République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université AMO de Bouira

Faculté des Sciences et des Sciences Appliquées

Département d'Informatique

Mémoire de master

en Informatique

Spécialité : *ISIL*

Thème

Titre de votre projet

Encadré par

— BOUDJELABA Hakim

Réalisé par

— GHODEBANE Hassiba

— BOUABID Mayada

2015/2016

Remerciements

Tout d'abord et avant tout, nous rendons grâce a notre dieu,le tout puissant qui nous avoir donné le courage et la patience durant ce travail.

Nos plus sincères remerciements s'adressent a notre encadreur **Mr Boudjelaba Hakim** pour ses précieux conseils et encouragements. Merci pour votre confiance, votre disponibilité et vos encouragement.

Nous adressons également nos vifs remerciements aux membres du jury qui ont accepté d'évaluer notre projet.

Ainsi que tous les enseignants qui ont assuré notre formation en cursus universitaire.

Finalement, nous aimerions aussi remercier notre famille et tous ceux qui nos ont encouragé et supporté avec les hauts et les bas tout au long de ce travail. Merci à tous et toutes.

Dédicaces

Je dédie ce mémoire tout d'abord a mes chers parents qui m'ont apportés tout le confort, mes chers frères, ma petite sœur, toutes ma famille et mes amis pour leurs encouragement et leurs soutiens. A tous ceux qui m'ont aidé dans la réalisation de ce travail et je ne manquerai pas tous ceux qui m'ont encouragé à aller jusqu'au bout.

Ghodbane Hassiba.

Dédicaces

Mes dédicaces vont particulièrement pour mes parents, mes deux chers frères, ma famille, tous ceux qui ont contribué de prés ou de loin pour la réalisation de ce travail et tous ceux qui ont m'ont accompagné et soutenu.

Bouabid Mayada

Résumé

Face à l'explosion de la production massive de données, les SGBD relationnels ont été rapidement limités par les grands acteurs du web comme Google, Facebook, Amazon ….. Ces limites ont conduit ces grands acteurs du web à développer et adopter de nouvelles technologies alternatives comme le NoSQL et l'apparition du terme Big Data. Ces nouvelles technologies viennent répondre aux exigences de gestion, de traitement, d'analyse de ces données gigantesques.

Mots clés : Bases de données relationnelles, Big Data, NoSQL, Hadoop, MapReduce, HDFS.

Abstract

In the face of the explosion of massive data production, relational DBMSs have been rapidly limited by major web players such as Google, Facebook and Amazon ….. These limits have led these major web players to develop and adopt new alternative technologies such as NoSQL and the emergence of the term Big Data. These new technologies come to meet the requirements of management, processing, analysis of these gigantic data

Key words : Relational database, Big Data, NoSQL, Hadoop, MapReduce, HDFS.

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations

SQL	Structured Query Language
ACID	Atomicité, Cohérence, Isolation, Durabilité
OLAP	Online Analytical Processing
SGBD	Système de Gestion de Base de Données
SGBDR	Un Système de Gestion de Base de Données Relationnelles
CPU	central processing unit
RAM	Random Access Memory
CAP	Coherence , ≪Availability , Partition tolérance
BASE	Basically Availabe, Soft-State, Eventuallty Consistent
jPS	JVM Process Status Tool
HDFS	Hadoop Distributed File System
JVM	Java virtual machine
JDK	Java Development Kit
IdO	Internet des Objets
LDD	Langage de Définition de Données
LMD	Langage de Manipulation de Données
LR	langage de Requête.
OQL	Object Query Lnaguage
NoSql	Non Seulement Sql

Introduction générale

Notre monde est actuellement confronté à une explosion importante de données. Chaque jour, $2,5$ de téraoctets de données sont générées dans le monde. Google reçoit 40000 requêtes chaque seconde, 72 vidéos sont mises en ligne chaque minute sur YouTube et 217 nouveaux utilisateurs de Smartphone sont enregistrés chaque minute. Aujourd'hui l'information nous parvient de toute part : des capteurs de géolocalisation, des données postées sur les réseaux sociaux, des transactions des clients ...[77].

Le développement et l'accès à ces données a conduit à l'apparition du terme **Big Data** qui possède ses origines dans le Data Science et le Cloud Computing. Ce phénomène impacte en particulier les entreprises qui sont amenées à manipuler des téraoctets voir des pétaoctets de données nécessitant une infrastructure spécifique pour leur création, leur stockage, leur traitement, leur analyse et leur récupération. En d'autres termes, il s'agit du développement en temps réel d'une masse de données volumineuse qui dépasse la capacité des outils de traitement et d'analyse traditionnels (bases de données relationnelles, ...).

Les entreprises sont habituées à utiliser les SGBDR pour stocker leur données structurées sur une seule machine, et lorsque elles font face à des problèmes de performance ils achètent du matériel très coûteux comme la mémoire, des processeurs plus performants et cela devient très vite pesant financièrement sur l'entreprise sans une amélioration significative des performances. L'approche SGBDR ne respecte pas les exigences des entreprises du Web 2.0 telles que Google, Amazon, Yahoo, Facebook et LinkedIn. Ces sociétés gèrent une grande quantité de données. Ces données générées sont non seulement volumineuses, mais ont une vélocité importante, et aussi diversifiées, ce qui à conduit à l'apparition du terme Big Data. Le Big Data impose de nouveaux défis à la manière traditionnelle de trai-

ter les données en utilisant les SGBDR. Un nouveau mouvement de gestion des données est né appelé **NoSQL** qui est l'abréviation de *Not Only SQL*. C'est une approche différente des SGBDR qui offre une mise à l'échelle linéaire. Les bases de données NoSQL peuvent traiter différents types de données, qui ne nécessitent pas un schéma strict. Le NoSQL a différente catégorisation pour répondre aux différents besoins de stockage du Big Data.

On s'intéresse dans ce mémoire à présenter les différentes solutions technologiques du Big Data qui sont utilisées pour remédier aux problèmes de stockage et de traitements de grande quantité de données. Ce mémoire est structuré comme suit :

— **Chapitre 1** Présentation générale des bases de données relationnelles et des SGBDR ainsi que leurs limites ;

— **Chapitre 2** Présentation et définition des termes Big Data et NoSQL et les différents concepts liés à ces deux termes ;

— **Chapitre 3** Présentation des différentes solutions Big Data et NoSQL existantes ainsi que les avantages et les inconvénients de chaque solution ;

— **Chapitre 4** Présentation des différentes étapes nécessaires pour la configuration, l'installation et le déploiement d'une solution Big Data suivi d'un programme test pour détailler le fonctionner de la solution choisi.

Chapitre 1

Les SGBDRs

1.1 Introduction

Dans ce chapitre, il sera présenté les différents systèmes de gestion de bases de données (SGBD), les SGBD relationnels, le schéma de données et le modèle de données. Ensuite on va présenter les différentes limites pratiques et théoriques liées à l'usage des bases de données relationnelles qui ont poussé les grands acteurs du web comme Google et Facebook vers de nouvelles technologies de stockage et de manipulation des données.

1.2 Définition d'une base de données

Une Base de Données est un ensemble structuré de données, centralisées ou non, servant pour les besoins d'une ou plusieurs applications, interrogeables et modifiables par un groupe d'utilisateurs. D'une autre façon, une Base de Données est un ensemble d'informations exhaustives, non redondantes, structurées et persistantes, concernant un sujet [1].

Une base de données seule ne suffit donc pas, il est nécessaire d'avoir également [2]

— Un système permettant de gérer cette base ;

— Un langage pour transmettre des instructions à la base de données (par l'intermédiaire du système de gestion).

1.2.1 Objectifs des Bases de Données

Une base de données répond aux exigences suivantes [3] :

— Centraliser l'information.

— Rendre indépendant les données et les traitement.

— Partage des données .

— Respecte l'intégrité et la cohérence des données.

— Sécurise l'information.

1.3 Les systèmes de gestion de bases de données (SGBD)

1.3.1 Définition

Un Système de Gestion de Bases de Données (SGBD) est un logiciel de haut niveau permettant aux utilisateurs de structurer, d'insérer, de modifier, de rechercher de manière efficace des données spécifiques qu'elle sont stockées et partagée de manière transparente par plusieurs utilisateurs. Plus précisément, les systèmes de gestion de bases de données (SGBD) sont des programmes permettant à l'utilisateur de créer et de gérer des bases de données [4].

SGBD se décompose en trois sous systèmes [5] :

— Système de gestion de fichier : Il permet le stockage d'informations sur support physique ;

— Le SGBD interne : Il gère l'ordonnance des informations ;

— Le SGBD externe :Il représente l'interface avec l'utilisateur.

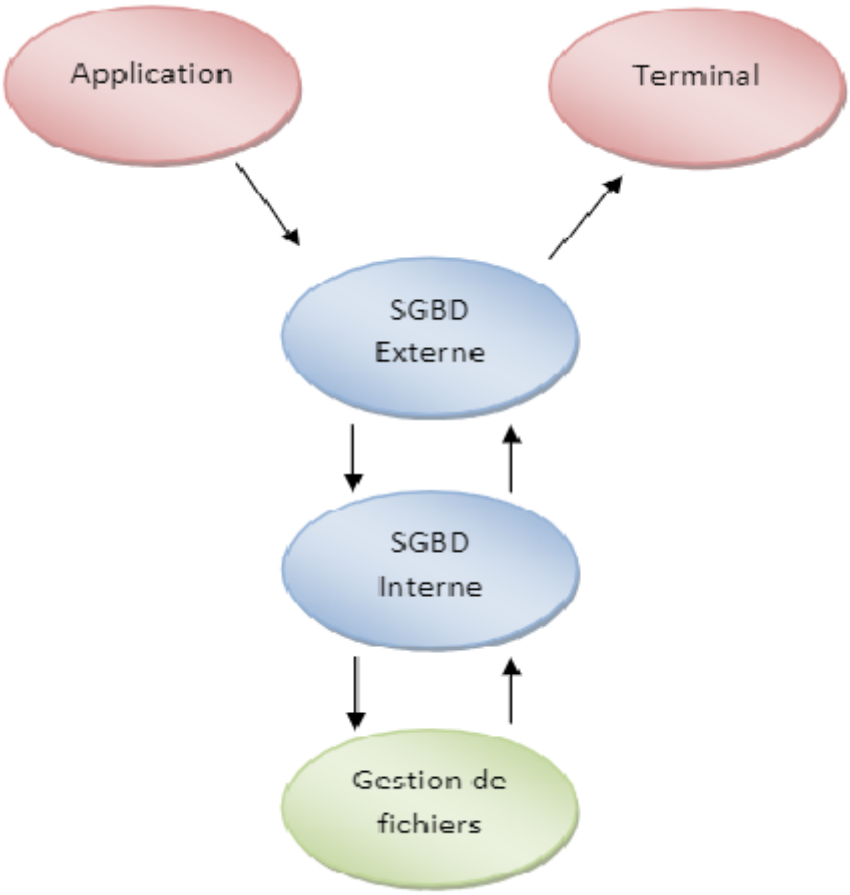


FIGURE 1.1 – les sous systèmes d'un SGBD [17]

1.3.2 Fonctionnalités d'un SGBD

Indépendance physique : le niveau physique peut être modifié indépendamment du niveau conceptuel.Cela signifie que la façon de définir les données doit être indépendante des structures utilisées pour leur stockage [1].

Indépendance logique : un utilisateur doit percevoir seulement la partie des données qui l'intéresse (une vue) et modifier la structure de celle-ci sans remettre en cause la majorité des applications [1].

Administration centralisée des données : le SGBD doit offrir aux administrateurs de données les outils pour vérifier la cohérence des données, l'éventuelle restructuration, la sauvegarde ou la réplication de la base de données. L'administration est centralisée et assurée seulement par des utilisateurs spécifiques pour des raisons de sécurité [6].

Manipulation des données : SGBD permet de création, recherche, suppression,

modification des données [7] .

Limitation de la redondance : le SGBD doit éviter l'informations redondantes, chaque donnée ne doit être présente qu'une seule fois dans la base, afin d'éviter d'une part un gaspillage d'espace mémoire mais aussi des erreurs [1].

Rapidité des accès : le système doit fournir les réponses aux requêtes le plus rapidement possibles, cela implique des algorithmes de recherche rapides [6].

Partage des données : le SGBD doit permettre un accès multi-utilisateur simultané aux mêmes données stockées dans la base de données [1].

Assurer la Sécurité de données : le SGBD doit présenter des mécanismes permettant de gérer les droits d'accès aux données selon les utilisateurs. Et robustesse vis-à-vis des pannes [1].

Respecter l'Intégrité de données : les données ne doivent présenter ni ambiguïté, ni incohérence, pour pouvoir délivrer sans erreur les informations désirées [1].

1.3.3 Niveaux et modèles de description des SGBD

1.3.3.1 Niveaux de description des SGBD

Trois niveaux de description (ou d'abstraction) des données sont définis (norme ANSI/SPARC) :

Niveau interne (ou physique) : le niveau interne correspond à la structure de stockage supportant les données. La définition du schéma interne nécessite au préalable le choix d'un SGBD. Elle permet donc de décrire les données telles qu'elles sont stockées dans la machine [8].

Niveau conceptuel (ou logique) : décrit la structure des données dans la base, leurs propriétés et leurs relations indépendamment de son implantation, c'est ce que l'on appelle le schéma conceptuel [7] [1] ; Langages de description des données (LDD) ; Consultation et mise à jour des données : Langages de Requêtes (LR) et Langage de Manipulation de Données (LMD) [4].

Niveau externe : décrit comment chaque utilisateur perçoit les données, c'est ce que l'on appelle le schéma externe ou vue, Le concept de vue permet d'obtenir l'indépendance logique.. Alors qu'au niveau conceptuel et interne les schémas décrivent toute une base

de données, au niveau externe ils décrivent simplement la partie des données présentant un intérêt pour un utilisateur ou un groupe d'utilisateurs [8] [1].

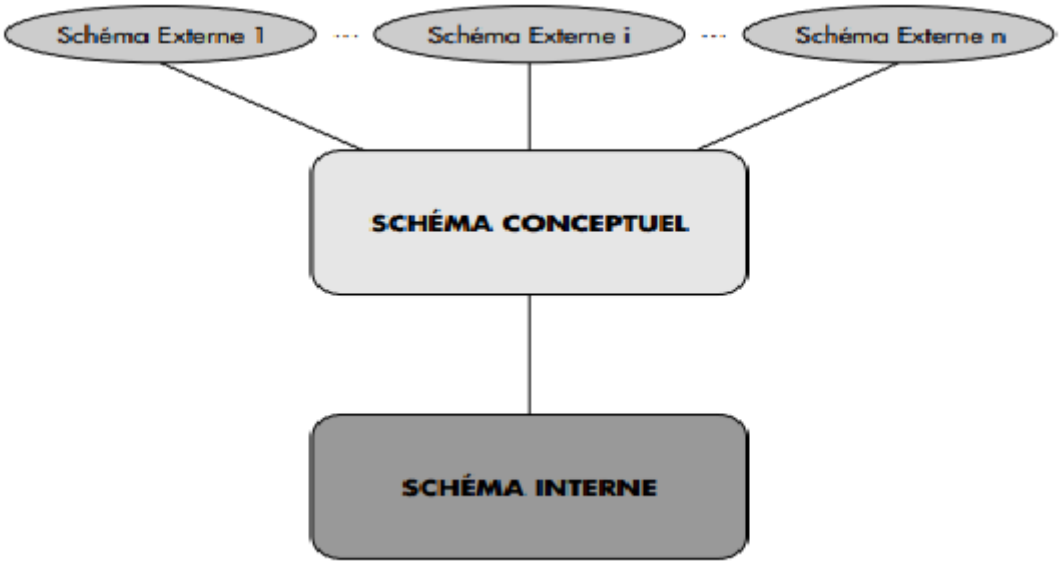


FIGURE 1.2 – Niveaux de représentation des données [8]

1.3.3.2 Modèles de description des SGBD

Les différents types de modèles sont utilisés pour La description des données, cette description des données est un résultat de la conception d'une base de données (appelée schéma) en termes de propriétés d'ensembles d'objets et d'organisation logique des données. Pour obtenir une telle représentation à partir d'un problème réel, on utilise un outil appelé modèle de description, basé sur un ensemble de concepts et de règles formant le langage de description. Un SGBD peut être caractérisé par le modèle de description qu'il supporte. Une fois la base de données ainsi spécifiée, il est possible de manipuler les données en réalisant des opérations de sélection, d'insertion, de modification et de suppression, et ce au moyen d'un langage spécifique de manipulation de données, mais aussi par des langages de programmation "classiques". Les modèles de La description sont [1] :

Hiérarchique : une base de données est vue comme un ensemble de fichiers reliés par des pointeurs, base de données construite selon un modèle en arbore arborescence, avec

une racine et plusieurs niveaux de sous arbres ; Chaque élément comporte juste un lien vers le niveau inférieur ; Les accès aux données commencent par la racine et descendent l'arborescence jusqu'aux détails recherchés. [1] [9]

Modèles réseau : A l'inverse du modèle hiérarchique, dans le modèle réseau toutes les entités représentées sont au même niveau et non décomposées hiérarchiquement. Pour décrire les entités, le modèle réseau offre aussi la notion d'enregistrement qui peut récursivement contenir d'autres enregistrements. En effet, un enregistrement peut avoir comme valeur d'un de ses attributs, un autre enregistrement. Un attribut peut non seulement être structuré en un nouvel enregistrement mais peut aussi être multi-value c'est à dire structuré sous forme d'un vecteur de valeurs atomiques ou structurées. [9]

Modèle relationnel : Le modèle relationnel est basé sur une organisation des données sous forme de tables. Permet une grande indépendance et entre les représentations logique et physique . La manipulation des données se fait selon le concept mathématique de relation de la théorie des ensembles, c'est-à-dire l'algèbre relationnelle. Supporte le langage SQL Les opérations relationnelles permettent de créer une nouvelle relation (table) à partir d'opérations élémentaires sur d'autres tables.
On représente ainsi les données dans des objets "table" à deux dimensions. Les colonnes (attributs) représentent les valeurs d'un type particulier pour un domaine particulier (ex :colonne "nom" de type varchar). Les lignes contiennent les enregistrements (tuples) de la table. Chacun d'entre eux possède un identifiant qui garantit son unicité. [10] [11].

Modèle Entité/Association : le modèle E/A est un modèle graphique permettant de décrire des concepts qu'on sera amené à manipuler dans la base de données. Il permet de distinguer les entités qui constituent la base de données, et les associations entre ces entités. Chaque entité possède des propriétés particulières appelées attributs. Chaque attribut peut prendre une (ou plusieurs) valeur(s). Ces concepts permettent de donner une structure à la base, ce qui s'avère indispensable. [10]

Modèle objet (SGBDOO, Système de Gestion de Bases de Données Orienté

Objet) : ce modèle vient de la conception orientée objet qui est apparu à la fin de 1980. Les données sont stockées dans des structures appelées objets de façon persistante contrairement aux objets des langages de programmation orientés objets. Afin d'assurer la portabilité des interfaces d'un SGBD objet à un autre. L'ODMG a proposé un modèle abstrait de données : OQL-Object Query Language qui a été ensuite implémenté pour les langages de programmation orienté objet, tels que C++, Smalltalk et Java .[12]

1.4 SGBDR (Systèmes de Gestion de Bases de Données Relationnelles)

Un SGBDR est un SGBD qui implémente la théorie relationnelle, les données sont contenues dans ce qu'on appelle des relations, qui sont représentées sous forme de tables. Une relation est composée de deux parties, l'en-tête et le corps [2] [11]

— Basés sur le modèle relationnel

— Un langage de requêtes standard : SQL

Parmi les logiciels SGBDR les plus connus et utilisés il est possible de citer : MySQL, PostgreSQL, Oracle, Microsoft SQL Server...

Tous les SGBDR présentent à peu près les mêmes fonctionnalités. Ils se distinguent par [2] :

— Leur coût.

— Le volume de données qu'ils sont capables de gérer.

— Le nombre d'utilisateurs qui peuvent interroger la base simultanément.

— La facilité avec laquelle ils s'interfacent avec les autres logiciels d'application.

1.4.1 Langages de manipulation des données relationnelles

Les langages utilisés dans les bases de données relationnelles se fondent sur l'algèbre relationnel et/ou le calcul relationnel. Ce dernier est basé sur la logique des prédicats

(expressions d'une condition de sélection définie par l'utilisateur). On peut distinguer deux grandes classes de langages [1] :

1.4.1.1 Langages algébriques ou algèbre relationnelle

L'algèbre relationnelle est le langage interne d'un SGBD relationnel. Ce langage consiste en un ensemble d'opérations qui permettent de manipuler des relations (tables et de colonnes). Son principe repose sur la création de nouvelles tables à partir des tables existantes, ces nouvelles tables devenant des objets utilisables immédiatement.
Cette algèbre se compose d'opérateurs de manipulation des relations. Ces opérateurs sont regroupés en deux familles : les opérateurs ensemblistes et les opérateurs relationnels. Chacune de ces familles contient quatre opérateurs.[13] [4]

1.4.1.2 Calcul relationnel

Construit à partir de la logique des prédicats il en existe deux types[1] :
le calcul de tuples : construits à partir à partir de la logique des prédicats, et dans lesquels les variables manipulées sont des tuples (ex : Query Language).
*c*alcul de domaine : également construit à partir de la logique des prédicats, mais en faisant varier les variables sur les domaines des relations.

1.4.2 Langage SQL (Structured Query Language)

Le SQL (Structured Query Language) il a été créé au milieu des années 70 par IBM et commercialisé par ORACLE en 1979. Est un langage de requête utilisé pour manipuler les bases de données relationnelles. Le langage est reconnu par la grande majorité des systèmes de gestion de bases de données relationnelles (SGBDR) : DB2, Oracle, Informix, Ingres, SQLServer, ACCESS. Chaque SGBDR a cependant sa propre variante du langage[13] [14]

Ses principales caractéristiques sont [15] :

1. Normalisation : SQL implémente le modèle relationnel.

2. Standard : la plupart des éditeurs de SGBDR intègrent SQL à leurs produits (Oracle, MS SQL Server, DB2, etc.) pour que les données, requêtes et applications soient facilement portables d'une BD à une autre.

3. Non procédural : SQL est un langage déclaratif non procédural permettant d'exprimer des requêtes dans un langage relativement simple. En contrepartie il n'intègre aucune structure de contrôle permettant par exemple d'exécuter une boucle itérative.

Les instructions SQL :

Sont regroupées en catégories en fonction de leur utilité et des entités manipulées :

La définition des données : SQL est un langage de définition de données (LDD), c'est-à-dire qu'il permet de : [3] [13]

- Créer une table : CREATE TABLE
- Créer une vue :CREATE VIEW
- Supprimer une table ou une vue DROP TABLE / VIEW
- Modifier une table ou une vue ALTER TABLE / VIEW.
- Renommer une table : RENAME

La manipulation de données : SQL est un langage de manipulation de données (LMD), cela signifie qu'il permet de [3] [14]

- Sélectionner des données dans la table : SELECT
- Insérer des données : INSERT INTO
- Modifier des données UPDATE...SET...WHERE
- Supprimer des données DELETE FROM...WHERE

SQL est un langage de protections d'accès : il est possible avec SQL de définir des permissions au niveau des utilisateurs d'une base de données. On parle de DCL (Data Control Language). [13]

Cette sous catégorie des commandes SQL définit le contrôle de données [3] :
GRANT : autorisation d'un utilisateur à effectuer une action ;
DENY : interdiction à un utilisateur d'effectuer une action ;
REVOKE : annulation d'une commande de contrôle de données précédente ;
COMMIT : validation d'une transaction en cours ;
ROLLBACK : annulation d'une transaction en cours ;
LOCK : verrouillage sur une structure de données.

1.4.3 Propriétés ACID

Les SGBD de type hiérarchique, réseau et relationnel fonctionnent selon les contraintes dites ACID (Atomicité, Cohérence, Isolation, Durabilité). Une transaction est réalisée avec succès si elle respecte ces quatre contraintes [5]

Atomicité : cela signifie que les mises à jour de la base de données doivent être atomiques, c'est-à-dire Lorsqu'une transaction est effectuée, elle doit être exécutée entièrement ou pas du tout , en effet, en cas d'échec d'une seule des opérations, toutes les opérations précédentes doivent être complètement annulées, peu importe le nombre d'opérations déjà réussies[5].

Cohérence : les données d'une base doivent toujours être dans un état coherent Avant et après l'exécution d'une transaction. Si le contenu final d'une base de données contient des incohérences, cela entraînera l'échec et l'annulation de toutes les opérations de la dernière transaction. Le système revient au dernier état cohérent. La cohérence est établie par les règles fonctionnelles[5].

Isolation : toute transaction doit s'exécuter comme si elle était la seule sur le système. Aucune dépendance possible entre les transactions. La propriété d'isolation assure que l'exécution simultanée de transactions produit le même état que celui qui serait obtenu par l'exécution en série des transactions. Chaque transaction doit s'exécuter en isolation totale : si A et B s'exécutent simultanément, alors chacune doit demeurer indépendante de l'autre [11].

Durabilité : une fois une transaction confirmée, les données correspondantes elles demeures sauvegardées dans la base même à la suite d'une panne d'électricité, d'une panne de l'ordinateur ou d'un autre probléme [11]

1.4.4 Limites des SGBD relationnels

Le modèle relationnel est fondé sur des concepts simples qui font sa force, en même temps sa faiblesse. Et ce modèle trouve ces limites pour certaines site web de grande ampleur, telles que Google, Facebook... [16] [17]. Les principales limites des SGBD Relationnels, sont présentées dans ce qui suit :

1.4.4.1 Problème de requête non optimale suite à l'utilisation des jointures

Imaginons qu'on a une table contenant toutes les personnes ayant un compte sur Facebook, soit plus de 800 millions, les données dans une base de données relationnelle classique sont stockées par lignes, ainsi si on effectue une requête pour extraire tous les amis d'un utilisateur donné, il faudra effectuer une jointure entre la table des usagers (800 millions) et celle des amitiés (chaque usager ayant au moins un ami donc au minimum on a 800 millions), puis on va parcourir le produit cartésien de ces deux tables .De ce fait, on perd en performances car il faut du temps pour stocker et parcourir une telle quantité de données[16].

1.4.4.2 Type de données

Les systèmes relationnels sont limités à des types simples (entiers, réels, chaines de caractères), Les seuls types étendus se limitant à l'expression de dates ou de données financières, ainsi que des conteneurs binaires de grande dimension (BLOB, pour Binary Large OBjects) qui permettent de stocker des images ainsi que des fichiers audio ou vidéo. Ces BLOBs ne sont toutefois pas suffisants pour représenter des données complexes non structurées. D'un côté les mécanismes de contrôle de ces types de données sont inexistants, d'un autre côté, le langage de requêtes (SQL) ne possède pas les opérateurs correspondant aux objets stockés dans ces BLOBs[17]

1.4.4.3 Langage de manipulation

Il est limité aux opérateurs de base de langage SQL aves ces différentes versions, qu'il n'est pas possible d'étendre à de nouveaux operateurs.[17]

1.4.4.4 Scalabilité limitée

Atteinte par les groupes comme Yahoo, Google, Facebook, etc. Cette limite est la plus gênante pour les entreprises. Les SGBD Relationnels ne sont pas adaptés aux environnements distribués requis par les volumes gigantesques de données et par les trafics tout aussi gigantesques générés par ces sites.[17].

1.4.4.5 Problème lié à l'application des propriétés ACID en milieu distribué

Il existe par ailleurs, une limitation plus théorique à ce que peuvent réaliser les bases de données relationnelles distribuées qui est décrite par les propriétés ACID. Ces propriétés bien que nécessaires à la logique du relationnel, elles nuisent fortement aux performances et en particulier à la propriété de cohérence. Cette dernière est très difficile à mettre en place dans les environnements distribués à plusieurs serveurs, puisque pour l'assurer, il faut que les serveurs doivent être des miroirs les uns des autres ; alors deux problèmes apparaissent :

— Le coût en stockage est énorme car chaque donnée est présente sur chaque serveur ;

— Le coût d'insertion/modification/suppression est très grand, la transaction n'est validée que si elle s'effectue sur tous les serveurs, ce qui fait patienter l'utilisateur durant ce temps [16].

1.5 Conclusion

Les SGBD relationnels montrent leur limite avec de très hauts débits et des données de types qui ne sont pas compatibles avec les schémas rigides du modèle relationnel. Une augmentation exponentielle des volumes de données qui se compte désormais en péta-octets. La gestion de ces volumes de données est donc devenue un problème que les bases de données relationnelles ne sont plus en mesure de gérer. A cause de ces limitations, dans le chapitre suivant nous avant présenter des nouvelles technologies ont été créées pour

répondre à ces besoins. De ce fait, et face à ces contraintes, des nouveaux systèmes de gestion de données nommés «NoSQL» viennent d'imposer en proposant des alternatives au modèle relationnel.

Chapitre 2

Big Data et NoSQL

2.1 Introduction

Avec le développement de l'Internet, du cloud computing, de l'Internet des Objets et des appareils mobiles, le Big Data est devenu très important et crucial.
Lorsqu'on parle de manipulation de gros volume de données, on pense généralement à des problématiques sur le volume des données et sur la rapidité de traitement de ces données. Une nouvelle organisation des données qui répond mieux aux contraintes de disponibilité de l'information face à l'augmentation exponentielle des données.
Par conséquent, ce chapitre concentre sur le concept de big data ainsi que les solutions NoSQL.

2.2 Big Bata

2.2.1 Définition de Big Data

Comme l'un des termes les plus utilisé sur le marché aujourd'hui, il n'y a pas de consensus quant à la façon de définir le Big Data. Le terme est souvent utilisé comme synonyme de concept associé tel que Business Intelligence (BI) et data mining [18], pour cela il existe plusieurs définitions d'un point de vue différent. La plus populaire c'est celle de la TechAmerica Foundation : « Big Data est un terme qui décrit des grands volumes de données à haute vitesse, complexes et variables qui requièrent des techniques et des technologies avancées pour permettre la capture, le stockage, la distribution, la gestion et l'analyse de l'information » [19]

2.2.2 Caractéristiques des Big Data

Le Big Data se caractérise par des termes qui commencent par la lettre « V ». Cependant, tous ne sont pas d'accord sur le nombre de « V » qu'il faut considérer.

2.2.2.1 Les « V » de base

Ce sont les «3V » (Volume, Vélocité, Variété) qui fournissent un cadre permettant de comprendre l'émergence du Big Data.

1)Le volume : le volume est largement accepté comme une caractéristique clé du Big Data[20]. Le volume ou la taille des données est maintenant supérieur à téraoctets et pétaoctets due au grand nombre des dispositifs technologiques dont nous faisons usage aujourd'hui. La grande échelle et la montée des données dépassent les techniques traditionnelles de stockage et d'analyse [24]

2)La variété : les données volumineuses proviennent d'une grande variété de sources telles que des capteurs, des objets intelligents, Internet et les technologies de collaboration sociale et sont généralement de trois types : structurées, semi-structurées et non structurées. [21]

Données structurées : il fait référence aux ensembles de données ayant un haut niveau d'organisation, en rendant facile leur inclusion dans une basse de données traditionnel et aussi permettant de les consulter avec des algorithmes simples, des moteurs de recherche ou d'autres opérations de recherche conventionnelles [22].

Données semi-structurées : ce sont de données qui ne s'adaptent pas complètement à la structure formelle d'une basse de données relationnel. Mais, elles ont des étiquettes ou des autres marqueurs qui vont permettre de les classer dans les champs nécessaires [22].

Données non-structurées : c'est le type de données le plus représentant pour l'analyse du Big Data, elles sont aléatoires et difficile à analyser. On trouve ici l'information dans une variété de formats différents tel que le texte, audio, vidéo, pages Web... [23]

3)Vélocité : la vélocité désigne tout d'abord la vitesse à laquelle il est possible d'ac-

tualiser les analyses De grandes quantités de données, qui sont provenant de transactions avec un taux de rafraîchissement élevé entraînant des flux de données à grande vitesse et le temps nécessaire pour agir sur la base de ces flux de données seront souvent très courts. [25]

2.2.2.2 Toujours plus de « V »

1)Véracité : est insistant sur l'importance de s'intéresser à la qualité et à la provenance des données qui sont analysées. La structure devra intégrer un système de collecte qui n'engendre pas trop de bruit qui n'est pas trop élevé pour pouvoir lire vos Big Data.

Si on analyse par exemple les données disponibles sur des réseaux sociaux comme Facebook, il est important de s'assurer qu'il ne s'agit pas de rumeurs ou encore de diffusions malveillantes pour nuire à l'analyse[26].

2)Valeur : la valeur est dépendante des caractéristiques vues précédemment, donc il ne suffit pas d'avoir une énorme quantité donnée procédant de différentes sources à une grande vitesse de génération, il est nécessaire que cet ensemble d'information donne la vraie valeur pour l'utilisateur ou une entreprise après l'avoir exploitée. On dit donc que le Big Data prend une grande importance selon la valeur qu'il peut avoir pour son exploiteur[27].

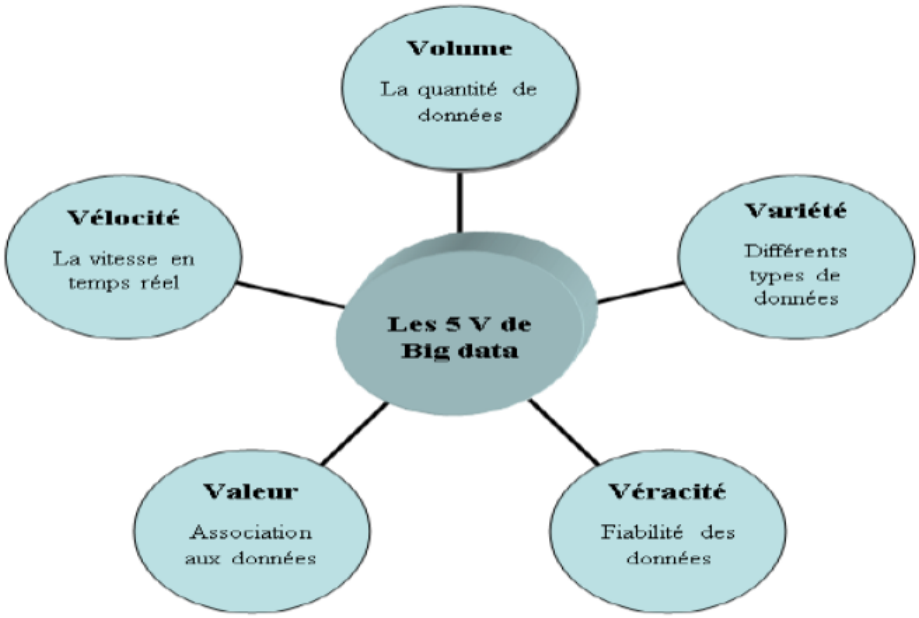


FIGURE 2.1 – Les 5V de Big Data

2.2.3 Intérêt des Big Data

L'utilisation de Big Data pourrait avoir un impact significatif sur le monde des affaires, pour que les entreprises pourront [34] :

— Améliorer la prise de décision.

— Réduire les coûts d'infrastructures informatiques via l'utilisation des serveurs standards et des logiciels open source.

— Développer la réactivité et l'interactivité avec les clients.

— Améliorer les performances opérationnelles.

2.2.4 Enjeux de big data

Trois problèmes fondamentaux qui doivent être abordées dans le traitement des données volumineuses : les problèmes de stockage, les problèmes de gestion et les problèmes de traitement. Chacun d'entre eux représente un vaste ensemble de problèmes de recherche technique [35] :

Problèmes de stockage

La quantité de données a explosé chaque fois qu'on a inventé un nouveau support de stockage. Ce qui est différent dans l'explosion la plus récente en grande partie due aux médias sociaux, c'est qu'il n'y a pas eu de nouveau support de stockage. De plus, des données sont créées par tout le monde. Le tableau ci-dessous montre les différents ensembles de données et leurs classifications.

Ensemble de données / domaine	**la description**
Communications Internet (Cisco)	667 exaoctets en 2013
médias sociaux	12+ Toctets de tweets tous les jours et en croissance.
nivers numérique humain	1,7 Zoctets en (2011), -¿ 7,9 Zoctets en 2015
British Library UK Website Crawl	110 Toctets par exploration de domaine à archiver
autre	compteurs électriques intelligents, 4,6 milliards de téléphones portables avec GPS

TABLE 2.1 – Les différents ensembles de données et leurs classifications[35]

Problèmes de gestion

La gestion sera peut-être le problème le plus difficile à résoudre avec des données massives. Ce problème est apparu il y a une dizaine d'années dans les initiatives eScience du Royaume-Uni où les données étaient distribuées géographiquement et «possédées» et «gérées» par plusieurs entités. La résolution des problèmes d'accès, de métadonnées, d'utilisation, de mise à jour, et de référence (dans les publications) s'est avérée être un obstacle majeur.

Problèmes de traitement

On ne sait pas comment accéder à de très grandes quantités de données semi-structurées ou non structurées et comment utiliser des modèles d'outils encore inconnus. Il est clair

que le problème ne peut être résolu par la modélisation dimensionnelle et le traitement analytique en ligne (OLAP), qui peuvent être lents ou avoir une fonctionnalité limitée, ou simplement en lisant toutes les données en mémoire. Les considérations techniques qui doivent être prises en compte dans la conception comprennent le rapport entre la vitesse de lecture des disques séquentiels et la vitesse de l'accès à la mémoire aléatoire.

2.2.5 Classification des Big Data

Les Big Data sont classés en différentes catégories pour mieux comprendre leurs caractéristiques. La classification est importante en raison des données à grande échelle dans le cloud. La classification est basée sur cinq aspects [33] :

Data sources (les sources de données)

Les médias sociaux : tel que facebook, twitter.

Les données générées par la machine : sont des informations générées automatiquement par un processus informatique, une application ou un autre mécanisme sans l'intervention active d'un humain.

Données de capteur : mesurer les quantités physiques et les transformer en signaux.

Le format de contenu

Concerne les données structurées, semi structurées et non structurées.

Entrepôt de données

Un entrepôt de données est une structure informatique dans laquelle est centralisé un volume important de données consolidées à partir des différentes sources de renseignements d'une entreprise et qui est conçue de manière que les personnes intéressées aient accès rapidement à l'information stratégique dont elles ont besoin [28].

Data staging

Une zone de transfert , ou zone d'atterrissage , est une zone de stockage intermédiaire utilisée pour le traitement des données pendant le processus d' extraction, de transformation et de chargement (ETL) [30], elle concentre sur :

Le nettoyage des données : permet de traiter, détecter et de supprimer les erreurs et les incohérences de données afin d'améliorer la qualité des données. [31]

Transformer : est le processus de transformation des données sous une forme appropriée pour l'analyse.

La normalisation : est la méthode de schéma de base de données de structuration afin de minimiser la redondance.[32]

2.2.6 Les usages de Big Data

Marketing

Le Big Data permet en effet aux professionnels du secteur de connaître leur client, par son parcours internet également par ses achats en magasin ou ses préférences affchées sur les réseaux sociaux, Anticiper leurs besoins pour cibler des offres personnalisées.[36]

Santé

Les perspectives de la recherche fondamentale et du ciblage des médicaments sont importantes. Les données sont essentielles à l'analyse des médicaments avant leur mise sur le marché. Les nouveaux appareils connectés qui mesurent en permanence notre rythme cardiaque, notre niveau de glycémie, les calories brulées, etc..., génèrent des flux d'information qui vont améliorer la prévention et réduire les coûts d'hospitalisation, en effectuant les mesures en ligne.[37]

Banque et finance

Une grande quantité d'applications Big Data tombe dans la catégorie de la classification et de la prédiction. Prenez les banques pour un exemple. Chaque jour, des millions de personnes demandent de nouvelles cartes de crédit, des prêts et des hypothèques. Dans le processus de prise de décision, les banques utilisent un numéro pour examiner les antécédents financiers d'une personne et évaluer sa probabilité de rembourser sa dette : un pointage de crédit. Ce score est calculé à partir de toutes les données que les banques connaissent à votre sujet. [36]

Recherche

Domaine d'application original du Big Data, Le Big Data permet à la science de réaliser des avancées importantes, lorsqu'il s'agit d'explorer des données complexes (ex : imagerie) ou d'effectuer des simulations (ex : domaine spatial). C'est d'ailleurs que le Big Data a

fait ses premières armes car ce secteur réclamait une approche à la fois quantitative et qualitative avancée.[37]

2.2.7 Les besoins principaux liés à l'émergence du Big Data

Parmis ces besoin, on cite :[11]

— La capacité à gérer une montée en charge horizontale.

— La parallélisassions des traitements des requêtes.

— Un schéma de données flexible.

— La capacité de stockage, traitement et lecture de données non structurées et semi-structurées.

— La performance des requêtes en lecture.

— La haute disponibilité des données.

2.3 Le NoSQL

2.3.1 Définition de NoSQL

Le terme « NoSQL » a été inventé en 2009, signifie 'non seulement SQL', c'est un système de gestion de base de données distribuées, non relationnel. L'unité logique n'y est plus la table, et les données ne sont en général pas manipulées avec le système générique SQL, évite généralement les opérations de jointures. Il permet de stocker et de récupérer des données de façon rapide.[38]

2.3.2 Pourquoi le NoSQL ?

Le besoin fondamental auquel répond le NoSQL est la performance. Afin de résoudre les problèmes liés à la gestion de grandes quantités de données « Big Data ». La gestion et le traitement de ces volumes de données sont considérés comme un nouveau défi de l'informatique, et les moteurs de bases de données relationnelles traditionnels, hautement transactionnels, semblent totalement dépassés. Ces données sont généralement non structurées, complexes et ne s'insèrent pas bien dans le modèle relationnel. [39] Voici quelques

exemples d'acteur concerné par ces gros volumes de données. Le réseau social Facebook c'est[5] :

— Plus de 845 millions de comptes à travers le monde.

— Plus de 2,7 milliards de « J'aime » et de messages postés par jour.

— Plus de 250 millions de photos « uploadées » chaque jour.

— Plus de 100 milliards de connexions par an.

2.3.3 Caractéristiques générales des BD NoSQL

Les caractéristiques des BD NoSQL sont [40] :

— Adoptent une représentation de données non relationnelle

— Ne remplacent pas les BD relationnelles mais sont une alternative, un complément apportant des solutions plus intéressantes dans certains contextes

— Apportent une plus grande performance dans le contexte des applications Web avec des volumétries de données exponentielle

— Utilisent une très forte distribution de ces données et des traitements associés sur de nombreux serveurs.

— Font un compromis sur le caractère « ACID » des SGBDR pour plus de scalabilité horizontale et d'évolutivité.

— Pas de schéma pour les données ou schéma dynamique.

— Données distribuées : partitionnement horizontal des données sur plusieurs noeuds (serveurs) généralement par utilisation d'algorithmes « MapReduce »

— Réplication des données sur plusieurs noeuds.

— Mode d'utilisation : peu d'écritures, beaucoup de lectures.

2.3.4 Le fonctionnement de NoSQL

2.3.4.1 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 [5].

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. Cette dernière ne peut pas assurer la flexibilité de système et Impossible de faire des mises à jour sans interrompre le système, donc auAucune tolérance à la panne. [5]

Scalabilité horizetale

Le principe de la scalabilité horizontale consiste à simplement augmenter l'échelle en ajoutant une nouvelle machine en tant que nouveau noeud du cluster. le système deviendrait robuste à mesure que de nouveaux noeuds seraient insérés pour répondre à l'augmentation de la charge ou de volumétrie de données[41]. La scalabilité horizentale assure la flexibilité du système et la possibilité de mises à jour sans interruption de service. Si un serveur tombe en panne ne peut pas pénaliser le système.

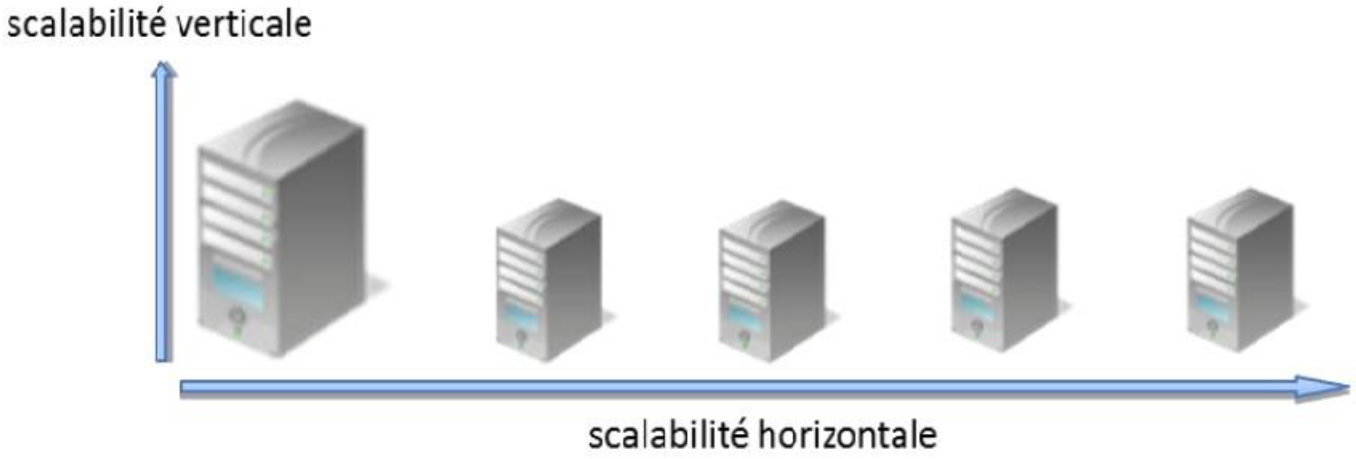


FIGURE 2.2 – Scalabilité verticale et horizontale [5]

2.3.4.2 Théorème de CAP

Le théorème de CAP est l'acronyme de « Coherence », «Availability » et « Partition tolérance ». Ce théorème énonce qu'une base de données d'un système distribué ne peut garantir les trois contraintes suivantes en même temps [35] :

- *Cohérence (Consistency) :* tous les noeuds (serveurs) du système voient exactement les mêmes données au même moment.
- *Haute disponibilité (Availability) :* garantie que toutes les requêtes reçoivent une réponse dans un délai raisonnable.
- *Tolérant à la partition (Partition Tolerence) :* le système doit répondre correctement à toute requête même durant le partitionnement.

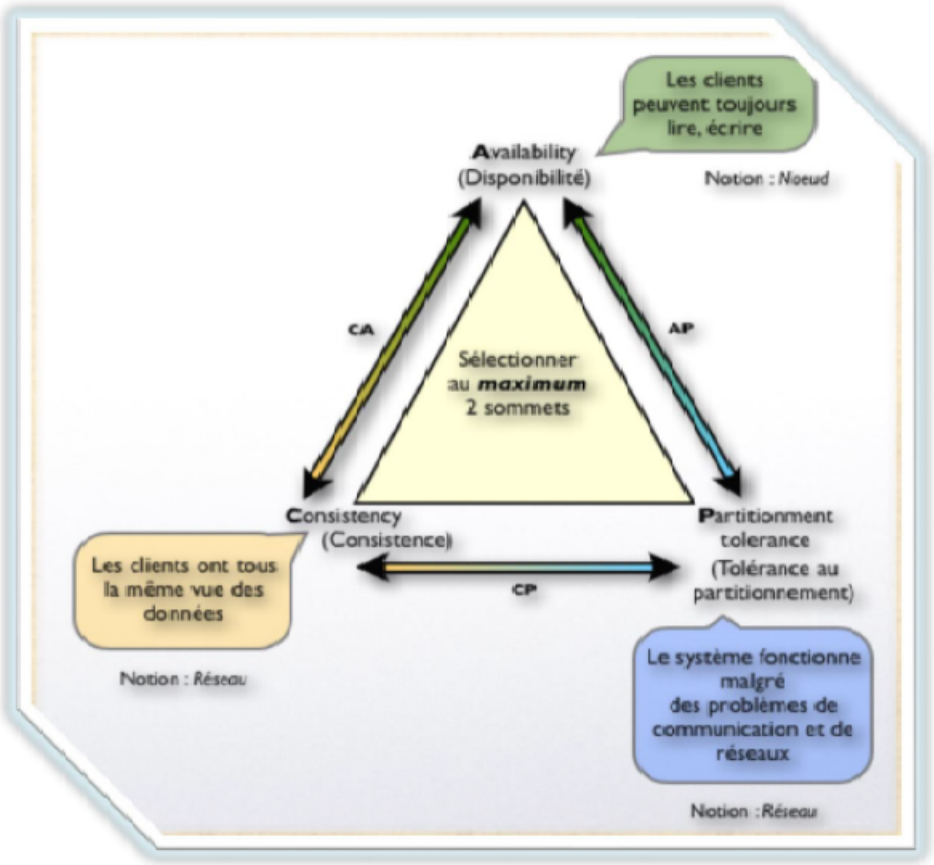


FIGURE 2.3 – Théorème CAP [42]

Comme mentionné plus haut, les bases de données NoSQL sont tenues de respecter deux principes sur trois. Elles choisiront un des trois cas en fonction des besoins :

— Cohérence et haute disponibilité (CA).

— Haute disponibilité et partition tolerance (AP).

— Cohérence et partition tolerance (CP).

2.3.4.3 Les propriétés de BASE

Les SGBD NoSQL, selon le théorème CAP, privilégient la disponibilité ainsi que la tolérance à la partition plutôt que la cohérence, répondent aux propriétés de BASE. Les caractéristiques de BASE (Basically Availabe, Soft-State, Eventuallty Consistent) sont fondées sur les limites que montrent les SGBD relationnelles [43]

— *Basically Available (Disponibilité basique) :* même en cas de désynchronisation ou de panne d'un des noeuds du cluster, le système reste disponible selon le théorème CAP.

— *Soft-state (Cohérence légère) :* indique que l'état du système peut changer à mesure que le temps passe, et ce sans action utilisateur.

— *Eventual consistancy (Cohérence à terme) :* spécifie que le système sera consisté à mesure que le temps passe, à condition qu'il ne reçoive pas une action utilisateur entre temps.

2.3.5 Différents types de base de données NoSQL

Base de données orientée documents : doivent être utilisées pour les applications dans lesquelles les données qui n'ont pas besoin d'être stockées dans une table avec des champs de taille uniforme, mais les données doivent être stockées en tant que document ayant des caractéristiques spéciales tel que MongoDB [44].

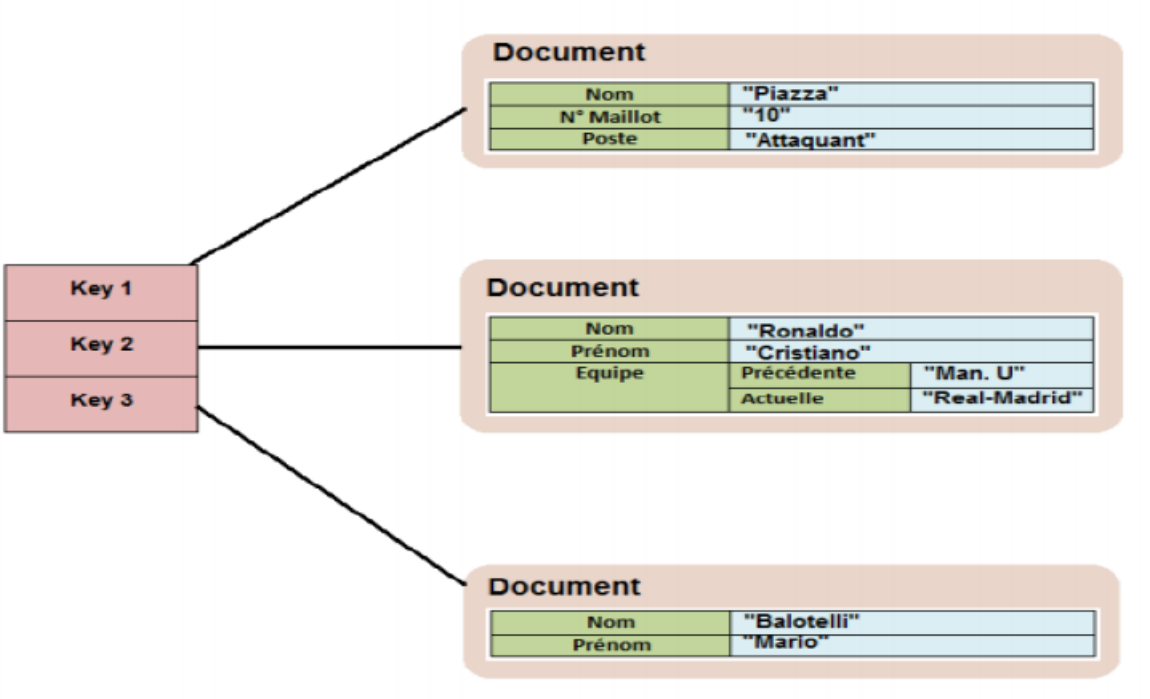


FIGURE 2.4 – Base de données orientée document [11]

Base de données orientées colonne : est une base de données qui stocke les données par colonne et non par ligne tel que HBase. [29]

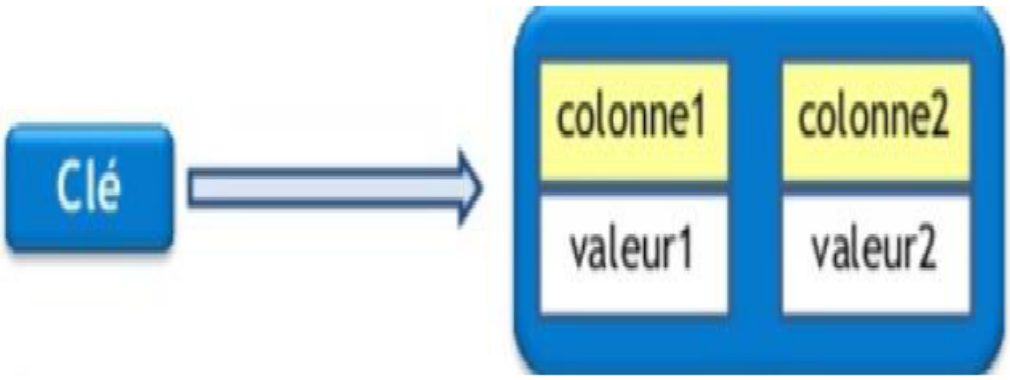


FIGURE 2.5 – Base de données orientée colonne[46]

Base de données orientées graphe : les données peuvent être représentées par un graphe avec des noeuds, des arêtes et des propriétés liées les unes aux autres par des relations tel que Neo4j. [29]

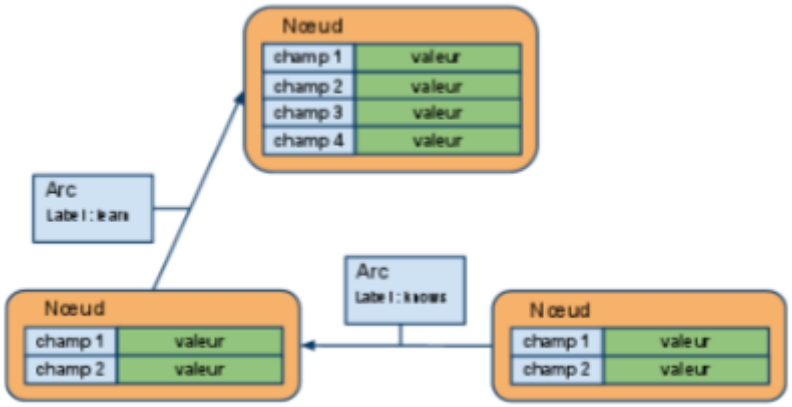


FIGURE 2.6 – Base de données orientées graphe(Neo4j)

Bases de données clé-valeur : les données sont représentées par un couple (clé, valeur), la valeur pouvant être une simple chaîne de caractères ou un objet tel que DynamoDB.[29]

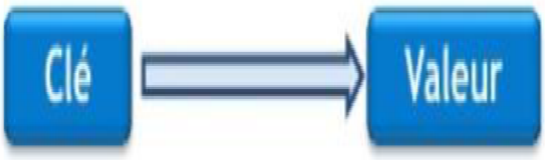


FIGURE 2.7 – Base de données clé-valeur[46]

2.3.6 Caractéristiques techniques

Les caractéristiques techniques principales sur lesquelles se repose le NoSQL :

L'architecture distribuée

Les moteurs NoSQL utilisent deux manières différentes pour distribuer les traitements et les données sur leurs divers serveurs. La distribution de données avec maître et celle sans maître.

Maître-esclave

Toutes les lectures et écritures arrivent au maître. Donc, Les requêtes des clients arrivent directement sur le serveur maître, pour ensuite être redirigées sur les serveurs « esclaves » qui contiennent l'information de la requête. Les esclaves prennent le relais et reçoivent des demandes seulement si le maître échoue. maximale. [46]

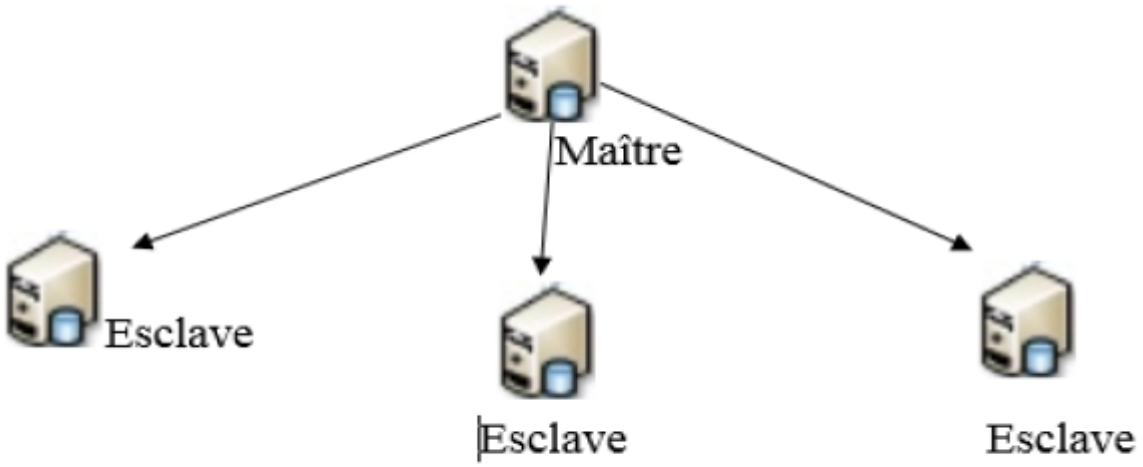


FIGURE 2.8 – Architecture maître esclave

Maître-Maître

Les lectures et les écritures peuvent se produire sur tous les noeuds gérant une clé. Les répliques maître-maître sont généralement cohérentes, le cluster effectuant une opération automatique pour déterminer la dernière valeur d'une clé et supprimer les anciennes valeurs périmées[46].

FIGURE 2.9 – Architecture maître / maître

2.3.7 Les avantages et les inconvénients du NoSQL

Les avantages

Les requêtes SQL et le modèle relationnel de bases de données restent souvent inefficaces dans l'optimalisation de la gestion de ces grandes données. En gros nous pouvons énumérer ci-dessous quelques avantages majeurs du NoSQL[45].

— Performance reconnue en termes de traitement de la volumétrie des données en ligne et de la montée en charge
— Bases de données de type NoSQL Open Source, donc facilité d'accès
— Multiplication et amélioration des offres des moteurs NoSQL
— Cohérence c'est-à-dire que tous les noeuds du système voient exactement les mêmes données au même moment
— Haute disponibilité qui permet de garantir l'accessibilité aux données en cas de panne ou de défaillance technique
— Les serveurs destinés aux bases de données NoSQL sont généralement bon marché et de faible qualité,
— Les bases de données NoSQL ne sont pas forcément moins complexes que les bases relationnelles, mais elles sont beaucoup plus simples à déployer.

Les inconvénients

Bien que le NoSQL soit un mouvement en mode avec beaucoup de bénéfice et de qualité, il existe néanmoins quelques petites difficultés dans son utilisation.Ces difficultés sont énumérées ci-dessous : [45]

— Le NoSQL est une technologie très jeune

— Il n'existe pas de langage d'interrogation standardisé

— Technologie adaptée à des besoins bien spécifiques

— La passage d'une base de données NoSQL vers une autre n'est pas transparente pour une application.

2.4 Conclusion

Le monde numérique se développe très vite et devient plus complexe dans le volume, la variété, la vitesse des données ce que signifié le Big Data. Un problème se pose alors quant au stockage et à l'analyse des données. La capacité de stockage des disques durs augmente mais le temps de lecture croît également. Les solutions NoSQL apparitaient ces dernières années pour solutionner ce problème, cça ce qu'on va vu dans le chapitre suivant.

Chapitre 3

Technologies des Big Data

3.1 Introduction

L'explosion quantitative des données a obligé les grands acteurs de big data comme Yahoo, Google et Facebook de trouver de nouvelles manières pour s'attaquer à ce phénomène, Il s'agit de concevoir de nouvelles technologies pour la capture, la recherche, le partage, le stockage, l'analyse et la présentation des données afin de contenir ces volumes énormes pour améliorer la performance et l'augmentation de la vitesse d'exécution des requêtes et des traitements.
Ce chapitre présente les différentes technologies les plus utilisées pour le Big Data

3.2 Les principales technologies de Big Data

Il existe différentes technologies qui peuvent entrer en jeu pour optimiser les temps de traitement sur des bases de données géantes. Elles peuvent globalement être catégorisées en deux familles :

3.2.1 Les technologies de stockage

La première solution permet d'implémenter les systèmes de stockage considérés comme plus performants que le traditionnel relationnel pour l'analyse de données en masse , le tableau suivant montre les technologies NoSQL :

famille	**Application**	**solution**
Clé-Valeur	Communication temps réel : messagerie en ligne	Riak, Redis, amazon dinamoDB
Orientée colonne	Donnée type liste	Cassandra, HBase, BigTable
Orientée document	Recherche d'information	MongoDB, CouchDB.
Orientée graphe	Réseaux sociaux	Neo4j, HypergraphDB

TABLE 3.1 – technologies NoSQL

Parmi les technologies de stockage les plus utilisées sont :

3.2.1.1 Apache Cassandra

A) Description

Apache Cassandra est un système de gestion de base de données distribuée open source conçu pour gérer une grande quantité de données sur de nombreux serveurs, offrant une haute disponibilité sans point de défaillance unique. C'est une solution NoSQL qui a été initialement développée par Facebook, elle offre une prise en charge robuste pour les clusters couvrant plusieurs DataCenter, avec une réplication asynchrone sans master permettant des opérations à faible latence pour tous les clients [47]

Auteur original	Avinash Lakshman, Prashant Malik
développeur	Apache Software Fondation
Première édition	2008
Version stable	1.1.5 / Septembre 10, 2012
Langage de programmation	Java
Type	Clé-valeur

TABLE 3.2 – Apache Cassandra [48]

B) Caractéristiques

Cassandra accorde beaucoup de valeur aux performances. cassandra présente les caractéristiques suivantes[48] :

Décentralisé

Chaque noeud du cluster a le même rôle. Il n'y a pas de point de défaillance unique. Les données sont distribuées à travers le cluster (donc chaque noeud contient des données différentes), mais il n'y a pas de maître car chaque noeud peut traiter n'importe quelle requête.

Prend en charge la réplication et la réplication multi-DataCenter

Les stratégies de réplication sont configurables. Cassandra est conçu comme un système distribué, pour le déploiement d'un grand nombre de noeuds dans plusieurs DataCenter. Les principales caractéristiques de l'architecture distribuée de Cassandra sont spécifiquement adaptées au déploiement de plusieurs DataCenter, à la redondance, au basculement et à la reprise après sinistre.

Évolutivité

Le débit en lecture et en écriture augmente linéairement au fur et à mesure que de nouvelles machines sont ajoutées, sans interruption ni interruption des applications. L'architecture des noeuds avec Cassandra se fait au travers ce qui est appelé un Hash Ring. Chaque noeud du cluster est disposé en file, dans un cercle directionnel. Lorsque l'on ajoute un nouveau noeud dans le cluster, ce dernier vient naturellement s'ajouter au Ring.

Tolérance aux pannes

Les données sont automatiquement répliquées sur plusieurs noeuds pour la tolérance aux pannes. La réplication entre plusieurs centres de données est prise en charge. Les noeuds défaillants peuvent être remplacés sans interruption.

Cohérence ajustable

Les écritures et les lectures offrent un niveau de cohérence réglable (one, quorum, all) [51] :

En lectures :

-*One :* le coordinateur reçoit la réponse du réplica le plus proche et la renvoie au client. Cette stratégie assure une haute disponibilité, mais au risque de renvoyer une ressource qui n'est pas synchronisée avec les autres réplicas. Dans ce cas, la cohérence des données n'est pas assurée.
-*Quorum :* le coordinateur reçoit la réponse de au moins la moitié des réplicas (somme du facteur de réplication / 2) + 1 , et renvoie au client la ressource avec le TimeStamp le plus récent. C'est la stratégie qui représente le meilleur compromis
-*All :* le coordinateur reçoit la ressource de tous les réplicas. Si un réplica ne répond pas, alors la requête sera en échec

En écritures :

-*One :* la réponse au client sera assurée si la ressource a été écrite sur au moins un des réplicas.
-*Quorum :* la réponse au client sera assurée si la ressource a été écrite sur au moins la moitié des réplicas.
-*All :* la réponse au client sera assurée lorsque la ressource aura été écrite dans tous les réplicas.

C)Architecture de Cassandra

Cassandra possède un seul mode de fonctionnement principal, qui peut se décrire comme une collection de noeuds indépendants qui sont interconnectés entre eux pour former un cluster[74].

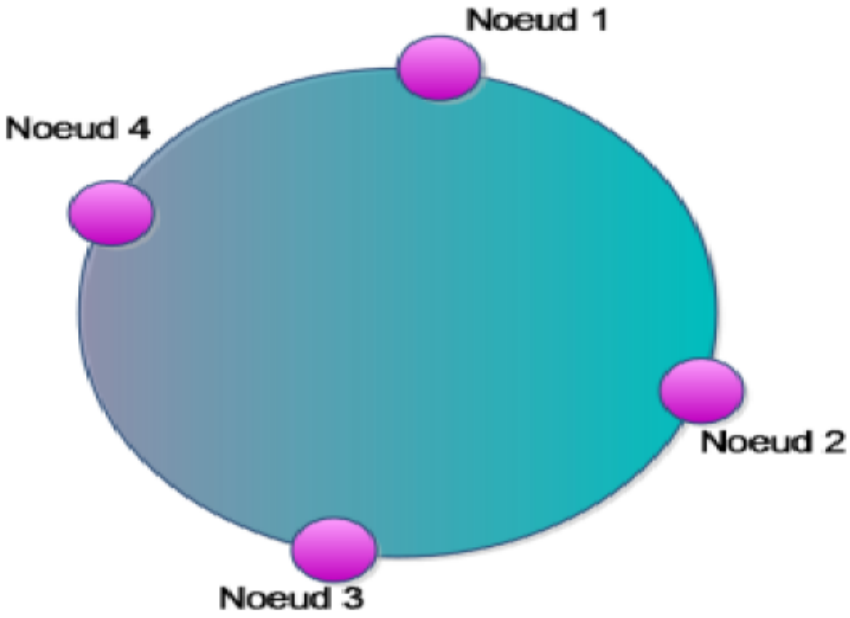


FIGURE 3.1 – architecture de Cassandra[74]

D)Réplication

Cassandra propose deux stratégies de réplication[74] :

— SimpleStrategy

— NetworkTopologyStrategy.

La principale différence entre ces deux stratégies réside dans le fait que la première s'utilise quand les noeuds du cluster se trouvent au sein du même Datacenter, alors que la seconde est optimisée pour la gestion de la réplication quand les noeuds se trouvent dans plusieurs Datacenter.

E)Structure de Cassandra

Il existe trois termes à connaître [74] :

— Keyspace : cette dernière peut s'apparenter à une base de données dans le monde relationnel. Comme une base de données classique, un serveur peut posséder plusieurs Keyspace.

— Famille de colonnes : toujours pour faire l'analogie avec le relationnel, une famille de colonnes se rapproche des tables telles qu'on les connait. C'est l'objet qui contiendra les données.

— Colonne : la colonne est le plus petit objet, et est définie par un nom de colonne, une valeur ainsi qu'un Timestamp.

F) Les avantages

Parmi les avantages de Cassandra [50] :

— Il n'a pas un seul point de défaillance (grâce aux mécanismes de réplication de données).

— Décentralisé.

— Élastique.

— Haute disponibilité.

— Open Source.

G) Les inconvénients

Les inconvénients majeurs de Cassandra sont [44] :

— Lenteur (les lectures sont comparativement plus lentes que les écritures).

— Pas d'interface graphique.

— Limitation de la taille des données.

— Cassandra a besoin d'une certaine structure (ne traite pas les images)

3.2.1.2 MongoDB

A) Description

Est la première base de données NoSQL, orientée document, sans schéma, qui stocke les données au format BSON. Un document basé sur un JSON, BSON est un format binaire qui permet une intégration rapide et facile des données avec certains types d'applications. MongoDB fournit également une évolutivité horizontale et n'a pas de point de défaillance unique. Un cluster MongoDB est différent d'un cluster Cassandra car il comprend un arbitre, un noeud maître et plusieurs noeuds esclaves. Depuis 2009, MongoDB est un projet open source avec licence AGPL détenu par la société 10gen.company[52].

Développé par	MongoDB, Inc
Première édition	2009
Version stable	V 4.0.1
Langage de programmation	C++
Type	Orientée document
Site web	www.mongodb.com

TABLE 3.3 – MongoDB[52]

B) Caractéristique de MongoDB

MongoDB a plusieurs caractéristiques parmi eux : [53]

Modèle de données de document :

Le modèle de données de MongoDB est orienté document, les données sont modélisées sous forme de documents JSON. Le modèle de type document réduit au maximum le nombre de relation dans la base de données

```
mayada
{
_id: ObjectID('mima1994'),
title: 'bdd NoSql',
url: 'http://example.com/databases.txt',
author: 'mayada',
vote_count: 20,
tags: ['databases', 'mongodb', 'cassandra'],
image: {
url: 'http://example.com/db.jpg',
caption: 'A database.',
type: 'jpg',
size: 75381,
data: 'Binary'
},
```

FIGURE 3.2 – document JSON

Le champ –id : représente la clé primaire

Tags : Tags stockés en tant que tableau de chaînes

Image : attribut pointant vers un autre document

En interne, MongoDB stocke les documents dans un format appelé Binary JSON ou BSON. BSON a une structure similaire, mais est destiné à stocker de nombreux

documents

MongoDB a des collections. En d'autres termes, MongoDB conserve ses données dans des collections de documents, que vous pouvez considérer comme un groupe de documents. Les collections sont un concept important dans MongoDB.

Requêtes ad hoc

Dire qu'un système supporte des requêtes ad hoc, c'est dire qu'il n'est pas nécessaire de définir à l'avance quelles sortes de requêtes le système acceptera. Un des objectifs de conception de MongoDB est de préserver la majeure partie de la puissance de requête qui a été si fondamentale dans le monde des bases de données relationnelles.

Une fois l'ensemble de données atteint une certaine taille, les requêtes ad hoc seules ne suffisent pas, les index deviennent nécessaires pour l'efficacité de la requête. Les index appropriés augmenteront les vitesses de requête et de tri ; par conséquent, tout système qui prend en charge les requêtes ad hoc doit également prendre en charge les index secondaires.

Index secondaires

Les index secondaires dans MongoDB sont implémentés en tant que B–trees. En autorisant plusieurs index secondaires, MongoDB permet aux utilisateurs d'optimiser pour une grande variété de requêtes en donnant la possibilité d'écrire des requêtes liées aux coordonnées géographiques. Avec MongoDB, on peut créer jusqu'à 64 index par collection. Les types d'index pris en charge incluent les index ascendants, descendants, uniques, composés-clés.

Réplication

MongoDB fournit une réplication de base de données via une topologie connue sous le nom de jeu de réplicas. Les ensembles de réplicas répartissent les données entre les machines pour la redondance et automatisent le basculement en cas de pannes de serveur et de réseau. De plus, la réplication est utilisée pour mettre à l'échelle les lectures de base de données. Si on a une application intensive en lecture, comme c'est généralement le cas sur le Web, il est possible de répartir les lectures de bases de données sur les machines du cluster de réplicas.

C)Architecture MongoDB

MongoDB possède différents modes de fonctionnement [74] :

Single : le mode Single, qui ne sert à faire fonctionner une base de données que sur un seul serveur.

Réplication : elle est découpée en deux modes :

Master / Slave : un serveur officie en tant que maître et s'occupe des demandes des clients. A côté de cela, il s'occupe de répliquer les données sur le serveur esclave de façon asynchrone. L'esclave est présent pour prendre la place du maître si ce dernier venait à tomber.

Replica Set : le Replica Sets fonctionne avec plusieurs noeuds possédant chacun la totalité des données, de la même manière qu'un réplica. Ces différents noeuds vont alors élire un noeud primaire, qui peut s'apparenter à un maître. Il faut qu'un noeud obtienne la majorité absolue afin d'être élu.

Sharding : le Sharding sert à partager les données entre plusieurs Shard, chaque Shard devant stocker une partie des données.

D)Les principaux composants de MongoDB

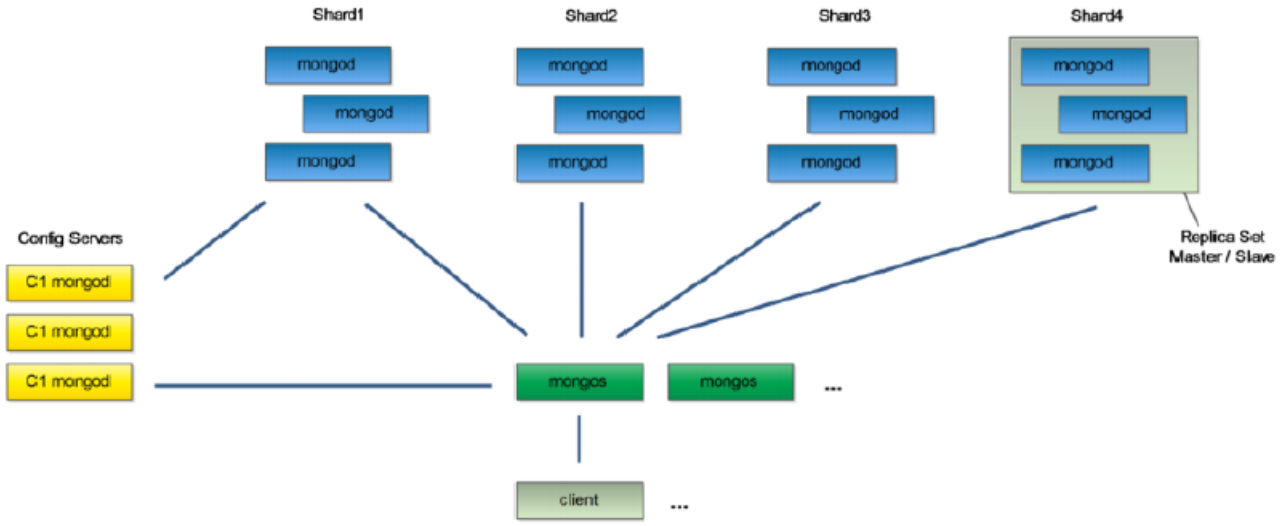


FIGURE 3.3 – Architecture de MongoDB[74]

Parmi les principaux composants on cite : [54]

Les Shards : ce sont un groupe de serveurs en mode Master / Slave ou Réplica

Sets.

Mongod, le processus de base de données principal :Il est responsable du système de stockage structuré. Mongod sera exécuté sur chaque réplique du cluster que vous déployez.

mongos, le contrôleur et routeur de requêtes des clusters distribués (sharded clusters) : Ce sont des serveurs qui savent quelles données se trouvent dans quel Shard et leur donnent des ordres (lecture, écriture).

mongo,le Shell MongoDB interactif : qui sert à émettre des commandes.

Configserver, ils connaissent l'emplacement de chaque donnée et en informent les mongos. De plus, ils organisent la structure des données entre les Shards.

E) Les avantages de MongoDB

Il y a plusieurs avantages importants dans MongoDB [55] :

— L'un des premiers avantages qui ont attiré les développeurs vers MongoDB est le moins d'efforts de développement initial nécessaires. De nombreux développeurs ont observé que l'écriture de commandes MongoDB est plus facile que l'écriture de ces longues requêtes SELECT *FROM.

— Un avantage supplémentaire de MongoDB est son bon à l'échelle. La mise à l'échelle est réalisée en utilisant Sharding dans MongoDB.

— Les index supportent l'exécution efficace des requêtes dans MongoDB. MongoDB définit les index au niveau de la collection et prend en charge les index sur n'importe quel champ ou sous-champ des documents d'une collection MongoDB.

— Schema-less, performance, disponibilité, flexibilité.

F) Les inconvénients

Bien que MongoDB a beaucoup d'avantages, il ne contredit pas les inconvénients [56] :

— L'indexation prend beaucoup de RAM.

— Une limite peut être atteinte dans la mise en place de requêtes analytiques complexes.

— MongoDB a tendance naturellement à utiliser plus de mémoire car il doit stocker les noms de clés dans chaque document.

— Pas de relation entre les données.

— Perte de données possible.

3.2.2 Les technologies de traitement ajustée

La deuxième est appelée le traitement massivement parallèle. Le Framework Hadoop en est un exemple.

3.2.2.1 Hadoop

A) Description :

Hadoop est un Framework open source écrit en java pour l'écriture et l'exécution d'applications distribuées qui traitent de grandes quantités de données [57]. Hadoop a été créé par Doug Cutting, le créateur d'Apache Lucene. Il a ses origines dans Apache Nutch, un moteur de recherche web open source [58]. Il inclut un système de fichier distribué HDFS et le modèle de programmation MapReduce. Tous les modules dans Hadoop sont conçus avec une hypothèse fondamentale que les défaillances matérielles sont courantes et devraient être traitées automatiquement par le Framework.[59]
Hadoop est utilisé par de nombreuses entreprises ayant de grands besoin en données, comme Yahoo, Facebook ou Ebay parmi les plus connus [60]

développeur	Apache Software Fondation
Langage de programmation	Java
Système d'exploitation	Cross-platform
Licence	Apache License 2.0
Site Web	hadoop.apache.org
Type	Système de fichiers distribué

TABLE 3.4 – Hadoop [59]

Parmi ces caractéristiques Hadoop est [57] :

Accessible : hadoop fonctionne sur de grandes grappes de machines de base ou sur des services de cloud computing tels que Elastic Compute Cloud (EC2) d'Amazon.

Robuste : parce qu'il est destiné à fonctionner sur du matériel de base, Hadoop est conçu avec l'hypothèse de dysfonctionnements matériels fréquents. Il peut gérer avec élégance la plupart de ses échecs.

Scalable : hadoop évolue linéairement pour gérer des données plus importantes en ajoutant plus de noeuds au cluster.

Simple : hadoop permet aux utilisateurs d'écrire rapidement du code parallèle efficace.

Souple : car il répond à la caractéristique de variété des données il est capable de traiter différents types de données .

B) Architecture

Le Framework Hadoop de base est composé des modules suivants [14] :

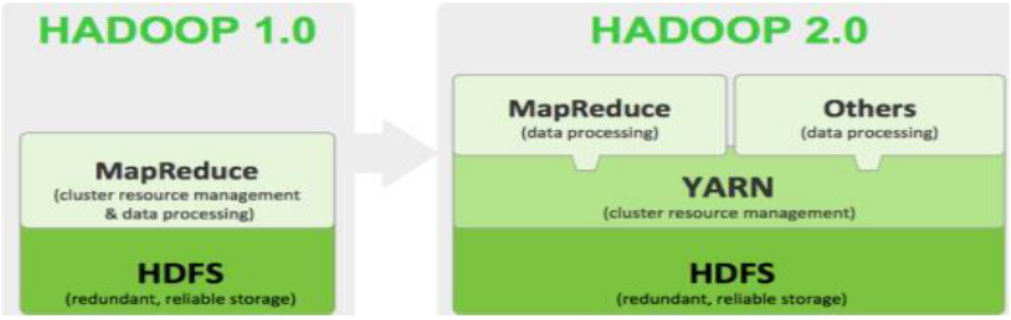


FIGURE 3.4 – versions de MapReduce [67]

— Hadoop Common : contient les bibliothèques et les utilitaires nécessaires aux autres modules Hadoop.

— Hadoop Distributed File System (HDFS) : un système de fichiers distribué qui stocke des données sur des machines de base, fournissant une bande passante agrégée très élevée à travers le cluster.

— Hadoop YARN : une plate-forme de gestion des ressources chargée de gérer les inclusions de ressources informatiques et de les utiliser pour la planification des applications des utilisateurs.

— Hadoop MapReduce : une implémentation du modèle de programmation MapReduce pour le traitement de données à grande échelle.

C) le système de fichier distribué d'Hadoop HDFS

HDFS est un système de fichiers distribué, influencée par GoogleFS, écrit en Java, il est conçu pour stocker de très grands ensembles de données de manière fiable et de les exécuter sur des clusters sur du matériel de base. Il présente de nombreuses similitudes avec les systèmes de fichiers distribués existants. Cependant, les différences par rapport aux autres systèmes de fichiers distribués HDFS est hautement tolérant aux pannes et peut être déployé sur du matériel à faible coût. Il fournit un accès à haut débit aux données d'application et convient aux applications disposant de grands ensembles de données. HDFS a été construit à l'origine en tant qu'infrastructure pour le projet Apache Nutch. Il fait partie du projet Ha-

doop, qui fait partie du projet Lucene Apache [61].

Organisation des fichiers dans HDFS

Un système de fichiers normal est séparé en plusieurs parties appelées blocs, qui sont les plus petites unités qui peuvent être lues ou écrites. Normalement, la taille par défaut est de quelques kilo-octets. HDFS a également le concept de blocs, mais d'une taille beaucoup plus grande, 64 Mo par défaut. La raison en est de minimiser les coûts de recherche pour trouver le début du bloc. Avec l'abstraction des blocs, il est possible de créer des fichiers plus grands que n'importe quel disque du réseau. Ces fichiers sont stockés sur un grand nombre de machines de manière à rendre invisible la position exacte d'un fichier. L'accès est transparent, quelle que soient les machines qui contiennent les fichiers. HDFS permet de voir tous les dossiers et fichiers de ces milliers de machines comme un seul arbre, comme s'ils étaient sur le disque dur local. HDFS ressemble à un système de fichiers Unix : il y a une racine, des répertoires et des fichiers. [62][63]

L'architecture de HDFS

HDFS repose sur une architecture maître / esclave, ce qui signifie qu'il existe un NameNode (le maître) et plusieurs des DataNodes (les esclaves). Un cluster HDFS est constitué de machines jouant différents rôles exclusifs entre eux [62] :

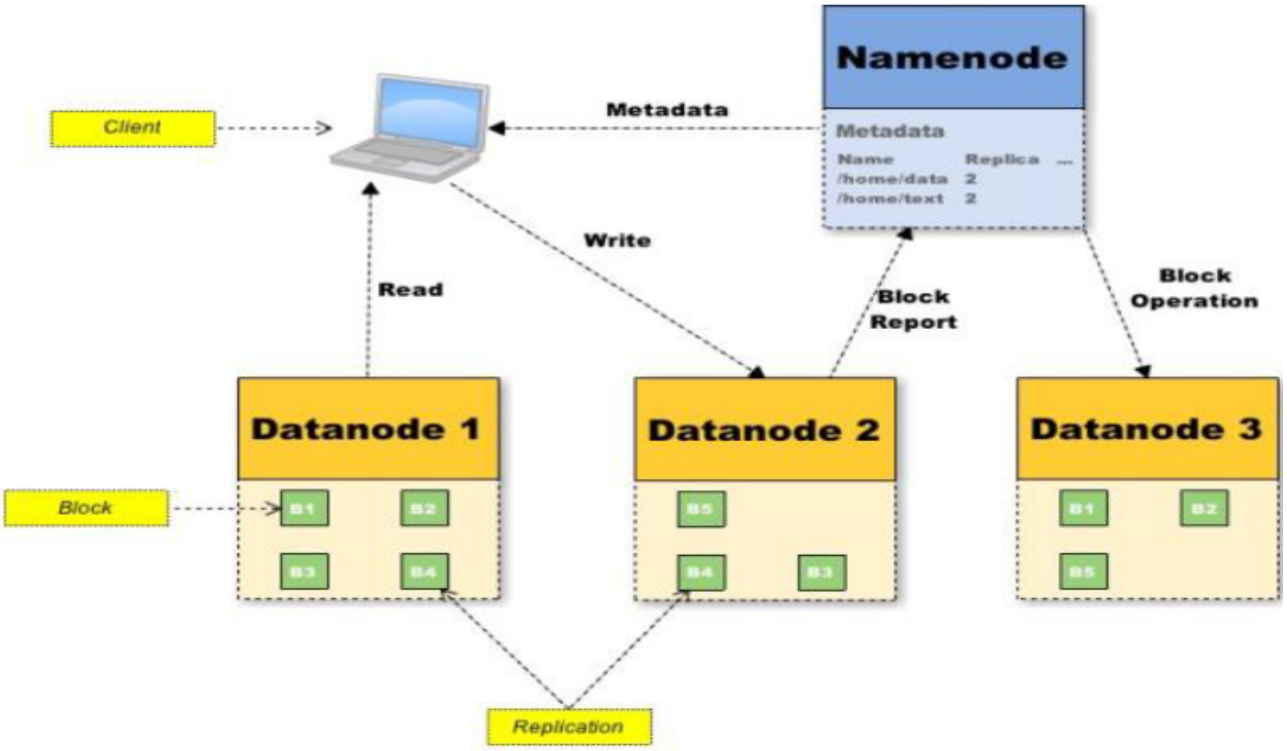


FIGURE 3.5 – architecture HDFS [62]

Le noeud principal ou NameNode : c'est le maître, le NameNode est responsable de la gestion de namespace du système de fichiers. Il gère l'arborescence du système de fichiers et stocke toutes les métadonnées des fichiers et des répertoires. De plus, les NameNodes savent où chaque partie (bloc) du fichier distribué est stockée.

Le noeud de données ou DataNodes : les DataNodes stockent et récupèrent tous les blocs lorsqu'ils sont signalés par le NameNode et servent les demandes de lecture et d'écriture des clients du système. En outre, ils rapportent périodiquement une liste de blocs qu'ils stockent.

Le secondary NameNode : est une sorte de NameNode, qui enregistre des sauvegardes de l'annuaire à intervalles réguliers, ce noeud secondaire s'exécute généralement sur une machine physique séparée, il communique avec NameNode pour prendre des instantanés des métadonnées HDFS à des intervalles définis par la configuration du cluster. Il conserve une copie de l'image de namespace

fusionné, qui peut être utilisée en cas d'échec du NameNode car ce dernier est un point de défaillance unique pour le cluster Hadoop [57].

Les NameNodes de secours : Hadoop version 2 propose une configuration appelée high availability dans laquelle il y a 2 autres NameNodes en secours, capables de prendre le relais instantanément en cas de panne du NameNode initial. Les NameNodes de secours se comportent comme des clones. Ils sont en état d'attente et mis à jour en permanence à l'aide de services appelés JournalNodes [63].

Lecture et écriture d'un fichier sur HDFS

Lecture d'un fichier sur HDFS

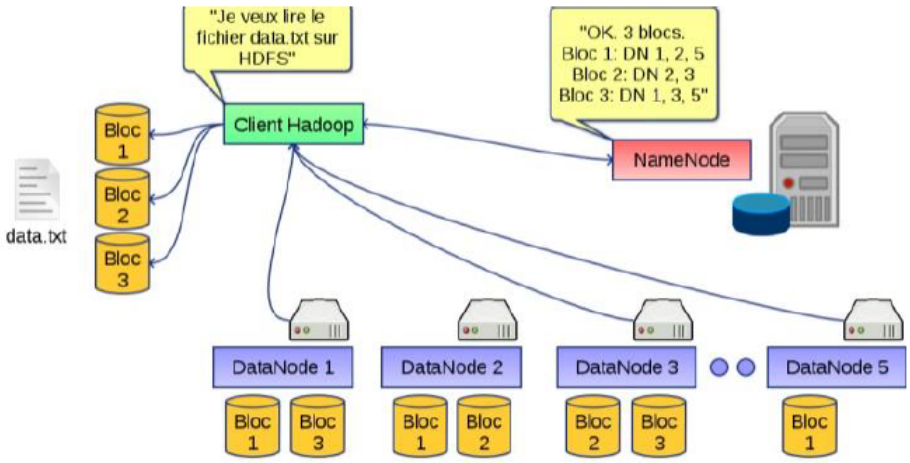


FIGURE 3.6 – Lecture HDFS [64]

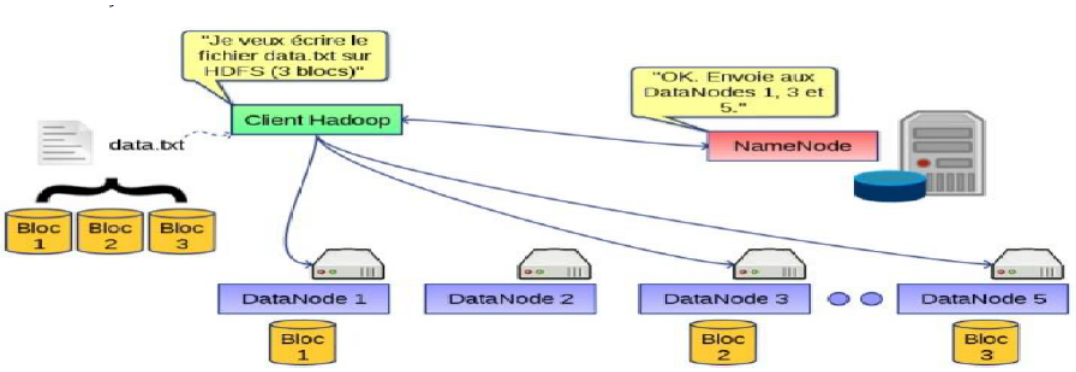
— Le client indique au NameNode qu'il souhaite lire un fichier.

— Celui-ci indique sa taille et les différents DataNodes contenant les N blocs.

— Le client récupère chacun des blocs à un des DataNodes.

— Si un DataNode est indisponible, le client le demande à un autre.

Écriture d'un fichier sur HDFS



s

FIGURE 3.7 – l'écriture sur HDFS [64]

— Le client indique au NameNode qu'il souhaite écrire un bloc.

— Celui-ci indique le DataNode à contacter.

— Le client envoie le bloc au DataNode.

— Les DataNodes répliquent le bloc entre eux.

— Le cycle se répète pour le bloc suivant.

D) MapReduce

MapReduce est un modèle de programmation et une implémentation associée pour le traitement et la génération de grands ensembles de données de manière parallèle, en les distribuant sur divers noeuds d'un cluster. Ce mécanisme a été inventé par Google pour la partie backend de leur moteur de recherche afin de rendre un grand nombre de serveurs capables de traiter efficacement l'analyse d'un grand nombre de fichiers de pages Web collectés dans le monde entier. Il a connu un très grand succès auprès des sociétés utilisant des Datacenters telles que Facebook ou Amazon [65].

Le principe de MapReduce

Le principe de MapReduce consiste essentiellement à imposer qu'un noeud, dans la phase "map", ne travaille que sur des données stockées dans un seul cluster pour produire en sortie un ordonnancement des données qui permet de les répartir de manière efficace sur des noeuds différents pour la phase de calcul (phase "reduce") Dans MapReduce, les données sont toujours lues ou écrites selon le format "key, value".

MapReduce est vu en deux étapes : la fonction map qui est chargé de lire les données stockées sur disque et les traiter et la fonction reduce que est chargé de consolider les résultats issus du mapper puis de les écrire sur disque [66].
Plus précisément, un programme MapReduce repose sur les opérations suivantes [63] :

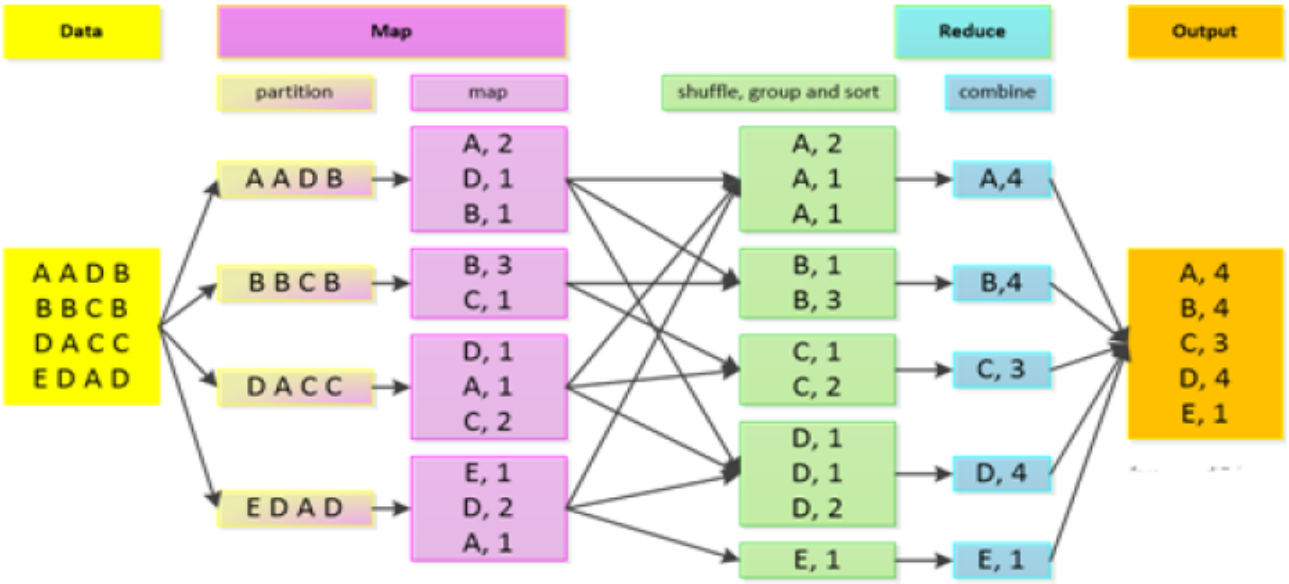


FIGURE 3.8 – fonctionnement de MapReduce [67]

— Prétraitement des données d'entrée.

— Split : séparation des données en blocs traitables séparément et mise sous forme de (clé, valeur).

— Map : application de la fonction map sur toutes les paires (clé, valeur) formées à partir des données d'entrée, cela produit d'autres paires (clé, valeur) en sortie.

— Shuffle et Sort : redistribution des données afin que les paires produites par Map ayant les mêmes clés soient sur les mêmes machines.

— Reduce : agrégation des paires ayant la même clé pour obtenir le résultat final.

Architecture de MapReduce

Le Framework Hadoop MapReduce se compose d'un noeud maître appelé JobTracker et de nombreux noeuds esclaves appelés TaskTrackers [57] [68] [69] :

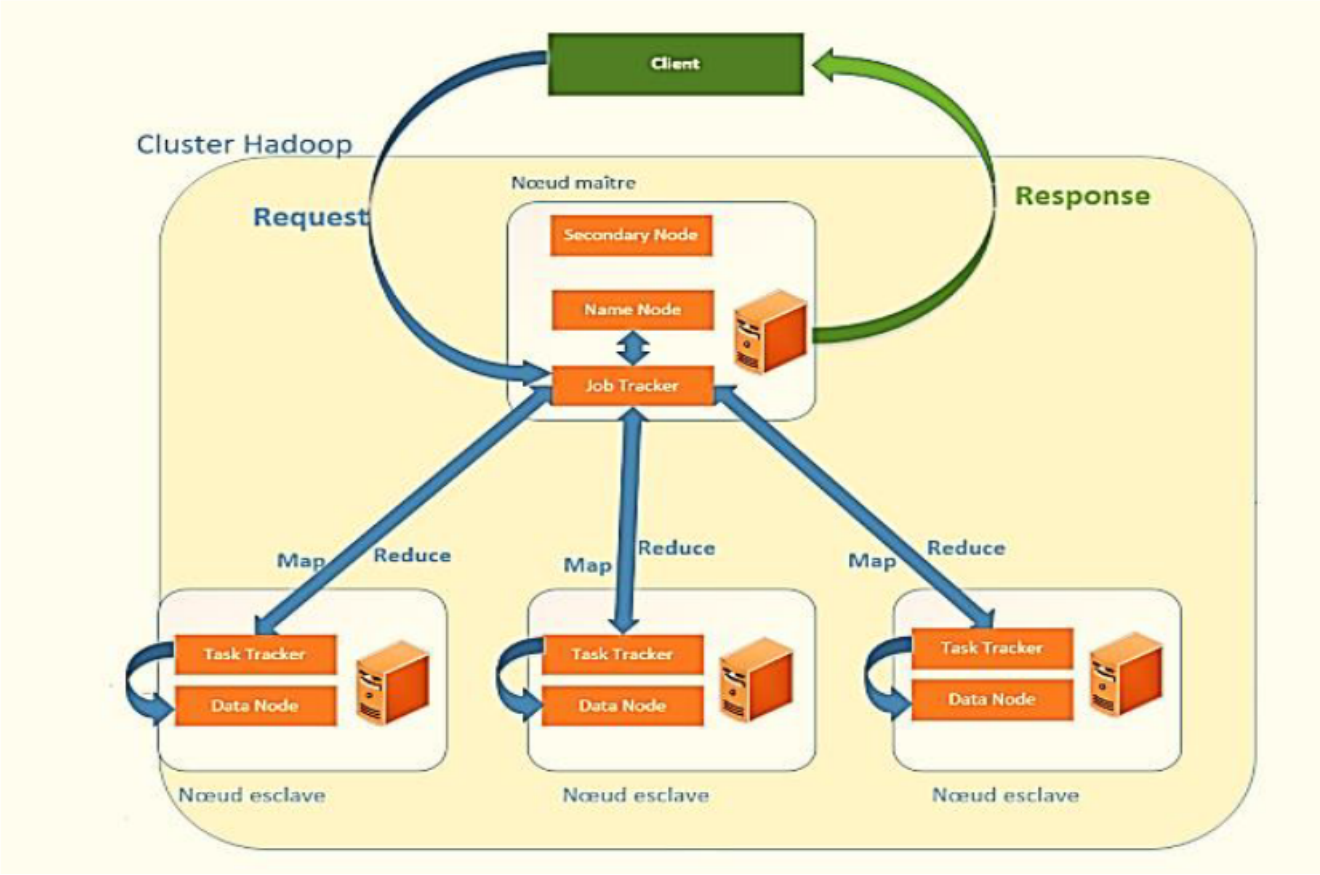


FIGURE 3.9 – Architecture de MapReduce [69]

JobTracker

Le JobTracker est la liaison entre l'application et Hadoop. Une fois qu'on a envoyé le code au cluster, JobTracker détermine le plan d'exécution en déterminant les fichiers à traiter, attribue des noeuds à différentes tâches et surveille toutes les tâches en cours d'exécution. Les travaux soumis par l'utilisateur sont donnés en entrée au JobTracker qui les transforme en nombres de tâches Map et Reduce (le JobTracker supervise l'exécution globale d'un travail MapReduce). Ces tâches sont assignées aux TaskTrackers. Le JobTracker assume la responsabilité de distribuer la configuration logicielle aux noeuds esclaves. Si une tâche échoue, JobTracker relancera automatiquement la tâche, éventuellement sur un autre noeud, jusqu'à une limite prédéfinie de tentatives. Il existe un seul JobTracker par cluster Hadoop. Il est généralement exécuté sur un serveur en tant que noeud principal du cluster.

TaskTrackers

Chaque TaskTracker est responsable de l'exécution des tâches individuelles que le JobTracker affecte. Il existe un seul TaskTracker par noeud esclave. Le TaskTracker exécute des tâches dans des processus java séparés de sorte que plusieurs instances de tâche peuvent être exécutées en parallèle en même temps. Une responsabilité du TaskTracker est de communiquer en permanence avec le JobTracker. Si le JobTracker ne parvient pas à recevoir un signal de présence d'un TaskTracker dans un laps de temps spécifié, il supposera que le TaskTracker s'est écrasé et soumettra à nouveau les tâches correspondantes aux autres noeuds du cluster.

Les TaskTrackers scrutent l'exécution de ces tâches et à la fin lorsque toutes les tâches sont accomplies ; l'utilisateur est informé de l'achèvement du travail.

YARN

Hadoop YARN est le nouveau remplaçant du composant MapReduce trouvé dans la version 1.La version 2 de Hadoop avec YARN offre une fonctionnalité MapReduce complète , elle ouvre également la porte à de nombreux autres « Framework d'applications » qui ne sont pas basés sur le traitement de MapReduce.

L'acronyme YARN est l'abréviation de « Encore un autre négociateur de ressources », qui est une bonne description de ce que fait réellement YARN.

Fondamentalement, YARN est un planificateur de ressources conçu pour fonctionner sur les clusters Hadoop. Ce planificateur permet une meilleure utilisation et une évolutivité du cluster, [74].

Dans ce nouveau contexte, MapReduce est juste l'une des applications fonctionnant au-dessus de YARN. Cette séparation offre une grande flexibilité dans le choix du cadre de programmation. Les Framework de programmation s'exécutant sur YARN coordonnent la communication intra-application, le flux d'exécution et les optimisations dynamiques comme bon leur semble, ce qui permet d'améliorer considérablement les performances [68].

E) Ecosystème d'Hadoop

Hadoop était à l'origine un système de fichiers et une implémentation MapReduce. Au cours des dernières années, il est devenu un écosystème complexe comprenant

une gamme de systèmes logiciels. Plusieurs sociétés proposent des distributions de Hadoop qui regroupent un certain nombre de ces projets. Cependant, l'objectif principal est facile à écrire du code et simplement créer un projet rentable[70].

Les composants de l'écosystème Hadoop

Plus les composants d'Hadoop ; L'écosystème Hadoop contient également différents projets dont il est question ci-dessous [68] :

Les Composants Hadoop	**Fonctionnalité**
-HDFS	-Stockage et réplication
-MapReduce	-Traitement de données et tolérance aux pannes
-HBase	-Accès rapide en lecture / écriture
-Pig	-Script
-Hive	-Langage HQL(SQL)
-Zookeeper	-Coordination
-HCatalog	-Métadonnées
-Oozie	- Workflow et planification

TABLE 3.5 – les composants de l'écosystème Hadoop [71]

Apache HBase : une base de données de données de clé / valeur distribuée et évolutive non-relationnel et orienté colonnes, conçu pour s'exécuter sur la plate-forme HDFS. Il est conçu pour évoluer horizontalement dans les clusters de calcul distribuées.

Apache Hive : une infrastructure d'entrepôt de données construite sur haut de Hadoop pour fournir des données résumé, l'interrogation ad hoc et l'analyse de grands ensembles de données stockés dans des fichiers Hadoop. Il n'est pas conçu pour offrir des requêtes en temps réel, mais il peut prendre en charge les fichiers texte et les fichiers de séquence.

Apache Pig : il fournit un mécanisme parallèle de haut niveau pour la programmation des travaux MapReduce à exécuter sur les clusters Hadoop. Il utilise un langage de script appelé Pig Latin, qui est un langage de flux de données orienté vers le traitement des données de manière parallèle.

Apache Zookeeper : c'est une interface de programme d'application (API) qui permet le traitement distribué des données dans les grands systèmes pour se synchroniser les uns avec les autres afin de fournir des données cohérentes aux demandes des clients.

Apache Sqoop : un outil conçu pour transférer efficacement des données en masse entre Hadoop et des magasins de données structurés tels que des bases de données relationnelles.

Apache Flume : service distribué pour la collecte, l'agrégation et le déplacement de grandes quantités de données de journal.

Avro : un système de sérialisation de données.

Cassandra : une base de données multi-maîtres évolutive sans points de défaillance uniques.

Chukwa : un système de collecte de données pour la gestion de systèmes distribués.

Mahout : une bibliothèque d'apprentissage automatique et d'exploration de données évolutive.

HCatalog : HCatalog gère HDFS. Il stocke les métadonnées et génère des tables pour de grandes quantités de données. HCatalog simplifie la communication de l'utilisateur en utilisant les données HDFS et est une source de partage de données entre les outils et l'exécution de plateformes.

Les distributions de Hadoop

Les distributions de Hadoop regroupent la majorité des services de l'écosystème de Hadoop et facilitent la gestion de tous les noeuds [27].

Cloudera : il est largement reconnu que Cloudera est le principal fournisseur de logiciels et de services Apache Hadoop dans le Big Data, Cloudera contribue plus de 50open source sous licence apache, comme Hive, Avro, HBase. Cloudera a également prospéré en particulier dans les technologies de données cloud et la recherche biologique et l'espace médical. [60]

HortonWorks : HortonWorks a été créée en juin 2011 par les membres de l'équipe Yahoo en charge du projet Hadoop. Leur objectif est de faciliter l'adoption de Plate-forme Apache Hadoop qui est la raison pour laquelle tous les composants sont open source et sous licence Apache. Le but économique de HortonWorks est

ne pas vendre la licence mais vendre du support et de la formation. Ainsi, cette distribution est la plus cohérente avec La plate-forme Apache Hadoop, et Horton-Works est un grand contributeur Hadoop [61]

MapR : MapR a été fondée en 2009 par Google membres. Bien que son approche soit commerciale, MapR contribue aux projets Apache Hadoop comme HBase, Pig, Hive, Zookeeper et surtout Drill. MapR diffère principalement de la version d'Apache Hadoop en prenant la distance avec le noyau de la plateforme. Ils proposent leur propre fichier distribué système ainsi que leur propre version de MapReduce : MapR FS et MapR MR.

F)Les avantages et les inconvénients de Hadoop

Si la mise en place de clusters Hadoop est une des solutions au problème de big data, elle ne convient pas à toutes les situations. Examinons quelques-uns des avantages et des inconvénients de ce type d'architecture [73].

Les avantages :

— Simple à utiliser et à déployer, portable (clusters hétérogènes), Indéfiniment scalable, très extensible et interconnectable avec d'autres solutions.

— Le framework Hadoop convient bien à l'analyse de type de données non structurées où le Big Data est le plus souvent non structuré et largement distribué.

— Les données n'ont pas à être uniformes : chaque élément de donnée est traité par un processus distinct d'un noeud distinct du cluster.

— Un autre avantage de Hadoop réside dans leur évolutivité. Un des problèmes de l'analyse du Big Data est, comme pour tous les types de données, sa croissance continue.

— Les capacités de traitement en parallèle de Hadoop participent à la rapidité de l'analyse.

— Un autre avantage de ce type d'architecture réside dans son coût.

— La résistance aux panne.

Les inconvénients :

Malgré ses nombreux avantages, la solution des clusters Hadoop ne convient pas aux besoins d'analyse de données de toutes les organisations. Pour les petits volumes de données, ses bénéfices seront inexistants, même s'il est besoin d'une analyse poussée de ces dernières, et aussi il est en développement.

3.3 conclusion

L'arrivée des technologies Big data dans le monde professionnel sont perçus comme une bonne nouvelle pour les entreprises génératrices de grand volume de données qui avaient du mal à les traiter. Ces technologies donnent une pluralité de solutions avec des fonctionnalités parfois diverses, parmi les plus utilisées Cassandra, MongoDB et Hadoop.

Dans le chapitre suivant nous avant presenter la meilleure solution que nous avons choisi pour l'implémenter dans le chapitre suivant est Hadoop, grâce à la structure de données de son système HDFS avec sa réplication entre les noeuds ainsi que la taille des blocs de données qui est supérieure à un système de fichier traditionnel donnent plus de flexibilité pour traiter des données plus rapidement d'une façon distribuée et parallèle.

Chapitre 4

Installation et Configuration de Hadoop

4.1 Introduction

Ce présent chapitre s'intéresse à l'installation et la configuration d'un cluster Hadoop. Nous décrirons comment réaliser une telle installation dans un système Linux et de tester la configuration de l'installation et fournir un environnement de développement MapReduce. Ainsi qu'une implémentation d'un exemples.

4.2 Les modes de configuration de Hadoop

Hadoop peut être installé en 3 modes différents : mode autonome, mode pseudo-distribué et mode entièrement distribué [75].

4.2.1 Le mode autonome

Le mode autonome est le mode de fonctionnement par défaut de Hadoop et s'exécute sur un seul nœud.

4.2.2 Le mode pseudo-distribué

Le mode pseudo-distribué se situe entre le mode autonome et le mode entièrement distribué sur un cluster. Il simule 2 nœuds, un maître et un esclave en exécutant un processus JVM. HDFS est utilisé pour le stockage en utilisant une partie de l'espace disque et YARN pour gérer les ressources sur cette installation Hadoop.

4.2.3 Le mode entièrement distribué

C'est le mode de production de Hadoop où plusieurs nœuds sont exécutés. Ici, les données sont distribuées sur plusieurs nœuds et le traitement est effectué sur chaque nœud.

4.3 La configuration pseudo-distribué du cluster Hadoop

4.3.1 Environnement de travail

Pour notre installation de Hadoop, nous avons construit un environnement basé sur :

— Système d'exploitation : linux-mint 17

— Apache Hadoop : v2.7.1

— JDK : 1.8.0_181

4.3.2 Installation de Hadoop 2.7.1

Apache Hadoop 2.7.1 est une version développée de 2.7.0. C'est une version stable après Apache Hadoop 2.6.0. Il prend en charge la fonctionnalité HDFS pour les fichiers avec un bloc de longueur variable. Sa fonction MapReduce permet d'accélérer les très gros travaux avec de nombreux fichiers de sortie. Voici les étapes à suivre :

4.3.2.1 Etape 1 : installation de Java

Java est la principale condition pour exécuter Hadoop sur n'importe quel système. Cette version supprime la prise en charge du support d'exécution JDK6 et fonctionne uniquement avec JDK8.

4.3.2.2 Etape 2 : création d'un utilisateur « hduser » dans un groupe « Hadoop »

Il est recommandé de créer un compte normal pour le fonctionnement de Hadoop

```
$ sudo addgroup hadoop

$ sudo adduser --ingroup hadoop hduser

$ sudo adduser hduser sudo

$ su hduser
```

4.3.2.3 Etape 3 : configuration de ssh

Hadoop nécessite un accès SSH pour gérer ses nœuds. Pour notre configuration de Hadoop, nous devons donc configurer l'accès SSH à localhost pour l' utilisateur « hduser ».

• Générer un SSH key pour hduser : cette commande créera une paire de clés RSA avec un mot de passe vide.

```
$ ssh-keygen -t rsa -P ""
```

Pour le fichier de sauvegarde de la clé ; valider le fichier proposé en tapant simplement sur « Entré ».

• Activer l'accès SSH à la machine locale avec cette clé nouvellement créée

```
$ cat $HOME/.ssh/id_rsa.pub >>$HOME/.ssh/authorized_keys
```

• Nous devons maintenant vérifier la connexion de « hduser » à « localhost »

```
$ ssh localhost
```

On doit répondre par (yes) sur la question (Are you sure you want to continue

connecting (yes/no) ?).

4.3.2.4 Etape 4 : configuration de Hadoop

• Déplacer le fichier d'installation « hadoop-2.7.1.tar.gz » à « hadoop » un répertoire personnel de hduser.

```
$ mkdir /home/hduser/hadoop

$ cp /…/ hadoop-2.7.1.tar.gz  /home/hduser/hadoop
```

• Décompresser « hadoop-2.7.1.tar.gz »

```
$ tar xzf hadoop-2.7.1.tar.gz
```

• Renommer « hadoop-2.7.1.tar.gz » par Hadoop

```
$ mv hadoop-2.7.1 hadoop
```

• Assurer que le propriétaire du fichier Hadoop est hduser.

```
$ sudo chown -R hduser:hadoop hadoop
```

4.3.2.5 Etape 5 : configuration de l'environnement

Nous devons définir les utilisations de variables d'environnement pour Hadoop.

• Editer le fichier « / .bashrc » et ajouter les valeurs suivantes à la fin du fichier.

```
$ sudo chown -R hduser:hadoop hadoop
```

```
# Run the .bashrc file in /etc first if it exists
 if [ -f /etc/bash.bashrc ]; then
 ./etc/bash.bashrc
 fi

 # Hadoop variables
 export HADOOP_HOME=/home/hduser/hadoop/hadoop
 export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
 export HADOOP_OPTS="$HADOOP_OPTS-Djava.library.path=$HADOOP_HOME/lib"

 export PATH=$PATH:$HADOOP_HOME/bin
 export PATH=$PATH:$HADOOP_HOME/sbin

 export HADOOP_MAPRED_HOME=$HADOOP_HOME
 export HADOOP_COMMON_HOME=$HADOOP_HOME
 export HADOOP_HDFS_HOME=$HADOOP_HOME
 export YARN_HOME=$HADOOP_HOME
```

• Appliquer les modifications ci-dessus dans l'environnement actuel.

```
$ source ~/.bashrc
```

• Vérification de l'installation de Hadoop avec la commande :

```
$ Hadoop version
```

4.3.2.6 Etape 6 : configuration des répertoire Hadoop

Maintenant, nous allons configurer les répertoires où Hadoop stocke les fichiers de données, les ports réseau, etc. Pour cela nous avons besoin de créer l'espace de stockage temporaire, et les fichiers de Storage de NameNode et DataNode pour HDFS.

```
$ sudo mkdir -p /app/hadoop/tmp

$ sudo mkdir -p /app/hadoop/usr/hduser/data/hdfs/namenode

$ sudo mkdir -p /app/hadoop/usr/hduser/data/hdfs/datanode

$ mkdir -p /home/hduser/data/hdfs/namenode

$ mkdir -p /home/hduser/data/hdfs/datanode
```

• Changer les propriétaires des répertoires et les permissions d'accès

```
$ sudo chown hduser:hadoop /app/hadoop/tmp

$ sudo chown -R hduser:hadoop /app/hadoop/usr/hduser

$ sudo chmod 750 /app/hadoop/tmp

$ sudo chmod -R 750 /app/hadoop/usr/hduser
```

4.3.2.7 Etape 7 : configuration des fichiers de configuration de Hadoop

Il existe de nombreux fichiers de configuration dans Hadoop, qui doivent être configurés conformément aux exigences de notre infrastructure Hadoop. Commençons maintenant avec la configuration de cluster Hadoop.

• hadoop-env.sh : éditer le fichier *$HADOOP–HOME / etc / hadoop / hadoop–env.sh*. Et définir la variable d'environnement JAVA_HOME. Modifier le chemin JAVA en fonction de l'installation de votre système.

```
$ cd /home/hduser/hadoop/hadoop/etc/hadoop/

$ $ nano $HADOOP_HOME/etc/hadoop/hadoop-env.sH
```

```
export JAVA_HOME = /usr/lib/jvm/jdk1.8.0_181
```

• core-site.xml : la configuration du core-site comprend des paramètres importants tels que l'emplacement du HDFS (HADOOP DATA DIR) sur chaque nœud et l'URI du serveur HDFS (qui inclut l'hôte et le port du maître). Il inclut également des paramètres de réglage supplémentaires pour la taille des tampons de lecture / écriture. [76]

```
$ nano core-site.xml
```

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

• hdfs-site.xml : La configuration du hdfs-site inclut des paramètres de configuration du système de fichiers distribués, tels que le nombre de réplications, la taille du bloc HDFS et le nombre de gestionnaires DataNode servant à traiter les demandes de bloc. [76]

```
$ nano hdfs-site.xml
```

```
<configuration>
<property>
 <name>dfs.replication</name>
 <value>1</value>
</property>
<property>
  <name>dfs.name.dir</name>
    <value>file:///home/hduser/data/hdfs/namenode </value>
</property>
<property>
  <name>dfs.data.dir</name>
    <value>file:///home/hduser/data/hdfs/datanode </value>
</property>
</configuration>
```

• mapred-site.xml : la configuration du MapReduce-site comprend l'hôte et le port de JobTracker (sur le maître), le nombre de copies parallèles pouvant être exécutées par les réducteurs, le nombre de mappes et les tâches à exécuter simultanément. [76]

```
$ nano mapred-site.xml
```

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

• yarn-site.xml : YARN prend en charge un modèle de ressource extensible. Par défaut, YARN effectue le suivi de l'UC et de la mémoire pour tous les nœuds, applications et files d'attente.

```
$ nano yarn-site.xml
```

```
<configuration>
 <property>
  <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
 </property>
</configuration>
```

4.3.2.8 Etape 8 : format le NameNode

Maintenant, formatez le NameNode en utilisant la commande suivante :

```
$ hdfs namenode -format
```

```
11/09/18 14:48:31 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = localhost/127.0.0.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 2.7.1
...
...
...
11/09/18 14:48:41 INFO common.Storage: Storage directory
/home/hadoop/hadoopdata/hdfs/namenode has been successfully formatted.
15/11/13 14:48:42 INFO namenode.NNStorageRetentionManager: Going to retain
1 images with txid >= 0
15/11/13 14:48:42 INFO util.ExitUtil: Exiting with status 0
15/11/13 14:48:42 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at localhost/127.0.0.1
```

4.3.2.9 Etape 9 : lancement de cluster Hadoop

Permet de démarrer le cluster Hadoop. On navigue simplement vers le répertoire de « hadoop sbin » et on exécute les scripts un par un.

```
$ cd $HADOOP_HOME/sbin/

$ ./start-dfs.sh

$ ./start-yarn.sh
```

• Start-dfs.sh : permet de lancer NameNode, DataNode, jps, secondary NameNode

• Start-yarn.sh : permet de lancer ResourceManager, NodeManager.

Pour vérifier on va lancer la commande :

```
$ jps
```

On va avoir :

```
hduser@mayada-Aspire-E5-573 ~/hadoop/hadoop/sbin $ jps
3458 NodeManager
2884 DataNode
3765 Jps
3305 ResourceManager
2732 NameNode
3135 SecondaryNameNode
```

Accès aux services Hadoop

• Accédez aux services Hadoop par le numéro de port 50070 par défaut sur le navigateur Web. Par exemple : http ://localhost :50070

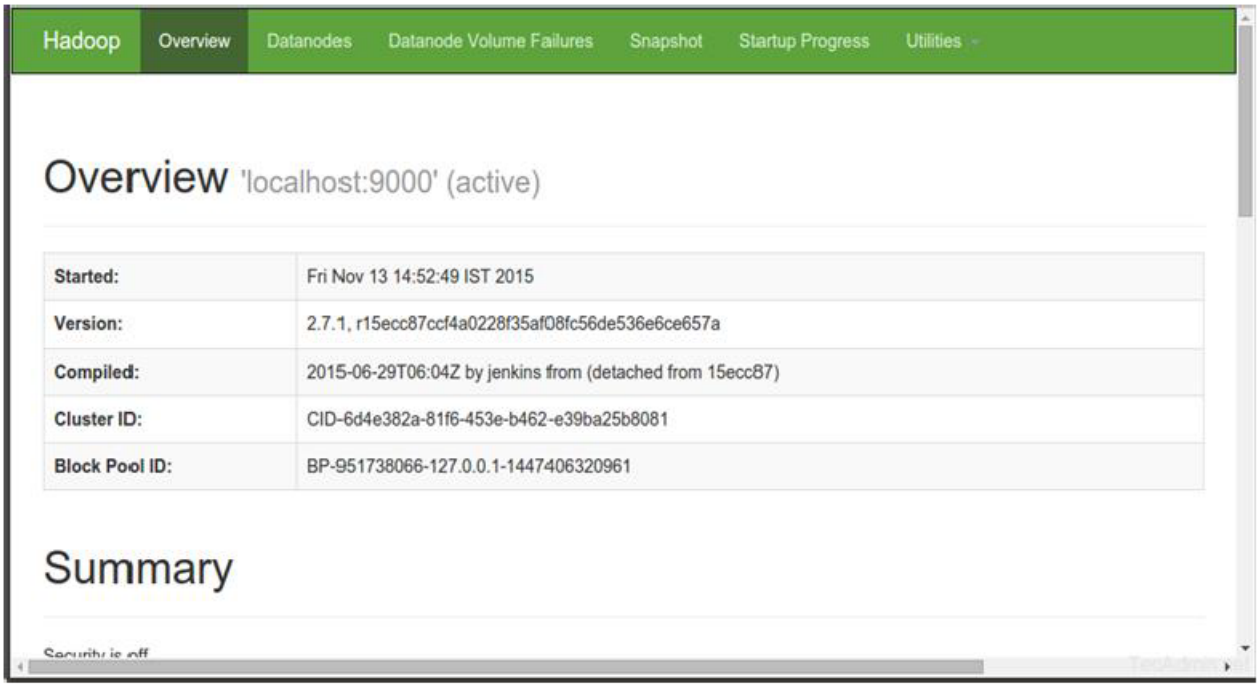


FIGURE 4.1 – aperçu de service Hadoop

• Et après avoir pris connaissance des détails sur le port 50070, accéder maintenant au port 8088 pour obtenir les informations sur le cluster et toutes les applications à l'aide de la commande suivante. http ://localhost :8088

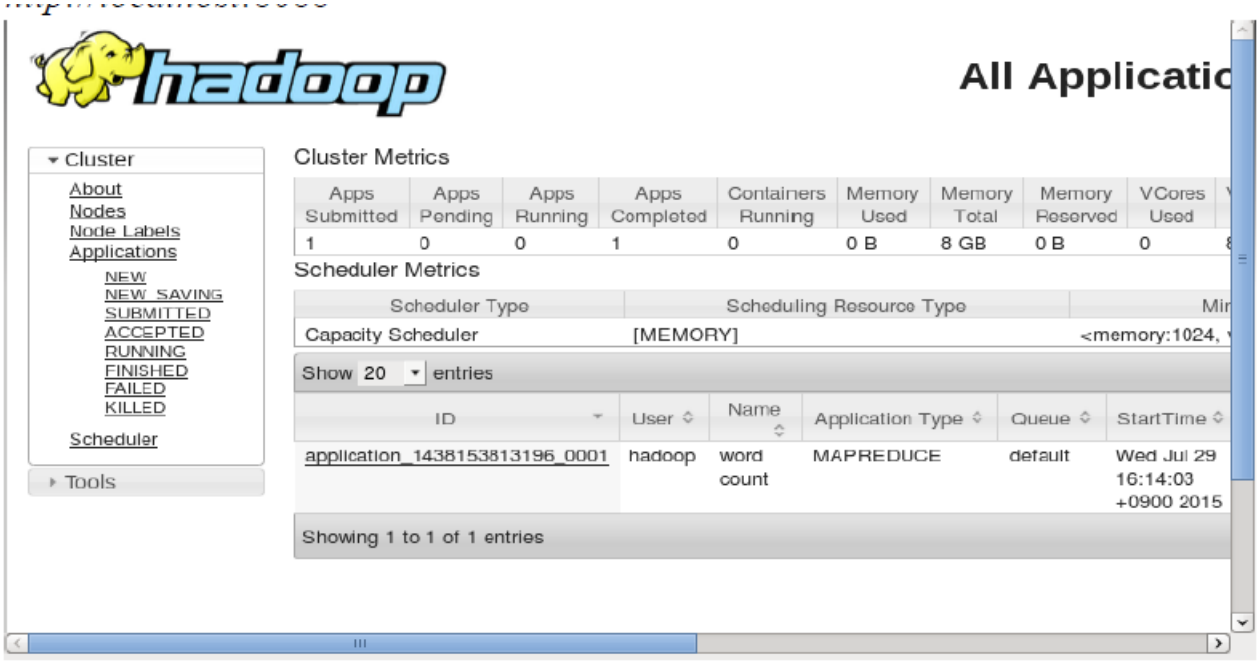


FIGURE 4.2 – information sur le cluster Hadoop

• Accédez au port 50090 pour obtenir des détails sur le NameNode secondaire.

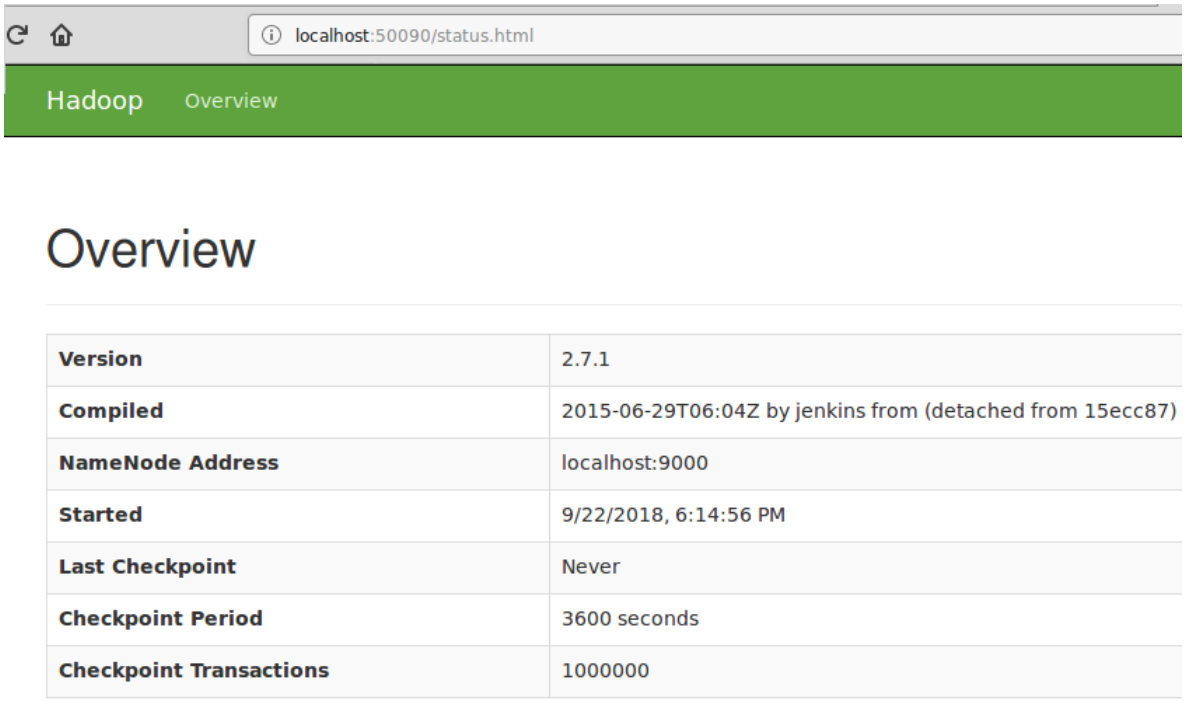


FIGURE 4.3 – information sur le cluster Hadoop

4.4 Implémentation d'un exemple sur Hadoop

Dans cette phase, on va implémenter un exemple d'application MapReduce pour obtenir un aperçu de son fonctionnement. Cet exemple est une application simple qui compte le nombre d'occurrences de chaque caractère dans un jeu d'entrées donné. Ce programme est le code de base utilisé pour comprendre le fonctionnement du paradigme de programmation de MapReduce.

• On commence par écrire un fichier texte (file.txt) .

• Dans eclipse, avant de commencer de programmer, il faut ajouter les librairies de Hadoop suivantes :

— home/hduser/hadoop/hadoop/share/hadoop/common/hadoopcommon-2.7.1.jar

— home/hduser/hadoop/hadoop/share/hadoop/mapreduce

— home/hduser/hadoop/hadoop/share/hadoop/mapreduce/lib

— home/hduser/hadoop/hadoop/share/hadoop/mapreduce/sources

• L'ensemble du programme MapReduce peut être divisé en trois parties (voir Annexe A) :

— La classe mapper : la tâche de mappage est chargée de marquer le texte en entrée en fonction de l'espace et de créer une liste de caractères, puis de parcourir tous les jetons et d'émettre une paire clé-valeur de chaque caractère avec un décompte de un.

— La classe reducer : cette fonction est appelée après la méthode map et reçoit les clés de la fonction map() correspondant à la clé spécifique. Les ajoute et les réduit à une seule valeur avant d'écrire finalement le caractère et le nombre d'occurrences du caractère dans le fichier de sortie.

— La classe du conducteur : cette classe appelée la classe du pilote. Elle contient la fonction main() et la méthode pour configurer et exécuter le travail. On doit enregistrer ce projet par exemple « wordcount.jar »

• On va créer un répertoire user dans le HDFS avec la commande suivante :

```
$ hadoop fs -mkdir /user
```

• Dans le serveur hadoop :localhost 50070, utilities —>browser the file system

FIGURE 4.4 – répertoire user dans hdfs

• On fait déplacer le fichier file.txt au répertoire « /user »

```
$ hadoop fs -put /home/hduser/Bureau/file.txt /user/file.txt
```

/user Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hduser	supergroup	86 B	9/15/2018, 4:58:17 PM	1	128 MB	file.txt
drwxr-xr-x	hduser	supergroup	0 B	9/17/2018, 12:22:17 PM	0	0 B	hduser

FIGURE 4.5 – le repertoire user

• On exécute le code avec la création d'un fichier de sortie , et il faut assurer que le fichier de sortie n'existe pas (par exemple wordcountoutput). Si c'est le cas, le programme générera une erreur

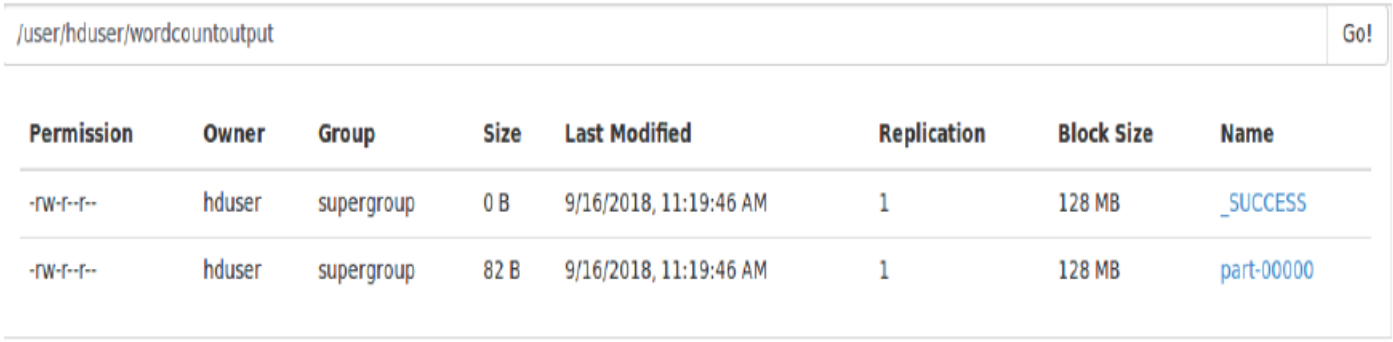
```
$ hadoop jar /home/hduser/Bureau/wordcount.jar
org.hadoop.trainings.WordCount /user/file.txt wordcountoutput
```

• Dans le cluster, on voit le fichier de sortie

drwxr-xr-x	hduser	supergroup	0 B	9/16/2018, 11:19:46 AM	0	0 B	wordcountoutput

FIGURE 4.6 – le fichier de sortie wordcountoutput

• On clique sur wordcountoutput,on voit qu'il est bien exécuté.



/user/hduser/wordcountoutput Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hduser	supergroup	0 B	9/16/2018, 11:19:46 AM	1	128 MB	_SUCCESS
-rw-r--r--	hduser	supergroup	82 B	9/16/2018, 11:19:46 AM	1	128 MB	part-00000

FIGURE 4.7 – l'exécution de fichier wordcountoutput

• On peut afficher le contenu de fichier de sortie dans le cluster Hadoop on clique sur « download », ou bien on utilise la commande suivante :

```
$ hadoop fs -cat wordcountoutput/part-00000
```

```
hduser@mayada-Aspire-E5-573 ~/hadoop/hadoop/sbin $ hadoop fs
t/part-00000
        15
a       5
b       1
d       1
e       4
f       2
g       1
h       5
i       6
j       1
l       1
m       2
n       2
o       11
r       5
s       2
t       1
u       5
w       5
y       6
```

4.5 conclusion

Nous pouvons conclure sur le fait que Hadoop reste aujourd'hui, une des meilleures solutions dans le traitement de gros volume de données. Hadoop a évolué à un rythme exponentiel au cours des dernières années. Cependant il faut noter que l'installation d'un environnement Hadoop peut-être réalisé de différentes façons, selon le type de cluster qui répond plus à vos besoins.

Conclusion générale et perspectives

La formation au big data nous a permis de savoir quelle est la tendance folle dans les industries informatiques et comment la technologie devient plus fructueuse pour le développement humain.

Les données devenant de plus en plus volumineuse et complexe, nos bases de données traditionnelles sont limitées face à l'analyse et au traitement de ces données. Actuellement, de nombreuses recherches sont en cours dans ce domaine. À mesure que les données augmentent à un rythme plus rapide, il existe un besoin énorme d'outils et de technologies capables de les gérer.

Dans un souci de gain de temps, de nouvelle technologie sont venu pour soulager les entreprises génératrices d'un grand nombre de données.

nous estimons que les technologies des Big Data seront de plus en plus utilisées pour répondre à de nouvelles problématiques pour la gestion de données dans des environnements à grande échelle. Les systèmes NoSQL se placent comme les solutions idéales pour gérer les grands volumes de données, la variété de leurs types caractérisant principalement le concept du Big Data. Plusieurs solutions NoSQL basées sur des architectures différentes sont développées dans tous les secteurs.

Notre travail a fait l'objet d'une étude des nouvelles technologies du Big Data, Nous nous sommes intéressés au Framework Hadoop. qui est l'infrastructure la plus émergente utilisée par la plupart des grandes entreprises telles que Facebook, Microsoft, IBM, Yahoo, Amazon et bien d'autres encore.

Bibliographie

[1] Olivier Losson. Introduction aux systèmes de gestion de bases de données relationnelles, cours master sciences et technologies. *Université Lille1.*

[2] DJebali Hadjer and KEnouze Ilyas. *Un Outil d'administration du Système de Gestion de Base de données Oracle (SGBD Oracle).* 2014.

[3] Wilfrid Niobet. Informatique documentaire et les bases de données. 2017.

[4] Antoine Cornuéjols. bases de données concepts et programmation. *AgroParisTech, Spécialité Informatique*, 2009.

[5] Matteo Di Maglie. *Adoption d'une solution NoSQL dans l'entreprise.* PhD thesis, Haute école de gestion de Genève, 2012.

[6] Jacqueline Konate. Principe des bases de données. 2018.

[7] Hala Skaf-Molli. Bases de données relationnelles.

[8] Georges Gardarin. *Bases de données.* Editions Eyrolles, 2003.

[9] Pascal Dechamboux. *Gestion d'objets persistants : du langage de programmation au système.* PhD thesis, Université Joseph-Fourier-Grenoble I, 1993.

[10] Philippe Rigaux. *Cours de bases de données.* CNAM/Médias, 2001.

[11] Hadrien Furrer. Sql, nosql, newsql stratégie de choix. PhD thesis, Haute école de gestion de Genève, 2015.

[12] Hondjack Dehainsala. *Explicitation de la sémantique dans les bases de données : Base de données á base ontologique et le modéle OntoDB.* PhD thesis, Université de Poitiers, 2007.

[13] Jérôme Gabillaud. *Oracle 11g : SQL, PL/SQL, SQL* Plus.* Editions ENI, 2009.

[14] Richard Grin. Langage sql. 2006.

[15] Nassim DENNOUNI. Base de données avancées.

[16] Kouedi Emmanuel. Approche de migration d'une base de données relationnelle vers une base de données nosql orientée colonne. *Mémoire master informatique, Université de Yaoundé I*, 2012.

[17] Xavier MALETRAS. Le nosql- cassandra, thèse professionnelle). *université paris 13*, 27/05/2012.

[18] Xiaomeng Su. Introduction to big data, institutt for informatikk og elæring ved ntnu. learning material is developed for course iini3012 big data. 2016.

[19] Aymen Chakhari. Presentation des auteurs. 2015.

[20] Amir Gandomi and Murtaza Haider. Beyond the hype : Big data concepts, methods, and analytics. *International Journal of Information Management*, 2015.

[21] S. Singh and N. Singh. Big data analytics. In *2012 International Conference on Communication, Information Computing Technology (ICCICT)*, Oct 2012.

[22] BrightPlanet. Structured vs unstructured data, 2012.

[23] Palak Gupta and Nidhi Tyagi. An approach towards big data—a review. In *Computing, Communication & Automation (ICCCA), 2015 International Conference on*. IEEE, 2015.

[24] Seref Sagiroglu and Duygu Sinanc. Big data : A review. In *Collaboration Technologies and Systems (CTS), 2013 International Conference on*. IEEE.

[25] Hana Mallek. Ingénierie des processus etl pour les big data. In *Actes du 8 e Forum Jeunes Chercheurs du congrès INFORSID*, page 65.

[26] Marie-Pierre Hamel and David Marguerit. Quelles possibilités offertes par l'analyse des big data pour améliorer les téléservices publics ? *Revue française d'administration publique*, 2013.

[27] O Polo. Big data : Une révision. *RÉSUMÉ*, 2015.

[28] Marie-Chantal Denis. *Conception et réalisation d'un entrepôt de données institutionnel dans une perspective de support à la prise de décision*. PhD thesis, Université du Québec à Trois-Rivières, 2008.

[29] Amandine Holemans et al. Implémentation d'une base de données nosql de données géospatiales de l'aide. 2017.

[30] Ralph Kimball and Margy Ross. *The data warehouse toolkit : the complete guide to dimensional modeling*. John Wiley & Sons, 2011.

[31] Erhard Rahm and Hong Hai Do. Data cleaning : Problems and current approaches. *IEEE Data Eng. Bull.*, 2000.

[32] John Quackenbush. Microarray data normalization and transformation. *Nature genetics*, 2002.

[33] Ibrahim Abaker Targio Hashem, Ibrar Yaqoob, Nor Badrul Anuar, Salimah Mokhtar, Abdullah Gani, and Samee Ullah Khan. The rise of "big data" on cloud computing : Review and open research issues. *Information Systems*, 2015.

[34] KONE. Angeline. Big data rapport de stage, 2013.

[35] Avita Katal, Mohammad Wazid, and RH Goudar. Big data : issues, challenges, tools and good practices. In *Contemporary Computing (IC3), 2013 Sixth International Conference on*, pages 404–409. IEEE, 2013.

[36] Blandine LAFFARGUE. guide du big data l'annuaire de référence à destination des utilisateurs. 2013/2014.

[37] assisté de Pierre-Y Groupe de travail Big Data. Contribution éditoriale : Philippe Roux. Big data : transformer les données en valeur business pour l'entreprise. 2014.

[38] Bogdan George Tudorica and Cristian Bucur. A comparison between several nosql databases with comments and notes. In *Roedunet International Conference (RoEduNet), 2011 10th*. IEEE, 2011.

[39] Rudi Bruchez. *Les bases de données NoSQL et le BigData : Comprendre et mettre en oeuvre*. Editions Eyrolles, 2015.

[40] Bernard Espinasse. Introduction aux systèmes nosql (not only sql). *Ecole Polytechnique Universitaire de Marseille*, 2013.

[41] Kanwar Renu and Trivedi Prakriti. Coherence a nosql revolution data distribution and replication in coherence). [En ligne].disponible sur :

https ://nanopdf.com/download/doc02082014133558_pdf . [consulté le :24-03-2018], 2018.

[42] Stefane Fermigier. Big data & open source : une convergence inévitable ?. 2011.

[43] M Leonard. L'avenir du nosql, 2014.

[44] Ameya Nayak, Anil Poriya, and Dikshay Poojary. Type of nosql databases and its comparison with relational databases. *International Journal of Applied Information Systems*, 2013.

[45] ibrahim BERTHE. A la découverte de nosql, 31/10/2015.

[46] Jean SEILER. Nosql panoramal, 2015.

[47] Nishant Neeraj. *Mastering Apache Cassandra*. Packt Publishing Ltd, 2013.

[48] Apache Cassandra. Apache cassandra. *Google Scholar*, 2015.

[49] Ellis Jonathan. Cassandra, réplication et langage des requêtes, 2016.

[50] Ruxandra Burtica, Eleonora Maria Mocanu, Mugurel Ionuţ Andreica, and Nicolae Ţăpuş. Practical application and evaluation of no-sql databases in cloud computing. In *Systems Conference (SysCon), 2012 IEEE International*. IEEE, 2012.

[51] Guillaume Payen. Analyse prédictive sur le site boredpanda. com.

[52] Cristina Băzăr, Cosmin Sebastian Iosif, et al. The transition from rdbms to nosql. a comparative analysis of three popular non-relational solutions : Cassandra, mongodb and couchbase. *Database Systems Journal*, 2014.

[53] Kyle Banker. *MongoDB in action*. Manning Publications Co, 2011.

[54] Miles Ward. Nosql database in the cloud : Mongodb on aws. *Amazon Web Services*, 2013.

[55] Sumitkumar Kanoje, Varsha Powar, and Debajyoti Mukhopadhyay. Using mongodb for social networking website deciphering the pros and cons. In *Innovations in Information, Embedded and Communication Systems (ICIIECS), 2015 International Conference on*.

[56] MABANZA Gilles. Comparaison entre mysql et mongodb, 2017.

[57] Chuck Lam. *Hadoop in action*. Manning Publications Co., 2010.

[58] Tom White. *Hadoop : The definitive guide.* " O'Reilly Media, Inc.", 2012.

[59] Apache Hadoop. Apache hadoop. *URL http ://hadoop. apache. org*, 2011.

[60] Philippe Besse and Nathalie Villa-Vialaneix. Statistique et big data analytics ; volum\'etrie, l'attaque des clones. 2014.

[61] Dhruba Borthakur. The hadoop distributed file system : Architecture and design, 2008.

[62] Thomas Kiencke. Hadoop distributed file system (hdfs), 2013.

[63] Pierre NERZIC. Outils hadoop pour le big data. 2018.

[64] amel ABID. Big data chapitre 2, 2017.

[65] Jeffrey Dean and Sanjay Ghemawat. Mapreduce : simplified data processing on large clusters. *Communications of the ACM*, 2008.

[66] Philippe Besse and Nathalie Villa-Vialaneix. Statistique et big data analytics ; volum\'etrie, l'attaque des clones. *arXiv preprint arXiv :1405.6676*, 2014.

[67] teknono KARDI. Mapreduce tutorial, 2017.

[68] Amogh Pramod Kulkarni and Mahesh Khandewal. Survey on hadoop and introduction to yarn. 2014.

[69] Amogh Pramod Kulkarni and Mahesh Khandewal. Survey on hadoop and introduction to yarn. *International Journal of Emerging Technology and Advanced Engineering*, 2014.

[70] Lizhe Wang, Jie Tao, Rajiv Ranjan, Holger Marten, Achim Streit, Jingying Chen, and Dan Chen. G-hadoop : Mapreduce across distributed data centers for data-intensive computing. *Future Generation Computer Systems*, 2013.

[71] Nawsher Khan, Ibrar Yaqoob, Ibrahim Abaker Targio Hashem, Zakira Inayat, Mahmoud Ali, Waleed Kamaleldin, Muhammad Alam, Muhammad Shiraz, and Abdullah Gani. Big data : survey, technologies, opportunities, and challenges. *The Scientific World Journal*, 2014.

[72] Allae Erraissi, Abdessamad Belangour, and Abderrahim Tragha. A big data hadoop building blocks comparative study. *International Journal of Computer Trends and Technology. Accessed June*, 2017.

[73] Brien POSEY. Clusters hadoop : avantages et limites pour l'analyse des big data, 2017.

[74] Arun C Murthy and Doug Eadline. *Apache Hadoop YARN : moving beyond MapReduce and batch processing with Apache Hadoop 2.* Pearson Education, 2014.

[75] Nidhin Mahesh. Configuration de hadoop -mapreduce, hdfs et yarn. mode autonome et pseudo-distribué, 2017.

[76] Sriram Krishnan, Mahidhar Tatineni, and Chaitanya Baru. myhadoop-hadoop-on-demand on traditional hpc resources. *San Diego Supercomputer Center Technical Report , University of California, San Diego*, 2011.

[77] Myriam Karoui, Grégoire Davauchelle, and Aurélie Dudezert. Big data. mise en perspective et enjeux pour les entreprises. *Ingénierie des Systèmes d'Information*, 2014.

Annexe A

Programme de test hadoop

La classe Mapper

```
package org.hadoop.trainings;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;

public class wordMapper extends MapReduceBase implements Mapper<LongWritable,Text, Text, IntWritable >{

        public void map(LongWritable key, Text value,
                        OutputCollector<Text, IntWritable> output, Reporter r)
                        throws IOException {
                String s = value.toString();
                for (String word : s.split("")){
                        if(word.length()>0){
                                output.collect(new Text(word), new IntWritable(1));
                        }
                }

        }

}
```

FIGURE A.1 – La classe Mapper

La classe Reducer

```
package org.hadoop.trainings;

import java.io.IOException;
import java.util.Iterator;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;


public class wordReducer extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable>{

        public void reduce(Text key, Iterator<IntWritable> values,
                        OutputCollector<Text, IntWritable> output, Reporter r)
                        throws IOException {

                int count = 0 ;
                while(values.hasNext()){
                        IntWritable i = values.next();
                        count+=i.get();
                }
            output.collect(key, new IntWritable(count));
        }

                }
```

FIGURE A.2 – La classe Reducer

La classe du conducteur

```
package org.hadoop.trainings;
import java.io.IOException;

import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class WordCount extends Configured implements Tool{

   public  int run(String[] args) throws IOException {
                if (args.length>2){
                        System.out.println("svp input");
         return -1;
                }
                JobConf conf = new JobConf(WordCount.class);
                FileInputFormat.setInputPaths(conf, new Path(args[0]));
                FileOutputFormat.setOutputPath(conf, new Path (args[1]));
                conf.setMapperClass(wordMapper.class);
                conf.setReducerClass(wordReducer.class);
                conf.setMapOutputKeyClass(Text.class);
                conf.setMapOutputValueClass(IntWritable.class);
                conf.setOutputKeyClass(Text.class);
                conf.setOutputValueClass(IntWritable.class);
                JobClient.runJob(conf);

                return 0;
        }

public static void main (String args[]) throws Exception {
                int exitCode = ToolRunner.run(new WordCount(), args);
                System.exit(exitCode);

        }
}
```

FIGURE A.3 – La classe du conducteur