Ministère de l'Enseignement Supérieur de Recherche Scientifique

Université KasdiMerbah Ouargla

Faculté des Nouvelles Technologies de l'Information et la Communication

Département de l'Informatique et Technologies de l'Information

Mémoire

de master Académique

Domaine : Informatique et Technologies de l'Information

Filière : Informatique

Spécialité : Informatique Industrielle

Présenté par : -Bencheikh Hadjer

-Khennour Aicha

Thème :

Application de sélection de candidats sous Hadoop

Soutenu publiquement le : 04/07/2019

Devant le jury :

Pr.Lallam Fatima Zohraa	Encadreur
M. Mahjoub Bachir	Membre du jury
M$^{lle.}$ ToumiChahrazad	Membre du jury

Année universitaire : 2018/2019

Table des matières

Liste des figures

Résumé

Actuellement au niveau des universités la sélection des candidats de master fait selon des critères appliqués sur leurs données stockées qu'elles sont augmentées de façon rapide. Par ailleurs, Les SGBDRs ne peuvent pas traiter ce volume des données, En effet l'utilisation de Big Data dans la sélection est apparu pour couvre les limites de l'approche traditionnelle dans le stockage et le traitement et des données grâce à ces outils de stockage Hadoop HDFS et de traitement Hadoop MapReduce et ces système de gestion des données.

Dans ce travail nous allons appliquer les outils de Big data Hadoop et un système de gestion de Big Data Hive pour améliorer la sélection des candidats de master dans les universités.

Mots clés: Big Data, Sélection des candidats, gestion de Big Data ,NoSQL, Hadoop, Hive.

Abstract

Currently at the university level the selection of master candidates made according to criteria applied on their stored data that they are increased quickly. Moreover, RDBMSs can't handle this volume of data, Indeed the use of Big Data in the selection appeared to cover the limits of the traditional approach in storage and processing and data through these tools of Hadoop HDFS storage and Hadoop MapReduce processing and these data management system.

In this work we will apply Big data tools like Hadoop and a Big Data management system Hive to improve the selection of master candidates in universities.

Key words: Big Data, Selection of candidates, Big Data Management, NoSQL, Hadoop,Hive.

الملخص:

حاليًا على مستوى الجامعة ، يتم اختيار المرشحين للماستر وفقًا للمعايير المطبقة على بياناتهم المخزنة والتي يتم زيادتها بسرعة. لا يمكن لنظام إدارة قواعد البيانات (RDBMSs) التعامل مع هذا الحجم من البيانات ، بل إن استخدام البيانات الكبيرة في الاختيار يبدو أنه يغطي حدود النهج التقليدي في التخزين والمعالجة والبيانات من خلال هذه الأدوات من Hadoop HDFS التخزين ومعالجة Hadoop MapReduce ونظام إدارة البيانات هذه.

في هذا العمل ، سنطبق Hadoop و Hive Big System Management System لتحسين اختيار المرشحين للماجستير في الجامعات.

الكلمات المفتاحية: البيانات الكبيرة ، اختيار المرشحين ، إدارة البيانات الكبيرة ،Hive,Hadoop,NoSQL.

Introduction générale

De nos jours, des très grandes quantités de données circulent de manière massive, à partir de diverses sources de réseaux sociaux, internet, Google, appareils mobiles, système GPS… etc. Ces données circulent d'une manière rapide et en temps réel et à grande échelle. Elles sont de différentes types et structures, On trouve des textes, des audio, des images et des vidéo, le big data est un terme apparu avec l'augmentation de ces données.

Le défi confronté dans ce contexte les systèmes classiques et les entrepôts de données rencontrent le problème de la dégradation des performances face à une quantité de données aussi importante en termes d'analyse et de traitement.

Le Big data résoudre ce problème et plusieurs domaines utilisent le Big data comme une solution optimal pour la gestion des données parmi ces domaine le domaine de sélection des candidats qu'il profite de Big Data pour améliorer la sélection dans les entreprises.

Plusieurs modèles de programmation ont apparu, et l'une des techniques les plus puissantes de traitement et l'analyse des données est le framework Hadoop, l'une des techniques les plus répandues et les plus utilisées Hadoop est un environnement d'exécution distribuée, performant et scalable, il propose un système de stockage distribué via son système de fichier HDFS (HadoopDistributed File System) et un système d'analyse et de traitement de données basé sur le modèle de programmation MapReduce pour réaliser des traitements parallèles et distribués sur des gros volumes de données.

Notre projet de fin d'étude a pour but d'étudier les techniques du Big Data, Nous nous intéresserons particulièrement aux technologies Hadoop avec ses composantes HDFS et MapReduce. à partir d'une base de données obtenue de l'université.Pour en tirer profit, analyser et traiter Nous allons implémenter tous les processus afin de stocker ces données sur le système de distribution de données HDFS.

En utilisant l'écosystème de Hadoop "Hive", qui donne la structure et l'organisation de ces données pour faciliter le processus de traitement effectué.

Notre mémoire est organisé comme suit :

Le premier chapitre : nous allons présenter une introduction du concept du Big Data avec ses définitions, ses caractéristiques, ses avantages et son importance, ses fonctionnements, ses technologies et les domaines dans lesquels on l'utilise. Ensuite, nous présenterons le Datawarehouse et les Base donné NoSQL, de leurs caractéristiques, leurs avantages et inconvénients et de leurs divers types.

Le deuxième chapitre: est consacré à la présentation du Framework Hadoop, les caractéristiques de ses principaux composants HDFS et de MapReduce, les écosystèmes de Hadoop.

Le troisième chapitre: Nous présenterons la conception du travail, implémentation de l'application et les résultats obtenus.

Enfin, Nous concluons par une conclusion générale.

Chapitre1: Big Data

I.1 Introduction

Actuellement le monde est conforté à une augmentation incroyable du volume des données avec ces divers types structurées et non structurées, résultant de réseaux des capteurs et des systèmes des GPS, et ceux générés par les utilisateurs à travers le partage de leurs données. Le problème qui se pose actuellement est que la croissance des données dépasse la puissance des SGBD traditionnelle, à cause du volume des données, la diversité des données, et le temps du traitement qui ne doit pas dépasser /la durée acceptable par l'utilisateur. En tant que tels, les développeurs ont le défi de développer des technologies capables de traiter d'énormes quantités de divers données en peu de temps, s'appelle BigData.

Dans cette partie, nous étudierons la signification des énormes Big Data et leur importance, les domaines d'utilisation, leurs outils et quelques technologies du Big Data.

I.2 Définition de Big Data

Contrairement aux données traditionnelles, le terme Big Data fait référence à de grands ensembles de données en croissance comprenant des formats hétérogènes: données structurées, non structurées et semi-structurées. Le Big Data a une nature complexe qui nécessite des technologies puissantes et des algorithmes avancés. Ainsi, les outils de Business Intelligence statiques traditionnels ne peuvent plus être efficaces dans le cas d'applications Big Data.[1]

Big data, littérairement les grosses données, est une expression anglophone utilisée pour désigner des ensembles de données qui deviennent tellement volumineux qu'ils en deviennent difficiles à travailler avec des outils classiques de gestion de base de données. Il s'agit donc d'un ensemble de technologies, d'architecture, d'outils et de procédures permettant à une organisation très rapidement de capter, traiter et analyser de larges quantités et contenus hétérogènes et changeants, et d'en extraire les informations pertinentes à un coût accessible.[2]

I.3 Les caractéristiques

La caractérisation de Big Data est généralement faite selon 3 «V», les V de Volume, de Variété et de Vélocité.

✓ **Volume**: Le Big Data vous oblige à traiter d'énormes volumes de données non structurées de faible densité. Il peut s'agir de données de valeur inconnue, comme des flux de données Twitter, des flux de clics sur une page internet ou une application mobile ou d'un appareil équipé d'un capteur. Pour certaines organisations, cela peut correspondre à des dizaines de téraoctets de données. Pour d'autre, il peut s'agir de centaines de pétaoctets.

✓ **vélocité**: La vitesse à laquelle les données sont traitées ou reçues. les données haute vitesse sont transmises directement à la mémoire au lieu d'être écrites sur le disque. Certains produits intelligents accessibles via Internet opèrent en temps réel ou quasi réel et nécessitent une évaluation et une action en temps réel.

✓ **Variété**: Les types de données traditionnels ont été des données ayant une structure et sont stockées dans une base de données relationnelle, mais avec l'apparence du Big Data, les données ne sont pas nécessairement structurées telles que les données texte, audio et vidéo requièrent un prétraitement supplémentaire pour dégager du sens et prendre en charge les métadonnées.[3]

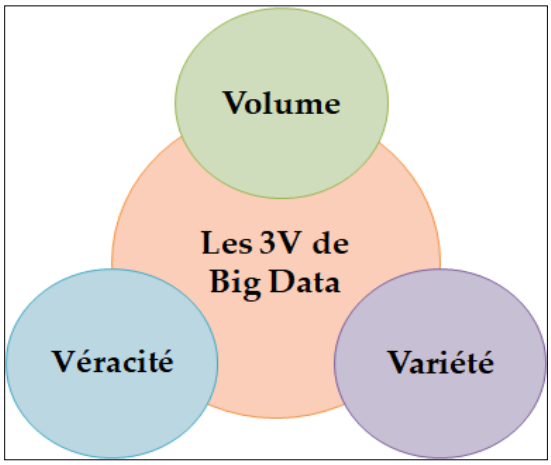


Figure I.1. Les 3V de Big Data.

I.4 Les avantages de Big Data

Plusieurs avantages peuvent être associés Big Data, nous pouvons citer par exemple :

1. **Prendre les bonnes décisions:** toute partie, qu'elle soit gouvernementale ou privée, rentable ou volontaire, peut économiser beaucoup d'argent en exploitant les données. En planifiant des décisions saines et correctes et des plans pour l'avenir.

2. **Augmentation des ventes:** les producteurs et les commerçants peuvent bénéficier d'informations stockées sur des réseaux sociaux tels que Facebook et Twitter pour voir la réactivité de leurs offres, campagnes publicitaires et autres éléments les aidant à planifier leurs produits et à augmenter leurs ventes.

3. **Meilleurs services de santé:** les données des patients enregistrés à l'hôpital, telles que les dossiers pré-médicaux, peuvent être utilisées pour fournir des services plus rapides. Et mieux pour les patients.

4. **Prévention des maladies:** les médecins peuvent éviter les données du génome humain (trois milliards de caractères contenant des informations humaines) Nombreuses maladies et traitement des maladies incurables.

5. **Réduire le coût de la publicité:** toute partie peut envoyer aux clients des publicités personnalisées en fonction de leurs intérêts, directement via le système appelé(Microtargeting).[4]

I. 5 Gestion de Big Data

Gestion de Big Data ou bien Big Data Management est une nouvelle discipline dans laquelle les techniques, outils et plates-formes de gestion de données y compris le stockage, le prétraitement, le traitement et la sécurité peuvent être appliqués.[5]

Le rôle de la gestion des données est assure un haut niveau de qualité des données et aide les entreprises à faire face à la quantité des données qui grandit.

a) Stockage des données

Stocker les données on pétaoctets de façon distribuée utilise les services de Cloud, le stockage consiste trois opérations principales(Clusting, Réplication, indexing).

b) Prétraitement

Avant l'analyse de Big Data on a besoin de vérifier la qualité des données et réparer les données au traitement par l'application des étapes suivantes(nettoyage des données, transformation, intégration, transmission, réduction, discrétisation).

c) Traitement

C'est aptitude de traiter un grand volume de données quel que soit le type ou bien la structure et l'emplacement de ces données, cet traitement peut être classification ou prédiction.

d) Sécurité:

Pour sécuriser un grand volume des données, plusieurs algorithmes de sécurité sont apparus pour la confidentialité, intégrité, disponibilité.

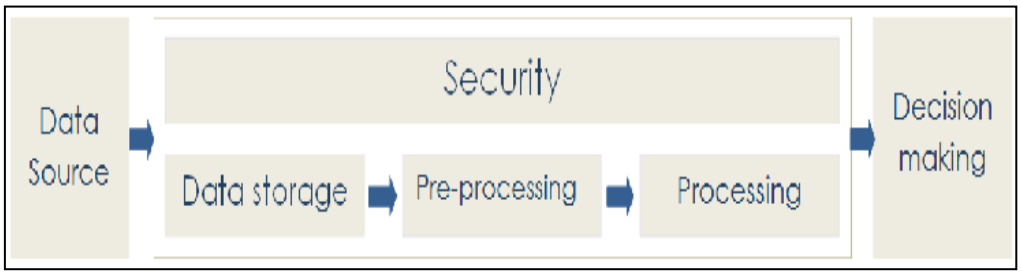


Figure I.2. Les étapes de gestion de Big Data.[6]

I.7 Quelques domaines d'utilisation du Big Data

Citons rapidement quelques domaines d'utilisation du Big Data. Le Big Data trouve sa place dans de nombreux domaines :

- ✓ **Les Banques:** la sanctuarisation de données anciennes dues à des contraintes réglementaires.
- ✓ **La Télécommunication:** l'analyse de l'état du réseau en temps réel .
- ✓ **Les Médias Numériques:** le ciblage publicitaire et l'analyse de sites web .
- ✓ **Les Marchés Financier:** l'analyse des transactions pour la gestion des risques et la gestion des fraudes, ainsi que pour l'analyse des clients.

La deuxième catégorie de secteur est plus hétérogène, les besoins, mais aussi l'utilisation qui est faite du Big Data, peuvent être très différents. On y trouve :

- ✓ **Les Services Publics**: l'analyse des compteurs (gaz, électricité, etc.) et la gestion des équipements.
- ✓ **Le Marketing**: le ciblage publicitaire et l'analyse de tendance.
- ✓ **La Santé**: l'analyse des dossiers médicaux et l'analyse génomique.[7]

I.8 Fonctionnement du Big Data

Le Big Data offre de nouvelles perspectives, qui ouvrent de nouvelles opportunités et favorisent de nouveaux Business Model. Son adoption implique trois actions principales :

Intégrer:

Le Big Data rassemble des données provenant de sources et d'applications disparates. Les mécanismes d'intégration des données classiques, comme ETL (extraire, transformer et charger) ne sont généralement pas à la hauteur. Pour analyser des jeux de Big Data à l'échelle de téraoctets, voire de pétaoctets, il est nécessaire d'adopter de nouvelles stratégies et technologies.

Lors de la phase d'intégration, vous devez importer les données, les traiter et vous assurer qu'elles sont formatées et disponibles sous une forme que vos analystes peuvent exploiter.

Gérer:

Le Big Data nécessite du stockage. Votre solution de stockage peut se trouver dans le cloud, sur site, ou les deux à la fois. Vous pouvez stocker vos données sous la forme de votre choix et imposer à ces jeux de données vos exigences de traitement, ainsi que les moteurs de traitement nécessaires, à la demande. Nombreux sont ceux qui choisissent leur solution de stockage en fonction de l'endroit où sont hébergées leurs données. Le cloud est de plus en plus adopté, car il prend en charge vos besoins informatiques actuels et laisse la possibilité d'augmenter les ressources en fonction des besoins.

Analyser:

Votre investissement dans le Big Data porte ses fruits dès lors que vous êtes en mesure d'analyser vos données et d'agir à partir de l'analyse. Forgez-vous un nouveau point de vue grâce à une analyse visuelle de vos divers jeux de données. Explorez davantage les données afin de faire de nouvelles découvertes. Partagez vos conclusions avec d'autres utilisateurs. Créez des modèles de données avec l'apprentissage automatique et l'intelligence artificielle. Exploitez vos données.[3]

I.9 Les Technologies de Big Data

Nous avons cité quelques technologies utilise le Big Data:

HADOOP : Hadoop un framework mis au point par la Apache Software Fundation afin de mieux généraliser l'usage du stockage et traitement massivement parallèle de MapReduce et de Google File System. Bien entendu, Hadoop possède ses limites. Quoi qu'il en soit, c'est une solution de Big Data très largement utilisée pour effectuer des analyses sur de très grands nombres de données.

Bases NoSQL: Les bases de données relationnelles ont une philosophie d'organisation des données bien spécifiques, avec notamment le langage d'interrogation SQL, le principe d'intégrité des transactions (ACID), et les lois de normalisation. Bien utiles pour gérer les données qualifiées de l'entreprise, elles ne sont pas du tout adaptées au stockage de très grandes dimensions et au traitement ultra rapide. Les bases NoSQL autorisent la redondance pour mieux servir les besoins en matière de flexibilité, de tolérance aux pannes et d'évolutivité.

Stockage "In-Memory" : Pour des analyses encore plus rapide, les traitements directement en mémoire sont une solution. Une technologie bien qu'encore trop coûteuse il est vrai pour être généralisée.[8]

I.10 Big Data et Data warehouse

Dans le passé, les entrepôts de données étaient des structures structurées stockant les informations dans une format normalisé et liées aux systèmes de leurs processus qui gèrent ces données depuis le stockage, le traitement, l'accès et la mise à jour. Ces entrepôts ensuite sont appelées bases de données et ont utilisé la structure relationnelle pour stocker ces informations et ont adopté les systèmes de gestion de bases de données relationnelles SGBDR. Ces règles et règlements sont apparus au

début des années 90.Avec l'augmentation des données stockées et la nécessité pour les entreprises de bénéficier de ces données. Au début des années 80, un système dédié à la prise de décisions pour l'entreprise à partir ses entrepôts de données et le terme Data Warehouse est apparu.

Data Warehouse est une base de données (données structurées) regroupant une partie ou l'ensemble des données fonctionnelles d'une entreprise. Il entre dans le cadre de l'informatique décisionnelle; son but est de fournir un ensemble de données servant de référence unique, utilisées pour la prise de décisions dans l'entreprise par les baies de statistiques et de rapports réalisés via des outils de Reporting. D'un point de vue technique, il sert surtout à 'délester' les bases de données opérationnelles des requêtes pouvant nuire à leurs performances. [9]

Les entreprises traditionnellement utilisent ses entrepôts de données pour la gestion des données relationnelles et structurée, donc il suffit d'utiliser les systèmes de gestion des bases de données relationnelles SGBDRs pour manipuler ce type de données.

Cependant, avec l'avènement du Big Data, le défi pour les entrepôts de données est de réfléchir à une approche complémentaire avec le Big Data, on pourrait concevoir un modèle hybride. Dans ce modèle les restes de données optimisées opérationnelles très structurées seront stockées et analysées dans l'entrepôt de données, tandis que les données qui sont fortement distribuées et non structurées seront contrôlées par Big Data (Hadoop ou NoSQL). [9][10]

On peut donc interfacer Big Data avec le DataWarehouse(DW), effectivement les données non structurées provenant de différentes sources peuvent être regroupées dans un HDFS avant d'être transformées et chargées à l'aide d'outils spécifiques dans le DataWarehouse et les outils traditionnels de BI. [10][11]

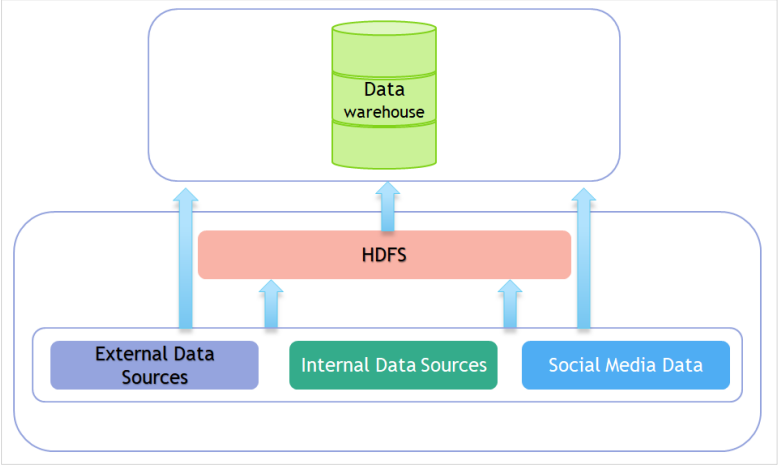


Figure I.3. Liens de Big data et DataWarehouse.

I.11 La Scalabilité

A ce moment les bases de données relationnelles répond aux besoins des utilisateurs de façon satisfaite, mais il y a un problème pour les grandes entreprises de web comme Google, Facebook et Amazon, la scalabilité est la meilleur solution [12]

I.11.1 Définition de scalabilité

La « scalabilité » est le terme utilisé pour définir l'aptitude d'un système à maintenir un même niveau de performance face à l'augmentation de charge ou de volumétrie de données, par augmentation des ressources matérielles.

Il y a deux façons de rendre un système extensible : la «scalabilité» horizontale (externe) ainsi que la «scalabilité» verticale (interne).

I.11.2 La scalabilité verticale

Le principe de la «scalabilité» verticale consiste à modifier les ressources d'un seul serveur, comme par exemple le remplacement du CPU par un modèle plus puissant ou par l'augmentation de la mémoire RAM. C'est ce que l'on appelle la croissance interne. On a une machine et on l'a fait évoluer dans le temps, mais les ressources reste toujours limitées et très coûteux. Et on peut résumer les limites de ce type dans ce qui suit:

- ✓ Aucune tolérance à la panne.
- ✓ Achat d'une machine coûteuse.

- ✓ Le système est limité dans la flexibilité.
- ✓ Impossibilité de faire des mises à jour sans interrompre le système.

Pour résoudre le problème des ressources on ajoute des serveurs interconnectés chaque serveur a son ressources, cette solution s'appelle la scalabilité horizontale.[11]

I.11.3 La scalabilité horizontale

Le principe de la « scalabilité » horizontale consiste à simplement rajouter des serveurs en parallèle, c'est la raison pour laquelle on parle de croissance externe. On part d'un serveur basique et on rajoute des nouveaux serveurs identiques afin de répondre à l'augmentation de la charge.

Cette solution ne donne pas une forte performance avec les SGBDRs dont les SGBDRs ne permettent pas de traiter des données distribuées en parallèle, pour ce raison on a besoin de changer la manière de traitement des données.[12]

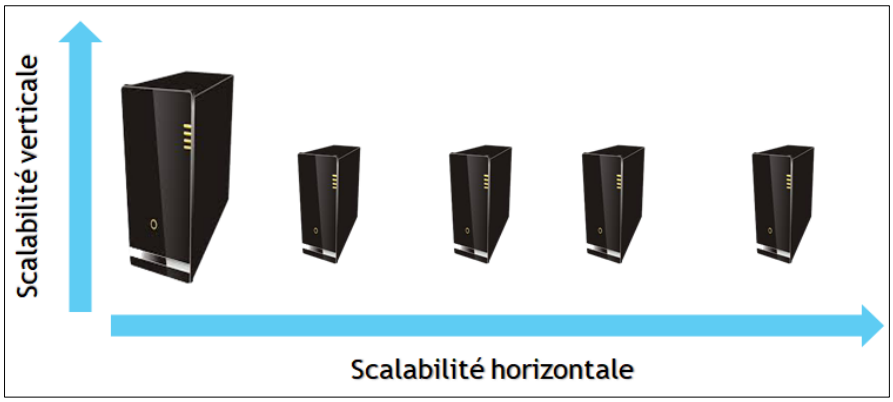


Figure I.4. la scalabilité horizontale et la scalabilité verticale.

I.12 Les bases de données NoSQL

NoSQL est un type de base de données, c'est une manière de stockage et de récupérer des données de façon rapide, un peu comme une base de données relationnelle, sauf qu'il n'est pas basé sur des relations mathématiques entre les tables comme dans une base de données relationnelle traditionnelle.

Le terme NoSQL est un terme apparu récemment dans le monde de traitement des données, il est très utilise dans le stockage et le traitement de Big Data et les

applications web en temps réel, les bases de données NoSQL sont des bases de données extensible permettent le traitement distribué des données.[12]

I.12.1 Les caractéristique des bases NoSQL

Les bases de données NoSQL sont développées pour couvrir les limites des bases de données relationnelle peut résumer les caractéristique de ces bases de données en:

- ✓ Stocker les données dans des documents au lieu des tables.
- ✓ Les structures non stables.
- ✓ N'utilise pas le langage SQL.
- ✓ N'utilise pas la normalisation et support la redondance des données.
- ✓ Traitement des données volumineuses et distribuées en temps réel.
- ✓ La rapidité de réponse à cause de ces requêtes.
- ✓ Permettent la structure Lecteurs/Ecritures.[13]

I.12.2 Les types des bases de données NoSQL

On a plusieurs façons de classification des bases de données NoSQL, mais selon le modèle de données on a 4 types:[14]

- ➢ **Les bases de données orientées colonne**

Dans ce type les données sont stockées se forme des colonnes et non lignes, l'orientation colonne permet l'ajoute des colonnes plus facilement aux table et permet aussi la compression par colonne. Parmi les bases de données orientées colonnes Hbase, Cassandra. Les Bd orientées colonne sont hybrides entre BD relationnels et clés-valeur.

- ✓ Les valeurs sont stockées dans des groupes de plusieurs colonnes mais ordonnés selon les colonnes.
- ✓ Les valeurs sont interrogées par correspondance de clé.
- ✓ Parmi les bases des données orientées colonne : Hbase (Facebook), Vertice.[14]

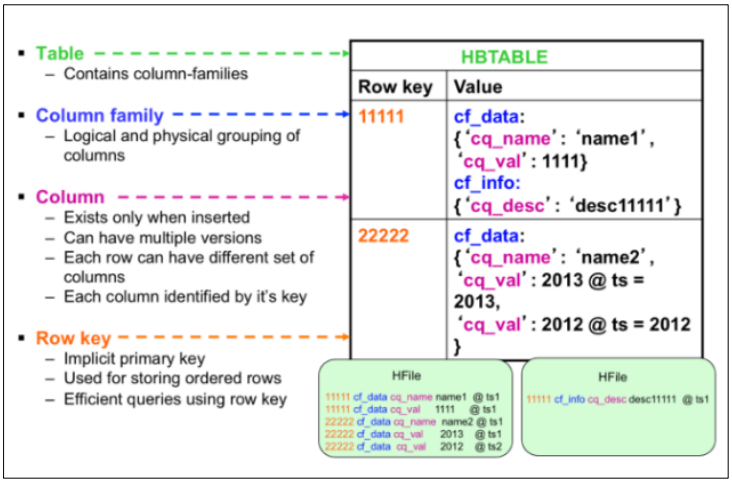


Figure I.5. Les bases de données orientées colonne.

➢ **Les bases de données orientées document**

Le stockage des données dans ce type au format des fichiers JSON ou XML, et puisque Les base de données NoSQL sont des bases non structurées, les fichiers ont des structures différents. Parmi les bases de données orientées documents on a MongoDB et CouchDB sont open source. La base de données orientée documents est évolution de la base de données clé-valeur.

- ✓ La clé n'est plus associée à une valeur sous forme de bloc binaire.
- ✓ Rendre la base de données consciente de la valeur qu'elle stocke.

On peut définir un document comme un ensemble de couples propriété/valeur dont la seul contrainte est respecter le format de représentation.

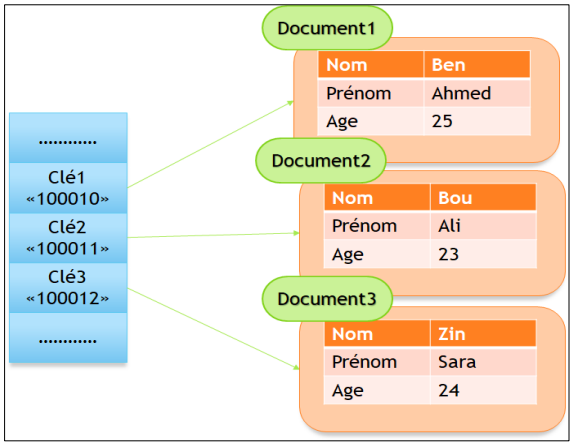


Figure I.6.Des bases des données orientées document.

➢ **Les bases de données orientées graphe**

Dans ce type les données stockées au forme de (Node-Relationship) nœud-association, parmi les bases de données orientées graph Neo4j et FlockDB...

✓ Traiter des données fortement connecté.

✓ Gérer facilement un modèle complexe et flexible.

✓ Offrir des performances exceptionnelles pour les lectures locales par parcours de graphe.

Types de graphe modélisés:

✓ Homogène, hétérogène.

✓ Orienté, non-orienté.

✓ Simple, Hypergraphe.

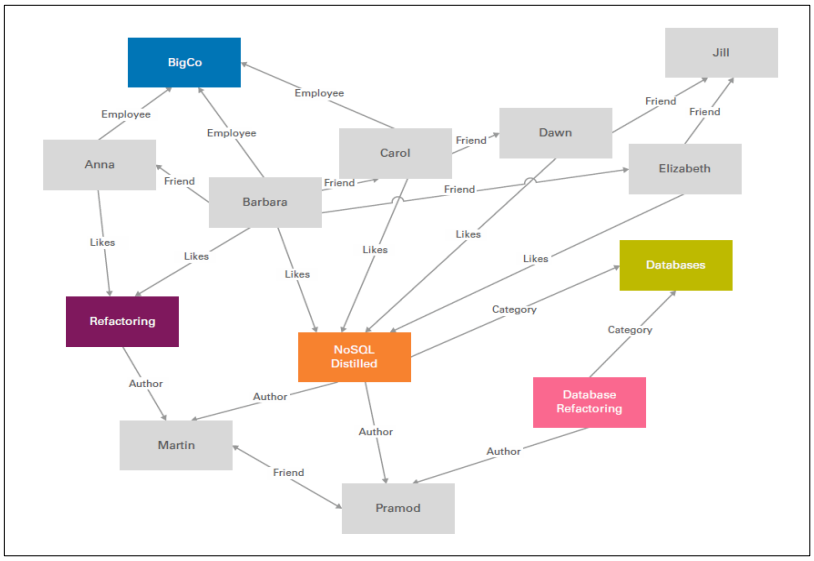


Figure I.7. Des bases des données orientées graph.

➢ **Les bases de données clé-valeur:**

Dans ce type les données sont stockées au format de clé a une valeur dont le clé n'est pas répété et la valeur non structurée (image, vidéo, texte, nombre…), la puissance des base de données clé-valeur est la rapidité de la lecture. Ce type de base de données est caractérisé par :

- ✓ La plus simple et flexibles des BD NoSQL.
- ✓ Associe des clés à des valeurs.
- ✓ Solution aux limitations des BD relationnelles.
- ✓ Les valeurs sont identifiées et accédées via la clé, Ex: Dans les réseaux sociaux a partir d'un utilisateur (clé) on peut accéder aux ses amis (la valeur).
- ✓ Pas de schéma.
- ✓ La valeur stockée peut être: Entier, chaine de caractères, Document (XML, JSON, HTML…), image, vidéo…etc.
- ✓ Parmi les bases de données clé-valeur : DynamoDB (amazon), Azure Table Storage(Microsoft), Riak, Redis.

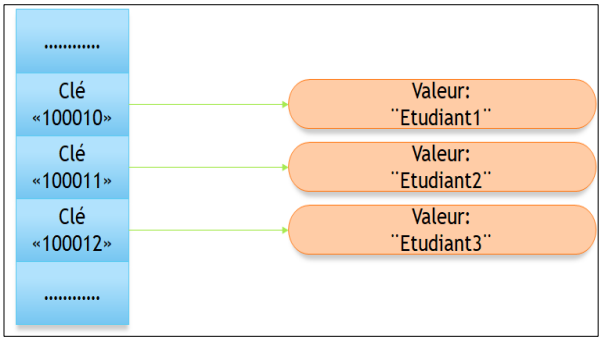


Figure I.8.Des bases des données clé-valeur.

I.12.3 Les avantages de BD NoSQL

- ➢ L'évolutivité se fait de manière horizontale (pour augmenter les performances on ajoute des nouvelles machines).
- ➢ Les données sont distribuées sur plusieurs machines (sharing) de ce fait on évite les goulets d'étranglements lors de la récupération des données (fortes performances de lecture).
- ➢ La représentation des données est notable par l'absence de schéma (schemaless).
- ➢ La majorité des solutions est Open Source. Néanmoins il existe des Support Pro pour répondre aux besoin des entreprises.[15]

I.12.4 Inconvénients de BD NoSQL

- Il n'existe pas de langage d'interrogation standardisé : chaque éditeur a mis en place le sien.

- La mise en œuvre d'un environnement fortement transactionnel (fort besoin d'écriture) où séquencement des écritures est primordial, reste complexe puisque l'architecture est distribuée compliquant l'atomicité et la cohérence des transactions.

- L'écriture de requêtes complexes est difficile à mettre en œuvre.

- Emergence des BD NoSQL :

- Développement des centres des données.

- Nouveaux paradigmes de traitement (MapReduce).

- Nécessité de nouveaux types de BD en 2009.

- Nouvelle génération de BD non relationnelle, distribuées, open source et scalable horizontalement.[16]

I.13 Le Big Data et la sélection de candidats

Le concept sélection ou recrutement n'est pas nouveau mais avec la croissance incroyable des données et l'impotence de chaque donnée pour la décision final, la sélection des candidats utilise les technologies de Big Data pour améliorer les résultats de recrutement ou sélection.

Une nouvelle génération des logiciels de sélection des candidats, plus variées et offrent la possibilité de prise en considération tous les données quelques soit les types, les structures, l'emplacement de ces données est nécessaire.

La manière traditionnelle de gestion des données ne sera pas efficace. Des nouvelles approches sont apparues comme les systèmes NoSQL. Seulement les systèmes posent des problèmes d'adaptation avec les données traditionnelles qui doivent être sujettes à une migration de basse donné traditionnelle à des bases NoSQL. Plusieurs travaux existent dans ces sens. Comme exemple:

Le travail de Boucetta Zouhil, préparé par le Département d'Informatique de l'université de Constantine consiste à fournir des services d'appariement entre les CVs annotés et les offres d'emploi à l'aide des ontologies en fonction des critères

(diplôme, expérience professionnelle, compétence, qualification personnelle, exigence). Son travail basé sur l'explicitation de contenu des documents à partir des éléments ontologiques formant un ontologie qui peut être utilisé par un recruteur pour annoter ces documents qui contiennent les éléments pertinents, ces annotations sont exploitées pour l'appariement entre ces documents et les CVs disponibles. [17]

Dans un autre travail fait par KOUEDI Emmanuel, à l'université de Yaounde I, il donne une solution pour le problème des limites des SGBDs traditionnels dans la cas où les données sont très volumineuses par proposer une approche de migration d'une base de données traditionnelles vers une base NoSQL, la solution montre comment partir d'un MCD classique pour construire son équivalent en BDNOC et faire l'extraction, le formatage et l'injection des données de la base source. Enfin donne un algorithme de migration de données qui a été élaboré et qui couvre les trois opérations majeures plus haut mentionnées.[18]

Dans ce mémoire, nous allons présenter un travail qui consiste en la sélection de candidats pour des spécialités Master bien déterminée. A partir d'une base de données concernant les étudiants en licence, nous allons faire l'extraction des données nécessaires pour la sélection. Ces données vont être stockées sous forme d'une base de données NoSQL. Vu le volume important des données, ces dernières doivent être stockées et traitées d'une manière distribuée.

I.14 Conclusion

Nous avons abordé dans ce premier chapitre les principes des Big Data, ses caractéristiques, son fonctionnement ainsi que les différents domaines dans lesquels elles sont utilisées.

On a aussi parlé de DataWarehouse et de l'augmentation des données des entreprises et les solutions proposées pour résoudre le problème. On a défini les types de l'extension ou la scalabilité utilisée dans les solutions et les raisons de l'apparition des bases de données NoSQL. On a présenté leurs modèles existants et l'utilisation de chaque type ou modèle. Nous avons aussi présenté en fin de chapitre le domaine de la sélection de candidat dans le contexte Big Data. En fin, nous avons introduit notre système de sélection que nous allons concevoir et implémenté par la suite.

Chapitre2: Hadoop

II.1 Introduction

Dans l'approche traditionnelle, les entreprises stockent et traitent le Big Data sur des ordinateurs, c'est-à-dire que les développeurs utilisent les bases de données et leurs ordinateurs personnels pour la manipulation des données, cette approche donne des résultats effectifs avec des petites quantités des données où le nombre des traitements est limités, mais si on a des données volumineuses c'est difficile de traiter dans un temps acceptable.

Google donne la solution pour résoudre ce problème avec l'algorithme MapReduce qui divise une tâche aux plusieurs parties et affecte chaque partie à une machine, puis collecte les résultats de chaque tâche pour donner la solution global grâce à un système de gestion des fichiers distribués s'appelle HDFS, A partir de la solution proposée les développeurs de Google invitent un projet open source appelé Hadoop.

II.2 Historique

Doug Cutting invite Hadoop, le Framework open source destiné à la gestion intensive des données, cherchait une solution quant à la distribution du traitement de Lucene afin de bâtir le moteur d'indexation web Nutch. Il décidait donc de s'inspirer de la publication de Google sur leur système de fichier distribué GFS (Google File System). Premièrement, renommer NDFS, il sera rebaptisé HDFS pour Hadoop Distributed File System.[19]

II.3 Présentation de Hadoop

Hadoop est une plateforme open source de la fondation Apache, ayant une capacité de gérer des données volumineuses, qui sont structurées et non structurées. Elle est conçue pour trouver une solution aux problèmes liés à la volumétrie et la variété des données en les traitants sur des différents serveurs simultanément. Cette architecture va offrir une puissance, et un stockage importants, les données vont être par la suite répliquées et réparties sur les machines du cluster, grâce à un système de réplication de façon à garantir une très haute disponibilité des données en cas de défaillance d'un ou de plusieurs serveurs.[20]

II.4 Caractéristiques de Hadoop

Hadoop a plusieurs caractéristiques, nous citons les suivants:

- **Résilience :** Pour assurer la tolérance aux incidents de Hadoop. Quand un des nœuds tombe en panne les données stockées dans un nœud du cluster sont répliquées dans les autres nœuds du cluster qui disposent toujours d'une copie et sauvegardent des données.

- **Évolutivité :** Hadoop fonctionne dans un environnement distribué. En cas de besoin de extensibilité la configuration peut être facilement étendue en installant d'autres serveurs, et la capacité de stockage peut ainsi atteindre plusieurs péta-octets.

- **Coût modéré :**Hadoop étant un Framework open source n'exigeant aucune licence, les coûts de cette solution sont nettement inférieurs à ceux des bases de données relationnelles classiques. Par ailleurs, l'utilisation d'un matériel standard peu coûteux joue explique le coût modéré de cette solution.

- **Vitesse :** Le système de fichiers distribué de Hadoop et les traitements concurrents de modèle MapReduce permettent d'exécuter les requêtes les plus complexes en temps très court.

- **Diversité des données:** Le HDFS peut stocker différents formats de données structurées, non structurées (par exemple, des vidéos) ou semi-structurées (par exemple, des fichiers XML). Lors du stockage des données, il n'est pas nécessaire de valider celles-ci par rapport à un schéma prédéfini : les données peuvent être téléchargées sous n'importe quel format. Lors de leur récupération, les données sont analysées et utilisées en appliquant le ou les schémas requis. Cette souplesse permet de dériver des connaissances différentes à partir des mêmes données.[10]

II.5 L'architecture Hadoop

Dans ce partie nous allons présente l'architecture de cluster dans Hadoop:

- **JobTracker:**

Le responsable de lancer des tâches distribuées à les autres machines ou bien les esclaves, ensuite, il contrôle l'état de ces esclaves et fait agréger les résultats de calculs.

- **NameNode:**

Est la machine qui fait la réplication et la répartition des données dans le cluster, possède toutes les informations des données et leurs emplacement, elle facilite donc l'accès des client aux fichiers stockés dans le cluster.

- **Secondary NameNode :**

Durant le traitement d'une opération: Chaque couple de minutes, Secondary NameNode va copier les nouveaux informations stockés dans de NameNode, Normalement, le Secondary NameNode doit être assure par une autre machine physique autre que le master afin d'assurer la continuité du fonctionnement du cluster.

Pour les machines esclaves, nous pouvons leur attribuer ces différents rôles :

- **TaskTracker:**

Permettre à l'esclave d'exécuter une tâche MapReduce sur les données qu'elle contient. JobTracker va envoyer les tâches à exécuter aux TaskTrackers. Nous pouvons remarquer qu'avec les solutions Big Data, les instructions sont amenées vers les données et non pas les données sont amenées aux instructions comme les programmes classiques.

- **DataNode:**

C'est la machine qui contient un bloc des données, en effet, les données sont généralement partitionnées et répliquées sur les différents nœuds du cluster pour garantir la disponibilité des données. Cette machine doit périodiquement informer NameNode par un rapport d'état.[21]

II.6 Les composants de Hadoop

II.6.1 HDFS (Hadoop Distributed File System)

HDFS est un système de fichiers aide à la stockage des données structurées ou non sur des machines distribués (cluster). Il s'appuie sur le système de fichier natif de l'OS pour présenter un système de stockage unifié reposant sur un ensemble de disques et de

systèmes de fichiers hétérogènes. Ce système est basée sur la redondance dont une donnée est stockée sur au moins N volumes différents.[10]

Les composants de HDFS

HDFS définit de deux types de nœud:

- **NameNode**: Il se caractérise par :
  - ✓ faire la partition et de la duplication des blocs des données.
  - ✓ Stocker et gérer les informations des blocs(métadonnées).
  - ✓ Sauvegarder la liste des blocs pour chaque fichier (dans le cas de lecture).
  - ✓ Contenir la liste des DataNodes pour chaque bloc (dans le cas de l'écriture).
  - ✓ Tenir les attributs des fichiers (ex : nom, date de création, facteur de réplication).
  - ✓ Logs toute métadonnée et toute transaction sur un support persistant.
  - ✓ Créations/suppressions.[21]
- **DataNode**: Il se caractérise par:
  - ✓ Stocker des blocs de données dans le système de fichier local.
  - ✓ Maintenir des métadonnées sur les blocs possédés.
  - ✓ Heartbeat avec le NameNode : Heartbeat est système permettant sous Linux la mise en clusters de plusieurs serveurs pour effectuer entre eux un processus de tolérance de panne. Le processus Heartbeat se chargera de passer un message-aller vers le NameNode indiquant : son identité, sa capacité totale, son espace utilisé, son espace restant. [21]
- **SecondaryNode**: Il caractérise par:
  - ✓ Télécharger régulièrement les logs sur le NameNode.
  - ✓ Crée une nouvelle image en fusionnant les logs avec l'image HDFS.
  - ✓ Renvoie la nouvelle image au NameNode.[21]

Ecriture dans un fichier ou volume HDFS

Pour écrire un fichier au sein d'HDFS:

- **Etape 1:** On va utiliser la commande principale de gestion de Hadoop: Hadoop, avec l'option fs. Admettons qu'on souhaite stocker un fichier . Ex: data.txt sur HDFS.

- **Etape 2:** Le programme va diviser le fichier en blocs de 64KB (ou autre, selon la configuration) – supposons qu'on ait ici 3 blocs.

- **Etape 3 :** Le NameNode lui indique les DataNodes à contacter.

- **Etape 4 :** Le client contacte directement le DataNode concerné et lui demande de stocker le bloc.

- **Etape 5 :** les DataNodes s'occuperont – en informant le NameNode – de répliquer les données entre eux pour éviter toute perte de données.

- **Etape 6 :** Le cycle se répète pour chaque bloc.[22]

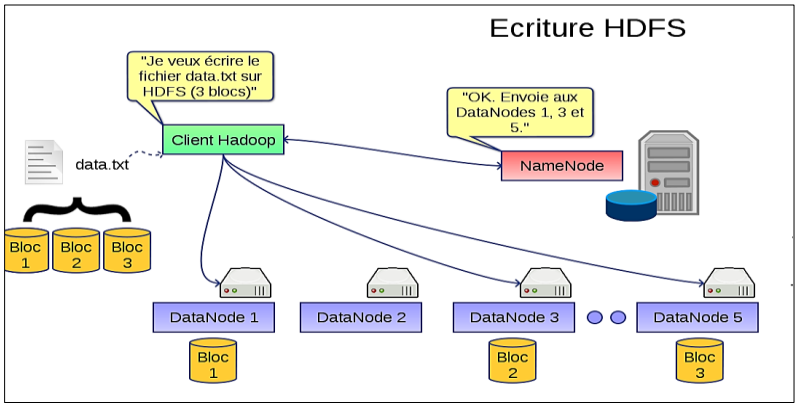


Figure II.1.Processus d'écriture dans un volume ou fichier HDFS [22]

Lecture d'un fichier HDFS

Pour lire un fichier existe dans HDFS, il faut suivre les étapes suivantes :

- **Etape1:** Le client indique au NameNode qu'il souhaite lire le fichier par exemple: data.txt.

- **Etape2:** Le NameNode lui indiquera la taille de fichier (nombre de blocs) ainsi que les différents DataNode hébergeant les n blocs.

- **Etape 3 :** Le client récupère chacun des blocs à un des DataNodes.

➢ **Etape 4 :** En cas d'erreur/non réponse d'un des DataNode, il passe au suivant dans la liste fournie par le NameNode.[21]

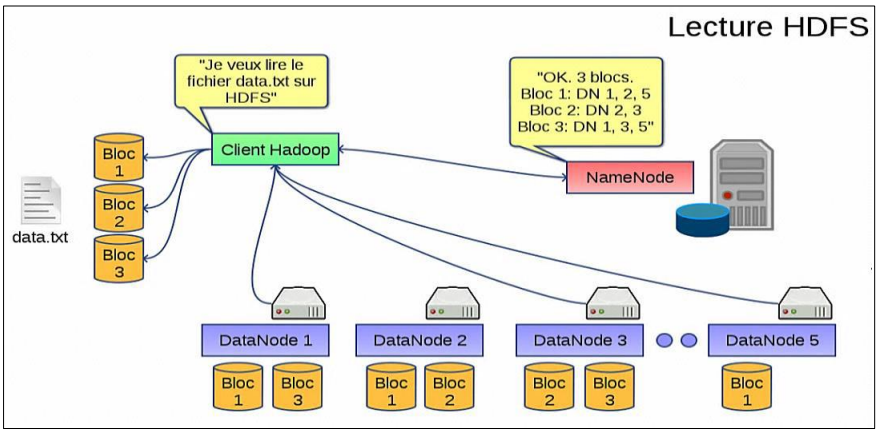


Figure II.2. Lecture d'un fichier HDFS [22]

II.6.2 MapReduce

Le principe de MapReduce est simple, il s'agit de découper une tâche manipulant un gros volume de données en plusieurs tâches traitant chacune un sous-ensemble de ces données. MapReduce a deux étapes Map et Reduce. Dans Map les tâches sont donc dispatchées sur l'ensemble des nœuds. Chaque nœud traite un ensemble des données concernés. Dans Reduce, les résultats sont fusionnées pour former le résultat final du traitement. Nous pouvons aussi distinguer des autres étapes intermédiaires comme suivant:[20]

➢ **Map**

La fonction mapper est pour lire les données stockées et de les découper et générer un autre ensemble de données sous forme de tuples (paire de clé/valeur).

Dans Hadoop, cela se traduit par plusieurs exécutions de la fonction Map, sur les machines esclaves qui contient les données.

➢ **Combiner**

Une étape intermédiaire gérée directement par Hadoop, son rôle est de trier regrouper les paires avec des clés identiques. Elle sert donc d'une part à réduire le résultat à la sortie du mapper et d'autre part à faciliter la vie du Reduce.

➢ **Shuffle**

Une étape intermédiaire aussi permet de regrouper les tuples ayant la même clé dans une seul tuple mais contient la fusion des autres résultats.

- **Reduce**

La fonction Reduce prend la sortie de la phase Shuffle pour agréger les données. Chaque tâche de Reduce produit un fichier de sortie qui sera stocké dans le système de fichiers HDFS.[20]

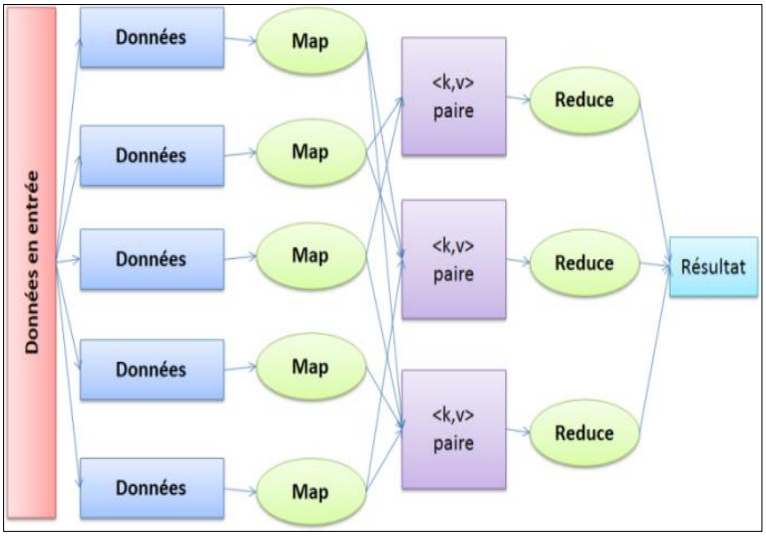


Figure II.3. Exemple des étapes de MapReduce.

L'architecture de MapReduce

Le MapReduce possède une architecture maître-esclave

- **Le maître MapReduce :** le JobTracker.
- **Les esclaves MapReduce :** les TaskTracker.

Le JobTracker

- ✓ Gérer l'ensemble des ressources du système.
- ✓ Recevoir les jobs des clients.
- ✓ Ordonnancer les différentes tâches des jobs.
- ✓ Assigner les tâches aux TaskTrackers.
- ✓ Réaffecter les tâches défaillantes.
- ✓ Sauvegarder des informations sur l'état d'avancement des jobs.

Le TaskTracker

- ✓ Exécute les tâches données par le JobTracker ;
- ✓ Exécution des tâches dans une autre JVM (Child) ;
- ✓ A une capacité en termes de nombres de tâches qu'il peut exécuter ;
- ✓ Heartbeat avec le JobTracker. [21]

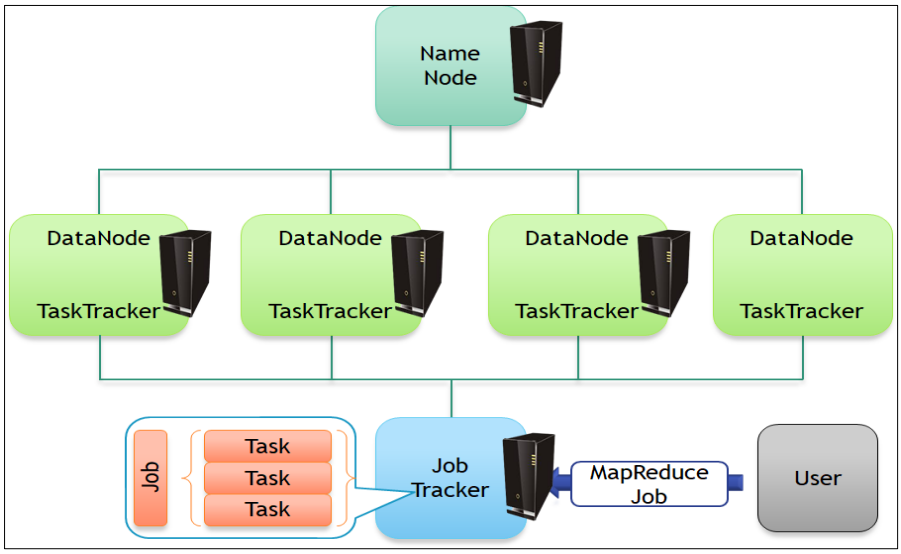


Figure II.4. L'architecture de MapReduce.

II.7 L'écosystème Hadoop

Nous allons identifier dans ce qui suit les composants de l'écosystème Hadoop et ses utilisations

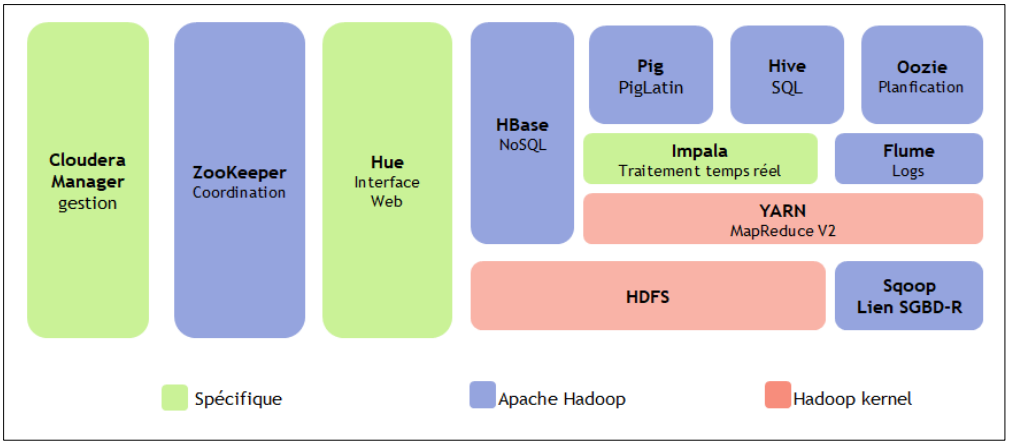


Figure II.5. Outils composant 1e noyau HADOOP.[23]

II.7.1 HDFS

Hadoop Distributed File System, est le système de fichiers Java qui assure le stockage des données structurées ou non structurées sur les différentes machines de notre cluster. Les fichiers sont distribués, scalables, et tolérants aux pannes. Pour y accéder avec un haut débit élevé, nous avons eu recours à MapReduce qui va nous aider à créer, supprimer et copier des données sans les modifier. En effet, nous avons gardé la réplication par défaut des données sur le cluster qui se fait en 3 copies.

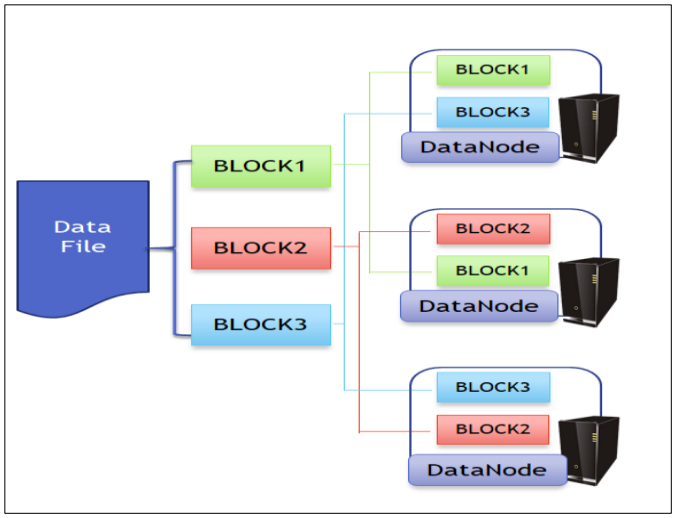


Figure II.6. Le fonctionnement de HDFS.

II.7.2 MapReduce

Initialement crée par Google pour son outil de recherche web. C'est un Framework qui permet la décomposition d'une requête importante en un ensemble de requêtes plus petites qui vont produire chacune un sous ensemble du résultat final : c'est la fonction Map.[24]

II.7.3 YARN (Horton Works)

YARN est un nouveau composant dans l'architecture maître-esclave à deux niveaux par rapport aux architectures précédentes des versions 0.x et 1.x. Le Yarn ressource manager, coordonne l'attribution des ressources de calcul sur le cluster.[11]

II.7.4 Hbase

Il s'agit d'un système en Java de gestion de bases de données non relationnelles distribuées, il assure un stockage structuré pour les grandes tables. En effet, nous pouvons définir Hbase comme une base de données NoSQL, qui est orientée colonnes, utilisée avec HDFS, ce dernier va faciliter la distribution des données de

Hbase sur les différents nœuds du cluster. Enfin, HBase sera capable de gérer les accès aléatoires en lecture et écriture en type temps réel.[23]

II.7.5 Cassandra (Facebook)

Cassandra est une base de données orientée colonnes développée sous l'impulsion de Facebook. Cassandra supporte l'exécution de jobs MapReduce qui peuvent y puiser les données en entrée et y stocker les résultats en retour (ou bien dans un système de fichiers).Cassandra, comparativement à Hbase, est meilleure pour les écritures alors que ce dernier est plus performant pour les lectures.[11]

II.7.6 Hive

Hive est une infrastructure d'entrepôt de données pour Hadoop. Il permet la synthèse, l'interrogation et l'analyse des données grâce à un langage de haut niveau semblable à SQL, qui s'appelle HiveQL. Ce langage nous a permis d'interagir facilement avec un cluster Hadoop. Par conséquent, l'exécution d'une requête HiveQL se traduit par la création et l'exécution d'un job MapReduce.[25]

II.7.7 Pig (Yahoo)

Apportent un modèle de développement de plus haut niveau, et donc beaucoup plus expressif et simple à appréhender, afin de démocratiser l'écriture de traitements MapReduce. Pig se rapproche plus d'un ETL où on part d'un ou plusieurs flux de données que l'on transforme étape par étape jusqu'à atteindre le résultat souhaité. Les différentes étapes de la transformation sont exprimées dans un langage procédural (Pig Latin). Pig est à l'origine un projet Yahoo qui permet le requêtage des données Hadoop à partir d'un langage de script.[11]

II.7.8 Zookeeper

Zookeeper est un logiciel open source de la fondation Apache Software, responsable à gérer la configuration des systèmes distribués. Il assure la distribution des tâches aux composants Hadoop par un seul maître actif en sauvegardant les régions de serveurs, d'où la facilité de les retrouver par les clients. En effet, Zookeeper est un bon coordinateur puisqu'il reçoit les battements de cœurs des régions de serveurs pour identifier l'état des régions de serveurs, ensuite, il veille à maintenir le cluster en bon état.[26]

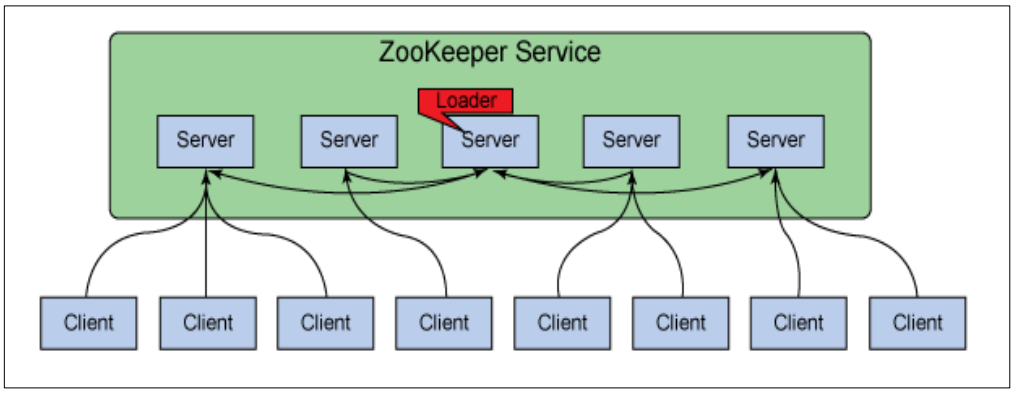


Figure II.6. zokeeper fonctionnement.[26]

II.7.9 Apache Ambari (HortonWorks)

Ambari est un projet d'incubation Apache initié par HortonWorks et destiné à la supervision et à l'administration de clusters Hadoop. C'est un outil web qui propose un tableau de bord, cela permet de visualiser rapidement l'état d'un cluster.[10]

II.7.10 Sqoop

Nous avons eu recours à un logiciel open source de la fondation Apache Software qui s'appelle Sqoop, pour importer les données d'un système de gestion de bases de données relationnelles telles que PostGreSQL dans notre cas, vers un système des fichiers distribués Hadoop (HDFS). En outre, Sqoop est capable à transformer les données dans Hadoop à l'aide de MapReduce pour assurer le fonctionnement en parallèle et la tolérance aux pannes, et à automatiser le processus de l'importation.[27]

II.7.11 Apache Oozie (Yahoo)

Oozie est une solution de workflow (au sens scheduler d'exploitation) utilisée pour gérer et coordonner les tâches de traitement de données à destination de Hadoop.

Oozie s'intègre parfaitement avec l'écosystème Hadoop puisqu'il supporte les types de jobs suivant : MapReduce (Java et Streaming),Pig, Hive,Sqoop.[11]

II.8 conclusion

Dans ce chapitre on a présenté l'outil Hadoop et ses composants principaux HDFS et MapReduce, l'architecture de cluster dans Hadoop, la façon de distribuer les données dans plusieurs machines (DataNode), les machines qui gèrent les métadonnées (NameNode), et la façon du traitement et de la gestion des données, et

tous les concepts qui concernent l'outil théoriquement, pour faciliter l'implémentation et la réalisation dans le prochain chapitre.

Chapitre III: Conception & implémentation

III.1 Introduction

Dans ce chapitre nous allons présenter la solution au problème posé. Nous présentons l'architecture d'un système de sélection des candidats pour continuer leurs études de master dans les spécialités compatibles avec leurs compétences, à partir des données existants déjà dans l'université. Après nous allons expliquer de manière détaillée l'installation de tous les outils utilisés et les étapes de la réalisation de ce projet.

III.2 Motivation et problématique

Le but de notre travail est d'améliorer la sélection des candidats par l'application des techniques de Big Data comme Hadoop pour résoudre le problème de stockage des données massive et faire les traitements distribués et un NoSQL système pour les prétraitement des données existées dans l'université donc on va commencer de répondre à la question comment profite de Big Data pour améliorer la sélection de candidats?

III.3 Présentation de solution

Notre système fonctionne selon le principe dont nous avons parlé précédemment dans le deuxième chapitre. Dans cette section nous présenterons l'architecture de notre système en générale et après nous expliquerons en détaillée le fonctionnement, les composants et les entrées/sorties de notre système.

III.3.1 L'architecture globale

D'abord notre système est un système de sélection des candidats dans le Big Data, qui en général fonctionne selon le principe présenté dans l'illustration suivante.

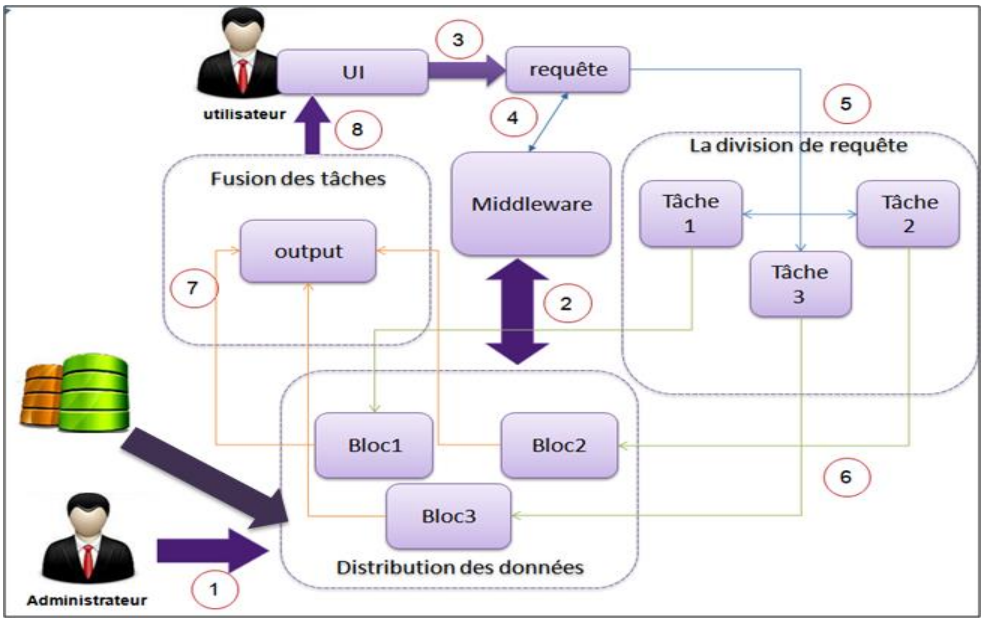


Figure III.1.Architecture globale de notre système

On a proposé une architecture qui permet :

1. Administrateur extrait la base de données NoSQL, partant d'une base de données relationnelle. Ensuite, il introduit ces données dans le système de stockage. Ce dernier partitionne les données introduites en blocs.
2. Le Middleware reçoit des informations sur l'emplacement des blocs.
3. Le client lance une requête à partir de l'interface utilisateur.
4. La requête est envoyée au Middleware. Ce dernier gère le plan d'exécution et renvoie la requête avec le plan.
5. Dans la phase de traitement la requête est divisée à des tâches selon le plan d'exécution.
6. Dans la phase divisionenvoyer chaque tâche à l'emplacement des données dont il a besoin et fait le traitement.
7. Dans la phase fusion complète le traitement par la collection et la fusion des résultats des tâches.
8. Afficher le résultat global à l'utilisateur dans interface utilisateur.

III.3.2 L'architecture détaillée

a) La phase de stockage

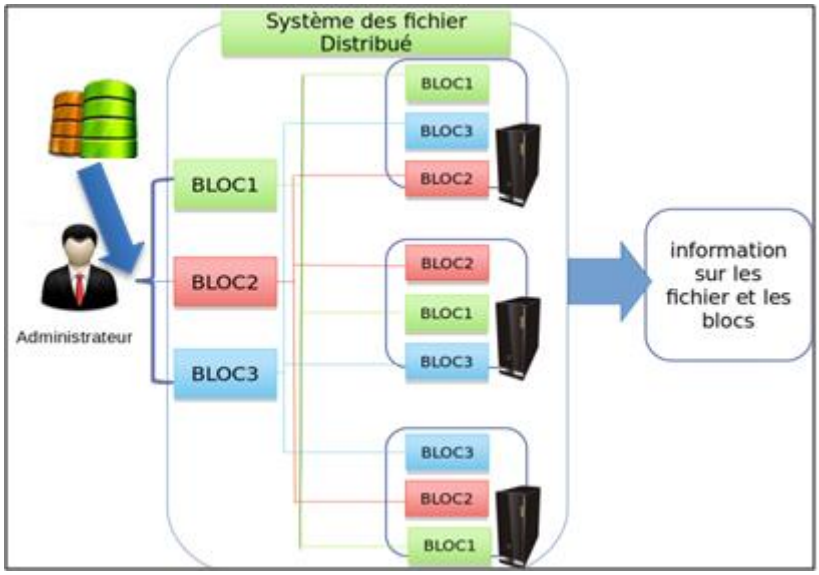


Figure III.2. Architecture de la partie stockage.

➢ **La base de Données**

Nos données sont des données qui concernent les étudiants des départements informatiques et technologie de l'information telles sous forme les fichiers **csv**. Un fichier **csv** est un fichier tableur contenant des données sur chaque linge séparé par un caractère de séparation.

```
Res,TG,IHM,DW,BDW,Mem,Comp,MoySem,MoyeSem,FiliereEtude
8.25,4.33,14.00,7.75,7.00,17.00,11.25,10.35,9.88,MINFF
7.25,7.50,11.00,17.50,15.00,17.50,9.63,12.39,10.36,MINFF
5.75,8.50,10.00,11.50,11.00,17.50,5.75,9.95,11.82,MINFF
10.50,5.75,13.00,17.50,15.00,16.00,8.25,11.40,10.60,MINFF
6.25,4.19,11.00,4.50,6.50,16.50,7.50,11.76,12.82,MINFF
8.00,5.69,13.00,14.00,3.50,15.00,7.50,10.57,11.09,MINFF
10.25,10.38,7.75,15.00,7.00,14.00,14.75,11.77,12.04,MINFF
9.25,7.00,10.50,13.50,10.00,16.75,7.00,10.42,10.25,MINFF
8.00,10.13,10.50,13.50,12.00,18.00,12.38,12.35,10.47,MINFF
5,0,14.00,13,5,17.00,10.13,9.98,10.23,MINFF
7.25,7.75,12.00,13.00,4.00,16.00,8.63,10.11,10.47,MINFF
13.75,7.00,11.00,10.00,13.00,12.00,8.25,6.18,8.29,MINFF
13.75,7.00,11.00,10.00,13.00,12.00,8.25,10.33,9.68,MINFF
8.13,11.25,14.50,6.00,8.00,17.00,7.88,10.44,10.99,MINFF
3.25,2.25,8.25,13.50,15.00,17.00,7.00,11.52,9.58,MINFF
10.00,1.50,12.00,18.75,14.50,12.00,3.75,8.99,11.01,MINFF
6.50,8.00,8.50,15.00,16.00,16.00,8.38,10.08,9.92,MINFF
5.25,7.75,10.50,11.00,7.50,17.00,7.13,11.28,9.31,MINFF
6.25,9.75,5.50,11.00,8.00,16.50,7.50,10.83,10.95,MINFF
4,6,13.00,8,9,18.00,6.00,10.21,10.11,MINFF
9.25,9.50,13.50,16.00,12.00,17.50,9.00,11.46,10.49,MINFI
8.50,12.25,12.50,15.00,9.00,17.00,10.38,11.80,11.43,MINFI
12.75,12.00,15.00,18.00,18.00,17.50,13.75,14.27,15.03,MINFI
6.00,13.25,8.00,16.50,16.00,18.00,12.75,11.81,11.29,MINFI
```

Figure III.3. Structure de notre base de données.

- **Fonctionnement de la partie stockage**

- ✓ Administrateur extraite les données à partir d'une base de données existante.
- ✓ Les données extraites seront entrées dans le système de fichiers distribué.
- ✓ Le système des fichiers va diviser les données sur plusieurs blocs.
- ✓ Ensuite les blocs seront dupliqués plusieurs fois généralement trois fois.
- ✓ Distribuer les blocs dans des différents emplacements dans le système.

Avant de commencer le traitement des données et lancer les requête par l'utilisateur notre système restructure les données stockées dans le système des fichiers distribuées par le **Middelware,** Cette opération reforme les données par une autre structure déterminée par **Middelware** pour faciliter le traitement.

b) La phase de traitement

Dans cette phase nous avons suivi le principe de MapReduce qui est définie dans les chapitres précédents. Le principe est simple. La requête lancée par l'utilisateur et des données concernées ne sont pas nécessairement stockés dans le même emplacement. Donc, la requête sera divisée obligatoirement sur des tâches. MapReduce a deux étapes Map et Reduce pour retourner le résultat final, lesquelles:

- **L'étape Map**: est responsable de lire les données stockées sur lesystème, de les découper et de générer un autre ensemble de données sous forme de tuples (paire de clé/valeur).
- **L'étape reduce**: L'étape reduce prend les sorties de la phase Map. Chaque tâche de Reduce produit un fichier de sortie qui sera stocké dans le système fichiers HDFS

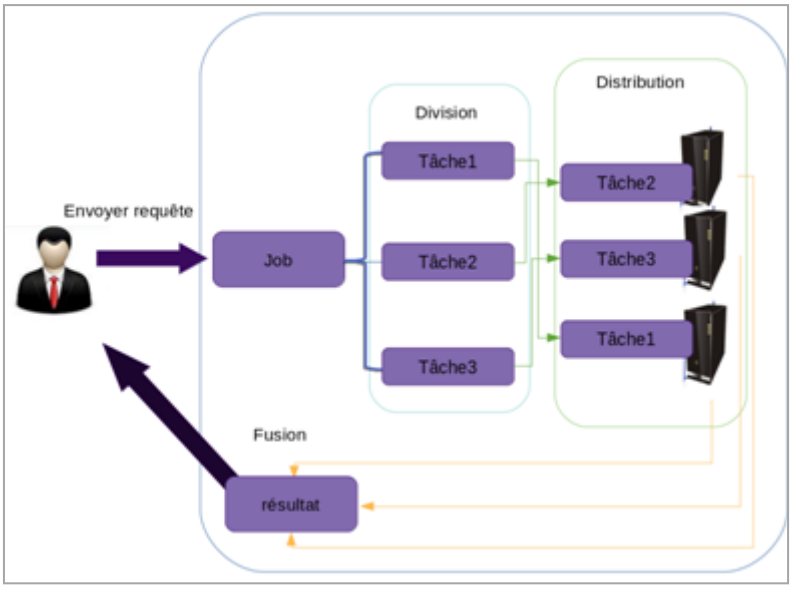


Figure III.4. Architecture de la partie traitement

➢ **Fonctionnement de la partie traitement**

✓ L'utilisateur lance la requête.

✓ La requête sera traduite en job par notre système. Ce job sera divisé à des tâches.

✓ La division du job est faite selon le plan d'exécution envoyé par Middleware.

✓ Le plan d'exécution est géré selon l'emplacement des données dont le job sera divisé à des tâches .Chaque tâche utilise les données stockées dans un seul emplacement.

✓ Envoyer chaque tâche à l'emplacement de bloc concerné.

✓ Chaque tâche fait le traitement sur le bloc de données concernées et retourne le résultat.

✓ La fusion des résultats des tâches est la sortie de cette phase.

c) Les acteurs de système

Dans ce système on a deux acteurs: utilisateur normal consulte la résultat individuel de sélection afin de faire la sélection par le deuxième acteur qu'il a le rôle d'administrateur qu'il peut charger les données dans le système et lancer la sélection.

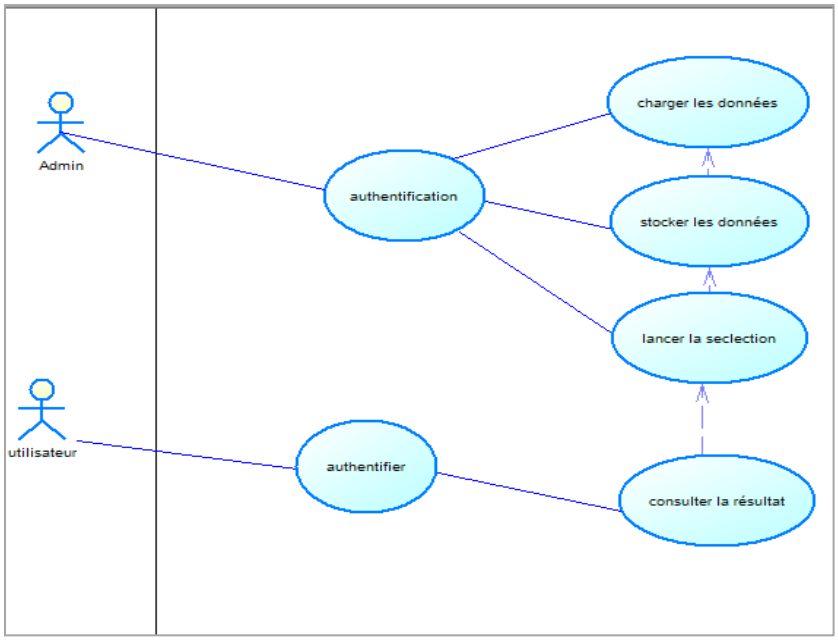


Figure III.5. Diagramme de cas d'utilisation.

d) Scénario d'utilisateur

Dans ce partie, l'utilisateur authentifie au système par un nom d'utilisateur et une mot de passe valide, le système vérifie si l'utilisateur est enregistré dans la base de données si l'utilisateur n'a pas un compte dans le système il peut s'inscrire, après l'authentification l'utilisateur peut consulter le système pour tester si il est accepté dans la sélection.

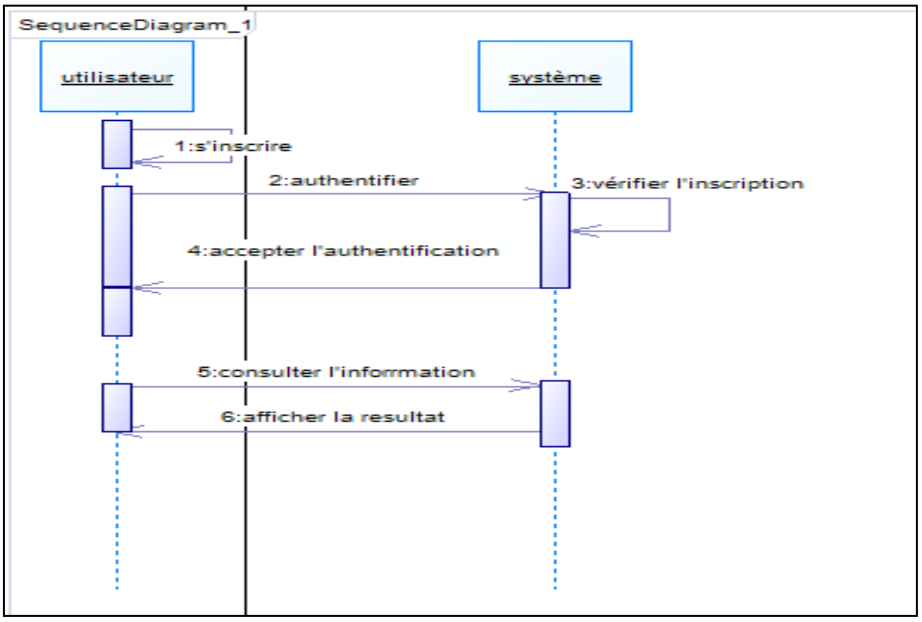


Figure III.6.Le scénario d'utilisateur.

e) Scenario d'administrateurs

- ✓ Administrateur connecte au système par l'authentification expliqué ci-dessus.
- ✓ L'administrateurs peut après l'authentification charger les données dans le système, ce dernier fait l'extraction et la distribution de ces données.
- ✓ Si les données est mise en place l'administrateur lance la sélection sur les données stockées.
- ✓ Le système retourne la résultat de sélection à l'administrateur.

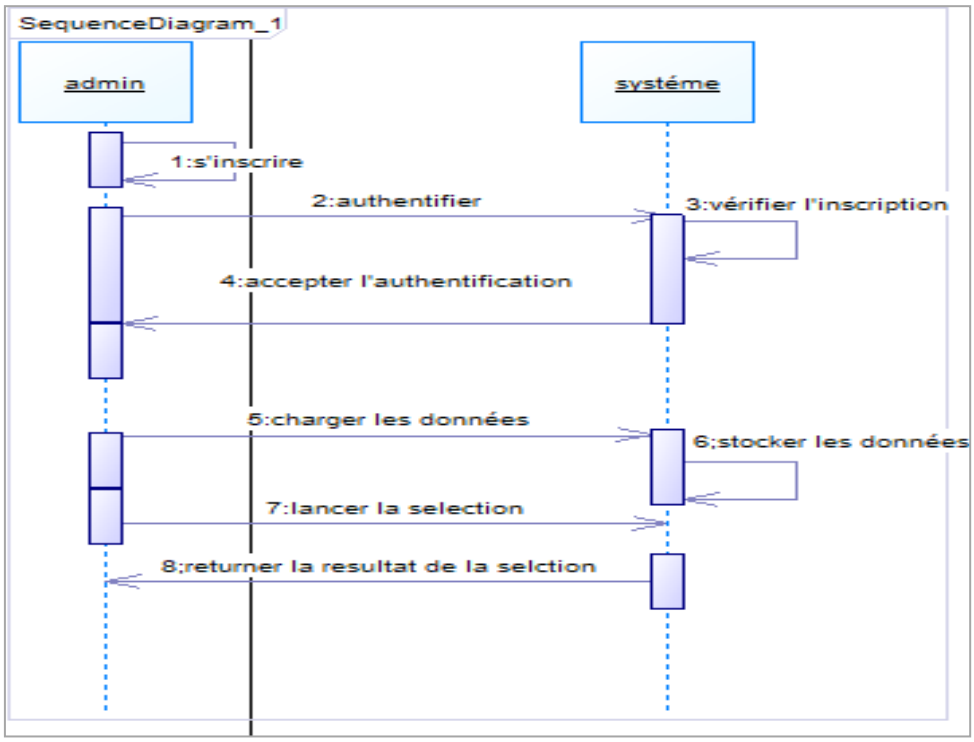


Figure III.7. Le scénario d'administrateur.

III.4 Notre algorithme de sélection

Dans notre travail, pour la sélection des candidats de master, les données sont des données numériques représentant des notes des étudiants dans des modules et les critères de sélection appliqués sont des formules de calcul des moyennes.

$$Moy_i [j]= (\Sigma_{q=[1..p]} Mod\_Spe[q]*Coeff) /\Sigma Coeff;$$

Où:

-**P** est le nombre de modules de spécialité.

-**Mod_Spe** [q] est le qième module d'une spécialité i.

-**Moy_i**[j] est la moyenne de l'étudiant j pour la spécialité i.

-**Coeff** est le coefficient d'un module.

Pour une spécialité, les modules sont divisés en deux: des modules de spécialité et des modules secondaires. On ne prend en considération que les modules de spécialité. En se basant sur les canevas des formations qui déterminent les conditions de sélection. On considère comme condition principale une moyenne requise. Un classement peut être considéré par la suite vu la contrainte de nombre de poste pour chaque spécialité. On montre dans ce qui suit notre algorithme de sélection.

Algorithme sélection()

p: le nombre de modules de spécialité;

MS_i: moyenne requise par la spécialité i

n: le nombres total d'étudiants;

k : nombre de spécialité ;

Mod_Spe: Les noms des modulede spécialité.

Moy_i[j] : moyenne de l'étudiant j pour la spécialité i

selecti : liste des étudiants sélectionnés pour la spécialité i

Spe_i : la liste des étudiants sélectionnés pour la spécialité i.

Pour chaque i← [1..k] faire // Pour toutes les spécialités,

pour chaque j← [1..n] faire // Pour chaque spécialité i, calcul des moyennes des étudiants

Moy_i [j]= ($\Sigma_{q=[1..p]}$ Mod_Spe[q]*Coeff) /ΣCoeff;

Si Moy_i [j] ≥ MS_i alors Select i ← j // sélection des étudiants pour la spécialité i

fin pour;

ordonner (Moy_i) ; // Classement des étudiants pour chaque spécialité i selon les moyennes

fin pour;

fin.

III.5 Implémentation

III.5.1 Outils et langages de programmation utilises

Pour la résiliation d'une application de sélection dans le Big data, nous avons utilisé quelques environnements de développement et langages de programmation, Citons :

- **Linux ubunu**

Ubuntu est un système d'exploitation développé par la communauté basé sur Linux. Dans le langage informatique, on dit qu'il est un système d'exploitation Open Source, Ubuntu fonctionne avec les ordinateurs de bureau, ordinateurs portables et les serveurs.

Ubuntu contient toutes les applications dont vous aurez besoin; soit des outils pour une utilisation de base comme le traitement de texte et les applications de courrier électronique ou plus compliquer comme les outils de programmation pour une utilisation avancée.

Ubuntu est un système d'exploitation gratuit. Vous ne payez aucun frais de licence. Vous pouvez télécharger, utiliser et partager Ubuntu librement.

Chaque six mois, une nouveau version Ubuntu est réalisée et mettre à la disposition des utilisateurs pour un téléchargement gratuit. La dernière version jusqu'à présente est la version Ubuntu 8.10. [28]

- **Hadoop**

Hadoop est un framework logiciel open source permettant de stocker des données, et de lancer ds applications sur des grappes de machines standards. Cette solution offre un espace de stockage massif pour tous les types de données, une immense puissance de traitement et la possibilité de prendre en charge une quantité de tâches virtuellement illimitée. Basé sur Java, ce framework fait partie du projet Apache, sponsorisé par Apache Software Foundation.[29]

- **Hive**

Hive est une infrastructure d'entrepôt de données pour Hadoop. Il permet la synthèse,
l'interrogation et l'analyse des données grace à un langage de haut niveau semblable à SQL, qui s'appelle HiveQL. Ce langage nous a permis d'interagir facilement avec un cluster Hadoop.
Par conséquent, l'exécution d'une requête HiveQL se traduit par la création et l'exécutiond'un job MapReduce.[25]

III.5.2 Environnement de travail

Pour installation et la mise en œuvre de Hadoop plateforme, On a choisi le système d'exploitation Linux avec la distribution ubuntu 18.10. installé :

- **sur une machine Toshiba**
  - ✓ Processeur Intel® Core™ i3-5005u 2.00GHz.
  - ✓ RAM 4.00 Go.
  - ✓ Disque 500 Go
- **sur une machine HP**
  - ✓ Processeur Pentium® Dual-Core™ 4-5005 @ 2.30GHz.
  - ✓ RAM 4.00 Go.
  - ✓ Disque 400 Go

III.5.3 Les étapes de notre solution

Pour l'implémentation de notre solution on va suivre les étapes suivantes

1. L'installation de Hadoop et du Hive.
2. Introduction d'un fichier de donnée par l'administrateur.
3. Création d'un fichier de donnée dans HDFS.
4. Restructuration des données par Hive.
5. Partitionner les données selon les mesures de la sélection.
6. Appliquer les requêtes nécessaires pour la sélection.
7. Afficher le résultat de sélection de l'utilisateur.

III.5.4 Installation de Hadoop

Nous ne pouvons pas installer Hadoop dans un cluster réel à cause du manque de matériel dans l'université. Ainsi, nous avons décidé d'installer Hadoop dans une seule machine et travailler de manière virtuelle.Puisque notre machine est de capacités limitées, nous avons simulé le travail d'un cluster qui contient trois machines seulement(Namenode et DataNode et SecondaryNode).

Dans les étapes suivantes on donne les détails de l'installation de Hadoop dans les machines présentes ci-dessus.

Avant l'installation de notre système Hadoop on a besoin d'adapter l'environnement par :

- ✓ La mise à jour du système..
- ✓ L'installation de Openssh-server qui facilite les connexions sécurisées entre le système d'architecture Client/Serveur.

➢ **La mise à jour du système d'exploitation Ubuntu:**

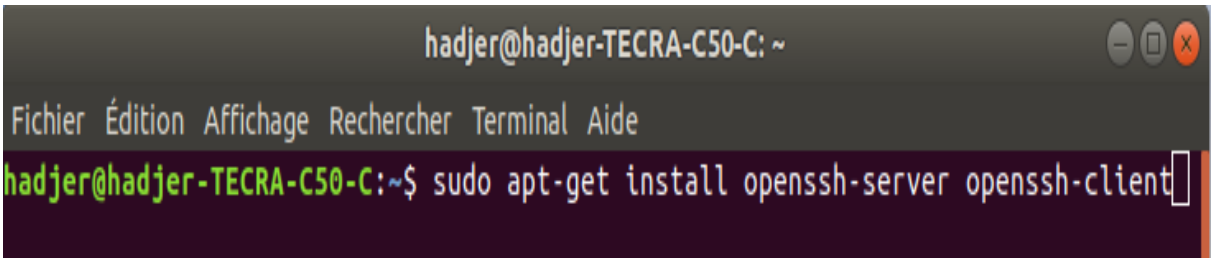
Avant toutes installation de nouveaux paquet pour mette à jour le cache des paquet sur notre machine.

➢ Mettre le système d'exploitation à jour par la commande suivante:

```
Sudoapt-get update
```

➢ **Installation du openssh –server:**

✓ Hadoop nécessite un accès SSH pour gérer les différents nœuds. On a Installé le Openssh server par l'exécution de la commande:



- ✓ Vérifier si open ssh-server est bien installer:

```
aicha@aicha-HP-620: ~
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
aicha@aicha-HP-620:~$ ssh localhost
aicha@localhost's password:
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.18.0-17-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage


198 paquets peuvent être mis à jour.
0 mise à jour de sécurité.

Your Hardware Enablement Stack (HWE) is supported until April 2023.
aicha@aicha-HP-620:~$
```

- ➢ **L'installation de hadoop:**
  - ✓ Nous Avons choisis la version-2.7.3 car elle est une des versions de Hadoop2 qui contient le core Yarn ou MapReduce2 pour séparer la gestion des ressources et le traitement MapReduce.
  - ✓ Pour installer Hadoop-2.7.3, nous l'avons téléchargé à partir du site Apache. On a exécuté la commande suivante.

```
hadjer@hadjer-TECRA-C50-C:~$ wget https://archive.apache.org/dist/hadoop/core/ha
doop-2.7.3/hadoop-2.7.3.tar.gz
```

  - ✓ Si le téléchargement est réussi décompresse le fichier téléchargé sera décompressé..

```
hadjer@hadjer-TECRA-C50-C:~$ tar -xvf hadoop-2.7.3.tar.gz
```

- ➢ **Configuration de Hadoop:**
  - ✓ L'installation de Hadoop est une configuration des fichiers suivants:

Le fichier .bashrc

- ✓ Avant de modifier le fichier **.bashrc** dans notre répertoire personnel, nous devons trouver le chemin où Java a été installée pour définir la variable d'environnement $JAVA_HOME en utilisant la commande suivante :

```
hadjer@hadjer-TECRA-C50-C:~$ update-alternatives --display java
```

- ✓ Ouvrir le fichier **.bashrc**

```
aicha@aicha-HP-620: ~
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
aicha@aicha-HP-620:~$ sudo gedit ~/.bashrc
```

- ✓ Pour informer les système d'exploitation par l'environnement Hadoop, on va ajouter les lignes suivantes dans le fichier:

```
export HADOOP_HOME=/home/aicha/hadoop-2.7.3
export HADOOP_CONF_DIR=/home/aicha/hadoop-2.7.3
export HADOOP_MAPRED_HOME=/home/aicha/hadoop-2.7.3
export HADOOP_COMMON_HOME=/home/aicha/hadoop-2.7.3
export HADOOP_HDFS_HOME=/home/aicha/hadoop-2.7.3
export YARN_HOME=/home/aicha/hadoop-2.7.3
export PATH=$PATH:/home/aicha/hadoop-2.7.3/bin
```

Le fichier hadoop-env.sh

- ✓ Nous devons définir $JAVA_HOME qui fait pour l'implémentation des jobHadoop, en modifiant le Le fichier **hadoop-env.sh**

```
# The java implementation to use.
export JAVA_HOME=home/aicha/jdk1.8.0_60
```

- ✓ L'ajout de la déclaration ci-dessus dans le fichier **hadoop-env.sh** assure que la valeur de la variable **$JAVA_HOME** sera disponible pour Hadoop chaque fois qu'il est mis en marche.

Le fichier core-site.xml.

- ✓ Le fichier **core-site.xml** contient les propriétés de configuration utilisées par Hadoop au démarrage.

```
<configuration>
<property>
<name>fs.default.name</name>
<value>hdfs://localhost:9000</value>
</property>
</configuration>
```

Le fichier mapred-site.xml.

- ✓ MapRed-site contient les paramètres de configuration de l'application MapReduce tels que le nombre des machines virtuel, pouvons s'exécuter en parallèle, la taille de mapper…etc

```
<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
</configuration>
```

Le fichieryarn-site.xml.

- ✓ Le fichier **yarn-site.xml** contient les paramètres de configuration de ResourceManager et NodeManager, tels que la taille de la gestion de la mémoire d'application, les opérations requises sur le programme et l'algorithme, etc.

```
<configuration>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
</configuration>
```

Le fichierhdfs-site.xml.

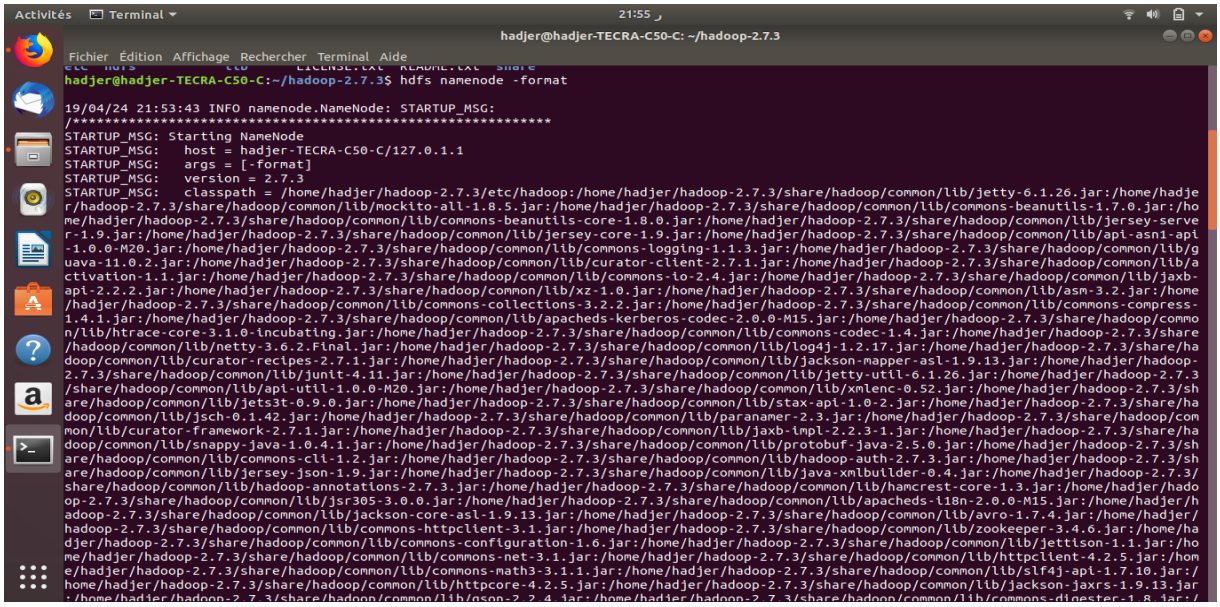
- ✓ Le fichier **hdfs-site.xml** doit être configuré pour chaque hôte du cluster qui va l'utiliser. Il est utilisé pour spécifier les répertoires qui seront utilisés comme NameNode et DataNode sur cette hôte.

```
<configuration>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>dfs.permission</name>
<value>false</value>
</property>
</configuration>
```

- **Formater le nouveau système de fichier HDFS pour Hadoop** :

Hfds namenode -format

- Avant de démarrer le serveur Hadoop dans le NameNode on doit formater le système fichier HDFS.



- **Lancement deHADOOP**:
- Nous passons au démarrage du Hadoop nouvellement installé, On peut lancer une des commandes suivantes:

(start-dfs.sh et start-yarn.sh) ou start-all.sh

```
aicha@aicha-HP-620: ~/hadoop-2.7.3/sbin
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
aicha@aicha-HP-620:~/hadoop-2.7.3/sbin$ ./start-all.sh
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
Java HotSpot(TM) Server VM warning: You have loaded library /home/aicha/hadoop-2
.7.3/lib/native/libhadoop.so.1.0.0 which might have disabled stack guard. The VM
 will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>',
or link it with '-z noexecstack'.
19/05/20 21:57:17 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
Starting namenodes on [localhost]
aicha@localhost's password:
localhost: starting namenode, logging to /home/aicha/hadoop-2.7.3/logs/hadoop-ai
cha-namenode-aicha-HP-620.out
aicha@localhost's password:
localhost: starting datanode, logging to /home/aicha/hadoop-2.7.3/logs/hadoop-ai
cha-datanode-aicha-HP-620.out
Starting secondary namenodes [0.0.0.0]
aicha@0.0.0.0's password:
0.0.0.0: starting secondarynamenode, logging to /home/aicha/hadoop-2.7.3/logs/ha
doop-aicha-secondarynamenode-aicha-HP-620.out
Java HotSpot(TM) Server VM warning: You have loaded library /home/aicha/hadoop-2
.7.3/lib/native/libhadoop.so.1.0.0 which might have disabled stack guard. The VM
```

✓ Pour vérifier l'exécution des processus JAVA et le bon fonctionnement d'Hadoop On tape la commande :

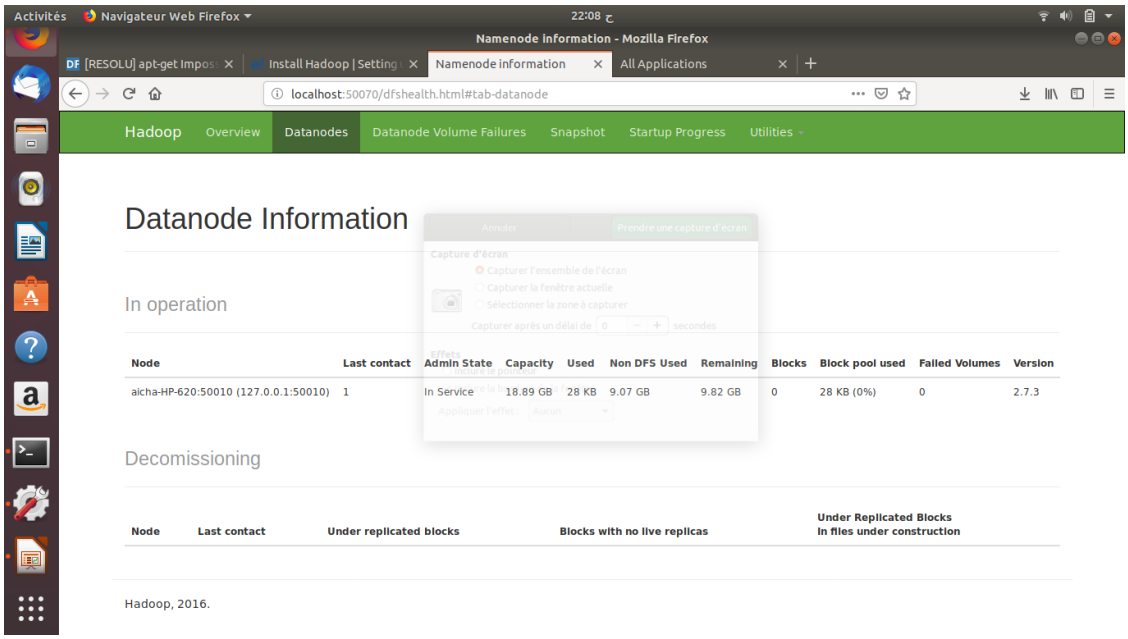
```
jps
```

```
aicha@aicha-HP-620:~/hadoop-2.7.3/sbin$ jps
17349 NodeManager
15031 NameNode
17015 ResourceManager
17677 Jps
16638 DataNode
16863 SecondaryNameNode
```

➢ Par défaut, Hadoop est configuré avec des interfaces WEB permetant d'obtenir des informations sur la capacité total et connaître l'état de disponibilité d'un nœud. Il permet également d'avoir des information sur les fichiers et de naviguer dans le HDFS cluster. On:

```
localhost:50070/dfshealth.html
```

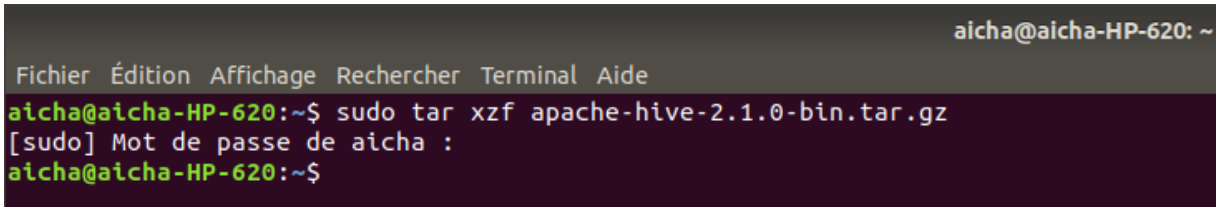
➢ peut accéder à cette interface via l'adresse suivante:



3.4.5 Installation de HIVE:

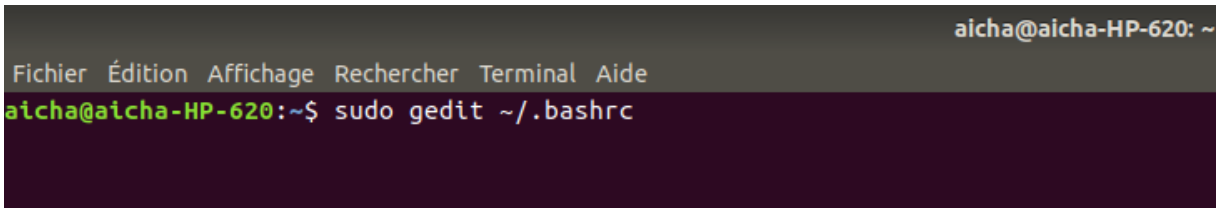
Télécharger Hive, version stable, depuis le site http://archive.apache.org/dist/hive/

Extraire le fichier téléchargé dans le chemin /usr/local, On exécute le commande suivante:



- Editez le fichier “.bashrc” pour mettre à jour les variables d'environnement pour l'utilisateur:

.bachrc



- Définir le chemin Hadoop dans **hive-env.sh**

```
# Set HADOOP_HOME to point to a specific hadoop install directory
# HADOOP_HOME=${bin}/../../hadoop
export HADDOP_HOME=home/aicha/hadoop-2.7.3

# Hive Configuration Directory can be controlled by:
export HIVE_CONF_DIR=512

# Folder containing extra ibraries required for hive compilation/execution can be controlled by:
export HIVE_AUX_JARS_PATH=/home/aicha/apache-hive-2.1.0-bin/conf
```

➢ Modifier Le fichier **hive-site.xml.**

```
<configuration>
<property>
<name>javax.jdo.option.ConnectionURL</name>
<value>jdbc:derby:;databaseName=/home/edureka/apache-hive-2.1.0-bin/metastore_db;create=true</value>
<description>
JDBC connect string for a JDBC metastore.
To use SSL to encrypt/authenticate the connection, provide database-specific SSL flag in the connection URL.
For example, jdbc:postgresql://myhost/db?ssl=true for postgres database.
</description>
</property>
<property>
<name>hive.metastore.warehouse.dir</name>
<value>/user/hive/warehouse</value>
<description>location of default database for the warehouse</description>
</property>
<property>
<name>hive.metastore.uris</name>
<value/>
<description>Thrift URI for the remote metastore. Used by metastore client to connect to remote metastore.</descriptio
</property>
<property>
<name>javax.jdo.option.ConnectionDriverName</name>
<value>org.apache.derby.jdbc.EmbeddedDriver</value>
<description>Driver class name for a JDBC metastore</description>
</property>
<property>
<name>javax.jdo.PersistenceManagerFactoryClass</name>
<value>org.datanucleus.api.jdo.JDOPersistenceManagerFactory</value>
<description>class implementing the jdo persistence</description>
</property>
</configuration>
```

➢ Par défaut, Hive utilise la base de données Derby. Initialiser la base de données Derby.

```
aicha@aicha-HP-620:~/apache-hive-2.1.0-bin$ bin/schematool -initSchema -dbType derby
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/aicha/apache-hive-2.1.0-bin/lib/log4j-slf4j-impl-2.4.1.jar!/org/slf
SLF4J: Found binding in [jar:file:/home/aicha/hadoop-2.7.3/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar
.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Metastore connection URL:        jdbc:derby:;databaseName=metastore_db;create=true
Metastore Connection Driver :    org.apache.derby.jdbc.EmbeddedDriver
Metastore connection User:       APP
Starting metastore schema initialization to 2.1.0
Initialization script hive-schema-2.1.0.derby.sql
Initialization script completed
schemaTool completed
```

Lancement hive

➢ Nous passons au démarrage hive:

```
aicha@aicha-HP-620:~$ cd apache-hive-2.1.0-bin
aicha@aicha-HP-620:~/apache-hive-2.1.0-bin$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/aicha/apache-hive-2.1.0-bin/lib/log4j-slf4j-i
SLF4J: Found binding in [jar:file:/home/aicha/hadoop-2.7.3/share/hadoop/common/lib/sl
.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/home/aicha/apache-hive-2.1.0-bin
 true
Java HotSpot(TM) Server VM warning: You have loaded library /home/aicha/hadoop-2.7.3/
tack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or li
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. C
ez) or using Hive 1.X releases.
hive> show databases;
```

III.4.6 Gestion de la base de données

- **La phase stockage de notre système**

Nous allons stocker les données dans HDFS par le procédure suivant:

- ✓ Pour distribuer nos données dans le système nous allons créer un dossier dans HDFS comme un emplacement par la commande suivante:

```
hadjer@hadjer-TECRA-C50-C:~$ hadoop dfs -mkdir -p /user/hive/warehouse
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
```

- ✓ Changer les droits d'accès de ce dossier pour permettre à l'utilisateur d'accéder aux données stockées dans l'emplacement, Par la commande:

```
hadjer@hadjer-TECRA-C50-C:~$ hadoop dfs -chmod 765 /user/hive/warehouse
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
```

tocker les données existes dans l'emplacement crée précédemment. Avec la commande suivante:

```
hadjer@hadjer-TECRA-C50-C:~$ hadoop dfs -put ~/LastMem/Results.csv  /home/hadjer/input
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
```

Donc les données sont entrées dans le système et stockées dans HDFS. Donc, nous allons modifier la structure de ces données par une autre structure adaptée avec notre travail. La manière de donner une nouvelle structure est détaillée dans la deuxième phase.

La phase préparation des données

Pour restructurer notre donnée, nous allons utiliser Hive fonctions suivantes:

- Nous déterminons la nouvelle structure par création d'un tableau contient la description de cette structure. Dans Hive nous pouvons créer un tableau comme suivant:

```
hive> Create table results(N_Ins string,Inf1 float,Inf2 float,SM float,TWEB float,TEC float,ASD2 float,SE1 float,TL float,GLPOO float,BD float,AO flo
at,ASD1 float,SI float,PL float,SE2 float,Res float,TG float,DW float,BDW float)row format delimited fields terminated by ',' stored as textfile;
OK
Time taken: 2.886 seconds
```

- Le tableau crée désigne la structure donc nous affectons seulement la donnée à partir de l'emplacement dans HDFS à la structure définie dans Hive, comme suivant:

```
hive> load DATA local INPATH '/home/hadjer/LastMem/Results.csv'OVERWRITE INTO TABLE results;
Loading data to table default.results
OK
Time taken: 1.065 seconds
hive>
```

Pour vérifier le succès de l'opération si dessus, nous pouvons lancer la requête «Select» pour afficher le contenu de tableau.

```
hive> SELECT * FROM results;
19/06/17 21:44:48 ERROR hdfs.KeyProviderCache: Could not find uri with key [dfs.encryption.key.provider.uri] to create a keyProvider !!
OK
19/06/17 21:44:51 INFO Configuration.deprecation: mapred.input.dir is deprecated. Instead, use mapreduce.input.fileinputformat.inputdir
19/06/17 21:44:51 INFO mapred.FileInputFormat: Total input paths to process : 1
N_Ins   NULL    NULL    NULL    NULL    NULL    NULL    NULL    NULL    NULL    NULL    NULL    NULL    NULL    NULL    NULL    NULL    NULL    NULL N
ULL
110030038       13.5    16.0    11.25   11.3    11.5    11.0    6.75    12.0    7.5     9.0     11.25   12.5    8.75    9.5     9.5     12.19   8.25 4
.33     7.75
120030182       17.5    18.25   17.69   9.0     18.25   10.0    11.5    12.88   7.0     8.0     10.72   17.0    12.75   11.25   10.5    11.75   7.25 7
.5      17.5
12030010        10.5    15.5    14.5    14.0    14.75   9.25    5.75    13.5    11.0    7.75    7.19    12.0    6.5     11.5    9.0     10.5    5.75 8
.5      11.5
70030001        17.75   15.5    14.5    10.35   13.5    15.0    9.0     7.5     10.63   1.13    8.38    14.38   13.5    13.0    19.0    10.88   10.5 5
.75     17.5
110030016       15.25   14.5    13.25   14.13   15.0    16.0    5.25    10.0    13.5    8.75    11.13   16.75   11.63   12.25   12.0    11.82   6.25 4
.19     4.5
110030048       15.13   12.0    13.0    12.7    13.75   16.0    6.18    12.88   11.0    9.0     12.13   11.5    10.25   11.25   10.5    10.5    8.0  5
.69     14.0
110030003       13.5    12.75   13.25   16.88   15.75   16.0    3.5     10.67   10.0    10.38   12.0    14.38   6.5     7.5     14.0    12.69   10.251
0.38    15.0
120030110       12.29   12.25   11.88   11.75   13.0    12.5    9.0     16.5    11.0    7.5     12.19   13.25   5.0     13.13   10.5    6.88    9.25 7
.0      13.5
120030070       14.09   14.5    13.0    8.63    6.75    13.0    13.63   11.19   17.0    5.25    8.13    13.25   8.75    9.63    12.25   11.75   8.0  1
0.13    13.5
120030100       13.5    11.0    10.75   15.75   14.25   9.5     5.25    14.75   7.5     7.25    9.97    14.25   2.75    12.5    10.0    12.0    NULL N
ULL     NULL
120030012       10.5    12.75   14.13   10.13   14.75   13.5    10.5    10.13   10.0    6.5     10.1    12.0    9.25    12.63   10.5    9.5     7.25 7
.75     13.0
110030047       15.25   10.5    10.0    11.58   14.0    17.5    14.5    8.25    11.0    11.5    11.38   12.0    7.5     10.0    12.5    8.13    13.757
.0      10.0
110030047       15.25   10.5    10.0    11.58   14.0    17.5    14.5    8.25    11.0    11.5    11.38   12.0    7.5     10.0    12.5    8.13    13.757
.0      10.0
120030078       11.75   8.0     11.0    16.63   10.25   12.0    6.75    10.69   9.0     9.0     13.45   11.75   7.0     11.5    10.5    10.25   8.13 1
1.25    6.0
120030027       17.5    16.25   13.25   5.63    14.75   13.0    10.75   12.5    7.0     11.0    11.0    14.25   5.5     10.75   9.0     7.88    3.25 2
```

III.4. 7 Traitement de la base de données

Dans cette partie et à partir de ce que nous avons fait précédent, nous allons mettre en œuvre l'interface graphique de notre système pour interaction avec les utilisateurs.

- Cette interface permet à l'administrateur ou l'utilisateur d'entrer dans notre système après la vitrification de leur compte.

Figure III.8.Interface d'authentification.

- Si l'utilisateur n'a pas un compte dans notre système il peut inscrire et créer un compte à partir l'interface suivant:

Figure III.9.Interface inscription.

- Cette interface est une interface administrateur lui permettant d'accéder à la base des données et extraire les données d'un emplacement et les stockées dans le système.

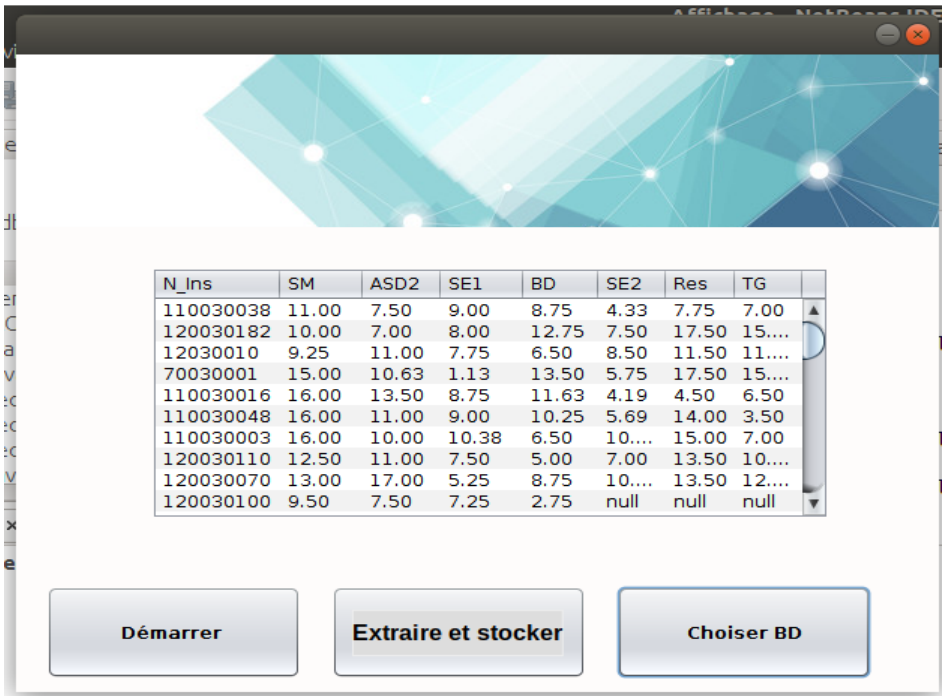


Figure III.10. Interface d'administrateur.

Cette interface permet à l'administrateur de déterminer la spécialité et ses critères de sélection.

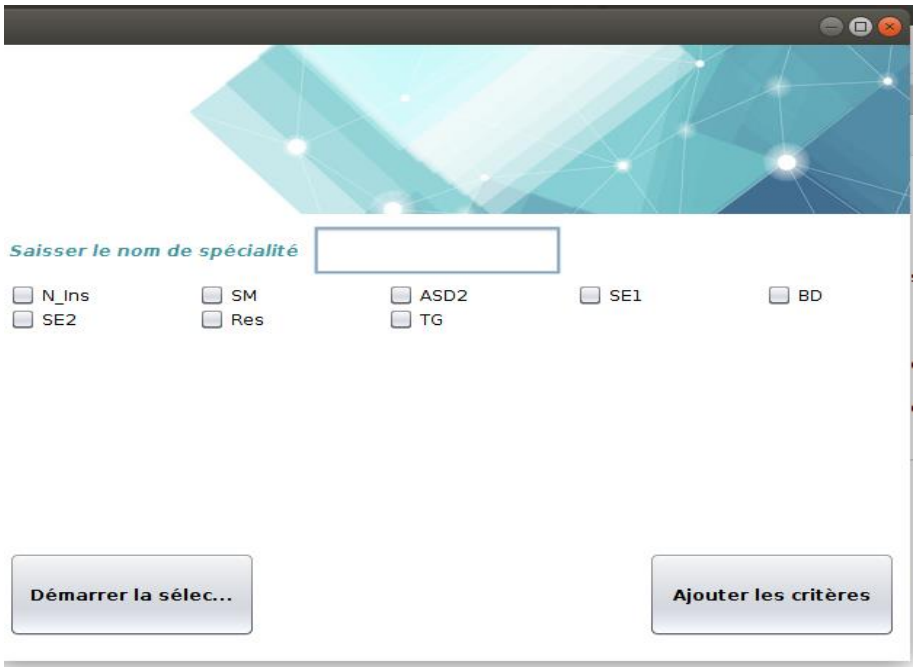
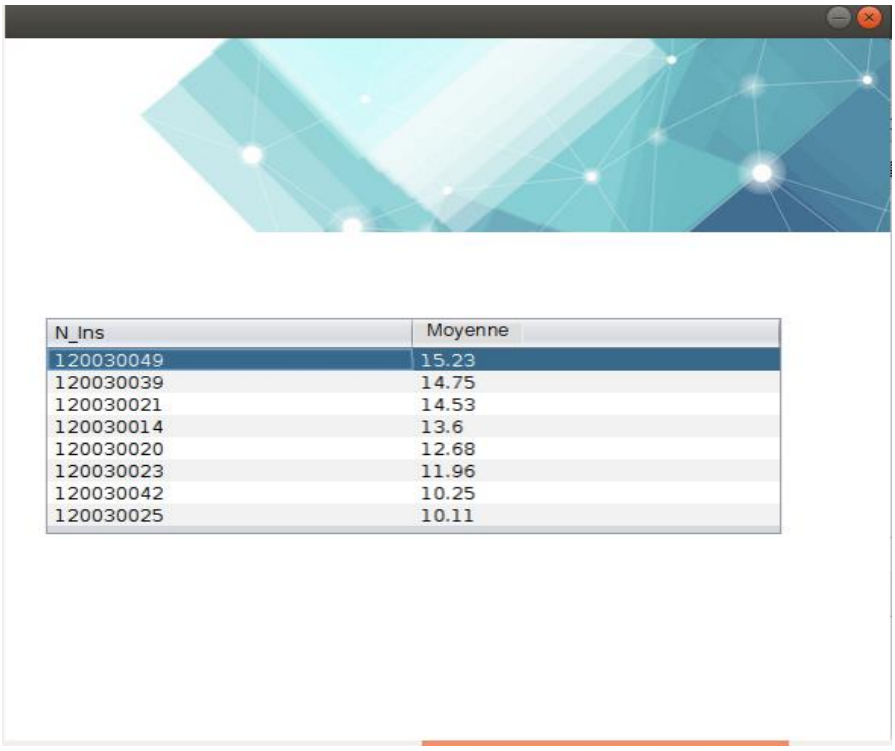


Figure III.11. Interface de choix des critères.

Cette interface contient le résultat final de sélection.



N_Ins	Moyenne
120030049	15.23
120030039	14.75
120030021	14.53
120030014	13.6
120030020	12.68
120030023	11.96
120030042	10.25
120030025	10.11

Figure III.12. Interface de résultat de la sélection

Cette interface permette à l'utilisateur de consulter le résultat final s'il est accepté pour un master ou non.

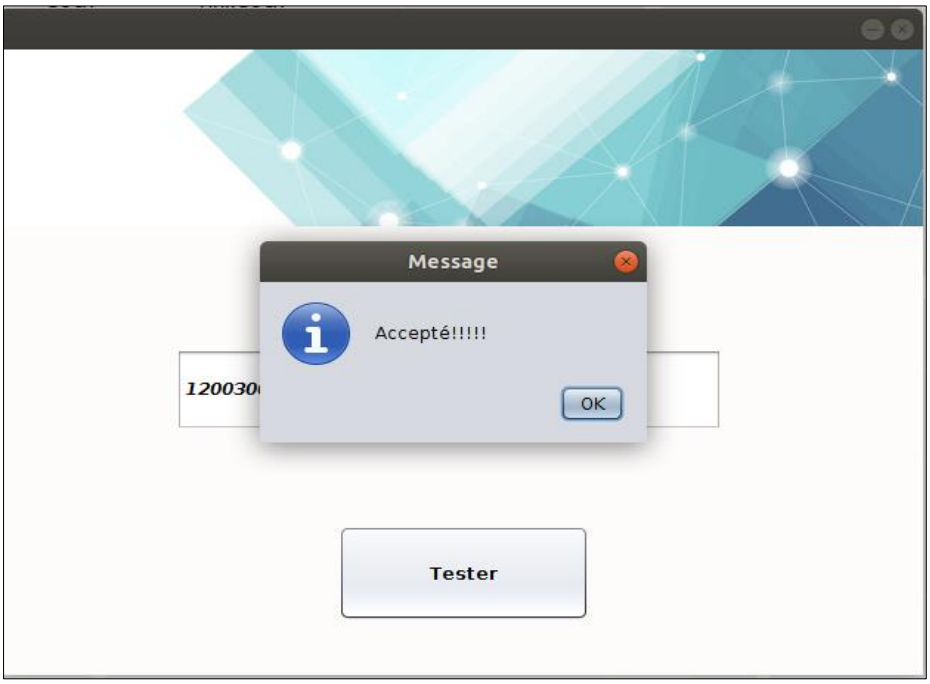


Figure III.13. Interface utilisateur.

III.3.5 conclusion

Dans ce chapitre nous avons présenté l'architecture globale et détaillé de notre système et les étapes de la mise en œuvre de notre projet avec tous les outils, les langages et les plateformes utilisés ainsi que la présentation avec l'explication du rôle de chaque outil. Enfin, nous avons présenté l'interface graphique de notre système et son utilisation par les différents utilisateurs et le résultat final de notre système.

Conclusion générale

Dans ce travail, nous avons mené une étude sur le Big Data et ses techniques dans le domaine de la sélection des candidats. Nous avons réalisé une application permettant aux universités la sélection les étudiants qualifiés pour terminer leurs études de master dans une spécialité bien précise, en appliquant un ensemble de critères aux données stockées dans la bases de données. Pour résoudre le problème de l'augmentation de la taille des données et leur impact sur le processus de sélection des candidats, nous avons réalisé une application Big Data. Nous avons utilisé Hadoop comme un outils de stockage et de traitement grâce a ces composants et Hive comme un système NoSQL pour la gestion de ces données.

Nous avons rencontré des difficultés et des défis en ce qui concerne la préparation de l'environnement de système et l'installation des outils (Hadoop et Hive) :premièrement, on a pas eu l'occasion de l'installer dans un cluster réel à cause de la limitation des ressources dans l'université. Deuxièmement, l'installation de Hadoop dans une seule machine et la création d'un cluster virtuel nous a causé beaucoup de problème dans le fonctionnement de l'outil Hadoop et Hive de plus elle cause la dégradation de la performance de notre machine .

Nous n'avons pas le choix alors nous avons installé les outils dans une machine et créé un cluster virtuel comme un solution temporaire.

En résume, nous pouvons dire qu'à travers ce projet, nous avons appris beaucoup de choses concernant un nouveau domaine le Big Data. Après avoir bien compris le domaine nous nous sommes concentrés sur l'apprentissage d'un outils pour bien comprendre le principe de stockage et de traitement des big data. Nous avons opter pour le frameworkHadoop qui est un outil très populaire. Nous avons pu l'installer avec les composants nécessaires malgré qu'on a mis beaucoup de temps. c'était un défi pour nous au début. A travers le développement d'une application, nous avons appris une nouvelle manière de gérer des données NoSQL utilisons la langage HQL.

Perspectives

Nous avons comme perspectives d'avoir l'occasion de test de notre application dans des architectures fortement distribuées. Donc d'installer Hadoop dans un cluster

réel de l'université pour permettre de résoudre les problèmes de stockage et d'augmentation des données et de les profiter autant que possible pour améliorer le performance de l'université. Une autre perspective est d'appliquer notre application sur des données plus volumineuses et plus variées.

Références bibliographiques

[1] Journal of king saud university – computer and information sciences, p 433-434

[2]http://www.redsen-consulting.com/2013/06/big-data

[3] https://www.oracle.com/fr/big-data/guide/what-is-big-data.html,consultéle 3/02/ 2019

[4]http://www.tutorialspoint.com/hadoop/hadoop_quick_guide.html,consulté le 24/02/ 2019

[5] Hao Zhang, Gang Chen, In-Memory Big Data Management and Processing: A Survey

[6] Tawfik Bourgi, Nesrine Zoghlami, Big Data for Transport and Logistics.

[7]M.CORINUS, T.Derey, J.Marguerie, W.Techer, N.Vic, Rapport d'étude sur le Big Data, SRS Day, 54p, 2012.

[8] https://www.piloter.org/business-intelligence/technologie-bigdata.html,consulté le 3/03/ 2019

[9]http://fr.wikipedia.org/wiki/Entrep%C3%B4t_de_donn%C3%A9es,consulté le 9/04/2015.

[10]Big data application and archetecteure; de himanshu et soumendramohanty.

[11]Mekideche Mounir, Conception etimplémentation d'un moteur de recherche à base d'une architecture Hadoop (Big Data), Avril 2015.

[12]Big data etNoSQL, http://administration-systeme.blogspot.com/?view=classic, visité en 01/03/2015 .

[13]KOUEDI Emmanuel, Approche de migration d'une base de données relationnelle versune base de données NoSQL orientée colonne, Mémoire master informatique, UNIVERSITE DE YAOUNDE I, mai 2012.

[14]Bernard ESPINASSE, Introduction aux systèmes NoSQL (Not Only SQL), Ecole Polytechnique Universitaire de Marseille, 19p, Avril 2013

[15] Redi Bruchez, « Les basesde données NoSQL et Big Data » , avril 2014.

[16]M. de formation d'IBM, « Chapitre map-reduce », 2016, [Accès le 10-Mai 2017]

[17] KOUEDI Emmanuel Approche de migration d'une base de données relationnelle vers une base de données NoSQL orientée colonne.

[18] Boucetta Zouhel, APPARIEMENT SÉMANTIQUE DES CVs/OFFRES D'EMPLOI DANS LE CADRE DU E-RECRUTEMENT

[19]Nalini Venkatasubramanian, Hadoop a distrbuted framework for BigData

[20]http://mbaron.developpez.com/tutoriels/bigdata/hadoop/introduction-hdfs-map-reduce/, consulté le 27/04/ 2019

[21] Jonathan Lejeune, Hadoop:uneplate-formed'exécution de programme Map-reduce, École des Mines de Nantes, 83p, Janvier 2015.

[22]Benjamin Renaut, Hadoop/Big Data, Université de Nice Sophia-Antipolis, 114p, 2013-2014

[23] L. R. JDN, « Hbase : le nosql au service du big data », 23-Avril-2013, [Accès le20Mai2017]adressehttp://www.journaldunet.com/developpeur/outils/comparatif- des-bases-nosql/hbase.shtml

[24] livre École des Mines de Nantes, 83p, Janvier 2015.

[25]http://blog.ippon.fr/2013/05/14/big-data-la-jungle-des-differentes-distributions-open-source-hadoop/, consulté le5/04/2019

[26] M. GROVER, « Zookeeperfundamentals, deployment, and applications »,Mai-2017.

[27] F. d'Apache software, « Introduction », [Accès le 05-Mai-2017].adresse : https://sqoop.apache.org/docs/1.4.1-incubating/SqoopUserGuide.html

[28]http://computers-maintenance.blogspot.com/2009/02/qu-est-ce-que-ubuntu-linx.html

[29]https://www.lebigdata.fr/hadooop,consulté le 3/05/ 2019