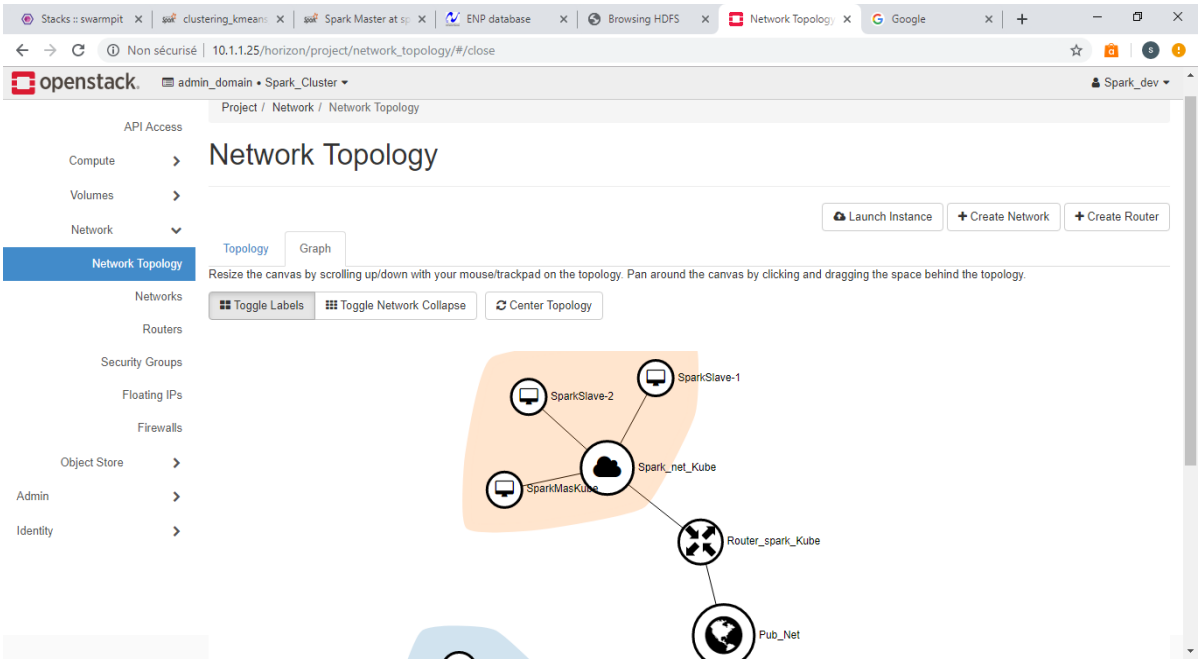


Figure 4.33 : La topologie réseau des machines virtuelle hébergeant le cluster Spark en conteneurs

Après avoir installé Docker et Kubernetes sur chacune des machines, on commence par écrire un Dockerfile pour créer l'image qui va être utilisée pour lancer des processus Spark (maitre ou esclaves) dans des conteneurs. Ce Dockerfile va contenir les paquets nécessaires à Spark pour fonctionner (JAVA et Scala et autres), des bibliothèques systèmes sont aussi nécessaires, des binaires systèmes pour l'exploitation des conteneurs, copier des scripts à

partir du local et les exécuter dès que les conteneurs soient créé (utile si on veut automatiser la connections du maitre et ses travailleurs). On compile le Dockerfile avec le démon Docker (Docker Engine), lui associer un tag et enregistrer dans le service cloud du Docker Hub pour stocker les images des conteneurs dans des répertoires associé à des utilisateurs. Après avoir créé l'image, on crée un cluster sur Kubernetes en spécifiant l'adresse du réseau qu'on va associer à notre cluster. Cette étape va générer des instructions à suivre pour établir les configurations du cluster créé par Kubernetes. Il est aussi impératif de déployer la CNI (Containers Network Interface), c'est un module d'extension (plug-in) qui va se charger de configurer les interfaces Linux des conteneurs et leurs connectivités, ainsi que supprimer les ressources alloué à des conteneurs en fin de cycle. Nous recommandons la CNI Calico pour cette tâche. Notre sur-infrastructure est prête pour créer le cluster en conteneurs avec Kubernetes passant par son serveur API kubectl. On utilise un fichier de configuration de déploiement yaml pour créer le maitre en mode déploiement (ceci assure la récréation du conteneur en état d'échec ou en cas supprimé en supprimant son pod), en spécifiant son nom, son image, ses ressources, ses ports et une commande qui exécute un script qui déploie le processus maitre de Spark lors de la création du conteneur. On crée aussi des services qui vont exposer le service du maitre Spark dans le port 7077, et l'interface web Spark dans le port 8080 du conteneur maitre, pour qu'il soit visible par les esclaves et par le développeur, l'interface web du history-server pour qu'on accède à l'historique d'exécution afin de pouvoir analyser les résultats. On crée aussi des esclaves répliqué autant de fois que l'on met en argument dans le fichier de configuration, en spécifiant leur image, leurs ports, leurs ressources etc. On demande au serveur kubectl d'exécuter ce fichier de déploiement yaml et de créer le cluster Spark, de nombreuses échecs peuvent surprendre même si le fichier de configuration est correctement écrit, ceci est généralement dû à des problèmes liés à l'image. Voici l'interface de Spark après l'avoir créer à travers Kubernetes :

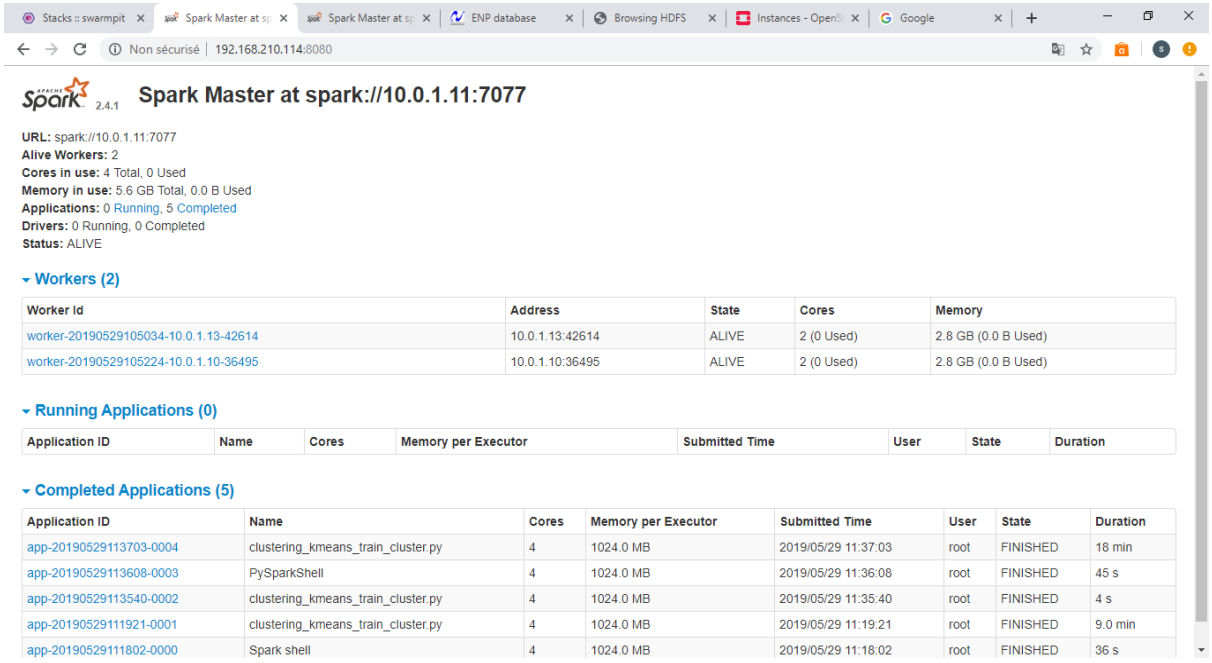


Figure 4.34 : WebUI du cluster en conteneur de Spark

Et voici le history-server :

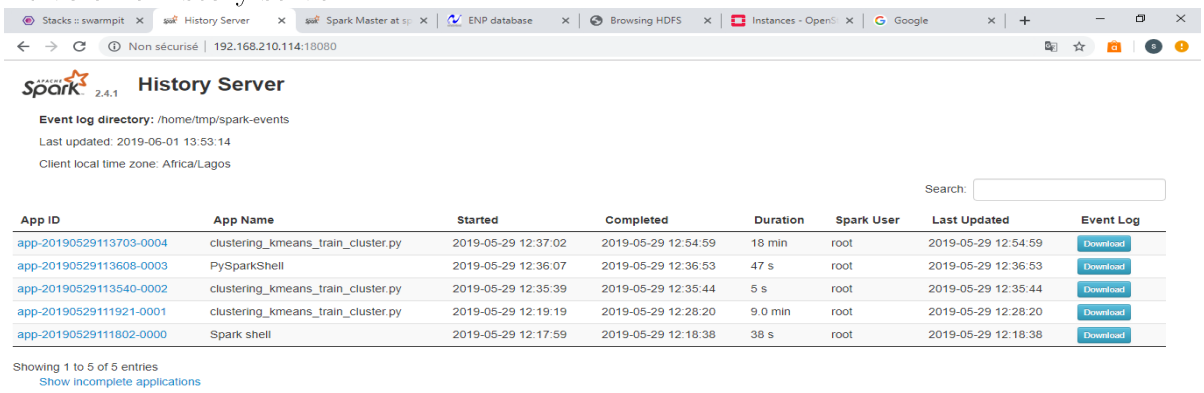


Figure 4.35 : WebUI du serveur de l'historique correspondant au cluster des conteneurs

Une fois ceci est achevé, nos deux architectures sont prêtes à être exploitées, en exécutant les applications Spark qu'on a développé, et en fournissant des données relatives aux exécutions pour permettre l'analyse de performance.

4.6.3.3 Comparaison de performance et explication du Benchmarking

Pour faire la comparaison ou du Benchmarking, on utilise :

- Une machine virtuelle master, et 2 machines virtuelles esclaves pour le cluster en VM utilisant toutes une image Ubuntu Xenial 16.04, et un gabarit de 2 VCPUs,

4GB de RAM, 20 GB de stockage, commun pour toutes les machines, et un hyperviseur QEMU

- Un conteneur master, 2 conteneurs esclaves, avec une image de base debian debian:stretch répartis sur 3 machines virtuelles Ubuntu Xenial 16.04 (modèle hybride), hypervisé par QEMU, avec un gabarit de 2 VCPUs, 4GB de RAM, 20 GB de stockage, orchestré par Kubernetes et Docker
- Kubernetes serveur API version 1.14.1, Docker Engine version 18.09.6, Docker Client 18.09.5
- Une charge de travail composé d'une base de données OBD (On-Board Diagnostics) de taille 2.99 GB (7.214.695 échantillons de 21 dimensions)
- Une classification non-supervisé sous l'algorithme « Kmeans » fournit par Spark-mllib API avec k=8 sur 80 % du volume totale des données comme jeu de données d'entraînement, avec et 20 % pour le jeu de données de validation.
- Une version de Spark 2.4.1-bin-hadoop-2.7
- Un serveur base de données NoSQL MongoDB V 3.4.20, sur une machine Ubuntu Xenial 16.04, avec 1 VCPU, 4GB de RAM et 40GB de stockage.
- Un système de fichiers distribué Hadoop 2.7.6 HDFS
- Il faut noter que d'après l'architecture réseau qu'on a créé et exploité pour ce projet, l'HDFS est créé en parallèle avec notre cluster en VM de référence, ainsi et le serveur de la base de données MongoDB qui se trouve dans le même réseau que le cluster en VM, ceci pourra causer que les échanges de données du cluster en conteneurs avec chacun des deux composants (HDFS et MongoDB), vont être plus couteux en terme de temps par rapport à celle en VM. On peut voir ceci dans la figure suivante :

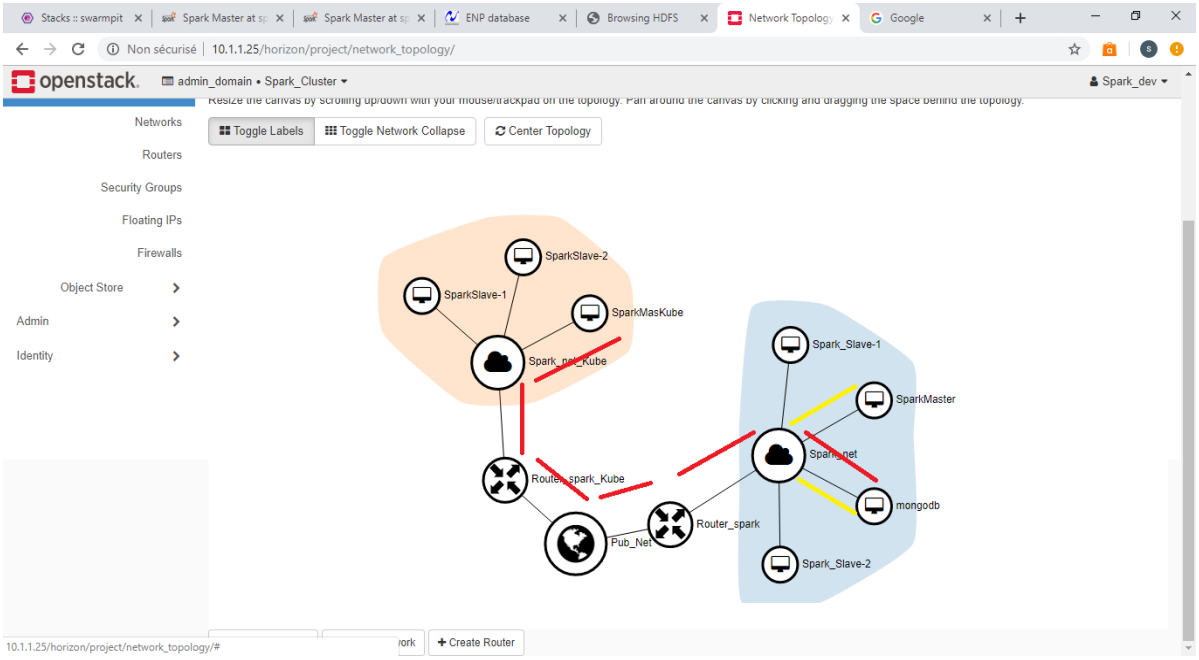


Figure 4.36 : L'architecture réseau montrant la différence du temps d'accès vers le serveur MongoDB du cluster des conteneurs (en rouge) par rapport au cluster en VM (en jaune)

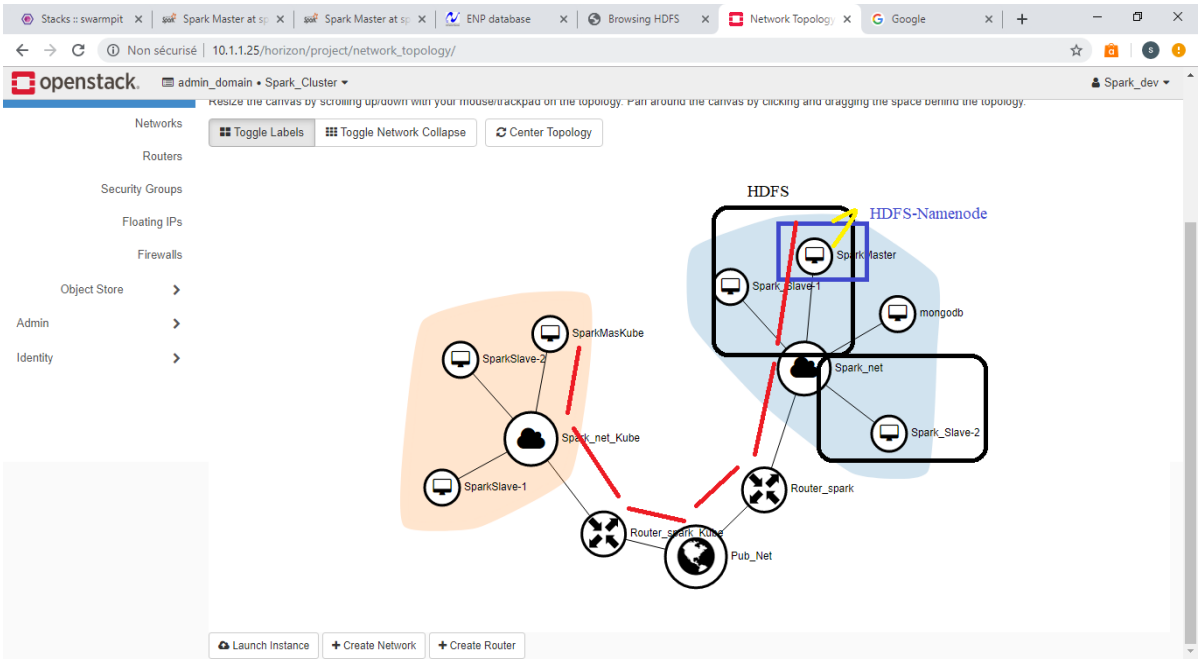


Figure 4.37 : L'architecture réseau montrant la différence du temps d'accès vers l'HDFS du cluster des conteneurs (en rouge) par rapport au cluster en VM (en jaune)

4.6.3.3.1 Comparaison des performances

On estime que le retard causé par cette architecture réseau peut être traduit avec un facteur de retard de 1.375. Ce facteur sera pris en compte lors de l'analyse comparative (que pour les jobs d'importation et d'insertion de données). Après avoir lancé les calculs

dans les deux architectures, et collectionné les contenus des fichiers logs générés lors de l'exécution des deux applications, et les faire passer par le serveur de l'historique, les résultats obtenus ont permis d'avoir quelque indices sur le fonctionnement. Voici quelques-uns :

Facteur	Temps totale d'exécution	Nombre de Cœurs CPU exploitable	Mémoire disponible	Ecart-type du temps retard de l'ordonnanceur de tâches pour la collection des données du modèle d'apprentissage selon le déséquilibre de données	Le temps moyen GC pour la collection des données du modèle d'apprentissage selon le déséquilibre de données
Cluster en VM	20.8 min	2	4 GB	7.15 ms	1.8 s
Cluster en conteneurs	15 min	4	5.6 GB	2.22 ms	5.1 s

Tableau 3 : Comparaison sur quelques paramètres temporels du Spark entre un cluster en conteneurs et un cluster en VMs

Sans avoir entré trop dans les détails comme un benchmark complet, on voit déjà l'avantage des systèmes de calculs en conteneurs par rapport aux systèmes de calculs basés intégralement sur la virtualisation. Pour faire un benchmark complet sur le fonctionnement et la consommation des ressources du système Spark, il faut intégrer d'autres framework de calculs spécialisés dans Spark. On propose dans le site officiel de Spark le framework troisième-partie Ganglia qui doit être compilé avec Spark avant son exploitation. Le framework le plus utilisés dans le domaine du benchmark de l'analyse de données massives est HiBench (Hadoop, Spark, Flink, Strom,..). Ils existent de nombreux articles scientifiques qui abordent cette problématique avec plus d'outils et de ressources, l'un des plus intéressants est intitulé « A Comparative Study of Containers and Virtual Machines

in Big Data Environment » publié par Cornell University en juillet 2018 et accepté par l'institut international de conférence sur le Cloud Computing IEEE. Cet article traite en détails les deux solutions (Conteneurs et VMs) avec des conditions similaires (sauf l'orchestration des conteneurs qui se fait par Docker dans l'article), mais plus démonstratives (plus de machines, plus de ressources physique, plus d'outils de mesure, plus de charges de travail, couvre plus de zones tels que la durée du déploiement et la difficulté du développement etc). On s'intéresse dans notre cadre qu'à estimer les ressources utilisées et le temps nécessaire à la construction de la solution complète. Voici quelques graphes qui donnent une idée sur la différence entre les deux approches au cours du temps, en utilisant plusieurs types d'application comme charges de travail :

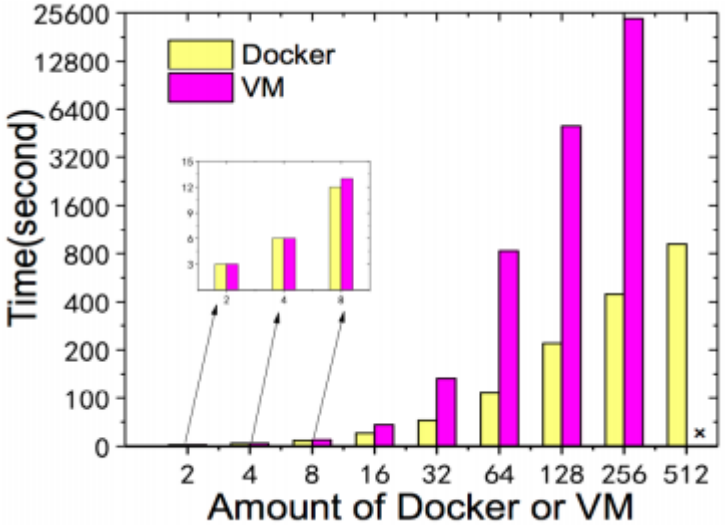


Figure 4.38 : La différence du temps du déploiement d'un cluster en VMs et un cluster en conteneurs Docker en fonction de la taille du cluster

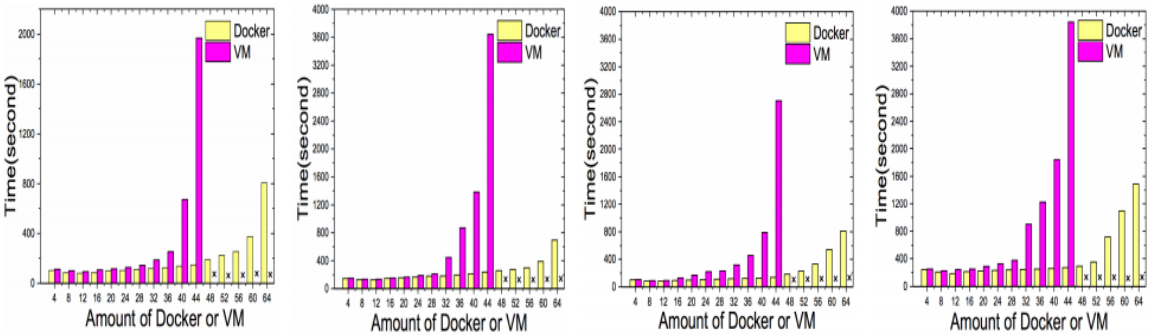


Figure 4.39 : la différence du temps nécessaire d'exécution entre un cluster en VMs et un cluster en conteneurs Docker en fonction de la taille du cluster pour plusieurs charges de travails

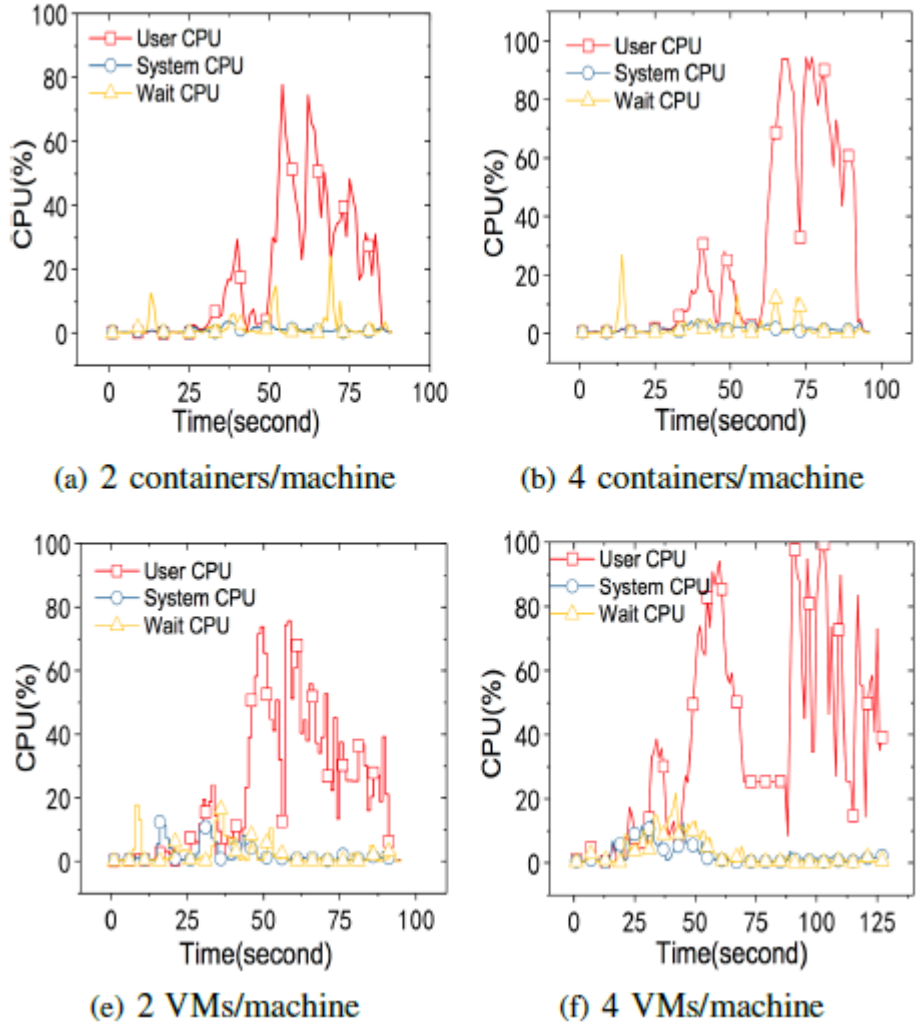


Figure 4.40 : La consommation CPU en % dans un cluster en VMs et un cluster en conteneurs Docker en fonction du temps pour plusieurs tailles du cluster

4.7 Conclusion

Dans ce chapitre, nous avons vu la construction de la globalité de la solution, en partant de l'explication vers le déploiement de chaque brique, ainsi que la comparaison de plusieurs solutions et l'analyse des performances des systèmes exploité, ce qu'il faut retenir :

- En couplant JUJU et MAAS, on peut déployer d'une manière automatique et centralisée les composants d'un Cloud OpenStack, JUJU utilise MAAS pour déployer les services d'OpenStack sur des serveurs physiques
- L'architecture orientée services du OpenStack facilite le développement et l'amélioration des différents services Cloud.
- Les bases de données NoSQL sont non-relationnelles, elles peuvent avoir un schéma dynamique comme elles peuvent être utilisées en relationnel.

- MongoDB est un serveur de bases de données No-SQL, ses principales caractéristiques sont la scalabilité, un temps de stockage optimisé, et l'indexation.
- Un système de fichier distribué est un système de stockage de fichiers partagé sur plusieurs disques physiques, son architecture fait de lui un système de fichier visible sur tout un réseau
- HDFS est un système de fichier distribué de Hadoop, il est scalable, résilient contre les défaillances, temps de stockage optimal
- Pour déployer Spark, on peut le faire sur des machines virtuelles ou sur des conteneurs ou dans un système hybride
- Un conteneur est un environnement d'exécution d'une ou plusieurs applications, sans le noyau du système d'exploitation, il partage le même noyau avec le système d'exploitation hébergeant
- Le déploiement des architectures en conteneurs prend beaucoup moins de temps qu'en machines virtuelles
- Les applications exécutées dans des conteneurs montrent beaucoup plus de performance que celles exécutées dans des machines virtuelles
- La différence de performance vient du fait que les conteneurs partagent le même noyau avec la machine hôte, contrairement aux machines virtuelles qui utilisent des systèmes d'exploitation complets sur une machine hôte.
- Pour créer et gérer des conteneurs, on utilise des orchestrateurs ou des gestionnaires des déploiements
- Docker est l'outil le plus utilisé pour créer des conteneurs, son propre orchestrateur s'appelle Swarm
- Kubernetes est l'un des orchestrateurs les plus puissants qui existent, il sert à déployer des grandes architectures des calculs des systèmes d'information en générale
- Kubernetes utilise Docker pour fournir un service d'orchestration de conteneur rapide, performant, mure, et doté d'une large plage de fonctionnalités

Chapitre 5 :

Mise en œuvre dans le domaine de l'analyse comportementale des conducteurs 4 roues

5 Mise en œuvre dans le domaine de l'analyse comportementale des conducteurs 4 roues

5.1 Introduction

Après avoir parlé de la conception et l'architecture globale de notre solution dans les chapitres précédents, nous allons entamer dans ce chapitre la mise en œuvre d'une application basée sur des données OBD (On-Board diagnostics) dédié à l'étude comportemental des conducteurs de véhicules à quatre roues. Pour le faire, nous allons donner d'abord une vue détaillée sur les données fournis par le OBD, évaluer par la suite la base de donnée choisie, développer une application sur Spark qui exploite l'algorithme K-means avec une approche Big Data, et l'exécuter dans les deux architectures possibles (VM et Conteneur), et essayer d'interprété les résultats obtenus. Nous montrons les résultats des tests par quelques captures d'écran à la fin du chapitre.

5.2 Bases de données OBDs

On-Board Diagnostics en anglais, le système de diagnostic embarqué (OBD) est un système informatisé intégré à tous les véhicules légers et camions légers depuis, comme l'exigent les modifications apportées en 1990 à la Clean Air Act. Les systèmes OBDs sont conçus pour surveiller la performance de certains des principaux composants d'un moteur, y compris ceux qui sont responsables du contrôle des émissions. Ce système est accessible par le connecteur de liaison de données (DLC). C'est le Datalogger d'un véhicule automobile. Il s'agit d'un connecteur à 16 broches qui peut vous indiquer avec quel protocole votre voiture communique, selon les broches qui y sont insérées. Ce système, généralement appelé OBD-II, est une standardisation du système connu OBD-I, qui utilisait des hardwares et des normes d'adaptateurs, aussi protocoles, différentes pour chaque constructeur de voiture. [30]

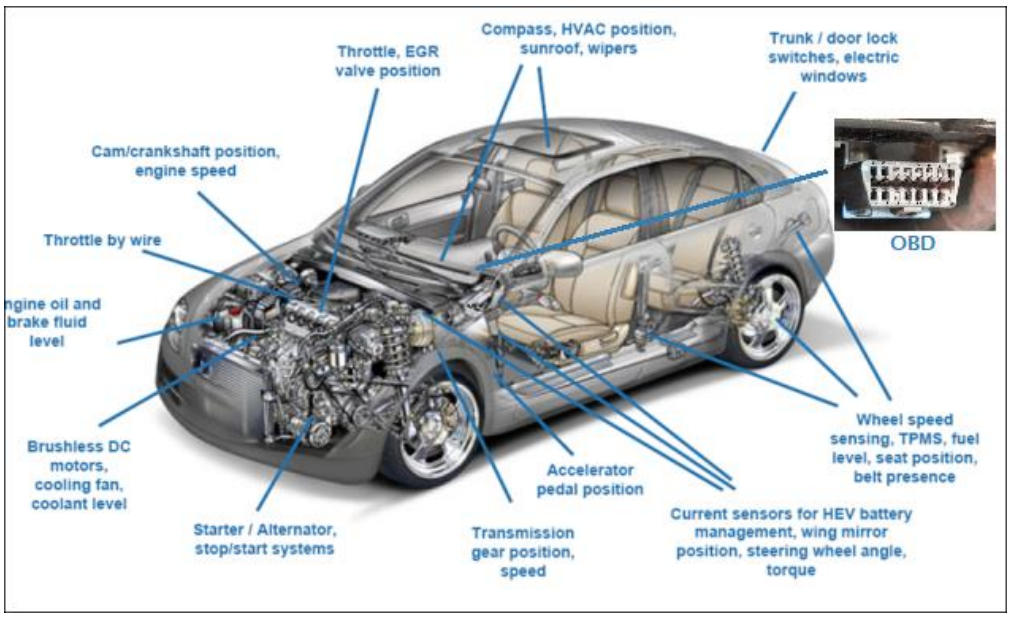


Figure 5.1 : L'ensemble des capteurs dans une voiture à 4 roues utilisés pour l'OBD-II

Dans le contexte de l'internet des objets (IoT), les OBDs présente une large source de donnée la plus volumineuse dans les applications qui visent à interconnecter les véhicules automobiles comme la gestion du trafic ou la lutte contre les accidents routiers, mais de nombreuses applications ont vu le jour avec l'apparition du Big DATA et l'apprentissage automatique dans l'industrie, notamment dans l'analyse de données des OBDs. OBD-II offre un moyen standard d'accéder à de nombreux types de données, y compris :

- Paramètres en temps réel : RPM, vitesse, accélération, niveau de batterie, position de la pédale, avance à l'allumage, débit d'air, débit du carburant, température du liquide de refroidissement, etc.
- Statut du témoin "Check Engine" (Vérifier le moteur)
- État de préparation aux émissions
- Freeze frame : un "snapshot" des paramètres au moment de la survenance d'un trouble.
- Codes de diagnostic des pannes (DTCs).
- Résultats des tests du capteur d'oxygène
- Numéro d'identification du véhicule (NIV)
- Nombre de cycles d'allumage
- Nombre de kilomètres par litre sur un certain régime moteur

5.3 Evaluation de la base de données de l'application

La base de données OBD mise en travail est fournie par une entreprise indienne appelé « **Yuñ** ». Une entreprise spécialiste dans l'internet des objets et l'analyse de donnée en travaillant sur des algorithmes d'apprentissage automatique et avec une expertise dans le domaine de la mécanique et l'automobile. Elle met des bases de données OBDs en public gratuitement pour une philosophie de partage de l'information et la démocratisation de la technologie. L'ensemble de données fourni ici est un échantillon de données récoltées en temps réel. Cet ensemble de données est recueilli sur une période de 4 mois sur environ de 30 véhicules à quatre roues. Ces données OBDs sont recueillies avec une fréquence de 1 Hz (1 enregistrement par seconde) tandis que les données d'accéléromètre sont recueillies à 25 Hz (25 points de données par seconde). Elle est fournie en format CSV sur 29 fichiers séparés, avec une taille totale de 7214695 lignes, ayant 21 dimensions, faisant un volume totale (après leurs insertions dans un serveur MongoDB) de 2.99 GB. Voici le descriptif du contenu de la base de données :

- Device Id : Chaque appareil a un identificateur unique. Device and Car est une cartographie un à un
- Time Stamp : L'horodatage fait référence à l'heure. La valeur correspond aux données recueillies à la seconde même. Format - Année - Mois - Jour Hrs:Min:Sec
- Trip ID : L'ID de déclenchement correspond à 1 déclenchement, le déclenchement commence lorsque le moteur est mis en marche et se termine lorsque le moteur de la voiture est arrêté.
- accData : Se rapporte aux données des capteurs accéléromètre et magnétomètre. Les données sont collectées à partir du dispositif OBD, les valeurs sont en termes de force G. Les données sont sur les axes X, Y, Z où l'axe X est horizontal, l'axe Y est vertical et l'axe Z est la direction de déplacement de la voiture.
- gps_speed : La vitesse en kmph telle que notée par le capteur GPS
- batterie : La tension de la batterie correspond à la tension de la batterie installée dans la voiture, qui fournit l'énergie électrique à un véhicule automobile.
- cTemp : La température du liquide de refroidissement du moteur d'un moteur à combustion interne. La température normale de fonctionnement de la plupart des moteurs est comprise entre 90 et 104 degrés Celsius (195 et 220 degrés Fahrenheit).

- dtc : Nombre de codes d'erreur de diagnostic. Les codes de diagnostic de panne, ou codes de diagnostic de panne, sont utilisés par les constructeurs automobiles pour diagnostiquer les problèmes liés au véhicule.
- eLoad : La charge du moteur mesure la quantité d'air (et de carburant) que le moteur aspire et compare ensuite cette valeur au maximum théorique.
- iat : Le capteur de température de l'air d'admission (IAT) a été utilisé comme signal d'entrée de l'unité de commande du moteur (ECU).il est considéré comme exigence pour calculer le volume de masse d'air pour la charge d'air entrant. Il s'agit d'aider à déterminer les besoins en carburant du moteur en fonction de la température de l'air de fonctionnement.
- imap : Le capteur de pression absolue du collecteur (capteur MAP) est l'un des capteurs utilisés dans le système de commande électronique d'un moteur à combustion interne. Le capteur de pression absolue du collecteur fournit des informations instantanées sur la pression du collecteur à l'unité de commande électronique (ECU) du moteur. Les données sont utilisées pour calculer la densité de l'air et déterminer le débit massique d'air du moteur, qui à son tour détermine le dosage de carburant nécessaire pour une combustion optimale (voir stœchiométrie) et influence l'avance ou le retard de l'allumage.
- kpl : KMPL est le kilométrage en kilomètres par litre. Il s'agit d'une mesure dérivée de la vitesse et du rapport débit massique carburant/air. Ce rapport est constant dans le cas des voitures à essence alors qu'il change pour les autres types de carburant. Par conséquent, la valeur KMPL est exacte pour les voitures à essence et contient quelques erreurs dans le cas d'autres types de carburant.
- maf : Un débitmètre massique d'air (MAF) est utilisé pour déterminer le débit massique d'air entrant dans un moteur à combustion interne à injection de carburant. L'information sur la masse d'air est nécessaire pour que l'unité de commande du moteur (ECU) puisse équilibrer et fournir la masse de carburant correcte au moteur.
- rpm : RPM ici signifie Révolutions Par Minute du moteur.
- speed : Les données de vitesse sont collectées à partir d'un dispositif OBD monté dans la voiture (km/h)

- tAdv : L'avance temporisée correspond au nombre de degrés avant le point mort haut (BTDC) que l'étincelle enflammera le mélange air-carburant dans la chambre de combustion pendant la course de compression.
- tPos : Se réfère à la position de la manette de l'air (Throttle position 0-100%)

La base de données contient naturellement pleins d'anomalies et d'erreurs, ainsi que des données manquantes d'une ligne à l'autre, ceci revient à l'incontrôlabilité de l'environnement ou se trouve le véhicule cible (zones non couverte par le réseau cellulaire), aussi à cause des pannes et incertitude des différents capteurs. Il est donc de faire l'évaluation de l'état de la base de données avant de construire le modèle d'apprentissage qui répond à une application donnée. Cette évaluation est qualifiée de valide les donnée non-manquante correspondant à un état mobile du véhicule (les états d'arrêts sont jugé peu utile pour la majorité des applications dans ce domaine), différentes de 255 (valeurs signifiant une erreur au niveau du capteur), et qui ne présentent pas d'anomalies indécidables. Voici les résultats obtenus :

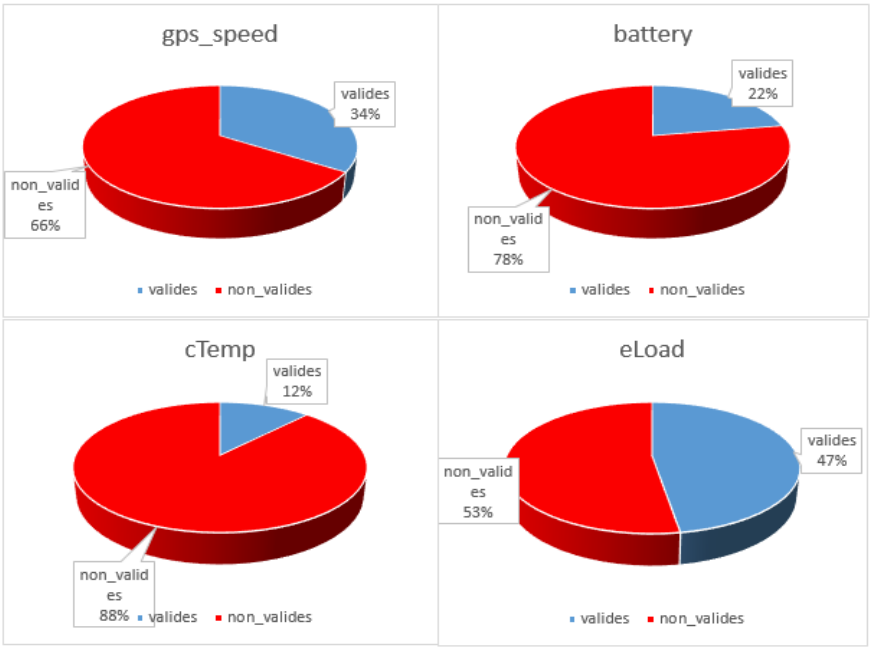


Figure 5.2 : Partie 1 de l'évaluation de la base de données

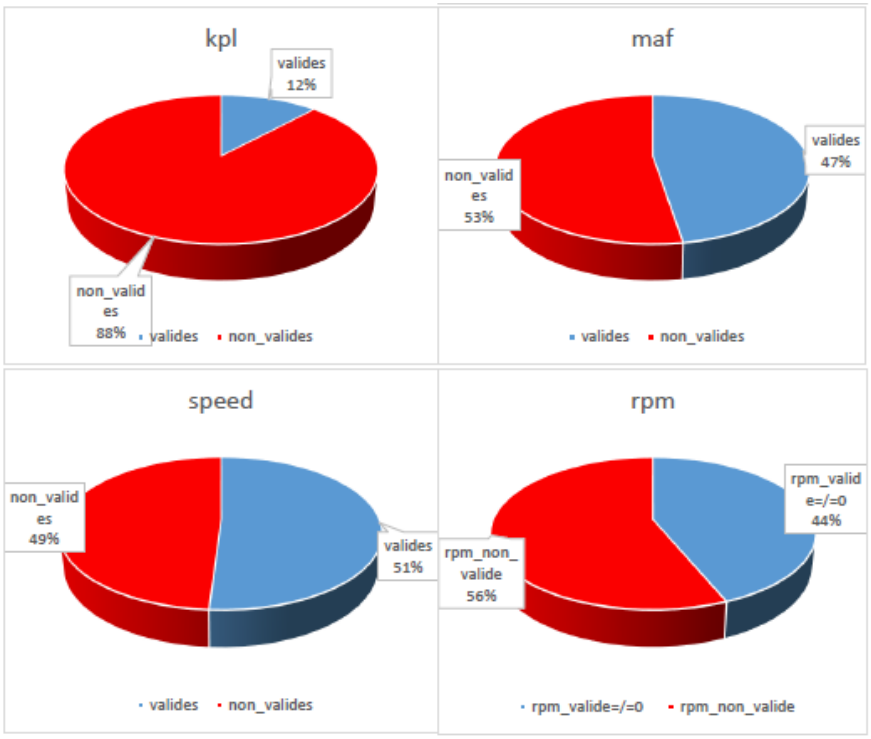


Figure 5.3 : Partie 2 de l'évaluation de la base de données

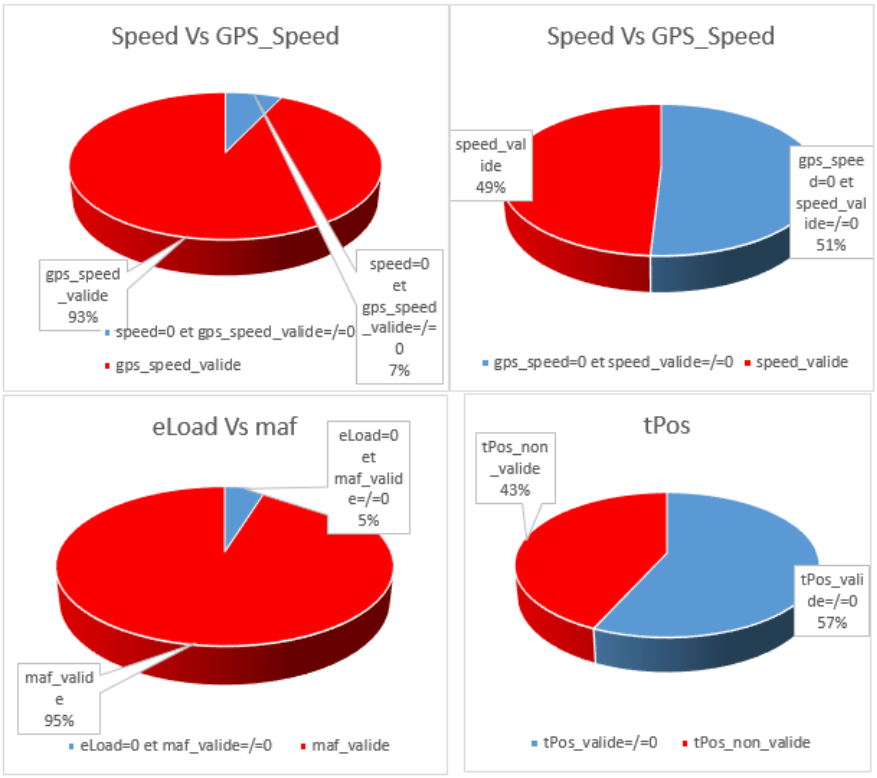


Figure 5.4 : Partie 3 de l'évaluation de la base de données

Les résultats tirés de cette évaluation sont :

- la vitesse fournie par le véhicule est beaucoup plus crédible que celle fournie par le GPS
- Les données pleinement exploitables (c.-à-d. leurs exploitation n'affectera pas la taille de la base de données) sont : eLoad, maf, speed, rpm, tPos.

5.4 Exploitation des données OBD pour l'analyse du comportement de conduite

Le nombre de véhicules augmente considérablement chaque année dans le monde entier. Cette augmentation entraîne de graves problèmes dans la vie de tous les jours, par example, des taux de pollution élevés, qui touchent tout le monde en causant des maladies comme l'asthme, le cancer pulmonaire et les problèmes cardiovasculaires, des embouteillages et des milliers de décès et blessés causés par des accidents chaque année, Le rapport officiel de la DGSN algérienne du premier trimestre 2019 déclare les chiffres suivants :

Désignation	Du 19/02/2019 Au 25/02/2019	Du 26/02/2019 Au 04/03/2019	Ecart	Pourcentage
Nombre d'accidents	309	293	-16	% - 05.17
Nombre de Blessés	373	379	+06	% + 01,60
Nombre de Décès	18	15	-03	% -16.66

LES PRINCIPALES CAUSES DES ACCIDENTS CORPORELS DURANT LA PERIODE ALLANT DU 26/02/2019 AU 04/03/2019 EN ZONES URBAINES.

La cause	Le nombre	Le Pourcentage
Le facteur humain	281	95.90%
Le véhicule	07	02.39%
La route et l'environnement	05	01.71%
Total	293	100%

Figure 5.5 : Rapport officiel de la DGSN algérienne des accidents routiers du premier trimestre 2019

Pour lutter contre ce phénomène destructif de la société, les pertes humaines et matérielles causés par les accidents de la route doivent être évalués en tenant compte de deux aspects différents, à savoir l'utilisation des véhicules par les conducteurs, et leurs comportements. Le premier aspect peut être étudié en analysant les données de détection du moteur d'un véhicule automobile qui peuvent être transmises à des smartphones ou à des tablettes via le système de diagnostic embarqué (OBD-II). D'autre part, le deuxième aspect est lié à la façon dont les conducteurs se comportent lorsqu'ils sont assis derrière le volant, par exemple, la maniabilité du volant et des pédales, la vitesse élevée et le changement de voie ou de route existante. En ce sens, les deux aspects sont devenus populaires dans des domaines tels que : la gestion de flotte et le marché de l'assurance automobile.

Par conséquent, ce travail vise à résoudre le problème des habitudes d'utilisation du conducteur en ne prenant en compte que les multiples capteurs de mesure ODB qui sont situés dans le moteur de la voiture[31].

Plusieurs approches ont été prises à travers des travaux de recherches publiques ou privés. Avoir l'avis d'un expert dans le domaine de l'analyse des données pour l'automobile est une pratique fortement recommandé pour ce genre d'application, afin de déterminer quelles sont les données qui vont construire notre modele, quel type et quelle technique d'algorithme d'apprentissage faut-il adopter pour avoir les meilleurs résultats possibles.

Pour ce travail, on se réfère partiellement à un article scientifique intitulé « Driving Behavior Analysis Based on Vehicle OBD Information and AdaBoost Algorithms » publié en mars 2015 de par IMECS (International MultiConference of Engineers and Computer Scientists). Pour construire son modèle d'apprentissage, l'article repose principalement sur ces quatre informations : la vitesse du véhicule (km/h), le nombre de tours par minute du moteur (r/min), La charge du moteur (le rapport entre la quantité d'air et de carburant que le moteur aspire sur le maximum théorique), et la position de la manette de gaz (throttle) en pourcentage.

On va essayer de trouver des patterns entre ces données le long de l'ensemble de donnée, en les classifiant dans un nombre finis de groupes, afin de pouvoir classer les instances de données dans ces groupes, qu'on va les interpréter de manière à pouvoir déterminer les groupes caractérisant la conduite la plus dangereuse vers la moins dangereuse. Ceci va être établie en utilisant un algorithme de classification non-supervisé, Kmeans.

5.5 Algorithme K-means

Nous commençons par examiner le problème de l'identification de groupes (dites souvent clusters), et de points de données dans un espace multidimensionnel. Supposons que nous ayons un ensemble de données {x1,... xN} consistant en N observations d'une variable euclidienne aléatoire D-dimensionnelle x. Notre but est de partitionner l'ensemble de données en un certain nombre K de clusters, où nous supposerons pour le moment que la valeur de K est donnée[32]. Intuitivement, on pourrait penser qu'un groupe comprend un ensemble de points de données dont les distances entre les points sont faibles par rapport aux distances aux points situés à l'extérieur du groupe. Nous pouvons formaliser cette notion en introduisant d'abord un ensemble de vecteurs D-dimensionnels μ_k, où k = 1,... K, dans lequel μ_k est un prototype associé au $k^{ième}$ cluster. Comme nous le verrons bientôt, nous pouvons penser que le μ_k représente les centres des clusters. Notre but est alors de trouver une affectation de points de données aux clusters, ainsi qu'un ensemble de vecteurs $\{\mu_k\}$, de sorte que la somme des carrés des distances de chaque point de données à son vecteur le plus proche μ_k, soit un minimum.

Il est commode à ce stade de définir une notation pour décrire l'affectation des points de données aux clusters. Pour chaque point de données x_n, nous introduisons un ensemble correspondant de variables indicateurs binaires $r_{nk} \in \{0, 1\}$, où k = 1, ... K décrivant à laquelle des K groupes le point de données x_n est affecté, de sorte que, si le point de données x_n est affecté au groupe k, alors $r_{nk} = 1$, et $r_{nj} = 0$ pour $j \neq k$. Ceci est connu comme le schéma de codage 1-of-K. On peut alors définir une fonction objective, parfois appelée mesure de distorsion, donnée par :

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|x_n - \mu_k\|^2 \quad (5)$$

Qui représente la somme des carrés des distances de chaque point de données à son vecteur assigné μ_k. Notre but est de trouver des valeurs pour $\{r_{nk}\}$ et $\{\mu_k\}$ afin de minimiser J. Nous pouvons le faire par une procédure itérative dans laquelle chaque itération implique deux étapes successives correspondant à des optimisations successives par rapport au r_{nk} et au μ_k. Tout d'abord, nous choisissons quelques valeurs initiales pour le μ_k. Ensuite, dans la première phase, nous minimisons J par rapport au r_{nk}, en gardant le μ_k fixe. Dans la deuxième phase, nous minimisons J par rapport au μ_k, en gardant r_{nk} fixe. Cette optimisation en deux étapes est ensuite répétée jusqu'à la convergence.

Considérons tout d'abord la détermination de la r_{nk}. Puisque J est une fonction linéaire de r_{nk}, cette optimisation peut être réalisée facilement pour donner une solution de forme fermée. Les termes impliquant différents n sont indépendants et nous pouvons donc optimiser pour chacun d'eux n séparément en choisissant r_{nk} comme étant 1 pour la valeur de k qui donne la valeur minimale de $\|x_n - \mu_k\|^2$. En d'autres termes, nous assignons simplement sur le nième point de données à la variable du centre de regroupement le plus proche. Plus formellement, cela peut s'exprimer comme suit :

$$r_{nk} = \begin{cases} 1 \; si \; k = argmin_j \left\| x_n - \mu_j \right\|^2 \\ 0 \; sinon \end{cases} \quad (6)$$

Considérons maintenant l'optimisation du μ_k avec le r_{nk} maintenu fixe. La fonction d'objectif J est une fonction quadratique de μ_k, et elle peut être minimisée en fixant sa dérivée par rapport à μ_k à zéro donnant :

$$2 \sum_{n=1}^{N} r_{nk}(x_n - \mu_k) = 0 \quad (7)$$

Alors que nous pouvons facilement résoudre pour que μ_k donne :

$$\mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}} \quad (8)$$

Le dénominateur dans cette expression est égal au nombre de points attribués au groupe k, et donc ce résultat a une interprétation simple, à savoir réglé μk égal à la moyenne de

tous les points de données xn attribués au groupe k. Pour cette raison, la procédure est connue sous le nom d'algorithme K-moyennes (K-means en anglais).

Voici une démonstration graphique sur une classification non supervisé en utilisant l'algorithme K-means :

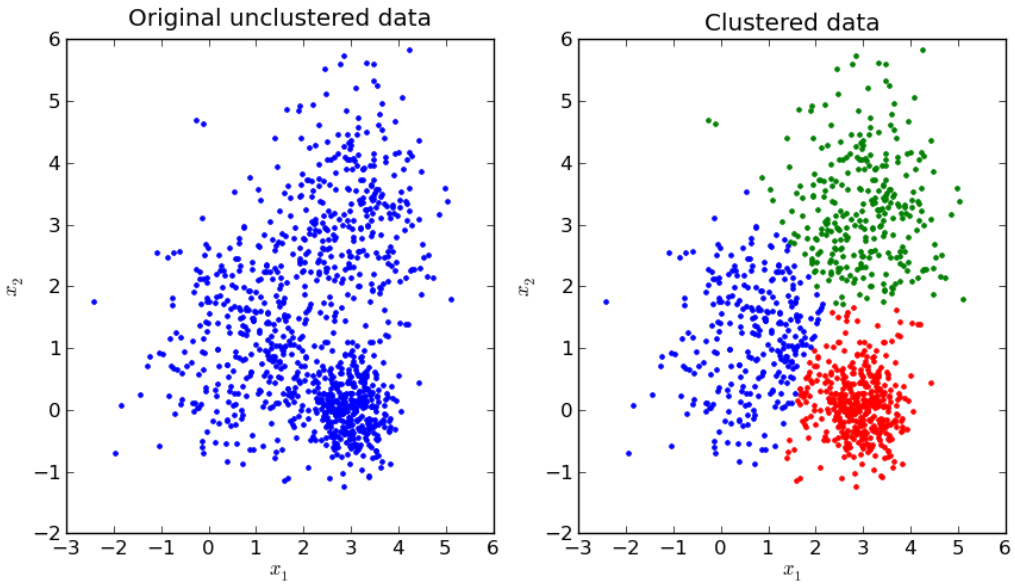


Figure 5.6 : Un jeu de données avant et après la classification K-means

5.6 La méthode d'Elbow

La méthode du coude est une méthode d'interprétation et de validation de la cohérence dans l'analyse des clusters conçue pour aider à trouver le nombre approprié de clusters dans un ensemble de données.

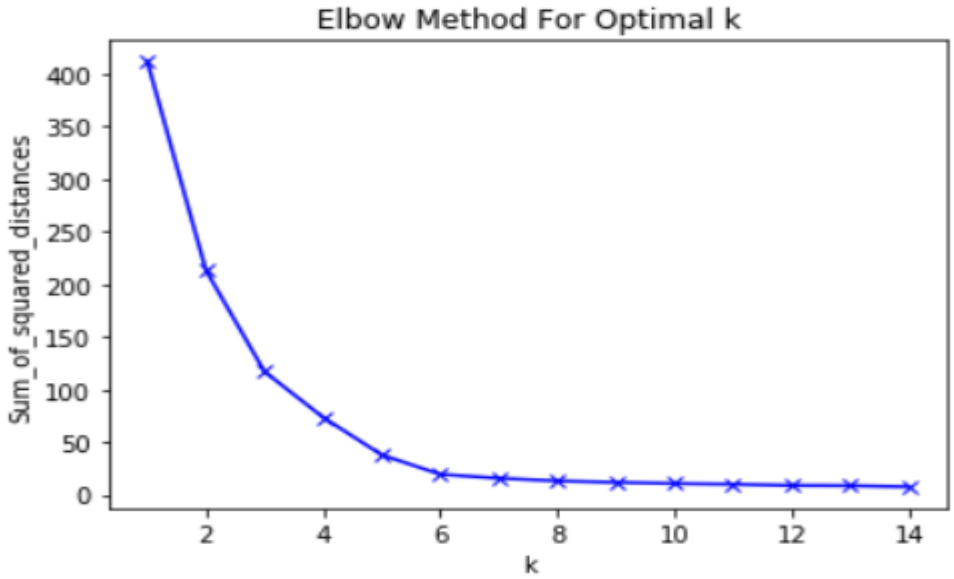


Figure 5.7 : Une courbe décrivant la relation entre la variance obtenue après chaque classification non-supervisé en fonction du nombre de groupes utilisés

Cette méthode examine le pourcentage de variance expliqué en fonction du nombre de clusters : Il faut choisir un certain nombre de clusters pour que l'ajout d'un autre cluster ne donne pas une meilleure modélisation des données. Plus précisément, si l'on trace le pourcentage de variance expliqué par rapport au nombre de clusters, les premiers clusters ajouteront beaucoup d'information (expliquent beaucoup de variance), mais à un certain point le gain marginal diminuera, donnant un angle sur le graphique. Le nombre de clusters est choisi à ce stade, d'où le "critère du coude". Ce "coude" ne peut pas toujours être identifié sans ambiguïté. Le pourcentage de variance expliqué est le rapport de la variance entre les clusters à la variance totale, également connu sous le nom de test F. Une légère variation de cette méthode trace la courbure de l'écart à l'intérieur du groupe.

5.7 Application de l'algorithme K-means dans l'analyse du comportement dangereux de la conduite et interprétation des résultats

Pour cette application, on exploite l'algorithme K-means sur un jeu de donnée quadridimensionnelles (vitesse (km/h), régime moteur (tours/min), charge du moteur (m^3/s), position du throttle (%)), et on applique la version K-means fournie par la librairie spark.mllib avec une initialisation K-means ++ parallèle (noté K-means||). On utilise 80 % du jeu de donnée pour l'apprentissage, et on laisse les 20 % pour la validation, choisis d'une manière aléatoire, pour un nombre donnée de k (en le variant de 1 à 8), et on calcule

la variance cumulée des données pour trouver le nombre optimal de clusters. Voici le graphe obtenu :

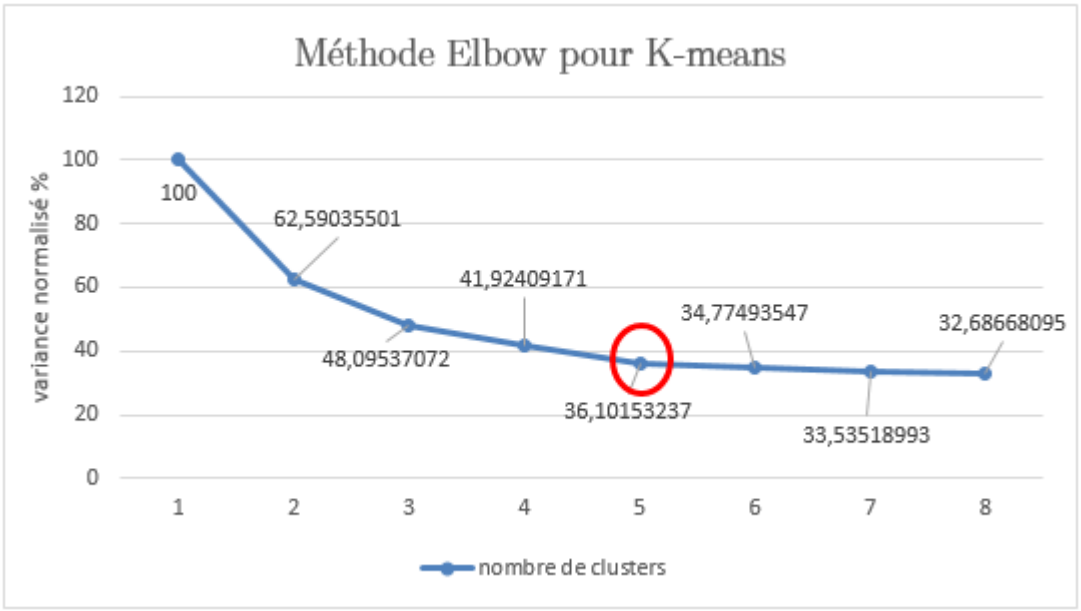


Figure 5.8 : La courbe d'Elbow obtenue lors des tests

On peut conclure d'après le graphe (visuellement) que l'augmentation du nombre de clusters par 3 (de 5 à 8) n'a diminué la variance totale des données autour des centre d'accumulation que de 4 % (36 % jusqu'à 32 %) de la variance maximale, ces clusters n'apporte pas trop d'information au modèle d'apprentissage. En revanche, on trouve qu'en ajoutant seulement un cluster (de 4 à 5 clusters), la variance diminue de presque 6%, ce qui est rentable. Sans sacrifier du temps et de puissance de calcul (nombre de clusters important) sans apporter suffisamment d'information, et sans sacrifier la précision de la classification (nombre de clusters petit) en perdant trop d'information, le nombre de cluster optimale sera 5 clusters avec une variance globale de 36.1 % (celle-ci dépend aussi de l'initialisation de l'algorithme et de l'etendue des valeurs des données).

L'un des inconvénient du K-means et des algorithme de classification non-supervisé est la difficulté d'interprétation du résultat. Dans notre cas, on se base sur la vitesse et le régime moteur(rpm) pour classifier les données correspondantes à des conduites dangereuses ou pas, c-à-d une vitesse haute avec un régime moteur éléver augmente la possibilités qu'une conduite sera dangereuse. Pour cela, on adopte un système de scorisation qui associe à chaque type de conduite un score allant de 0 à 100 qui quantifie sa dangerosité (plus le score est haut, plus la conduite sera dangereuse). Pour la commodité, on ajoute un commentaire qui explicite ce degré dans un langage humain. Voici le plan de classification :

Score	Commentaire
100	« Too aggressive »
75	« Aggressive »
50	« Unsafe »
25	« Safe »
0	« Too safe »

Tableau 4 : Liste des scores et les commentaires correspondants associés à chaque groupe de classification

Après avoir développé l'application sur Spark, et insérer les résultats obtenues dans une base de données, on peut les consulter à travers l'interface graphique du serveur MongoDB abordé dans le chapitre 5. Voici un échantillon des résultats des différents groupes de conduite :

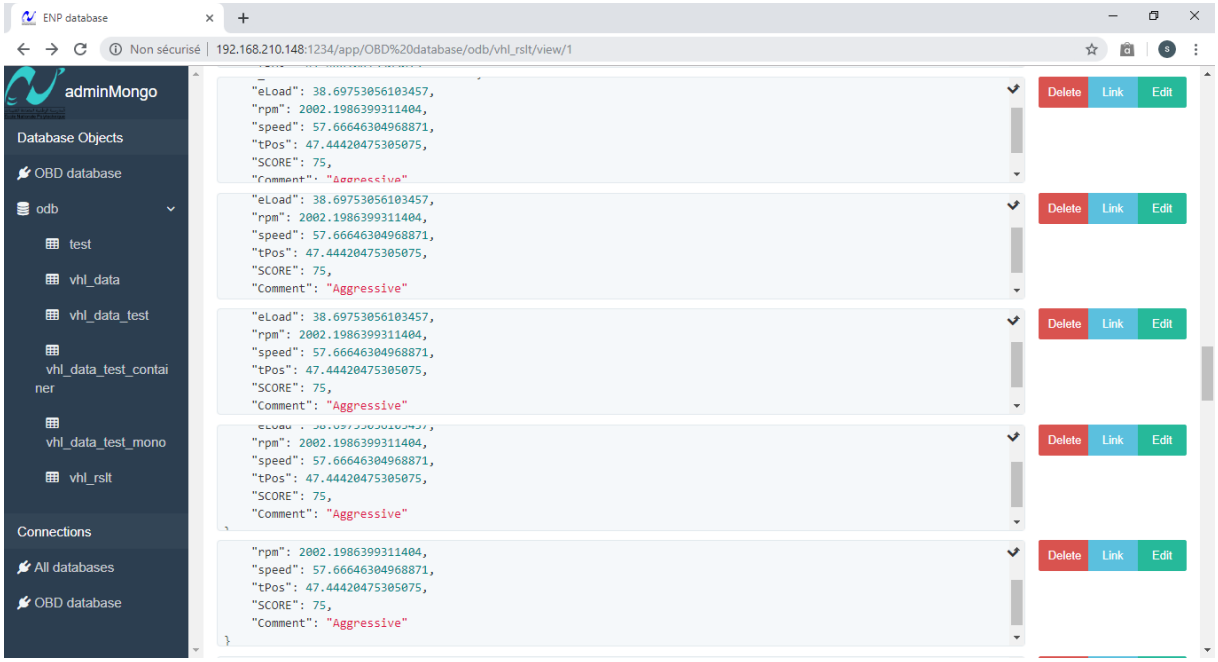


Figure 5.9 : Exemple 1 des résultats obtenus après classification non supervisé

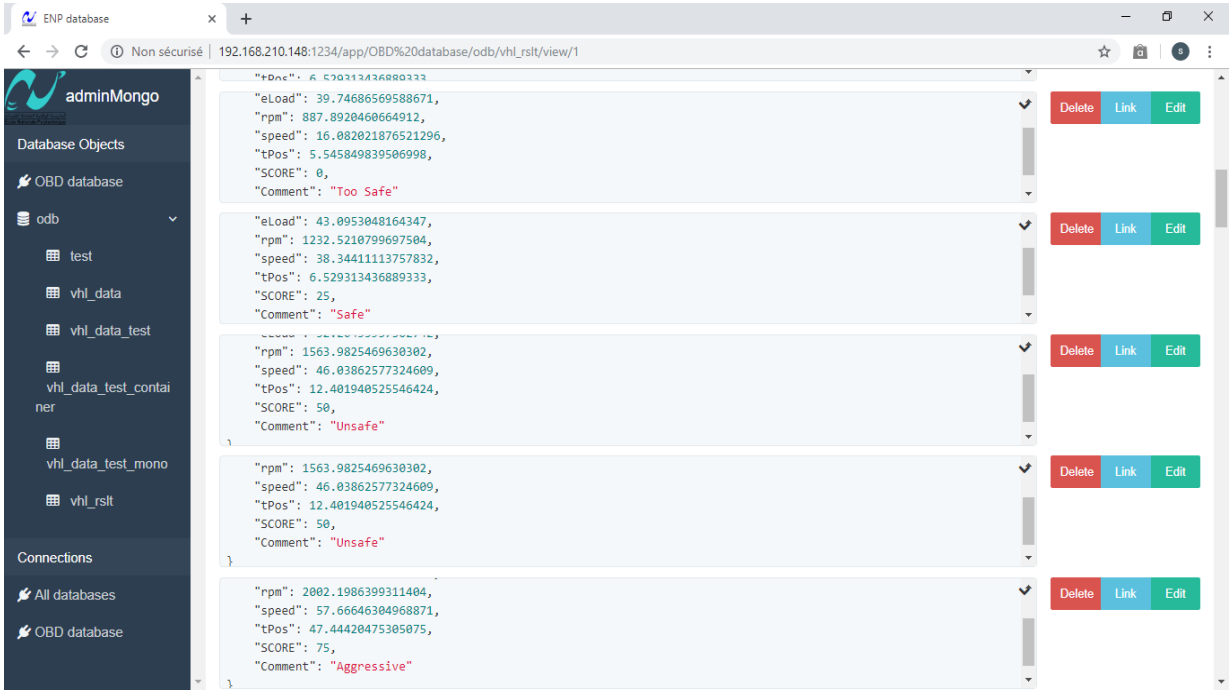


Figure 5.10 : Exemple 2 des résultats obtenus après classification non supervisé

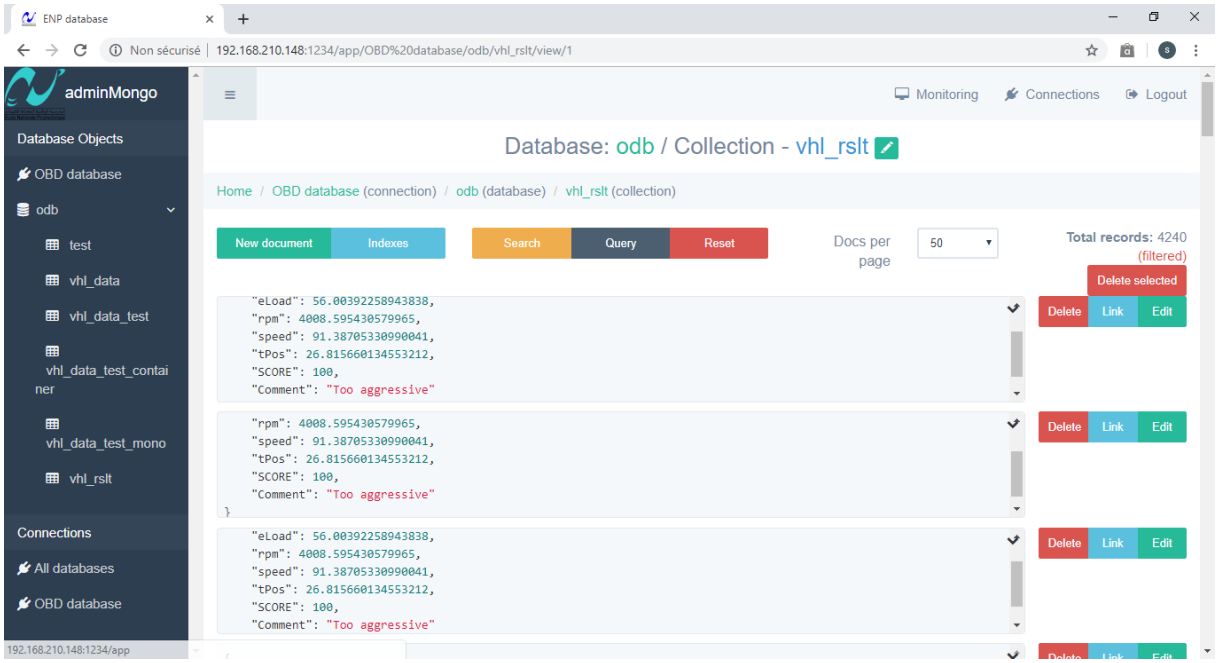


Figure 5.11 : Exemple 3 des résultats obtenus après classification non supervisé

5.8 Conclusion

La lecture de ce chapitre va devoir amener à conclure les résultats suivants :

- Avec les données OBD, nous pouvons faire de l'analyse sur le comportement des conducteurs, et prendre des décisions en fonction des résultats obtenues, ou bien

évaluer et scorer l'agressivité de la conduite. Cette application peut créer de nombreuses valeurs économiques et sociales, comme la gestion des risques au sein des sociétés d'assurance, levier de contrôle puissant et performant des flots de véhicules, une meilleure approche pour la surveillance des routes et la lutte pour une meilleure comportement de conduite.

- L'utilisation de l'algorithme K-means a permis de tirer des patterns sur des données d'une nature ambiguë et difficile à interpréter, et sans avoir besoin d'avoir une information considérable sur ces patterns auparavant.
- En revanche, l'interprétation des patterns extraits qui regroupent l'ensemble des données dans des groupes, est parfois une tâche délicate et déroutante. Il est nécessaire d'avoir l'avis d'un spécialiste du domaine avec un niveau expert dans l'analyse des données pour pouvoir valider l'exactitude du model apprentis.
- En termes de performance de calcul, l'algorithme K-means présente plusieurs avantages dans le domaine du Big Data (scalabilité, complexité algorithmique, ...) dû à sa simplicité comparant à d'autres algorithmes comme GaussianMixture et autres.
- La méthode du coude (Elbow Method), permet de choisir un nombre de groupes k optimale pour un algorithme de classification non supervisé, elle se base sur le calcul de la variance des données dans tous les groupes en fonction du nombre de ces derniers.

Conclusion générale

L'internet des objets fait partie des technologies révolutionnaires que la 4éme révolution industrielle pourra s'en servir pour permettre d'avoir un monde connecté, efficace et plus facile pour vivre. L'IoT s'intègre dans tout un système de technologies qui portent cet aspect révolutionnaire chacun dans son domaine, en allant de l'intelligence artificielle vers le Big Data, jusqu'au Cloud computing ou Edge computing. L'intérêt de ce projet de fin d'études, n'est pas seulement de développer des applications Big Data dans un système IoT, mais aussi, de développer l'architecture logicielle qui va former une plateforme pouvant accueillir de nombreuses applications dans plusieurs domaines. L'infrastructure logicielle pour une solution Big Data-IA-IoT est l'étape fondatrice qui va permettre un développement productif, riche et fonctionnelle au sein de cet écosystème. L'une des applications les plus pertinentes dans le contexte de IoT et en termes de besoin pour notre société en Algérie, est d'essayer de sécuriser nos routes et de mettre fin aux pertes humaines et matérielles que le trafic algérien génère tous les jours. Ce projet propose l'idée de faire de l'analyse pour quantifier l'agressivité du comportement des conduites pour des véhicules à 4 roues. Une application qui pourra servir comme un levier d'ajustement de comportement puissant pour les autorités publiques, ainsi que des systèmes d'assurances, et donc va permettre de sauver des vies, et d'économiser de l'argent.

Le premier chapitre porte le contexte technologique du projet, en abordant des définitions pour les quatre écosystèmes suivants : Internet des objets, l'intelligence artificielle, Big Data et Cloud computing.

Dans les deux chapitres suivants, deux et trois, on a pu voir, expliquer et comprendre, aussi évaluer, les outils de calculs et d'hébergement qui ont été maitrisés est mis en œuvre au sein de l'architecture globale de la solution : Apache Spark dans le contexte de l'analyse Big Data, et OpenStack dans le contexte du Cloud computing.

Le chapitre 4 va aborder le cœur du travail technique du projet, les systèmes nécessaires pour assurer le fonctionnement de la plateforme et la solution proposé. En commençant par le déploiement du Cloud OpenStack qui va héberger l'intégralité de la solution, allant vers le serveur de base de données NoSQL qui va interagir avec le réseau IoT pour stocker les données, vers le système de fichier distribué Hadoop qui va porter la plateforme AI de la solution, un tableau de bord pour faciliter l'interaction du système-Client, et finalement, l'implémentation de l'outil de calcul Spark sur plusieurs architectures (en VMs et en conteneurs) avec plusieurs types d'orchestration, pour essayer d'évaluer la performance et de rendre la solution plus scalable. La conteneurisation des applications améliore considérablement leur performance. La globalité de cette architecture forme la plateforme du développement IoT-Big Data-AI. Elle est cependant fonctionnelle pour des solutions en batch,

et non pas du calcul en temps réel. Ajouter une brique pour le calcul en temps réel sera aussi plus intéressant en termes de capacités.

Le dernier chapitre montre l'exploitation de la solution pour une application dans le domaine de l'analyse de données pour le trafic routier. Il montre aussi le type d'analyse, ainsi que son point d'entrée, tout en évaluant les performances algorithmiques de l'analyse. La difficulté dans un modèle d'apprentissage automatique non-supervisé reste toujours de pouvoir interpréter les résultats. Il est souhaitable en pratique d'avoir de l'expertise dans l'analyse de donnée dans le domaine pour pouvoir valider une application donnée, les calculs mathématiques montrent une certaine stérilité la dessus.

Bibliographie

- [1] Breiman, Leo. 2001. “Random Forests.” *Machine Learning* 45(1): 5–32.
- [2]Cornuéjols, Antoine., and Laurent. Miclet. 2010. *Apprentissage Artificiel*. Eyrolles.
- [3] Breiman, Leo. 1984. *Classification and Regression Trees*. Wadsworth International Group.
- [4] Chen, Sheng, Haibo He, and Edwardo A. Garcia. 2010. “RAMOBoost: Ranked Minority Oversampling in Boosting.” *IEEE Transactions on Neural Networks* 21(10): 1624–42.
- [5] Imed Romdhani Mohammed Riyadh Abdmeziem, Djamel Tandjaoui. 2015 Architecting the internet of things : State of the art.
- [6] Le tendanceur. Les objets connectes : histoire et définitions. [en ligne] http ://www.letendanceur.bzh/les-objets-connectes-histoire-et-de_nitions-de-cette-revolution-technologique/.(consulté le 14/04/2019)
- [7] Barker, J. S. W. a. A. 2013.Undefined By Data: A Servey of Big Data Definitions.
- [8] Laney, D. 2001. 3D Data Management: Controlling Data Volume, Velocity, and Variety. Meta Group.
- [9] Gandomi, A.,& Haider, M. 2015/Beyond the hype: Big Data concepts, methods, and analytics. International Journal of information Management, 35(2), 137-144 doi: https://doi.org/10.1016/j.infomgt.2014.10007 (consulté le 12/03/2019)
- [10] Cloud-Computing_Top-Threats.pdf.2015
- [11] Cloud Computing Top Threats in 2016 The Treacherous 12. Cloud Security Alliance, (February), 1–35. Disponible sur
- [12]https://downloads.cloudsecurityalliance.org/assets/research/top threats/Treacherous-(consulté le 13/02/2019)
- [13] Garivier, A. 2017.Big Data, Machine Learning :qu’est-ce que la science des données ?
- [14] Bastien, L. Machine Learning et Big Data : définition de l’apprentissage automatique, https://www.lebigdata.fr/machine-learning-et-big-data (consulté le 20/03/2019)
- [15] What is Apache Hadoop? http://hadoop.apache.org/ (consulté le 10/02/2019)
- [16] Apache, F. 2018a. https://spark.apache.org (consulté le 10/02/2019)

- [17] Ryza, Uri Larserson, Sean Owen & Johs. Advanced Analytics with Spark (Sandy Wills 2nd Edition.)
- [18] Srinivas Duvvuri, Bikramaditya Singhal 2016. Spark for Data Science (Packt Publishing)
- [19] Alex Liu .Apache Spark Machine Learning Blueprints .
- [20] Andrey Markelov 2016.Certified OpenStack Administrator Study Guide (Apress)
- [21] Openstack :https://www.openstack.org/(consulté le 15/02/2019)
- [22]Michael Solberg, Ben Silverman (2017) ,Openstack for Architects (Packt Publishing)
- [23] Kevin Jackson , Cody Bunch, OpenStack Cloud Computing Cookbook, 2nd Edition_ Over 100 recipes to successfully set up and manage your OpenStack cloud environments with complete coverage of Nova, Swift, Keystone, Glance.
- [24] Dan Radez , 2015 OpenStack Essentials_ Demystify the cloud by building your own private OpenStack cloud (Packt Publishing)
- [25] Juju : https://jaas.ai/(consulté le 20/05/2019)
- [26] Maas : https://docs.maas.io/1.9/en/(consulté le 20/05/2019)
- [27] Oskar Hane (2015)Build Your Own PaaS with Docker. (Packt Publishing)
- [28] . Shrikrishna Holla.(2015).Orchestrating Docker. (Packt Publishing)
- [29] . Deepak Vohra (2016).Kubernetes Microservices with Docker. (Apress)
- [30] Qi Zhang1, Ling Liu2, Calton Pu2, Qiwei Dou3, Liren Wu3, and Wei Zhou3.(2015)A Comparative Study of Containers and Virtual Machines in Big Data Environment
- [31] Cephas Alves da Silveira Barreto1, Joao C. Xavier-Junior, Anne M.P. Canuto and Ivanovitch M.D. da Silva (2016).A Machine Learning Approach Based on Automotive Engine Data Clustering for Driver Usage Profiling Classification.
- [32] Shi-Huang Chen, Jeng-Shyang Pan, and Kaixuan Lu (2018).Driving Behavior Analysis Based on Vehicle OBD Information and AdaBoost Algorithms.
- [33] Arthur, David, and Sergei Vassilvitskii. (2007). "K-Means++: The Advantages of Careful Seeding." *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* 8: 1027–1025.