REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA

RECHERCHE SCIENTIFIQUE

UNIVERSITE IBN KHALDOUN - TIARET

MEMOIRE

Présenté à :

FACULTÉ MATHEMATIQUES ET INFORMATIQUE

DÉPARTEMENT D'INFORMATIQUE Pour l'obtention du diplôme de :

MASTER

Spécialité : Réseau et télécommunication

Par :

Ameur Hadjira

Dahmani Rachid

Sur le thème

[Annotation des images médicales dans un environnement parallèle et distribué]

Soutenu publiquement le …. /09 / 2021 à Tiaret devant le jury composé de :

Mr Nassan Samir	Grade Université MCA	Président
Mr Meratti Mjeded	Grade Université MCB	Encadreur
Mr Belarbi Mostefa	Grade Université MAA	Examinateur

2020_2021

Dédicaces

Je dédie ce travail

A ma famille, elle qui m'a doté d'une éducation digne, son amour a fait de moi ce que je suis aujourd'hui…

A mes chers parents, pour tous leurs sacrifices, leur amour, leur tendresse, leur soutien et leurs prières tout au long de mes études.

A mes chères sœurs ……. pour leurs encouragements permanents, et leur soutien moral.

A mes chers frères……. pour leur appui et leur encouragement.

A tous mes amis de promotion de 2eme année Master.

A tous les personnes qui ont m'aider à réaliser ce travail.

Remerciements

Je voudrais tout d'abord exprimer mes plus profonds remerciements à **ALLAH** *le tout-puissant, pour m'avoir accordé vie, santé et paix de l'esprit sans quoi je n'aurais pu achever ce travail ;*

Ce mémoire n'aurait jamais pu voir le jour sans le soutien actif des membres de nos famille, surtout nos parents qu'ils nous ont toujours encouragé moralement et matériellement et à qui on tient à les remercier.

Notre remerciement s'adresse à notre encadreur Mr **MERATTI MDJEDED** *et notre Co-encadreur Mr* **BELARBI Mustapha** *pour leur encadrement.*

Nos remerciements s'adressent également à tous nos professeurs pour leurs générosités et la grande patience dont ils ont su faire preuve malgré leurs charges académiques et professionnelles.

Nous tenons à remercier sincèrement les membres du jury qui nous font le grand honneur de juger notre travail.

Enfin on tient à exprimer vivement nos remerciements avec une profonde gratitude à toutes les personnes qui ont contribué de près ou de loin à sa réalisation, car un projet ne peut pas être le fruit d'une seule personne.

Résumé

Face à l'explosion de la production massive de données, les grands acteurs du web comme Google, Facebook et Amazon développent et adoptent de nouvelles technologies qui viennent répondre aux exigences de gestion, traitement et d'analyse de ces données gigantesques.

Un Framework Hadoop est l'un de ces technologies et la principale plateforme du

Big Data ; sert à réaliser un stockage distribué et un traitement parallèle afin de réduire le temps d'exécution, garantir la fiabilité et la disponibilité d'un système du cluster, et c'est ce que nous avons abordé principalement dans notre projet de fin d'étude qui a déroulé en deux étapes : une installation et configuration d'Hadoop sur trois machines virtuelles et l'exécution d'un programme qui calcule le nombre des mots dans un texte donnée, ce programme nommé wordcount.

ملخص

في مواجهة الانفجار الهائل في إنتاج البيانات ، يعمل كبار اللاعبين على الويب مثل قوقل و فيسبوك و أمازون على تطوير واعتماد تقنيات جديدة تلبي متطلبات إدارة هذه البيانات الضخمة ومعالجتها وتحليلها.

يعد إطار العمل هادوب أحد هذه التقنيات والنظام الأساسي الرئيسي

البيانات الكبيرة؛ يستخدم لتحقيق التخزين الموزع والمعالجة المتوازية من أجل تقليل وقت التنفيذ ، وضمان موثوقية وتوافر نظام الكتلة ، وهذا ما تناولناه بشكل أساسي في نهاية مشروع الدراسة الذي تم على مرحلتين: التثبيت والتكوين من هادوب على ثلاثة أجهزة افتراضية وتنفيذ برنامج يقوم بحساب عدد الكلمات في نص معين ، يسمى هذا البرنامج عدد الكلمات.

Abstract

Faced with the explosion of massive data production, major web players such as Google, Facebook and Amazon are developing and adopting new technologies that meet the demands of managing, processing and analyzing this gigantic data.

A Hadoop Framework is one of these technologies and the main platform of

Big Data; is used to achieve distributed storage and parallel processing in order to reduce execution time, guarantee the reliability and availability of a cluster system, and this is what we mainly addressed in our end of study project which took place in two stages: an installation and configuration of Hadoop on three virtual machines and the execution of a program which calculates the number of words in a given text, this program called wordcount.

Table des matières

Introduction générale .....

Chapter I: Big Data et Cloud Computing

I.1 Big Data 4

I.1.1 Introduction 4

I.1.2 Definition de Big Data 4

I.1.3 Caractéristiques du Big Data 5

I.1.3.1 Volume 5

I.1.3.2 Vitesse 5

I.1.3.3 Variété 5

I.1.4 Processus de chargement et de collecte de données dans Big Data 6

I.1.5 Différence entre BI (Business intelligence) et Big Data 7

I.1.6 Architecture Big Data 8

I.1.7 Avantages de l'architecture Big Data 8

I.1.8 L'analyse : le point clé du Big Data 9

I.1.9 Quelques domaines d'utilisation du Big Data 9

I.1.10 Big Data et Datawarehouse 10

I.1.11 Big Data et les ETL (extraction, transformation et chargement) 11

I.1.12 Les principales technologies de Big Data 12

I.1.13 Bases données NoSQL 13

I.1.13.1Caractéristiques NoSQL 13

I.1.13.2 Les types des bases NoSQL 14

I.1.13.3 Les principales bases de données NoSQL 15

I.2 Le Cloud Computing 15

I.2.1. Définition 16

I.2.2. Les différents services 16

I.2.3. Les formes de déploiement du Cloud Computing 17

I.3 Conclusion 17

Chapitre II: Hadoop et ses composants

II.1 Introduction 19

II.2 Présentation d'Hadoop 19

II.2.1 Le système de fichier distribué d'Hadoop HDFS 20

II.2.1.1 Les composantes d'HDFS 21

II.2.1.2 HDFS et tolérance aux fautes 22

II.2.1.3 Lecture d'un fichier HDFS 23

II.2.1.4 Ecriture dans un fichier ou volume HDFS 23

II.2.1.5 HDFS : Stratégies de placement de bloc 24

II.2.2 MapReduce 24

II.2.2.1 MapReduce dans Hadoop 26

II.2.2.2 Architecture du Framework MapReduce (purement maître-esclave) 26

II.2.2.3 Fonctionnement du Framework MapReduce 27

II.2.2.4 Hadoop MapReduce 2.x: YARN 29

II.2.2.5 MapReduce et HDFS 30

II.2.3 Écosystème d'Hadoop 30

II.2.4 Outils composant le noyau Hadoop 31

II.2.5 Les outils de Requêtage et de Scripting des données dans Hadoop 32

II.2.6 L'outil d'intégration SGBD-R (Relationnel) Sqoop (Cloudera) 32

II.2.7 Les outils de gestion et de supervision du cluster Hadoop 32

II. 2.8Outil d'ordonnancement et de coordination : Apache Oozie (Yahoo) 33

II.3 Conclusion 34

Chapitre III : Mise en œuvre, Test et Evaluation

III.1 Introduction 35

III.2 Motivation et problématique 35

III.3 Qu'allons-nous installons pour créer le cluster Hadoop multi-node 35

III.4 L'environnement de travail 35

III.4.1 Virtual Box 35

III.4.2 Ubuntu 20.04 TLS 36

III.4.3 Choix d'Ubuntu 36

III.5 Les étapes d'installation et de configuration de notre systéme 37

III.5.1 Configuration du réseau. 37

III.5.2 Installation d'SSH 37

III.5.3 Installation d'PDSH 37

III.5.4 L'ouvre de fichier bashrc 38

III.5.5 Configuration d'SSH 39

III.5.6 La copie de la clé publique 39

III.5.7 Vérification d'SSH 40

III.5.8 Installation de Java8 40

III.5.9 Vérification que Java est correctement installé 40

III.5.10 Téléchargement d'Hadoop 41

III.5.11 Décompression du fichier hadoop-3.3.1.tar.gz 41

III.5.12 Changement du nom de dossier hadoop-3.3.1 en hadoop 41

III.5.13 Edition de JAVA_HOME 41

III.5.14 Remplacement du répertoire du dossier hadoop 42

III.5.15 L'ouvre de fichier d'environnement 42

III.5.16 L'ajout d'un utilisateur hadoopuser....42
III.5.17 Vérification de l'adresse IP de la machine....43
III.5.18 Installation des configurations réseau....44
III.5.19 Création du node master....44
III.5.20 Insertion du nom de master....45
III.5.21 Configuration d'SSH sur hadoop-master....46
III.5.22 Création de la clé SSH....47
III.5.23 Copie de la clé pour tous les utilisateurs....47
III.5.24 L'ouvre du fichier core-site.xml....49
III.5.25 L'ouvre du fichier hdfs-site.xml....50
III.5.26 L'ouvre du fichier workers....51
III.5.27 La copie des configurations hadoop-master sur les slaves....52
III.5.28 Formate du HDFS....52
III.5.29 Démarrage de HDFS....53
III.5.30 Vérification de fonctionnement....54
III.5.31 Configuration YARN....54
III.5.32 L'ouvre du fichier fil-site.xml....55
III.5.33 Commencement du fil....56
III.5.34 Test de navigateur....56
III.6 Les problèmes rencontrés....57
Conclusion générale....59
Références bibliographiques....60
Annexes....63

Liste des Figures

Figure I.1: Le Big Data, les 3 V 5
Figure I.2 : Couche de chargement des données dans le Big Data 6
Figure I.3: Architecture de Big Data 8
Figure I.4 : Lien entre Big Data et DW 11
Figure I.5 : Utilisation d'ETL Informatica pour Big Data 12
Figure I.6: Les solutions de stockage 13
Figure I.7: Services Cloud Computing 16
Figure II.1 : L'architecture d'HDFS..... 21
Figure II.2 : Fonctionnement du Secondary NameNode 22
Figure II.3 : Lecture d'un fichier HDFS. 23
Figure II.4 : Processus d'écriture dans un volume ou fichier HDFS.....24
Figure II.6 : Schéma de fonctionnement de MapReduce 27
Figure II.7 : Schéma de fonctionnement de MapReduce 2. 29
Figure II.8 : Architecture maître-esclave du MapReduce 30
Figure II.9: Ecosystème d'Hadoop 33

Liste des abréviations

BI : Business Intelligence.

DW: Datawarhouse

ETL: Extraction, transformation et chargement

GFS : Google file système.

HDFS : Système du fichier distribués (Hadoop distributed file système).

JVM: Java Virtual machine.

SQL: Structured Query Language.

SGBDR: SGBD Relationnel.

MPP: Massively Parallel Processing.

RAID: Redundant Array of Independent Disks.

SGBD : Système de Gestion de Base de données.

SQL TLS : Protocoles de transaction sécurisée.

VM : Virtuale Machine.

INTRODUCTION GENERALE

Contexte

Nous sommes confrontés actuellement à une explosion de données structurées ou non structurées produites massivement par les différentes sources de données numériques. D'une part les applications qui génèrent des données issues des logs, des réseaux de capteurs, des traces de GPS, etc., et d'autre part, les utilisateurs produisent beaucoup de données telles que des photographies, des vidéos et des musiques. Selon IBM, chaque heure 2.5 trillions d'octets de données sont générées. Selon les prévisions faites, d'ici 2020 cette croissance sera supérieure à 40 Zettaoctets, alors qu'un Zettaoctet de données numériques, seulement, ont été générées de 1940(début de l'informatique) à 2010. Beaucoup de concepts «inséparables» dominent actuellement le marché de l'IT : «Cloud Computing», «Big Data», «NoSQL» ou «MapReduce».

Problématique

De rudes contraintes opposent les différents chercheurs dans le domaine, quant au stockage et à l'analyse de ces masses de données. Les prévisions de taux de croissance des volumes de données traitées dépassent les limites des technologies traditionnelles à savoir les bases de données relationnelles ou les Datawarehouses. On parle de pétaoctet

(billiard d'octets 10^{15}), voire d'exaoctet (trilliard d'octets 10^{18}) et encore le Zettaoctet(10^{21}) ou le Yottaoctet(10^{24}).

Notre problématique est comment prendre en charge l'accroissement rapide des volumes de données géographiquement éloignées et comment gérer cette puissante montée en charge. Quel sont les technologies et les modèles de programmation proposées pour pallier ces différents problèmes engendrés par ce déluge de données ?

Solution

Cette révolution scientifique qui envahisse le monde de l'information et l'Internet a imposé aux différents chercheurs depuis quelques années, de nouveaux défis et les a poussé à concevoir de nouvelles technologies pour contenir, traiter ces volumes énormes de données. Plusieurs modèles de programmation parallèle et systèmes de gestion de fichiers distribués ont émergé, principalement, le Framework Hadoop se place comme la solution la plus répandue dans le marché informatique.

Hadoop est un environnement d'exécution distribuée, performant et scalable, il propose un système de stockage distribué via son système de fichier HDFS (Hadoop Distributed File

System) et un système d'analyse et traitement de données basé le modèle de

programmation MapReduce pour réaliser des traitements parallèles et distribués sur des gros volumes de données.

Notre projet de fin d'étude a pour but d'étudier les méthodes et les technologies du Big Data, Nous nous intéresserons particulièrement aux technologies Hadoop avec ses composantes HDFS et MapReduce.

La partie applicative, va consister à un déploiement d'Hadoop avec tous ses composants dans un environnement distribué, configuration d'un cluster de plusieurs machines, exécution de plusieurs jobs MapReduce sur des fichiers HDFS et rapprochement des différentes statistiques des opérations effectuées. L'installation d'Hadoop a été effectuée à partir de la distribution Apache sur des machines virtuelles avec Linux Ubuntu20.04 LTS comme système d'exploitation.

Objectif

Mettre en œuvre une architecture complète d'un cloud computing et montrer l'influence du parallélisme sur les performances globales du système par l'annotation des images médicales dans ce cluster hadoop, d'un côté. D'un autre côté, la familiarisation avec l'environnement Hadoop et la maitrise de nouvelles technologies, MapReduce, HDFS et NoSQL sont les objectifs principaux ciblés par notre projet de fin d'études.

Organisation du mémoire

Nous commencerons dans ce projet par une introduction générale décrivant le

contexte de travail, la problématique, la solution adoptée et l'objectif de ce sujet.

La deuxième partie s'articule autour de trois chapitres qui se présentent comme suit :

Dans le premier chapitre, on a introduit les concepts des Big Data et du Cloud

Computing, on a également présenté le NoSQL, l'un des principales technologies du Big Data.

Le deuxième chapitre est consacré à la présentation du Framework Hadoop, la description de ses principaux composants HDFS et sa propre version de MapReduce.

Dans le troisième chapitre, on a abordé la partie technique qui consiste à déployer, installer et configurer un cluster hadoop et tester l'ensemble des éléments logiciels nécessaires pour la mise en œuvre de la solution.

Nous achèverons avec une conclusion, dans laquelle on va synthétiser tous les résultats obtenus et les enseignements acquis durant la réalisation de ce projet.

CHAPITRE I

Big Data et Annotation des Images.

I.1 Big Data

I.1.1 Introduction

De grandes quantités d'informations sont mises en ligne sur le web par des milliers d'entreprises, d'organisations et d'individus, la charge ainsi que le volume de données à gérer ont crû de façon exponentielle, pour cela les sociétés ont recouru au

Datawarehouse ou entrepôt de données pour l'analyse et le stockage de données. Généralement le Datawarehouse est centralisé dans un serveur connecté à une baie de stockage, cette solution est difficilement scalable (ajout de puissance à la demande) en plus du fait qu'elle ne gère que les données structurées dans des SGBD.

Pour faire face à l'explosion du volume des données, on parle actuellement de pétaoctet (billiard d'octets) voir de zettaoctet (trilliard d'octets) et aussi face à la grande variété des données (image, texte, web, etc.) un nouveau domaine technologique a vu le jour : le Big Data inventé par les géants du web, au premier rang comme Yahoo, Google et Facebook, qui ont été les tous premiers à déployer ce type de technologie.

Ce concept apporte une architecture distribuée et scalable pour le traitement et le stockage de données. Ce nouveau paradigme a pour principal objectif l'amélioration des performances et l'augmentation de la vitesse d'exécution des requêtes et des traitements.

I.1.2 Définition de Big Data

Le terme de Big Data a été évoquée la première fois par le cabinet d'études Gartner en 2008 mais la naissance de ce terme effective remonte à 2001 et a été évoquée par le cabinet Meta Group.

Il fait référence à l'explosion du volume des données (de par leur nombre, la vitesse à laquelle elles sont produites et leur variété) et aux nouvelles solutions proposées pour gérer cette volumétrie tant par la capacité à stocker et explorer, et récemment par la capacité à analyser et exploiter ces données dans une approche temps réel. [12]

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

hétérogènes et changeants, et d'en extraire les informations pertinentes à un coût accessible. [16]

I.1.3 Caractéristiques du Big Data

Le Big Data (en français "Grandes données") regroupe une famille d'outils qui répondent à une triple problématiques : C'est la règle dite des **3V**. [10]

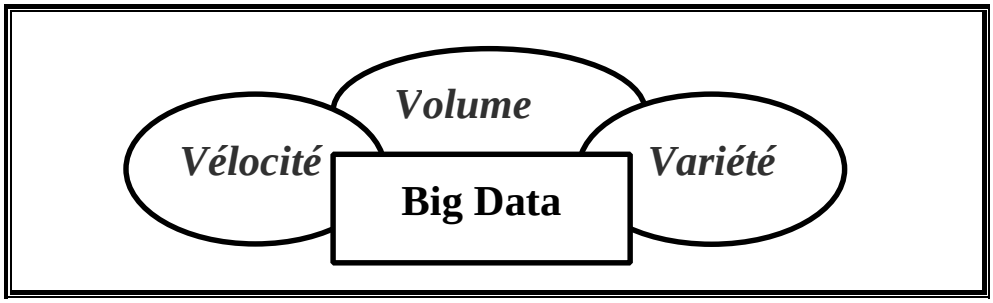


Figure I.1: Le Big Data, les 3 V [16]

I.1.3.1 Volume

Le Big Data est associé à un volume de données vertigineux, se situant actuellement entre quelques dizaines de téraoctets et plusieurs péta-octets en un seul jeu de données.

Les entreprises et tous les secteurs d'activités confondus, devront trouver des moyens pour gérer le volume de données en constante augmentation qui est créé quotidiennement. Les catalogues de plus de 10 millions de produits sont devenus la règle plutôt que l'exception.

Voici quelques chiffres pour illustrer ce phénomène :

- 90% des données actuelles ont été créées dans les deux dernières années seulement ;
- Twitter comme exemple, génère 7 To de données chaque jour.

I.1.3.2 Vitesse

La vitesse décrit la fréquence à laquelle les données sont générées, capturées et partagées. Les entreprises doivent appréhender la vitesse non seulement en termes de création de données, mais aussi sur le plan de leur traitement, de leur analyse et de leur restitution à l'utilisateur en respectant les exigences des applications en temps réel.

I.1.3.3 Variété

La croissance de la variété des données est la conséquence des nouvelles données multi structurelles et de l'expansion des types de données provenant de différentes sources

hétérogènes. Aujourd'hui, on trouve des capteurs d'informations aussi bien dans les appareils électroménagers, les trains, les automobiles ou les avions, qui produisent des informations très variées.

Ces nouvelles données dites **non-structurées** sont variées :

- Des photos ;
- Des mails (avec l'analyse sémantique de leur contenu) ;
- Les données issues des réseaux sociaux (commentaires et avis des internautes sur Facebook ou Twitter par exemple) ;

Ces trois caractéristiques illustrées par les trois « V », sont les principes définissant le Big Data. Avant tout, il s'agit d'un changement d'orientation sur l'utilisation de la donnée. En somme, le point clé du Big Data est de donner un sens à ces grosses données et pour cela, il faut les analyser.

I.1.4 Processus de chargement et de collecte de données dans Big Data La couche responsable du chargement de données dans Big Data, devrait être capable de gérer d'énorme volume de données, avec une haute vitesse, et une grande variété de données. Cette couche devrait avoir la capacité de valider, nettoyer, transformer, réduire (compression), et d'intégrer les données dans la grande pile de données en vue de son traitement. La Figure illustre le processus et les composants qui doivent être présent

dans la couche de chargement de données. [2][5]

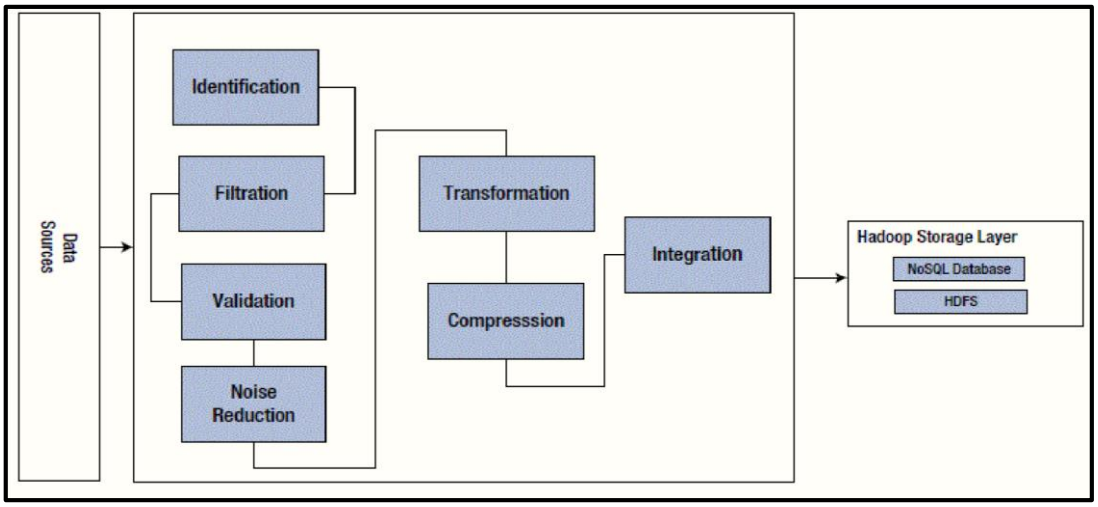


Figure I.2 : Couche de chargement des données dans le Big Data [2]

La couche de chargement de données de Big Data collecte les informations pertinentes finales, sans bruit, et les charge dans la couche de stockage de Big Data (HDFS ou NoSQL base).Elle doit inclure les composants suivants :

- **Identification** des différents formats de données connues, par défaut Big Data cible les données non structurées ;
- **Filtration et sélection** de l'information entrante pertinente pour l'entreprise ;
- **Validation** et analyse des données en permanence ;
- **Réduction** de bruit implique le nettoyage des données en supprimant le bruit ;
- **La transformation** peut entrainer le découpage, la convergence, la normalisation ou la synthèse des données ;
- **Compression** consiste à réduire la taille des données, mais sans perdre de la pertinence des données ;
- **Intégration** consiste à intégrer l'ensemble des données dans le stockage de données de Big Data (HDFS ou NoSQL base).

I.1.5 Différence entre BI (Business intelligence) et Big Data

La méthodologie BI traditionnel fonctionne sur le principe de regrouper toutes les données de l'entreprise dans un serveur central (Datawarehouse ou entrepôt de données). Les données sont généralement analysées en mode déconnecté.

Les données sont généralement structurées en SGBDR avec très peu de données non structurées. [2][5]

Une solution Big Data, est différente d'une BI traditionnel dans les aspects suivants :

- Les données sont conservées dans un système de fichiers distribué et scalable plutôt que sur un serveur central ;
- Les données sont de formats différents, à la fois structurées ainsi que non structurées ;
- Les données sont analysées en temps réel ;
- La technologie Big Data s'appuie sur un traitement massivement parallèle (concept MPP).

I.1.6 Architecture Big Data

On distingue principalement les couches suivantes :

- **Couche matériel** (infrastructure Layer) :peut-être des serveurs virtuels VMware, ou des serveurs lame blade ;
- **Couche stockage** (Storage layer) : les données seront stockées soit dans une base NoSQL, ou bien directement dans le système de fichier distribué ou les Datawarehouse ;
- **Couche management et traitement** : on trouve dans cette couche les outils de traitement et analyse des données comme MapReduce ou Pig ; [2]

Couche visualisation Pour la visualisation du résultat du traitement.

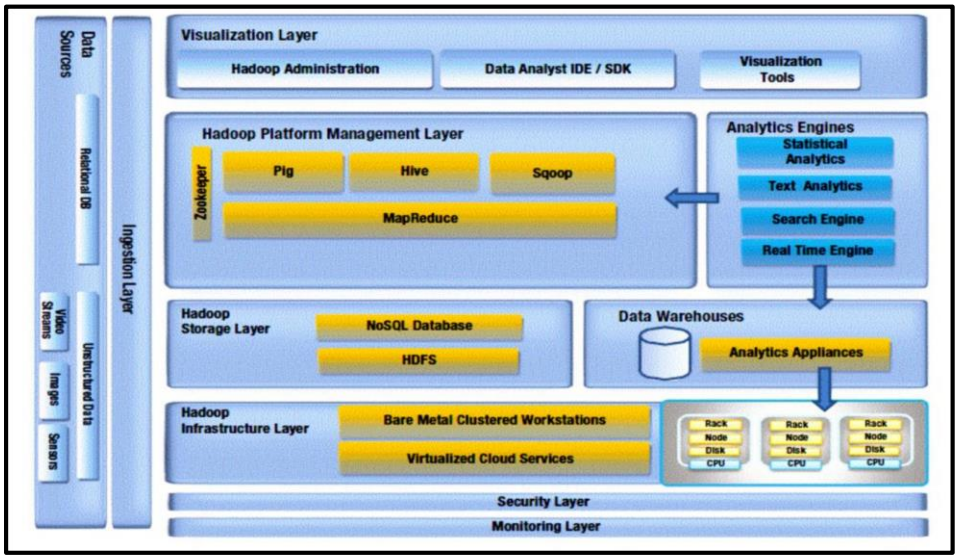


Figure I.3: Architecture de Big Data [2]

I.1.7 Avantages de l'architecture Big Data

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

- **Evolutivité (scalabilité)** : Quelle est la taille que devra avoir votre infrastructure ? Combien d'espace disque est nécessaire aujourd'hui et à l'avenir ? le concept Big Data nous permet de s'affranchir de ces questions, car il apporte une architecture scalable.
- **Performance** : Grâce au traitement parallèle des données et à son système de fichiers distribué, le concept Big Data est hautement performant en diminuant la latence des requêtes.
- **Coût faible** : Le principal outil Big Data à savoir Hadoop est en Open Source, en plus on n'aura plus besoin de centraliser les données dans des baies de stockage souvent excessivement

chère, avec le Big Data et grâce au système de fichiers distribués les disques internes des serveurs suffiront.

- **Disponibilité** : On a plus besoin des RAID disques, souvent coûteux.

L'architecture Big Data apporte ses propres mécanismes de haute disponibilité.

I.1.8 L'analyse : le point clé du Big Data

Le Big Data répond à de nombreux objectifs précis parmi lesquels on trouve l'**extraction** d'informations utiles des données stockées, l'**analyse** de ces données, la **restitution** efficace des résultats d'analyse ou encore, l'accroissement de **l'interactivité** entre utilisateurs et données.

La combinaison de ce déluge d'informations et d'algorithmes logiciels intelligents ouvre la voie à de nouvelles opportunités de business. Prenons, par exemple, Google et Facebook qui sont des « entreprises Big Data », mais aussi IBM ou JDA Software.

L'analyse est le point clé de l'utilisation du Big Data. Elle permet de mieux connaître sa clientèle, d'optimiser son marketing, de détecter et prévenir des fraudes, d'analyser son image sur les réseaux sociaux et d'optimiser ses processus métiers. [10]

I.1.9 Quelques domaines d'utilisation du Big Data

Avant de conclure, citons rapidement quelques domaines d'utilisation du Big Data. Le Big Data trouve sa place dans de nombreux domaines :

Dans la première catégorie, on retrouve des secteurs qui manipulent quotidiennement des volumes de données très important, avec des problématiques de vitesse associées. On y trouve

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

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

- **Les Services Publics** : l'analyse des compteurs (gaz, électricité, etc.) et la gestion des équipements ;
- **Le Marketing** : le ciblage publicitaire et l'analyse de tendance ;
- **La Santé** : l'analyse des dossiers médicaux et l'analyse génomique. [10]

I.1.10 Big Data et Datawarehouse

Les entrepôts de données sont traditionnellement des supports des données structurées et ont été étroitement lié aux systèmes opérationnels et transactionnels de l'entreprise (SGBDR). Ces systèmes qui sont soigneusement construits sont maintenant au milieu d'importants changements après l'émergence de Big Data. [3][5]

Datawarehouse est une base de données (données structurées) regroupant une partie ou l'ensemble des données fonctionnelles d'une entreprise. Il entre dans le cadre de l'informatique décisionnelle ; son but est de fournir un ensemble de données servant de référence unique, utilisées pour la prise de décisions dans l'entreprise par les baies de statistiques et de rapports réalisés via des outils de reporting. D'un point de vue technique, il sert surtout à 'délester' les bases de données opérationnelles des requêtes pouvant nuire à leurs performances. [17]

Les organisations continueront inévitablement à utiliser des entrepôts de données pour gérer le type de données structurées et opérationnelles qui caractérise les systèmes relationnels(SGBDR). Ces entrepôts seront toujours fournis aux analystes avec la capacité pour analyser les données clés, les tendances, et ainsi de suite.

Cependant, avec l'avènement du Big Data, le défi pour les entrepôts de données est de réfléchir à une approche complémentaire avec le Big Data, on pourrait concevoir un modèle hybride. Dans ce modèle les restes de données optimisées opérationnelles très structurées seront stockées et analysées dans l'entrepôt de données, tandis que les données qui sont fortement distribuées et non structurées seront contrôlées par Big Data (Hadoop ou NoSQL). [3][5]

La tendance serait de stocker la grande masse de données non structurées dans une vaste gamme de serveurs Big Data (Hadoop/MapReduce) pour tirer profit de la scalabilité et la rapidité d'analyse de Big Data, ensuite à l'aide d'outils, ces données seront déplacées dans le modèle relationnel de sorte qu'elles peuvent être interrogées avec le langage SQL traditionnel (SGBDR et Datawarehouse).

On peut donc interfacer Big Data avec le Datawarehouse(DW), effectivement les données non structurées provenant de différentes sources peuvent être regroupées dans un HDFS avant d'être

transformées et chargées à l'aide d'outils spécifiques dans le Datawarehouse et les outils traditionnels de BI. [2][5]

Comme les DW traditionnels ne gèrent pas les données non structurées, Big Data peut servir comme un moyen de stockage et d'analyse des données non structurées qui seront chargées dans les DW.

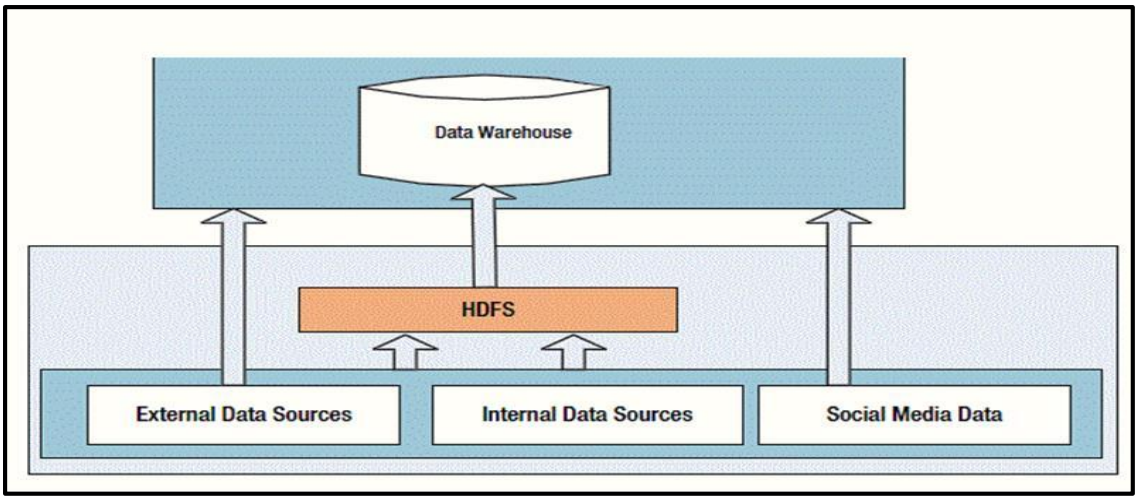


Figure I.4 : Lien entre Big Data et DW [2]

I.1.11 Big Data et les ETL (extraction, transformation et chargement)

Certains outils ETL traditionnels comme Talent commence à s'adapter avec le monde Big Data. Les outils ETL sont utilisés pour transformer les données dans le format requis par l'entrepôt de données(Datawarehouse). La transformation est effectivement faite dans un endroit intermédiaire avant que les données ne seront chargées dans l'entrepôt de données.

Pour le Big Data des outils ETL comme Informatica ont été utilisés pour permettre une solution d'ingestion rapide et flexible des données non structurées (supérieure à 150 Go/jour). [2].

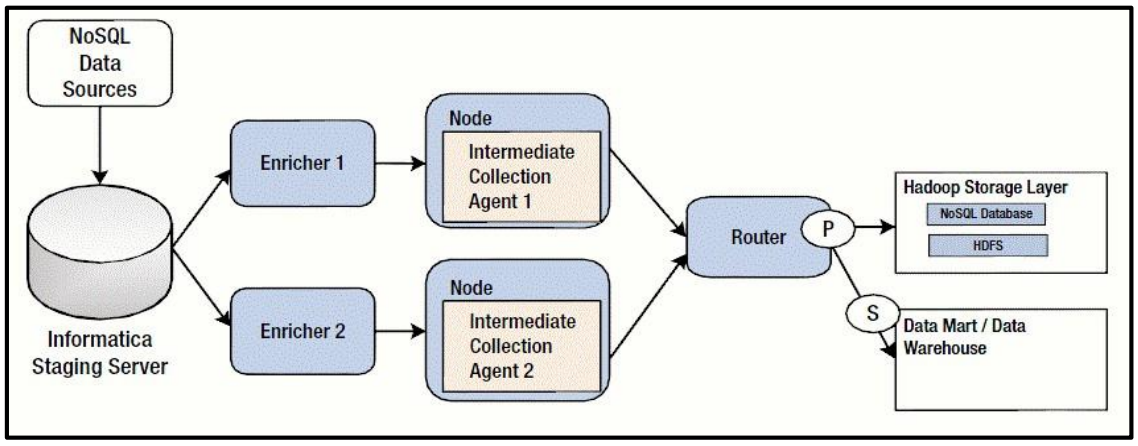


Figure I.5 : Utilisation d'ETL Informatica pour Big Data [2]

I.1.12 Les principales technologies de Big Data

Elles sont nombreuses. Pour optimiser les temps de traitement sur des bases de données géantes, plusieurs solutions peuvent entrer en jeu :

- **Des bases de données NoSQL** (comme MongoDB, Cassandra ou Redis) qui implémentent des systèmes de stockage considérés comme plus performants que

le traditionnel SQL pour l'analyse de données en masse (orienté clé/valeur, document, colonne ou graphe).

- **Des infrastructures de serveurs pour distribuer les traitements** sur des dizaines, centaines, voire milliers de nœuds. C'est ce qu'on appelle le traitement massivement parallèle. Le Framework Hadoop est sans doute le plus connu d'entre eux. Il combine le système de fichiers distribué HDFS, la base NoSQL HBase et l'algorithme MapReduce. (Hadoop, HDFS et MapReduce seront présentés dans le chapitre II).

- **Le stockage des données en mémoire :** On parle de traitement in-memory pour évoquer les traitements qui sont effectués dans la mémoire vive de l'équipement informatique, plutôt que sur des serveurs externes. L'avantage du traitement in-memory est celui de la vitesse

puisque les données sont immédiatement accessibles. En revanche, ces données ne sont pas stockées

sur le long terme, ce qui peut poser des problèmes d'historisation. [11]

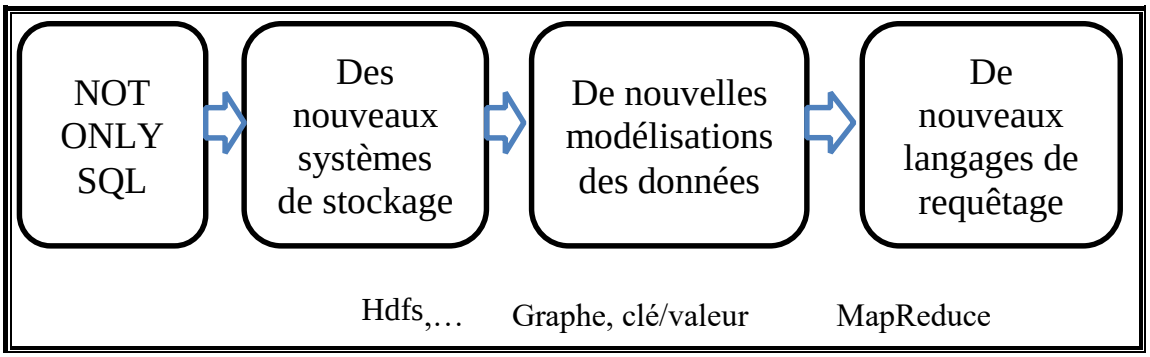


Figure I.6: Les solutions de stockage

I.1.13 Bases données NoSQL

Les bases de données NoSQL (No-SQL ou Not Only SQL) sont un sujet très à la mode en ce moment. Le terme NoSQL désigne une catégorie de systèmes de gestion de base de données destinés à manipuler des bases de données volumineuses pour des sites de grande audience. Les bases données NoSQL sont scalables, elles permettent de traiter les données d'une façon distribuée. Parmi les avantages du NoSQL on trouve :

- Leurs performances ne s'écroulent jamais quel que soit le volume traité. Leur temps de réponse est proportionnel au volume ;
- Elles se migrent facilement. En effet, contrairement aux SGBDR classiques, il n'est pas nécessaire de procéder à une interruption de service pour effectuer le déploiement d'une fonctionnalité impactant les modèles des données ;
- Elles sont facilement scalable. A titre d'exemple, le plus gros cluster de NoSQL fait 400 To, tandis qu'Oracle sait traiter jusqu'à une vingtaine de Téraoctet (pour des temps de réponse raisonnable).

I.1.13.1Caractéristiques NoSQL

- Gros volume de données ;
- Réplication scalable et distribution ;
  - – Des centaines de machines voire des milliers;
  - – Distribuées partout dans le monde.
- Des requêtes qui exigent une réponse rapide;

- Asynchronicité des insertions et updates ;
- Acidité non respectée dans la plupart du temps ;
- Des performances en lectures/écritures ;
- Centaines de milliers de lectures/secondes ;
- Centaines de milliers d'écritures/secondes ;

I.1.13.2 Les types des bases NoSQL

Il en existe 4 types distincts qui s'utilisent différemment et qui se prêtent mieux selon le type données que l'on souhaite y stocker. [7]

Clé-Valeur

Les BD NoSQL fonctionnant sur le principe Clé-Valeur sont les plus basiques que l'on peut trouver.

- Elles fonctionnent comme un grand tableau associatif et retourne une valeur dont elle ne connaît pas la structure ;
- Leur modèle peut être assimilé à une table de hachage (hashmap) distribuée **;**
- Les données sont simplement représentées par un couple clé/valeur ;
- La valeur peut être une simple chaîne de caractères, ou un objet sérialisé.
- **Document**

Elles sont basées sur le modèle « clé-valeur » mais la valeur est un document en format semi-structuré hiérarchique de type JSON ou XML (possible aussi de stocker n'importe quel objet, via une sérialisation). Elles stockent une collection de

"documents" .

- **Colonnes**

Les données sont stockées par colonne, non par ligne, on peut facilement ajouter des colonnes aux tables, par contre l'insertion d'une ligne est plus coûteuse quand les données d'une colonne se ressemblent, on peut facilement compresser la colonne.

C’est un modèle proche d’une table dans un SGBDR mais ici le nombre de colonnes:

- est **dynamique ;**

- peut **varier d'un enregistrement à un autre,** ce qui évite de retrouver des colonnes ayant des valeurs NULL.

- **Graphe**

Elles permettent la modélisation, le stockage et la manipulation de données complexes liées par des relations non-triviales ou variables

- modèle de représentation des données basé sur la théorie des graphes
- s'appuie sur les notions de nœuds, de relations et de propriétés qui leur sont rattachées.

I.1.13.3 Les principales bases de données NoSQL

- **MongoDB**

La plus populaire des bases NoSQL documentaires est écrite en C et n'utilise pas de machine virtuelle JAVA. Cette base est idéale pour débuter car elle est à la fois polyvalente et simple. Aussi à l'aise pour le stockage massif de données que pour le développement rapide orienté Web. Elle possède également une documentation de premier ordre. [18] .

- **Cassandra**

Cassandra est le projet open source qui découlent de la technologie de stockage

Facebook. À l'origine, elle a été écrite spécifiquement pour répondre à la croissance explosive de cette entreprise. Elle est assez complexe à configurer, mais elle permet d'adresser toutes les situations ou la performance et le traitement de la volumétrie est critique. Cassandra est une base de données en colonnes écrite en JAVA. [18]

- **HBase**

Hbase est inspirée des publications de Google sur BigTable. Comme BigTable, elle est une base de données oriente colonne. Basée sur une architecture maitre/esclave, les bases de données HBase sont capables de gérer d'énormes quantités d'informations (plusieurs milliards de lignes par table). [5]

I.2 L'annotaion des images

Définition

L'annotation peut être définie comme le processus d'interprétation explicite du document. La création des métadonnées à partir des documents annotés est l'une des principales techniques

permettant aux données d'être compréhensible par la machine. Les métadonnées associées aux images peuvent être classés comme suit :

i) les métadonnées de contenu indépendant, où les métadonnées sont liées à l'image, comme les noms des auteurs, les dates, le lieu.

ii) le contenu en fonction de métadonnées, où les métadonnées sont liés aux caractéristiques de bas niveau et / ou niveau intermédiaire, par exemple: la couleur, la texture, la forme.

iii) les métadonnées descriptives de contenu, où les métadonnées sont liées au contenu sémantique. [24]

Le contenu descriptif des métadonnées peut être fourni à deux niveaux de spécificité:

i) un contenu descriptif associé à l'image complète.

ii) la segmentation d'images avec des liens vers le contenu descriptif de chaque segment. Les systèmes d'annotions des images par un contenu descriptif basées sur les ontologies, utilisent généralement deux types d'ontologies : une permettant de définir un schéma d'annotation et une autre pour définir les concepts de domaine [24] .

Dans la section suivante nous allons rapidement présenter les ontologies ainsi les ontologies utilisé dans la recherche des images médicales.

I.3 Conclusion

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

On a aussi recensé les différents modèles de bases de données NoSQL qui existent, actuellement, dans le marché en accentuant sur les solutions les plus populaires. Nous avons présenté brièvement en fin du chapitre les différents concepts liés au Cloud Computing, à savoir les types de services Cloud connus jusqu'à présent et ses différents modes de déploiement.

CHAPITRE II

Hadoop et ses composants

II.1 Introduction

Le Big data regroupe plusieurs nouvelles technologies et d'outils pour répondre à une triple problématique : un **V**olume de données important à traiter, une grande **V**ariété d'informations (structurées ou non structurées), et un certain niveau de **V**élocité à atteindre. Pour répondre à ces besoins, il s'avère que l'écosystème Hadoop serait la solution open source par excellence.

Apache Hadoop (High-availability distributed object-oriented platform) est un système distribué qui répond à ces problématiques. D'une part, il propose un système de fichier distribué HDFS (Hadoop Distributed File System) pour assurer le stockage et l'intégrité des données en dupliquant plusieurs copies d'un même bloc à travers des dizaines, des centaines, voire des milliers de machines différentes, ce qui amène un cluster Hadoop à être configuré sans requérir un système RAID. D'autre part, Hadoop fournit un système d'analyse de données appelé MapReduce pour réaliser des traitements sur des gros volumes de données grâce à sa répartition efficace du travail sur différents nœuds de calcul.

II.2 Présentation d'Hadoop

Hadoop est un Framework Java open source d'Apache pour réaliser des traitements sur des volumes de données massifs, de l'ordre de plusieurs pétaoctets (soit plusieurs milliers de To).

Hadoop a été conçu par Doug Cutting en 2004, également à l'origine du moteur Open Source Nutch. Doug Cutting cherchait une solution pour accroître la taille de l'index de son moteur. Il eut l'idée de créer un Framework de gestion de fichiers distribués. Yahoo! en est devenu ensuite le principal contributeur, le portail utilisait notamment l'infrastructure pour supporter son moteur de recherche historique. Comptant plus de 10 000 clusters Linux en 2008, il s'agissait d'une des premières architectures Hadoop digne de ce nom. Crée spécialement pour les gros volumes. Facebook pour l'analyse des logs, Google pour l'analyse des requêtes, etc... Il est caractérisé par :

- **Robuste** : si un nœud de calcul tombe, ses tâches sont automatiquement réparties sur d'autres nœuds. Les blocs de données sont également répliqués;
- **Coût** : il optimise les coûts via une meilleure utilisation des ressources présentées;
- **Souple** : car il répond à la caractéristique de variété des données en étant capable de traiter différents types de données;

- **Virtualisation** : ne plus se reposer directement sur l'infrastructure physique (baie de stockage coûteuse), mais choisir la virtualisation de ses clusters Hadoop.

Trois principales distributions Hadoop sont aujourd'hui disponibles : Cloudera, Hortonworks, MapR.

Nous allons présenter deux concepts fondamentaux d'Hadoop : Sa propre version de l'algorithme MapReduce à savoir **Hadoop MapReduce** et son système de fichiers distribué **HDFS**.

II.2.1 Le système de fichier distribué d'Hadoop HDFS

Hadoop utilise un système de fichiers virtuel qui lui est propre : le HDFS (Hadoop Distributed File System).

HDFS est un système de fichier distribué, extensible et portable inspiré par le Google File System (GFS).

Il a été conçu pour stocker de très gros volumes de données sur un grand nombre de machine équipées de disques durs banalisés, il permet de l'abstraction de l'architecture physique de stockage, afin de manipuler un système de fichier distribué comme s'il s'agissait d'un disque dur unique. [19]

Toutefois, HDFS se démarque d'un système de fichiers classique pour les principales raisons suivantes [15]:

- HDFS n'est pas dépendant du noyau du système d'exploitation. Il assure une portabilité et peut être déployé sur différents systèmes d'exploitation. Un de ses inconvénients est de devoir solliciter une application externe pour monter une unité de disque HDFS ;

- HDFS est un système distribué sur un système classique, la taille du disque est généralement considérée comme la limite globale d'utilisation. Dans HDFS, chaque nœud d'un cluster correspond à un sous-ensemble du volume global des données du cluster. Pour augmenter ce volume global, il suffira d'ajouter de nouveaux nœuds. On retrouvera également dans HDFS, un service central appelé NameNode qui aura la tâche de gérer les métadonnées.

- HDFS utilise des tailles de blocs largement supérieures à ceux des systèmes classiques. Par défaut, la taille est fixée à 64 Mo. Il est toutefois possible de monter à 128 Mo, 256 Mo, 512 Mo voire 1 Go. Alors que sur des systèmes classiques, la taille est généralement de 4 Ko, l'intérêt de fournir des tailles plus grandes permettant de réduire le temps d'accès à un bloc.

Notez que si la taille du fichier est inférieure à la taille d'un bloc, le fichier n'occupera pas la taille totale de ce bloc ;

- HDFS fournit un système de réplication des blocs dont le nombre de réplications est configurable. Pendant la phase d'écriture, chaque bloc correspondant au fichier est répliqué sur plusieurs nœuds. Pour la phase de lecture, si un bloc est indisponible sur un nœud, des copies de ce bloc seront disponibles sur d'autres nœuds.

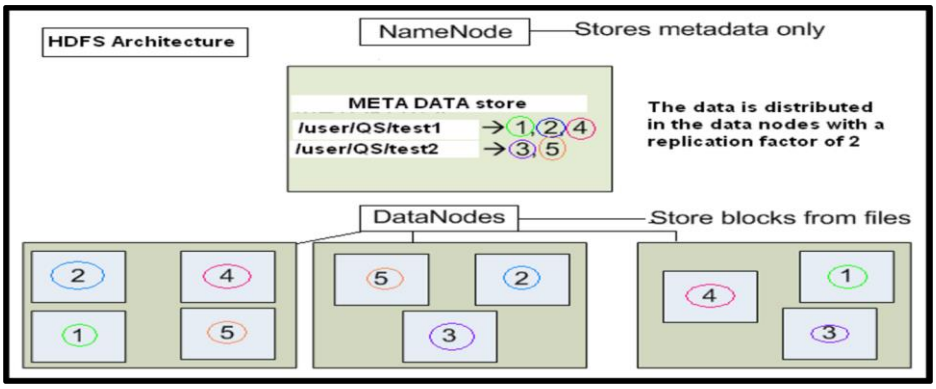


Figure II.1 : L'architecture d'HDFS [21]

II.2.1.1 Les composantes d'HDFS

HDFS définit deux types de nœuds :

Le nœud principal ou NameNode

Il se caractérise par :

- Responsable de la distribution et de la réplication des blocs ;
- Serveur d'informations du Hdfs pour le client Hdfs ;
- Stocke et gère les métadonnées ;
- Comporte la liste des blocs pour chaque fichier (dans le cas de lecture) ;
- Contient la liste des DataNodes pour chaque bloc (dans le cas de l'écriture) ;
- Tenir les attributs des fichiers (ex : nom, date de création, facteur de réplication) ;
- Logs toute métadonnée et toute transaction sur un support persistant ;
- Lectures/écritures ;
- Créations/suppressions ;
- Démarre à partir d'une image d'HDFS (fsimage).

Le nœud de données ou DataNode

Il se caractérise par :

- Stocke des blocs de données dans le système de fichier local ;
- Maintenir des métadonnées sur les blocs possédés (ex : CRC) ;
- Serveur de bloc de données et de métadonnées pour le client hdfs ;
- Heartbeat avec le Namenode : Heartbeat est système permettant sous Linux la mise en clusters de plusieurs serveurs pour effectuer entre eux un processus de tolérance de panne. Le processus Heartbeat se chargera de passer un message-aller vers le NameNode indiquant : son identité, sa capacité totale, son espace utilisé, son espace restant.

Secondary NameNode

Le Name Node dans l'architecture Hadoop est un point unique de défaillance. Si ce service est arrêté, il n'y a pas moyen de pouvoir extraire les blocs d'un fichier donné. Pour résoudre ce problème, un NameNode secondaire appelé Secondary NameNode a été mis en place dans l'architecture Hadoop. Son rôle consiste à :

- Télécharger régulièrement les logs sur le NameNode ;
- Crée une nouvelle image en fusionnant les logs avec l'image HDFS ;
- Renvoie la nouvelle image au NameNode. [9]

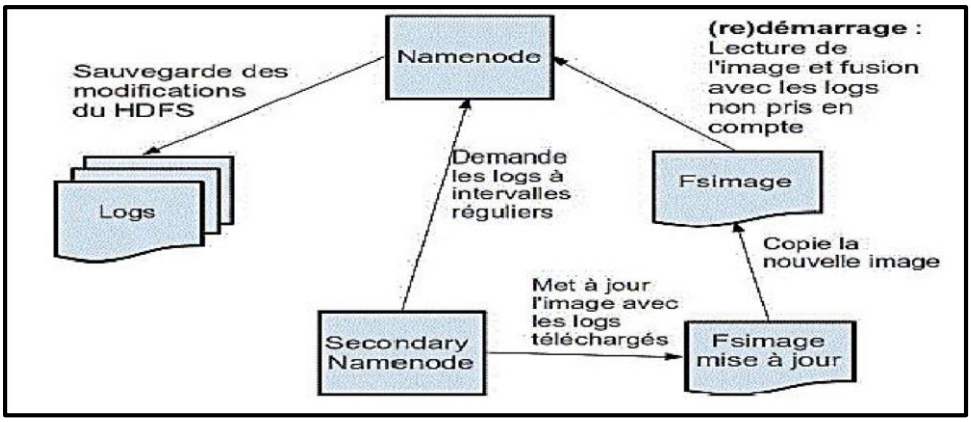


Figure II.2 : Fonctionnement du Secondary NameNode [9]

II.2.1.2 HDFS et tolérance aux fautes

Pour éviter un crash du DataNode la solution serait :

- Plus de Heartbeat (détection par le NameNode) ;

- Plus de réplication distribuée (robustesse des données).

Dans le cas d'un crash du NameNode (voir Figure II.2**)** :

- Sauvegarde des logs de transaction sur un support stable ;
- Redémarrage sur la dernière image du hdfs.

II.2.1.3 Lecture d'un fichier HDFS

Pour lire un fichier au sein de HDFS, il faut suivre les étapes suivantes :

Etape 1 : Le client indique au NameNode qu'il souhaite lire le fichier data.txt

Etape 2 : Le NameNode lui indiquera la taille de fichier (nombre de blocs) ainsi que les différents Data Node hébergeant les n blocs.

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

Etape 4 : En cas d'erreur/non réponse d'un des DataNode, il passe au suivant dans la liste fournie par le NameNode. **[9]**

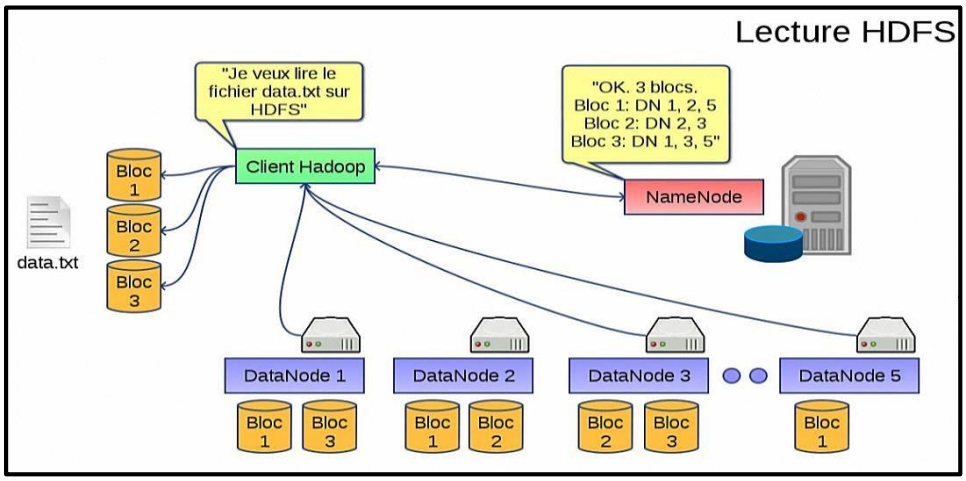


Figure II.3 : Lecture d'un fichier HDFS. [6]

II.2.1.4 Ecriture dans un fichier ou volume HDFS

Pour écrire un fichier au sein d'HDFS:

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

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

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

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

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

Etape 6 : Le cycle se répète pour le bloc suivant.

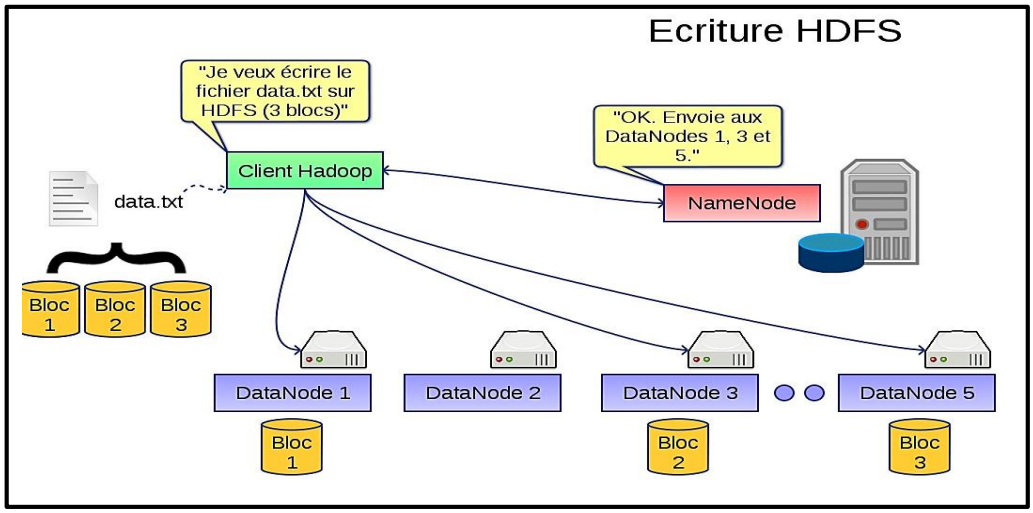


Figure II.4 : Processus d'écriture dans un volume ou fichier HDFS. [3]

II.2.1.5 HDFS : Stratégies de placement de bloc

Stratégie actuelle

- Une réplication sur un nœud aléatoire dans le rack ;
- Deuxième réplication sur un rack distant ;
- Troisième réplication sur le même rack distant ;
- Réplication additionnel placé aléatoirement ;

Le client lit le plus proche réplica. [8]

II.2.2 MapReduce

MapReduce est un paradigme (modèle) de programmation parallèle proposé par Google. Il est principalement utilisé pour le traitement distribué sur de gros volumes de données aux seins d'un cluster de nœuds. Il est conçu pour la scalabilité et la tolérance aux pannes.

Le modèle de programmation fournit un cadre à un développeur afin d'écrire une fonction Map et une fonction Reduce. Tout l'intérêt de ce modèle de programmation est de simplifier la

vie du développeur. Ainsi, ce développeur n'a pas à se soucier du travail de parallélisation et de distribution du travail. MapReduce permet au développeur de ne s'intéresser qu'à la partie algorithmique. [15]

Un programme MapReduce peut se résumer à deux fonctions Map () et Reduce ()

- La première, **MAP**, va transformer les données d'entrée en une série de couples clef /valeur. Elle va regrouper les données en les associant à des clefs, choisies de telle sorte que les couples clef/valeur aient un sens par rapport au problème à résoudre. Par ailleurs, cette opération doit être parallélisable: on doit pouvoir découper les données d'entrée en plusieurs fragments, et faire exécuter l'opération MAP à chaque machine du cluster sur un fragment distinct. La fonction Map s'écrit de la manière suivante :

Map (clé1, valeur1) → List (clé2, valeur2).

- La seconde, **REDUCE**, va appliquer un traitement à toutes les valeurs de chacune des clefs distinctes produite par l'opération MAP. Au terme de l'opération REDUCE, on aura un résultat pour chacune des clefs distinctes. Ici, on attribuera à chacune des machines du cluster une des clefs uniques produites par MAP, en lui donnant la liste des valeurs associées à la clef. Chacune des machines effectuera alors l'opération REDUCE pour cette clef. La fonction Reduce s'écrit de la manière suivante : Reduce (clé2, List (valeur2)) → List (valeur2).

II.2.2.1 MapReduce dans Hadoop

Il existe Deux versions notables du MapReduce :

- Version 0.x et 1.x : architecture purement maître-esclave ;
- Version 2.x YARN : architecture maître-esclave à deux niveaux

(Stable depuis oct. 2013). [9]

II.2.2.2 Architecture du Framework MapReduce (purement maître-esclave) Le MapReduce possède une architecture maître-esclave

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

1. Le JobTracker

- Gère l'ensemble des ressources du système ;
- Reçoit les jobs des clients ;
- Ordonnance les différentes tâches des jobs soumis ;

- Assigne les tâches aux TaskTrackers ;
- Réaffecte les tâches défaillantes ;
- Maintient des informations sur l'état d'avancement des jobs. [9]

2. Un TaskTracker

- Exécute les tâches données par le Jobtracker ;
- Exécution des tâches dans une autre JVM (Child) ;
- A une capacité en termes de nombres de tâches qu'il peut exécuter ;
- Heartbeat avec le JobTracker. [9]

II.2.2.3 Fonctionnement du Framework MapReduce

L'exécution d'un job MapReduce, concerne quatre entités indépendantes :

- Le client, qui soumet le job MapReduce;
- Le JobTracker, qui coordonne l'exécution et le partage du job en sous tâches. Le Jobtracker est une application Java dont la classe principale est JobTracker;
- Les TaskTrackers, qui gèrent les tâches que le JobTracker lui a confiées. Les Tasktrackers sont des applications Java dont la classe principale est TaskTracker;
- Le système de fichiers distribué, qui est utilisée pour le partage de fichiers

de travail entre les autres entités (tasktrackers). [4][5]

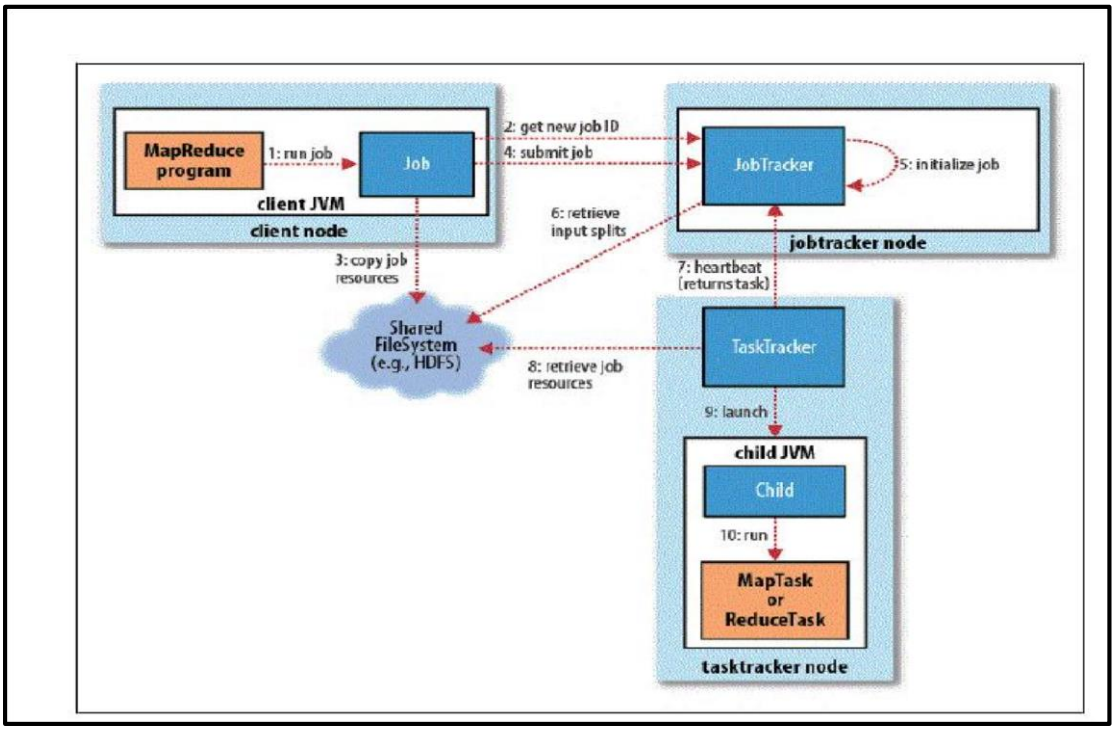


Figure II.6 : Schéma de fonctionnement de MapReduce [5]

L'exécution de MapReduce se fait selon la démarche suivante :

1- Lancement du job

Le processus de lancement du job est établi par JobSummitter qui effectue les opérations suivantes (**étape 1** de la figure):

- Demande aux JobTrackers l'ouverture d'un nouvel job ID(en appelant getNewJobId () sur JobTracker) (**étape 2** de la figureII.6) ;
- Vérifie si les fichiers en input et output existent bien, si ce n'est pas le cas il retourne une erreur ;
- Etablit la liste des sous tâches qui seront données aux tasktrackers, et renvoi une erreur en cas de problème (exemple : fichier de l'input introuvable) ;
- Copie les ressources nécessaires pour exécuter le travail, y compris le fichier JAR (programme MapReduce) du job, les paramètres de configuration Hadoop (hdfs.xml et core-site.xml) et les sous-tâches. La copie se fait dans le système de fichiers du JobTracker dans un répertoire nommé d'après l'ID du job. Le JAR du job est copié avec une réplication haute (contrôlé par le paramètre Mapred.submit.replication dans le fichier mapred-site.xml, qui est

par défaut (10), de sorte qu'il y'est beaucoup de copies à travers le cluster pour les tasktrackers (**étape 3** de la Figure II.6) ;

- Indique au JobTracker que le job est prêt pour l'exécution en appelant submitJob () sur JobTracker (**étape 4** de la figure II.6).

2- Initialisation du job

Une fois que le jobtracker exécute le submitJob () les actions suivantes sont déclenchées :

- Création d'un objet pour représenter le job ou la tâche à exécutée et lance un processus pour garder une trace de l'état d'avancement du job (**étape 5**) ;

- Récupération de la liste des sous tâches (input split compute) crées précédemment lors de l'étape3, récupération du nombre maximum de Reduce tasks crées est déterminé par la variable Mapred.Reduce.tasks dans {$HADOOP_HOME}/conf/mapred-site.xml(**étape 6**). **3- Affectation des tâches**

- Les TaskTrackers envoient régulièrement des Heartbeat au JobTracker pour lui signifier leurs disponibilités à exécuter des jobs, si le Jobtracker possède des jobs en fille d'attente il confiera la tâche au TaskTracker ;**(étape 7)**

- Après attribution du job au TaskTracker, il commence tout d'abord par localiser le fichier JAR (copié lors de **l'étape 3**) en le copiant depuis le système de fichiers partagé. Il copie également tous les fichiers nécessaires à partir du cache distribué par l'application sur le disque local. **(étape 8)**

4- Exécution du job

TaskRunner lance une nouvelle machine virtuelle Java (JVM, **étape 9**) pour exécuter chaque tâche (**étape 10**)

II.2.2.4 Hadoop MapReduce 2.x: YARN

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

- Les YARN node managers, lancent et contrôlent les conteneurs de calcul sur les machines de la grappe;

- Le MapReduce application master, coordonne les tâches en cours d'exécution de MapReduce. Ce dernier avec les tâches MapReduce sont exécutés dans des conteneurs qui sont programmés par le YARN ressources manager et gérés par le Node Managers;
- Le système de fichiers distribué HDFS, est utilisé pour le partage de fichiers de travail entre les autres entités. [4]

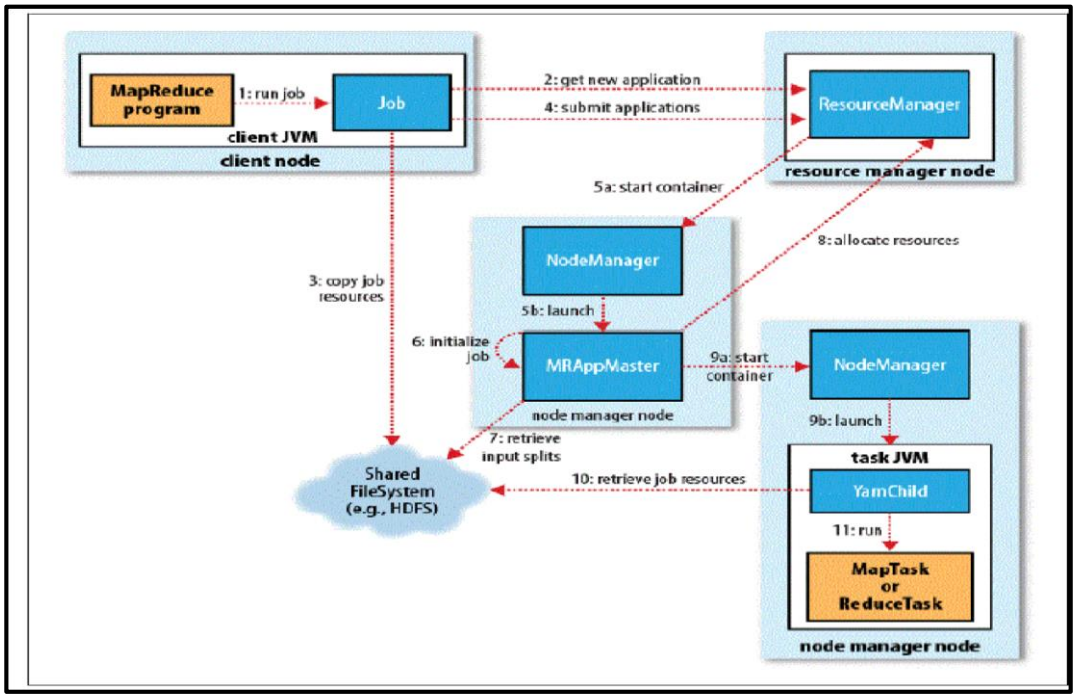


Figure II.7 : Schéma de fonctionnement de MapReduce 2. [5]

II.2.2.5 MapReduce et HDFS

Dans un écosystème d'Hadoop multi nœuds, on peut avoir sur une même machine physique l'exécution du JobTracker et du NameNode, aussi l'exécution sur une même machine physique d'un TaskTracker avec un DataNode (voir figure9).

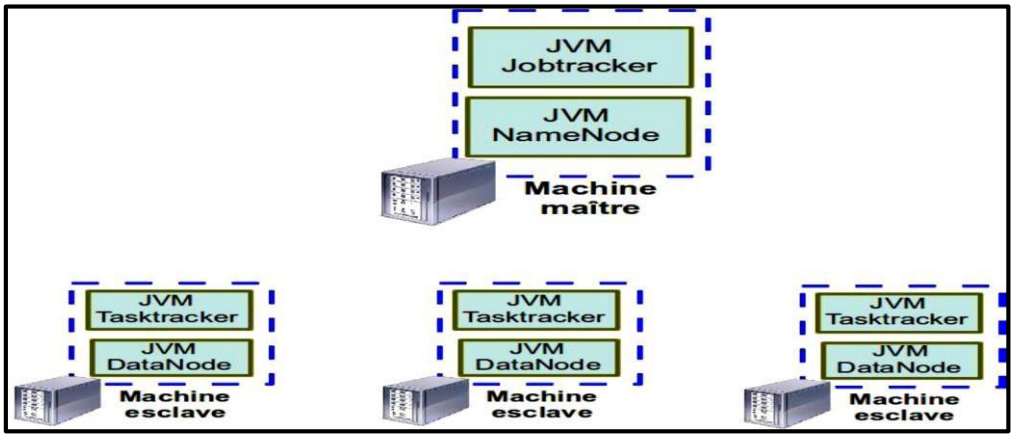


Figure II.8 : Architecture maître-esclave du MapReduce [9]

II.2.3 Écosystème d'Hadoop

Aujourd'hui il est difficile de se retrouver dans la " jungle " d'Hadoop pour les raisons suivantes :

- Ce sont des technologies jeunes ;
- Beaucoup de travaux d'acteurs différents (spécialistes du web, start-up,…) qui veulent prendre le train Big Data en marche ;

Dans une distribution Hadoop on va retrouver les éléments suivant sous leurs équivalences : HDFS, MapReduce, Zookeeper, Hbase, Hive, HCatalog, Oozie, Pig,

Sqoop,…etc.

On peut toutefois distinguer trois distributions majeures qui sont Cloudera, Horton Works et MapR, toutes les trois se basant sur Apache Hadoop :

- MapR : Noyau Hadoop mais repackagé et enrichi de solutions propriétaires ;
- Cloudera: Fidèle et open source en grande partie sauf pour les outils d'administration ;
- Horton Works: Fidèle à la distribution Apache et donc 100% open source. [20]

II.2.4 Outils composant le noyau Hadoop

- **HDFS (Hadoop Distributed File System)** : HDFS est un système de fichiers Java utilisé pour stocker des données structurées ou non sur un ensemble de serveurs distribués. HDFS s'appuie sur le système de fichier natif de l'OS pour présenter un système de stockage unifié reposant sur un ensemble de disques et de systèmes de fichiers hétérogènes. la consistance des données est basée sur la redondance. Une donnée est stockée sur au moins n volumes différents.
- **MapR** : En mai 2011, MapR a annoncé une alternative au système HDFS. Ce système permet d'éviter le SPOF (Single Point Of Failure) représenté par le

Name Node. Ce système n'est pas inconnu car il s'agit de HBase, dont elle propose une version propriétaire.

- **HBase (Apache)** : HBase est un sous-projet d'Hadoop, c'est un système de gestion de base de données non relationnel distribué, écrit en Java, disposant d'un stockage structuré pour les grandes tables. HBase est inspirée des publications de Google sur BigTable. Comme

BigTable, c'est une base de données orientée colonnes. HBase est souvent utilisé conjointement au système de fichiers HDFS, ce dernier facilitant la distribution des données de HBase sur plusieurs nœuds. Contrairement à HDFS, HBase permet de gérer les accès aléatoires read/write pour des applications de type temps réel.

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

- **MapReduce**: Initialement crée par Google pour son outil de recherche web.

C'est un Framework qui permet la décomposition d'une requête importante en un ensemble de requêtes plus petites qui vont produire chacune un sous ensemble du résultat final : c'est la fonction Map.

- **YARN (HortonWorks)** YARN (Yet-Another-Resource-Negotiator) est aussi appelé MapReduce 2.0, ce n'est pas une refonte mais une évolution du Framework. [5]

II.2.5 Les outils de Requêtage et de Scripting des données dans Hadoop

- **Hive (Facebook)** : Hive est à l'origine un projet Facebook qui permet de faire le lien entre le monde SQL et Hadoop. Il permet l'exécution de requêtes SQL sur un cluster Hadoop en vue d'analyser et d'agréger les données. Le langage SQL est nommé HiveQL. C'est un langage de visualisation uniquement, c'est pourquoi seules les instructions de type "Select" sont supportées pour la manipulation des données. Dans certains cas, les développeurs doivent faire le Mapping entre les structures de données et Hive.

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

Contrairement à Hive, Pig est basé sur un langage de haut niveau Pig Latin qui permet de créer des programmes de type MapReduce. Il ne dispose pas d'interface web, Pig se base sur HDFS et MapReduce pour exécuter le script en backround (en arrière-plan). [5]

II.2.6 L'outil d'intégration SGBD-R (Relationnel) Sqoop (Cloudera)

Sqoop permet le transfert des données entre un cluster Hadoop et des bases de données relationnelles. C'est un produit développé par Cloudera, Il permet d'importer/exporter des données depuis/vers Hadoop et Hive. Pour la manipulation des données Sqoop utilise MapReduce et des drivers JDBC.

II.2.7 Les outils de gestion et de supervision du cluster Hadoop [5]

- **Apache ZooKeeper** : ZooKeeper est un service de coordination des services d'un cluster Hadoop, en particulier, le rôle de ZooKeeper est de fournir aux composants Hadoop les fonctionnalités de distribution, pour cela il centralise les éléments de configuration du cluster Hadoop, propose des services de clustérisassions et gère la synchronisation des différents éléments (événements).

ZooKeeper est un élément indispensable au bon fonctionnement de Hbase.

- **Apache Ambari (HortonWorks)** : Ambari est un projet d'incubation Apache initié par HortonWorks et destiné à la supervision et à l'administration de clusters Hadoop. C'est un outil web qui propose un tableau de bord, cela permet de visualiser rapidement l'état d'un cluster. [5]

Ambari dispose d'un tableau de bord dont le rôle est de fournir une représentation :

- De l'état des services ;
- De la configuration du cluster et des services ;
- De l'exécution des jobs ;
- Des métriques de chaque machine et du cluster.

• II.2.8 Outil d'ordonnancement et de coordination : Apache Oozie (Yahoo)

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

Oozie s'intègre parfaitement avec l'écosystème Hadoop puisqu'il supporte les types de jobs suivant :

- MapReduce (Java et Streaming) ;
- Pig ;
- Hive ;

- Sqoop.

Ainsi que des jobs spécifiques du système telles que des programmes java et des scripts Shell.

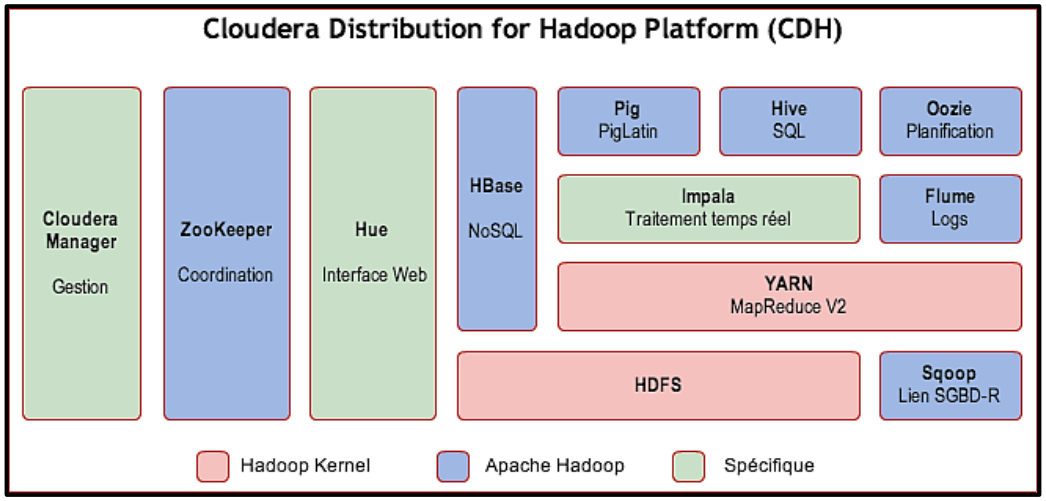


Figure II.9 : Ecosystème d'Hadoop [14]

II.3 Conclusion

Nous avons présenté dans ce chapitre les différents composants d'Hadoop, principalement, HDFS et le modèle de programmation MapReduce. On a étalé sur les aspects théoriques pour pouvoir réaliser et implémenter la solution complète dans le chapitre qui suit.

Chapitre III

Mise en œuvre, Test et Evaluation

III.1 Introduction

Dans ce chapitre, nous allons aborder la partie pratique de notre projet qui consiste à annoter et à traiter des données (images médicales) dans un cluster Hadoop; ainsi la conception et la mise en œuvre complète de notre environnement distribué.

Nous définirons les différentes étapes d'installation et de configuration de notre cluster Hadoop et les différents tests effectués sur des tâches MapReduce.

III.2 Motivation et problématique

Le but de notre travail est de réduire le temps de traitement des données par l'application de notre cluster Hadoop ; pour résoudre le problème de stockage des données massives et faire les traitements distribués à partir de notre système, donc on va répondre à la question comment installer un cluster Hadoop et profiter de Big Data pour atteindre un objectif de faire un stockage distribué et un traitement parallèle ?

III.3 Qu'allons-nous installer pour créer le cluster Hadoop Multi-Node ?

Java 8 ;

SSH ;

PDSH ;

- Utilisation de trois machines virtuelles (VM) avec un système d'exploitation Linux Ubuntu 20.04
- Installation d'Hadoop à partir de la distribution Apache qui est considérée, actuellement, comme la distribution la plus utilisée dans les entreprises, du fait qu'elle propose une version open source complète qui utilise les principaux composants d'Hadoop (HDFS, MapReduce, Hbase, Pig, Zookeeper).

III.4 L'environnement de travail

Dans la section suivante, on va décrire les outils et systèmes composant notre environnement de travail.

III.3.1 Virtual Box

VirtualBox est un logiciel open source pour virtualiser l'architecture informatique x86.

Il agit comme un hyperviseur, créant une VM (machine virtuelle) où l'utilisateur peut exécuter un autre système d'exploitation (Operating system).

III.3.2 Ubuntu 20.04

Ubuntu : est un système d'exploitation1) libre, gratuit, sécurisé et convivial, qui peut facilement remplacer ou cohabiter avec un système actuel (Windows, MacOs, autre GNU/Linux, ...). Avec Ubuntu, nous pouvons naviguer sur Internet, lire et écrire des courriels, créer des documents, des présentations et des feuilles de calculs, gérer votre bibliothèque multimédia et bien plus encore.

La distribution de base a eu trois périodes de diffusion : La période 1993-2004 : stabilisation du noyau Linux et apparition des variantes (Arch, Debian,...)

La période 2004-2017 : Debian sert de base à sa variante : Ubuntu.Unity est développée en vue de la convergence desktop-phone dans sa version 8.

La période 2017-actuelle : Unity est abandonnée au profit de Gnome.

Le terme « ubuntu » n'a pas été choisi au hasard : il véhicule des notions de communauté et de partage, qui sont les valeurs fondamentales du logiciel ouvert et libre.

Le mot « ubuntu » correspond à une idéologie provenant d'Afrique sub-saharienne qui s'articule autour des relations et des obligations des hommes les uns envers les autres.

Il recouvre un concept qui se développe depuis la Renaissance en Occident et qu'on appelle Humanisme au XVIème siècle avec l'apparition de l'imprimerie. Cette technologie sera la pierre angulaire de la diffusion des connaissances pendant cette période dénommée « le Siècle des Lumières ». La philosophie qui caractérise cette époque est traduite dans la maxime : le salut est dans le savoir et dans Autrui, il est donc indispensable d'acquérir la connaissance et de pratiquer l'ouverture d'esprit envers ses semblables. [22]

III.3.3 Choix d'Ubuntu

Le système d'exploitation open source Ubuntu offre plusieurs opportunités, parmi lesquelles, On trouve:

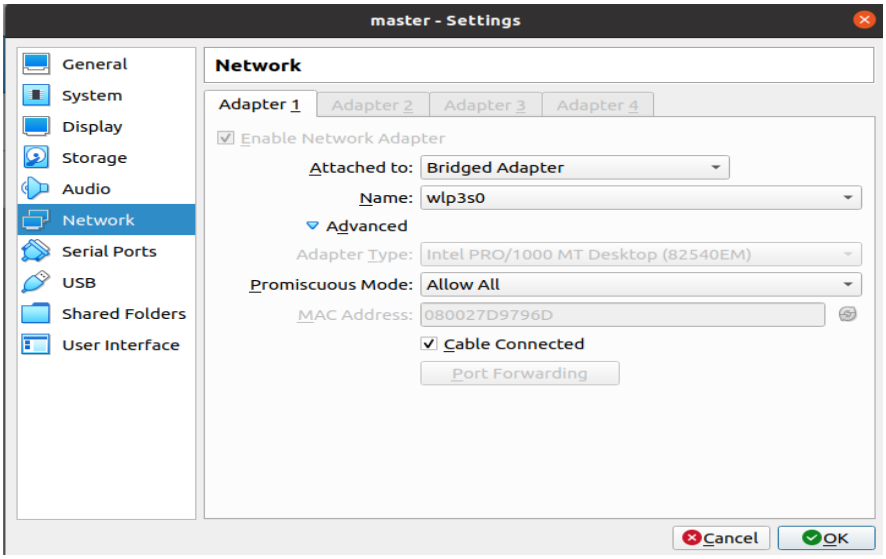
- Facile à installer : un environnement graphique clair vous guide durant cette phase
- délicate et tout est prêt en 40 mn environ, y compris l'installation des dernières mises
- à jours de sécurité et corrections de paquets logiciels.
- Moins gourmand et plus léger : après installation du système et des applications,
- Ubuntu n'occupe que 6 Go sur mon disque dur, contre le double ou le triple pour un

- système tel que Windows Seven. Ubuntu est également moins exigeant en termes de
- puissance et fonctionnera donc parfaitement sur un modèle d'ordinateur un peu ancien.
- est un système multi utilisateur.
- Pas de menaces et n'a pas besoin d'antivirus, il est sécurisé.
- est multitâche – gère plusieurs tâches en même temps comme Windows.
- Ubuntu vous propose de nombreux logiciels – Firefox, OpenOffice, logiciels de jeux,
- de formations, bureautiques, de sciences, d'éducations etc...
- Il facilite la communication en ligne – Yahoo, Google, Skype.
- Gérer parfaitement comme tout système d'exploitation les images et vidéos. [23]

III.5 Les étapes d'installation et de configuration de notre système

Etape01

La configuration de notre réseau : où in est dit "Mode promiscuité", sélectionnez l'option "Autoriser tout".



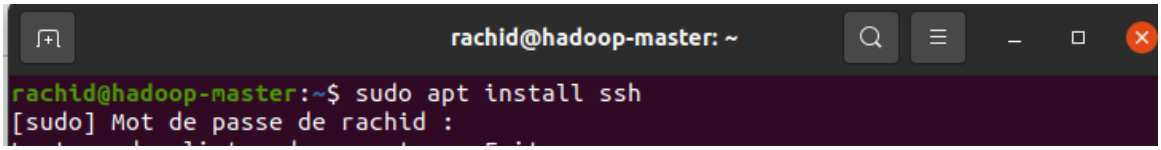
Etape 02.

Installez SSH à l'aide de la commande suivante :

sudo apt install ssh

Il vous demandera le mot de passe. Quand il demande une confirmation, donnez-le simplement.

Etape 03



Installer PDSH a l'aide de la commande :

sudo apt install pdsh

Tout comme avant, donnez une conformation en cas de besoin

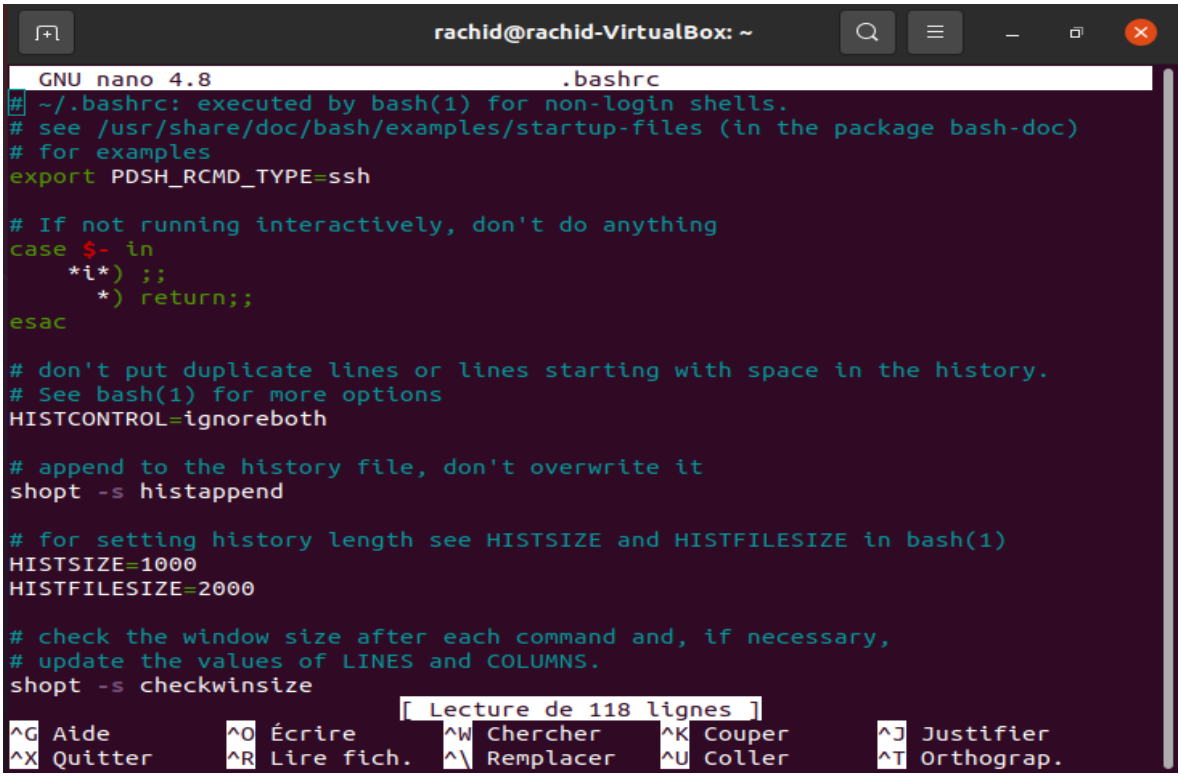
Etape 04

Ouvrez le fichier .bashrc avec la commande suivante :

nano .bashrc

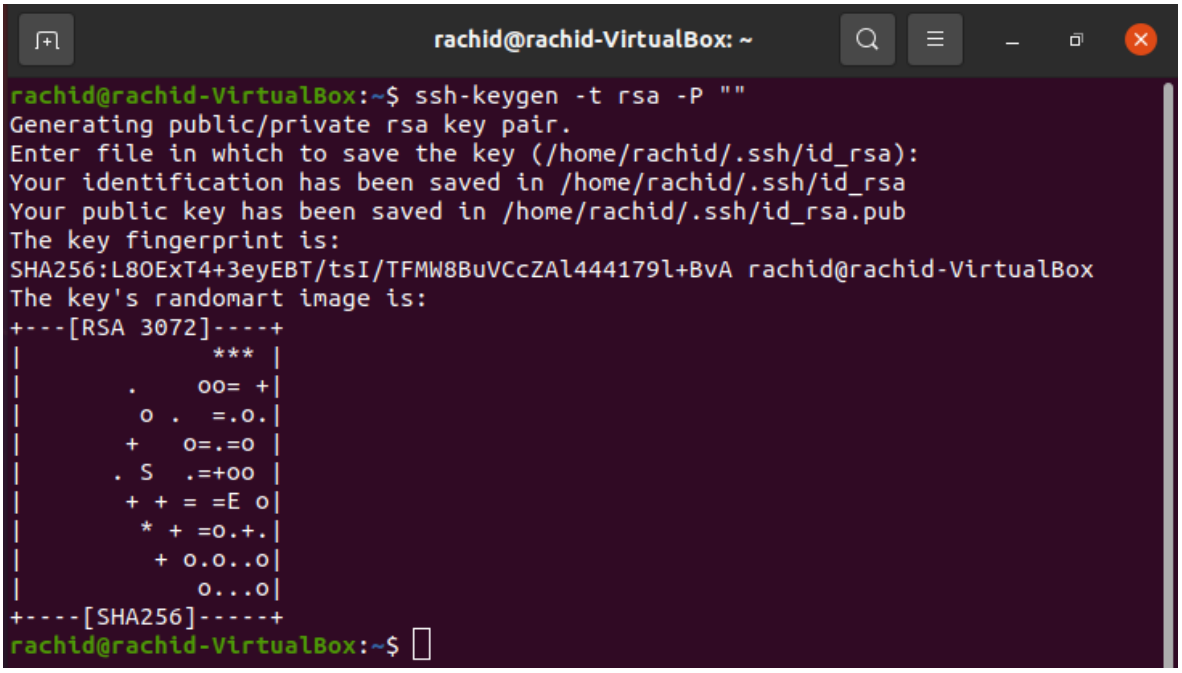
A la fin du fichier, il suffit d'écrire la ligne suivante :

export PDSH_RCMD_TYPE=ssh



Etape 05

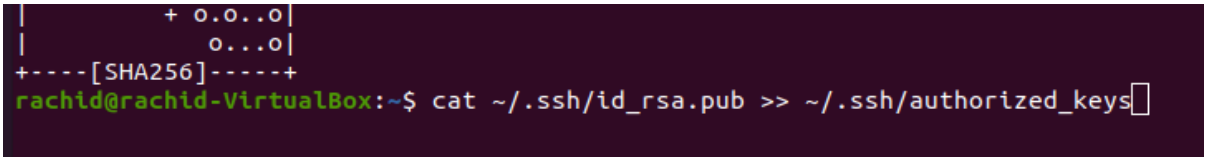
Configuration maintenant SSH, création d'une nouvelle clé à l'aide de la commande suivante : Appuyez simplement sur Entrée à chaque fois que cela est nécessaire.



Etape 06

Maintenant, nous devons copier la clé publique dans le fichier allowed_keys avec la commande suivante :

cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys

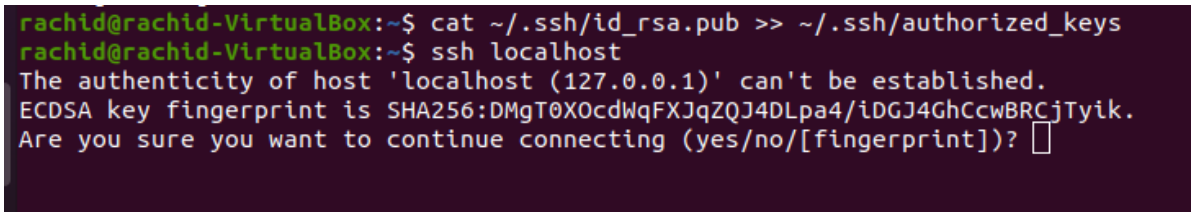


Etape 07

Nous pouvons maintenant vérifier la configuration SSH en nous connectant a l'hôte local :

ssh localhost

Tapez simplement « yes » et appuyez sur Entrée si nécessaire.

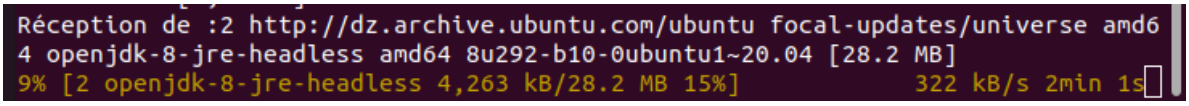


Etape 08

C’est l’étape où nous installons Java 8. Nous utilisons cette commande

sudo apt install openjdk-8-jdk

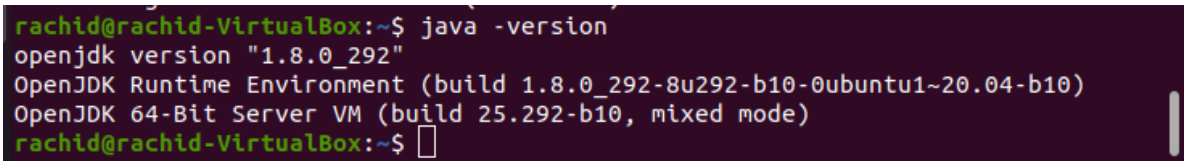
Tout comme précédemment, donnez une confirmation en cas de besoin



Etape 09

Cette étape n’est pas vraiment une étape, c’est juste pour vérifier si Java est maintenant correctement installé :

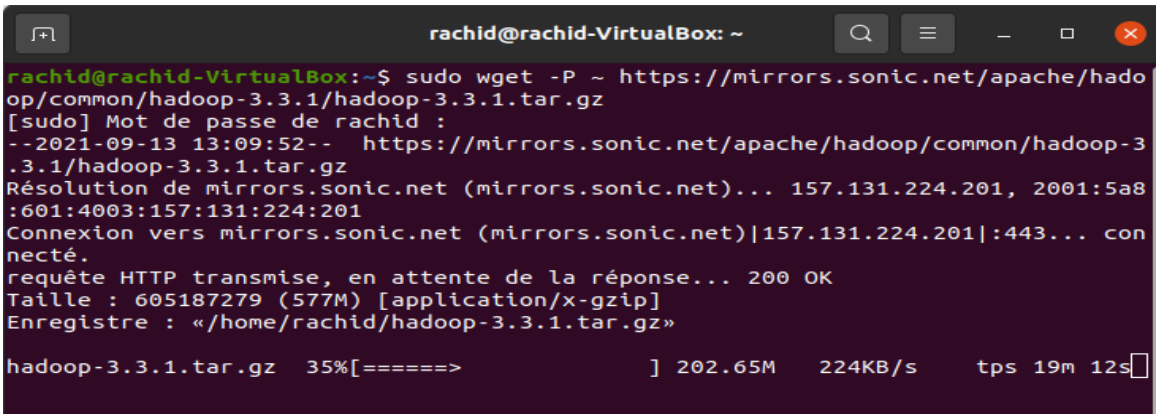
java -version



Etape 10

Télécharger Hadoop a l'aide de la commande suivante :

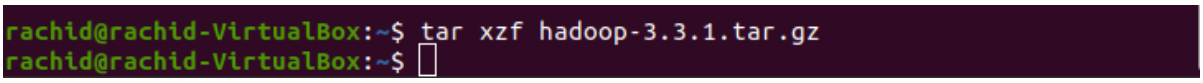
sudo wget -P ~https://mirrors.sonic.net/apache/hadoop/common/hadoop-3.3.1/hadoop-3.3



Etape 11

Nous devons décompresser le fichier hadoop-3.3.1.tar.gz avec la commande suivante :

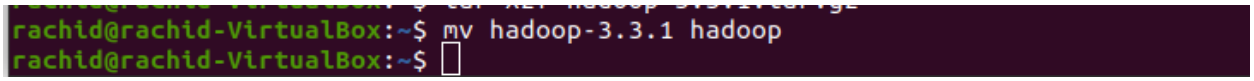
tar xzf hadoop-3.3.1.tar.gz



Etape 12

Changez le nom du dossier hadoop-3.3.1 en hadoop (cela a rendu son utilisation plus facile Utilisez cette commande :

mv hadoop-3.3.1 hadoop



Etape 13

Ouvrez le fichier hadoop-env.sh dans l'éditeur nano pour éditer JAVA_HOME :

nano ~/hadoop/etc/hadoop/hadoop-env.sh

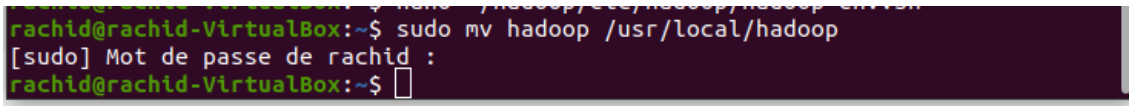
Collez cette ligne dans JAVA_HOME

export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/

Etape 14

Remplacez le répertoire du dossier hadoop par /usr/local/hadoop. C'est la commande :

```
sudo mv hadoop /usr/local/hadoop
```



Fournissez le mot de passe si nécessaire.

Etape 15

Ouvrez le fichier d'environnement sur nano avec cette commande :

```
sudo nano /etc/environment
```

Ensuite les configurations suivantes :

```
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/usr/local/hadoop/bin:/usr/local/hadoop/sbin"JAVA_HOME="/usr/lib/jvm/java-8-openjdk-amd64/jre"
```

Etape 16

Maintenant, nous allons ajouter un utilisateur appelé hadoopuser, etnous allons mettre ses configurations :

```
sudo adduser hadoopuser
```

Fournissez le mot de passe et vous pouvez laisser le reste vide, appuyez simplement sur Entrée.

```
rachid@rachid-VirtualBox:~$ sudo adduser hadoopuser
Ajout de l'utilisateur « hadoopuser » ...
Ajout du nouveau groupe « hadoopuser » (1001) ...
Ajout du nouvel utilisateur « hadoopuser » (1001) avec le groupe « hadoopuser »
...
Création du répertoire personnel « /home/hadoopuser »...
Copie des fichiers depuis « /etc/skel »...
Nouveau mot de passe :
Retapez le nouveau mot de passe :
passwd : le mot de passe a été mis à jour avec succès
Modification des informations relatives à l'utilisateur hadoopuser
Entrez la nouvelle valeur ou « Entrée » pour conserver la valeur proposée
        Nom complet []:
        N° de bureau []:
        Téléphone professionnel []:
        Téléphone personnel []:
        Autre []:
Ces informations sont-elles correctes ? [O/n] O
rachid@rachid-VirtualBox:~$
```

Tapez maintenant ces commandes :

sudo usermod -aG hadoopuser hadoopuser

sudo chown hadoopuser:root -R /usr/local/hadoop/

sudo chmod g+rwx -R /usr/local/hadoop/

sudo adduser hadoopuser sudo

```
rachid@rachid-VirtualBox:~$ sudo usermod -aG hadoopuser hadoopuser
rachid@rachid-VirtualBox:~$ sudo chown hadoopuser:root -R /usr/local/hadoop/
rachid@rachid-VirtualBox:~$ sudo chmod g+rwx -R /usr/local/hadoop/
rachid@rachid-VirtualBox:~$ sudo adduser hadoopuser sudo
Ajout de l'utilisateur « hadoopuser » au groupe « sudo »...
Ajout de l'utilisateur hadoopuser au groupe sudo
Fait.
```

Etape 17

Nous devons maintenant vérifier l'adresse IP de la machine :

ip addr

```
rachid@hadoop-master: ~
rachid@hadoop-master:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
Rhythmbox <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP gr
oup default qlen 1000
    link/ether 08:00:27:d9:79:6d brd ff:ff:ff:ff:ff:ff
    inet6 fe80::22fd:ccf:86e6:e2f1/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
rachid@hadoop-master:~$
```

Maintenant, comme vous pouvez le voir, l'adresse IP est 192.168.43.238 ; n'oubliez pas que ce sera différent pour vous, vous devez agir en conséquence lorsque les adresses IP seront utilisées plus tard.

Notre réseau sera le suivant :

Master : 192.168.43.238

Slave1 : 192.168.43.239

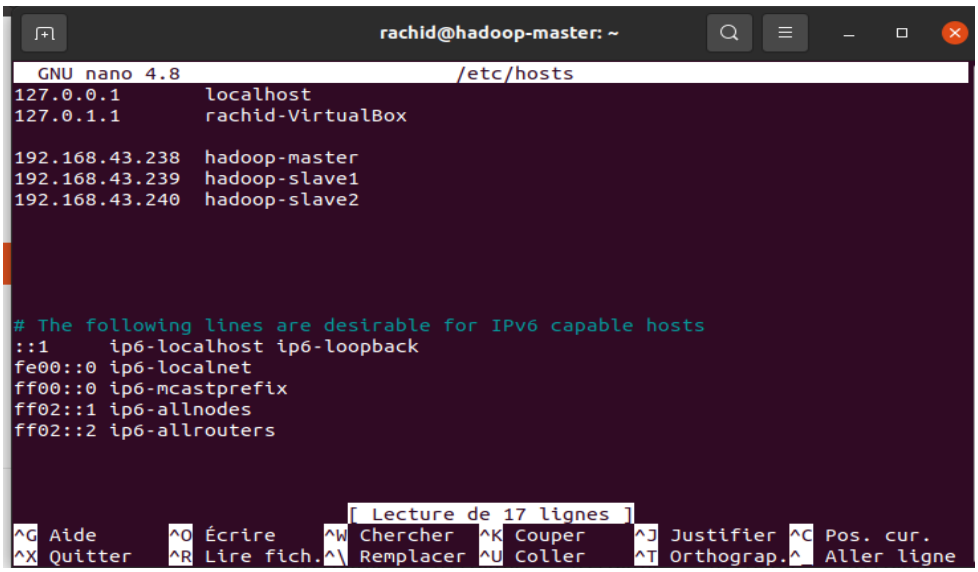
Slave2 : 192.168.43.240

Dans votre cas, continuez d'ajouter 1 au dernier numéro de l'adresse IP que vous obtenez sur votre machine, comme nous avons fait.

Etape 18

Ouvrez le fichier hosts et insérez vos configurations réseau :

sudo nano /etc/hosts



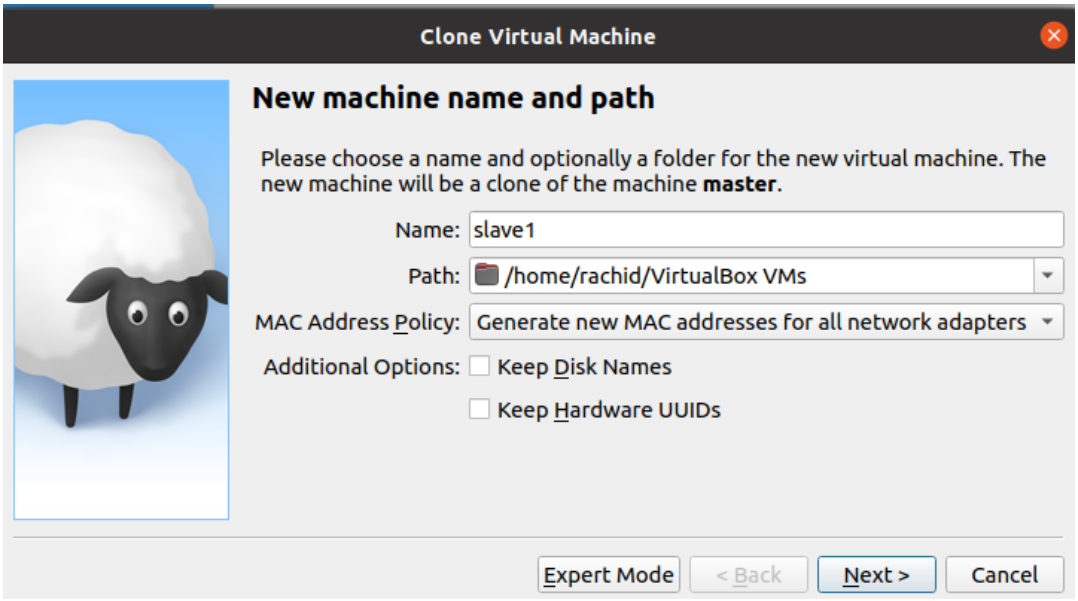
Etape 19

Il est maintenant temps de créer les masters.

Arrêtez votre machine virtuelle maître et clonez-la deux fois,en nommant un slave1 et l'autre slave2.

Assurez-vous que l'option « Générer de nouvelles adresses MAC pour toutes les cartes réseaux » est choisie.

Faites également un clone complet.

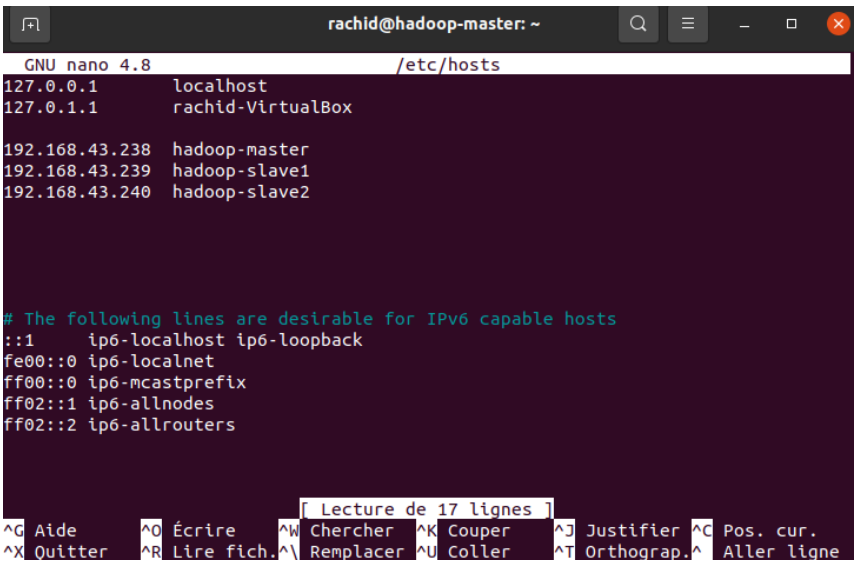


Etape 20

Sur la VM (machine virtuelle) principale, ouvrez le fichier de nom d'hôte sur nano :

sudo nano /etc/hostname

Insérez le nom de votre master machine virtuelle principale. (attention c'est le même nom que vous avez entrée précédemment dans le fichier hosts



Faires la même chose sur les slaves.

De plus, vous devez tous les redémarrer pour que cette configuration prenne effet :

Sudo reboot

Etape21

Configurez le SSH sur hadoop-master, avec hadoopuser. C'est la commande :

```
sudo - hadoopuser
```

```
rachid@hadoop-master:~$ su - hadoopuser
Mot de passe :
To run a command as administrator (user "root"), use "sudo <c
ommand>".
See "man sudo_root" for details.
```

Etape 22

Créer une clé SSH par la commande :

ssh-keygen -t rsa.

```
hadoopuser@hadoop-master:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hadoopuser/.ssh/id_rsa):
Created directory '/home/hadoopuser/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/hadoopuser/.ssh/id_rsa
Your public key has been saved in /home/hadoopuser/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:DZ3I1fdFXKNnjfBsxdvVPWPaBftOGYnw6hTVx5OHL1o hadoopuser@hadoop-
master
The key's randomart image is:
+---[RSA 3072]----+
|           .o..oOO|
|        . + .=+=@/|
|         + o. =XB&|
|          o  ooE+*|
|         S .o o oo|
|           o .  o |
|            .    .|
|                  |
|                  |
+----[SHA256]-----+
```

Etape 23

Maintenant, nous devons copier la clé SSH pour tous les utilisateurs .Utilisez cette commande :

```
ssh-copy-id hadoopuser@hadoop-master

ssh-copy-id hadoopuser@hadoop-slave1

ssh-copy-id hadoopuser@hadoop-slave2
```

```
hadoopuser@hadoop-master:~$ ssh-copy-id hadoopuser@hadoop-master
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/
hadoopuser/.ssh/id_rsa.pub"
The authenticity of host 'hadoop-master (192.168.43.238)' can't be es
tablished.
ECDSA key fingerprint is SHA256:DMgT0XOcdWqFXJqZQJ4DLpa4/iDGJ4GhCcwBR
CjTyik.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s),
 to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you
 are prompted now it is to install the new keys
hadoopuser@hadoop-master's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'hadoopuser@hadoop-mas
ter'"
and check to make sure that only the key(s) you wanted were added.
```

```
hadoopuser@hadoop-master:~$ ssh-copy-id hadoopuser@hadoop-slave1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/
hadoopuser/.ssh/id_rsa.pub"
The authenticity of host 'hadoop-slave1 (192.168.43.239)' can't be es
tablished.
ECDSA key fingerprint is SHA256:DMgT0XOcdWqFXJqZQJ4DLpa4/iDGJ4GhCcwBR
CjTyik.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s),
 to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you
 are prompted now it is to install the new keys
hadoopuser@hadoop-slave1's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'hadoopuser@hadoop-sla
ve1'"
and check to make sure that only the key(s) you wanted were added.
```

```
hadoopuser@hadoop-master:~$ ssh-copy-id hadoopuser@hadoop-slave2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/
hadoopuser/.ssh/id_rsa.pub"
The authenticity of host 'hadoop-slave2 (192.168.43.240)' can't be es
tablished.
ECDSA key fingerprint is SHA256:DMgT0XOcdWqFXJqZQJ4DLpa4/iDGJ4GhCcwBR
CjTyik.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s),
 to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you
 are prompted now it is to install the new keys
hadoopuser@hadoop-slave2's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'hadoopuser@hadoop-sla
ve2'"
and check to make sure that only the key(s) you wanted were added.
```

Etape 24

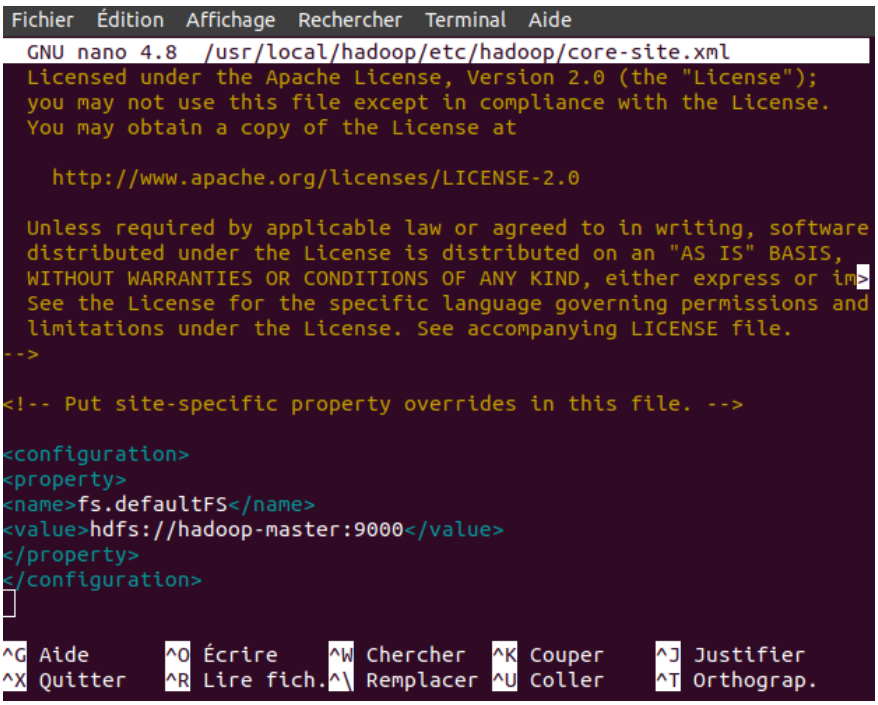
Sur hadoop-master, ouvrez le fichier core-site.xml sur nano :

```
sudo nano /usr/local/hadoop/etc/hadoop/core-site.xml
```

```
hadoopuser@hadoop-master:~$ sudo nano /usr/local/hadoop/etc/hadoop/core-site.xm
l
[sudo] password for hadoopuser:
```

Ajoutez ensuite les configurations suivantes:

```
<configuration>

<property>

<name>fs.defaultFS</name>

<value>hdfs://hadoop-master:9000</value>

</property>

</configuration>.
```



Etape 25

Toujours sur hadoop-master, ouvrez le fichier hdfs-site.xml :

sudo nano /usr/local/hadoop/etc/hadoop/hdfs-site.xml

```
hadoopuser@hadoop-master:~$ sudo nano /usr/local/hadoop/etc/hadoop/hdfs-site.xm
l
```

Ajoutez les configurations suivantes:

<configuration>

<property>

<name>dfs.namenode.name.dir</name><value>/usr/local/hadoop/data/nameNode</value>

</property>

<property>

<name>dfs.datanode.data.dir</name><value>/usr/local/hadoop/data/dataNode</value>

</property>

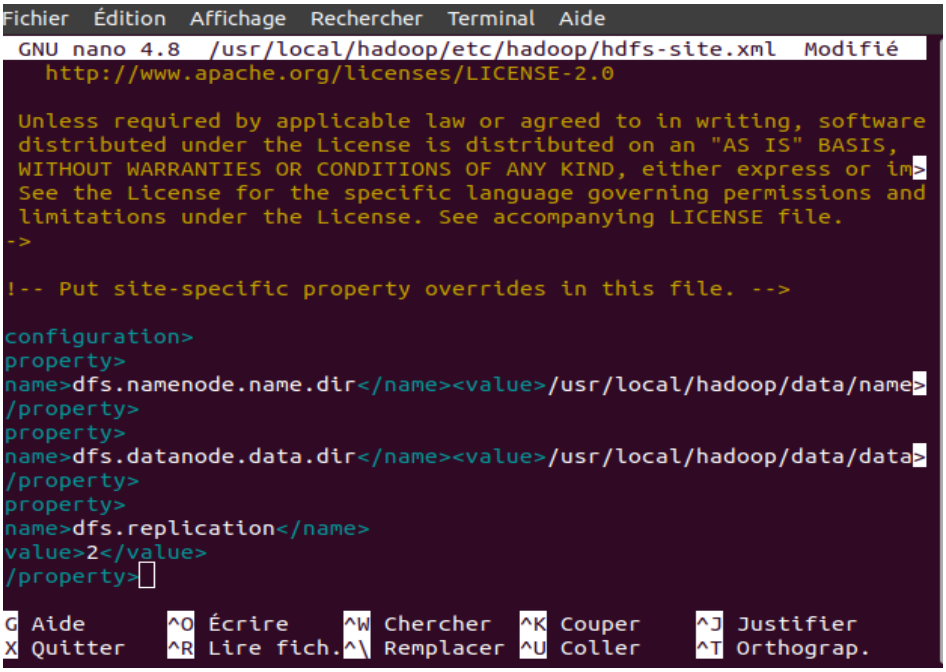
<property>

<name>dfs.replication</name>

<value>2</value>

</property>

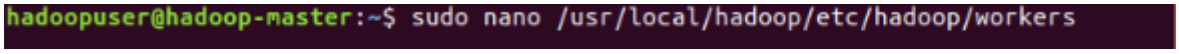
/configuration>.



Etape 26

On est toujours sur hadoop-master, ouvrez le fichier **workers** :

sudo nano /usr/local/hadoop/etc/hadoop/workers



Ajoutez ces deux lignes: (les noms des slaves, vous vous souvenez du fichier hosts ?)

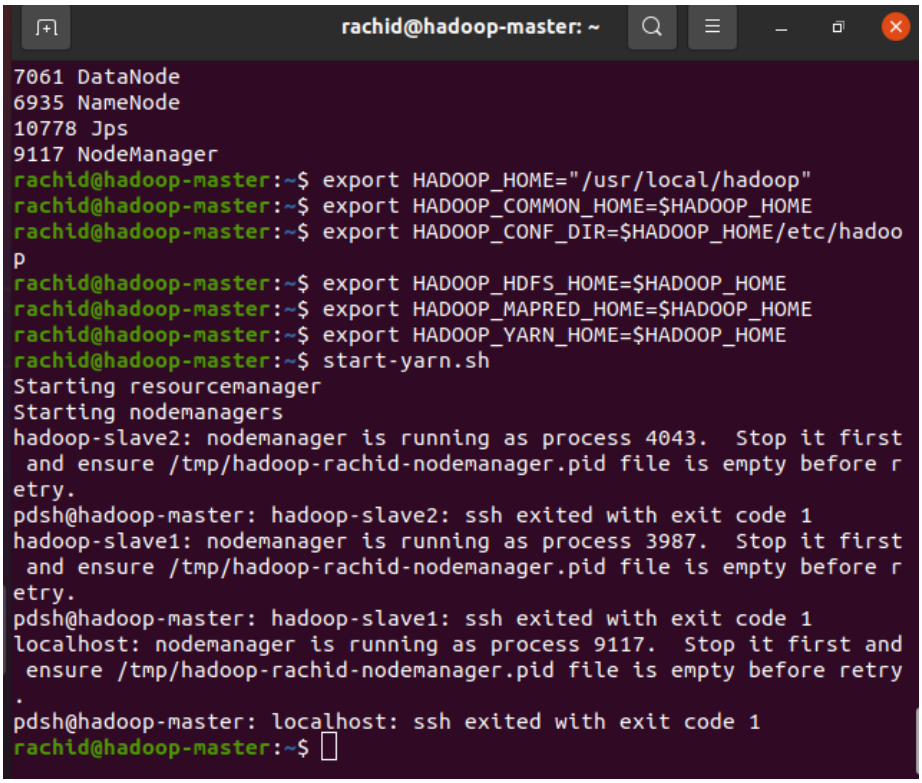
hadoop-slave1

hadoop-slave2

Etape 27

Vous devez copier les configurations hadoop-master sur les slaves pour cela vous utilisez ces commandes :

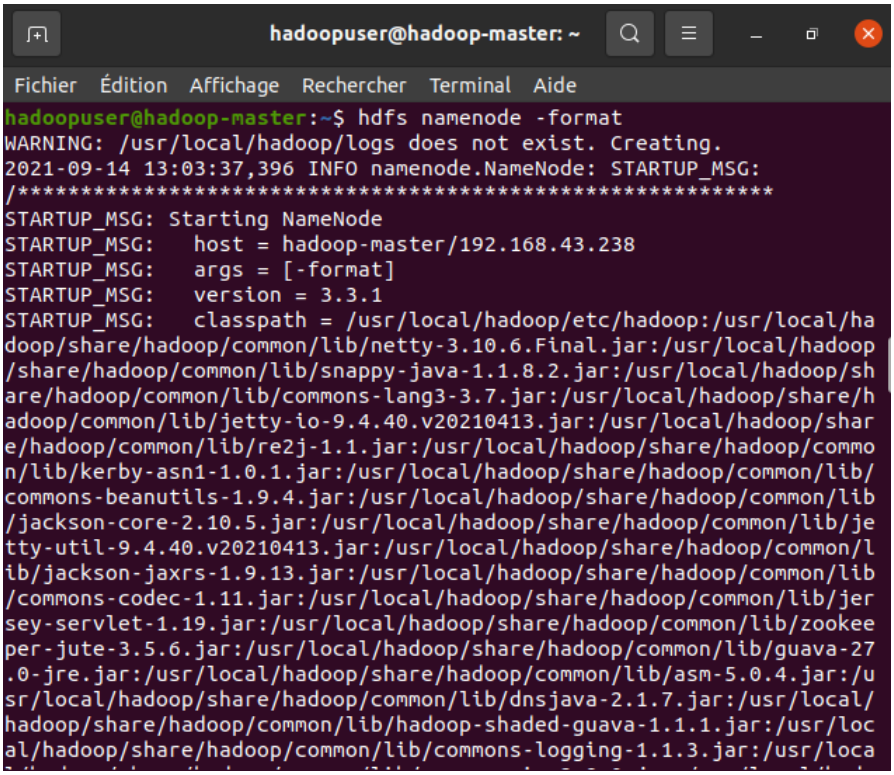
```
scp /usr/local/hadoop/etc/hadoop/* hadoop-slave1:/usr/local/hadoop/etc/hadoop/

scp /usr/local/hadoop/etc/hadoop/* hadoop-slave2:/usr/local/hadoop/etc/hadoop/
```



Etape 28

Vous devez maintenant formatter le système de fichiers HDFS, Executez ces commandes :

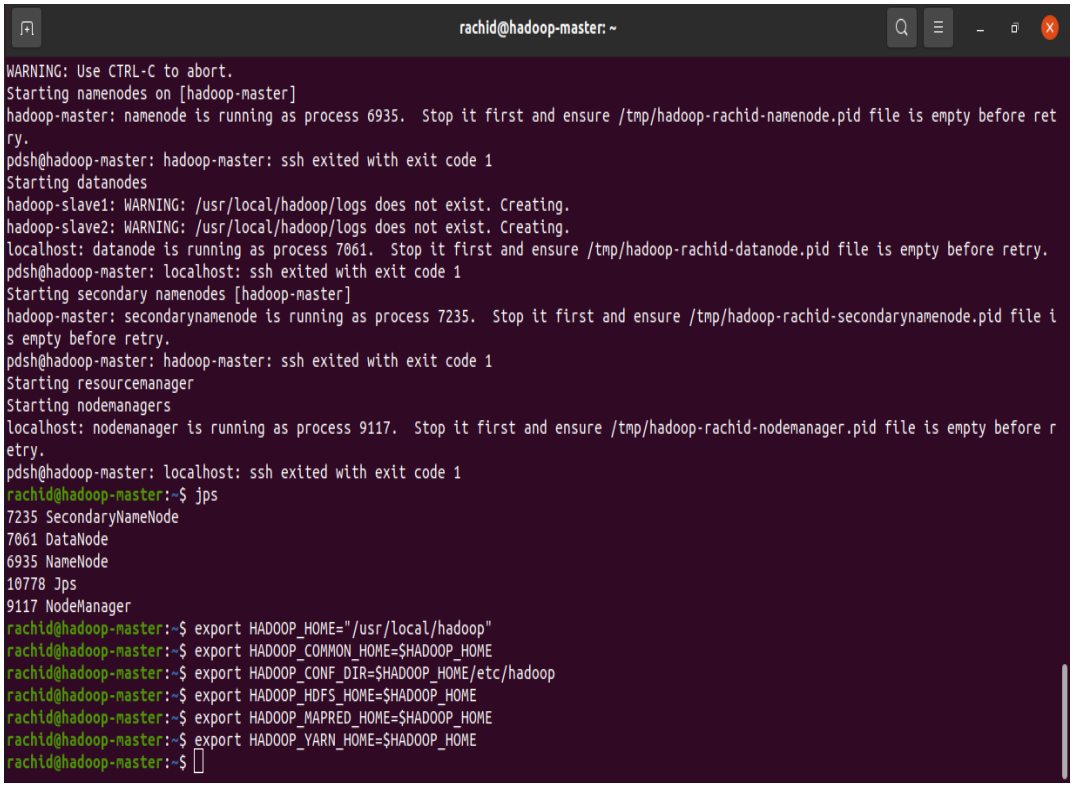
```
source /etc/environment

hdfs namenode -format
```



Etape 29

Démarrez HDFS avec cette commande:

```
start-dfs.sh
```



Pour vérifier si cela a fonctionné, exécutez la commande suivante☹cela vous indiquera quelles ressources ont été initialisées:

jps

Maintenant, vous devez faire la même chose dans les slaves :

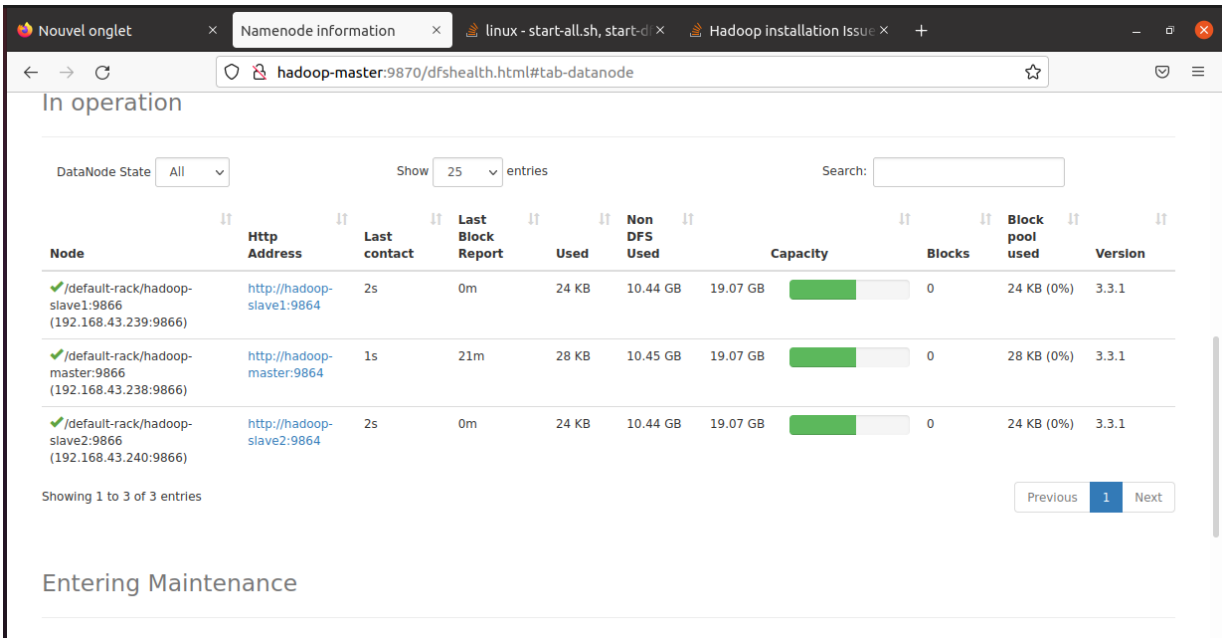
```
rachid@hadoop-slave1:~$ jps
3987 NodeManager
3845 DataNode
4071 Jps
rachid@hadoop-slave1:~$
```

```
rachid@hadoop-slave2:~$ jps
4043 NodeManager
3901 DataNode
4127 Jps
rachid@hadoop-slave2:~$
```

Etape 30

Voyez si cela a fonctionné :

Ouvrez votre navigateur et tapez : hadoop-master :9870. C'est ce que vous verrez.

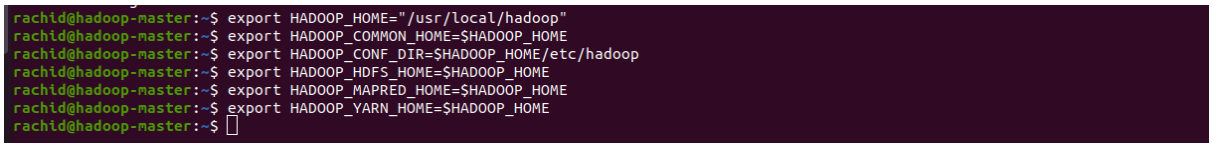


Comme vous pouvez le voir, les deux nœuds sont opérationnels !

Etape 31

Configuration **Yarn,** exécutez simplement les commandes suivantes :

```
export HADOOP_HOME="/usr/local/hadoop"

export HADOOP_COMMON_HOME=$HADOOP_HOME

export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop

export HADOOP_HDFS_HOME=$HADOOP_HOME

export HADOOP_MAPRED_HOME=$HADOOP_HOME

export HADOOP_YARN_HOME=$HADOOP_HOME
```



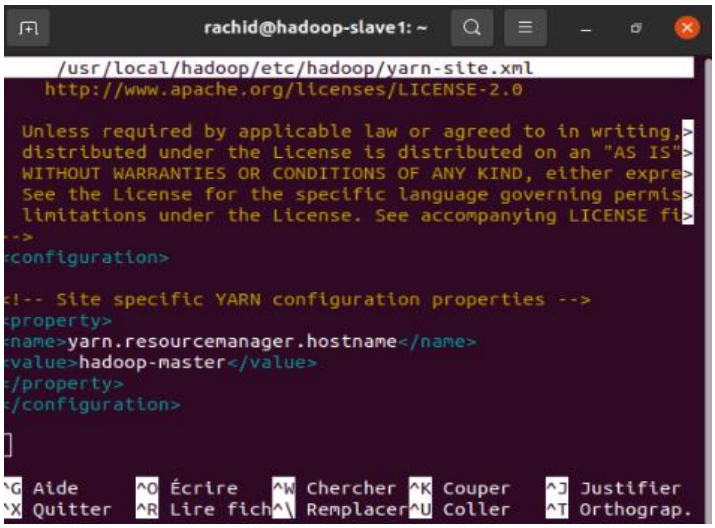
Etape 32

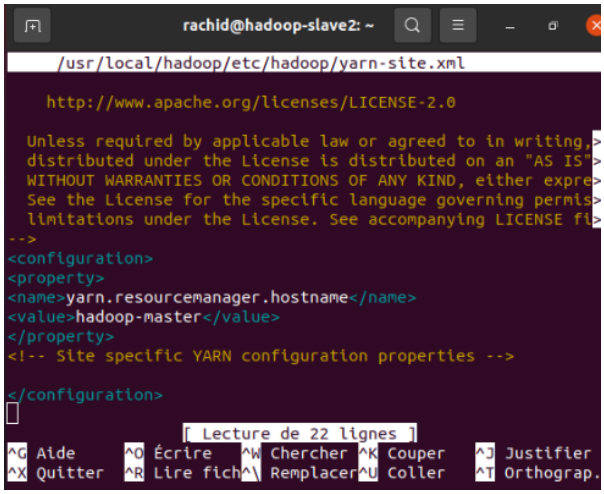
Dans les deux slaves, ouvrez **fil-site.xml** sur nano :

```
sudo nano /usr/local/hadoop/etc/hadoop/yarn-site.xml
```

Vous devez ajouter les configurations suivantes sur les deux slaves:

```
<property>

<name>yarn.resourcemanager.hostname</name>

<value>hadoop-master</value>

</property>
```

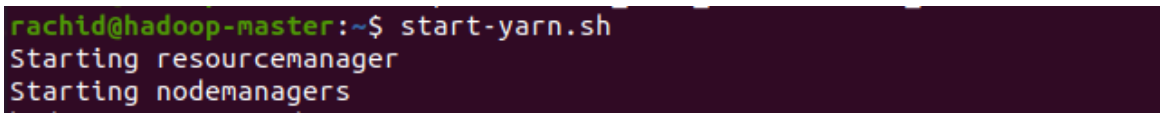




Etape 33

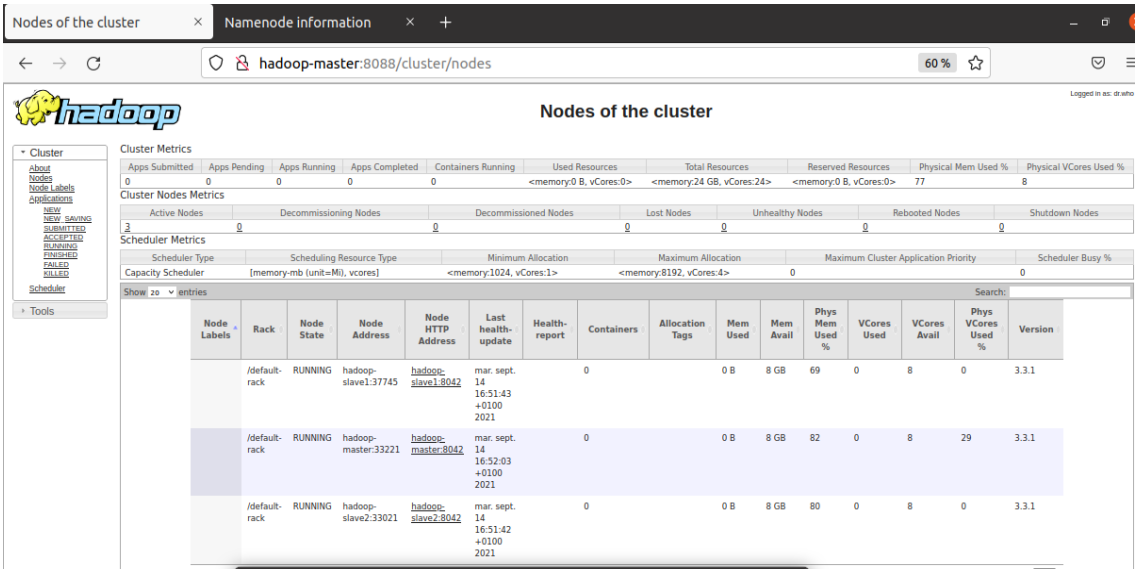
Sur le master, commencez-vous le fil. Utilisez cette commande :

start-yarn.sh



Etape 34

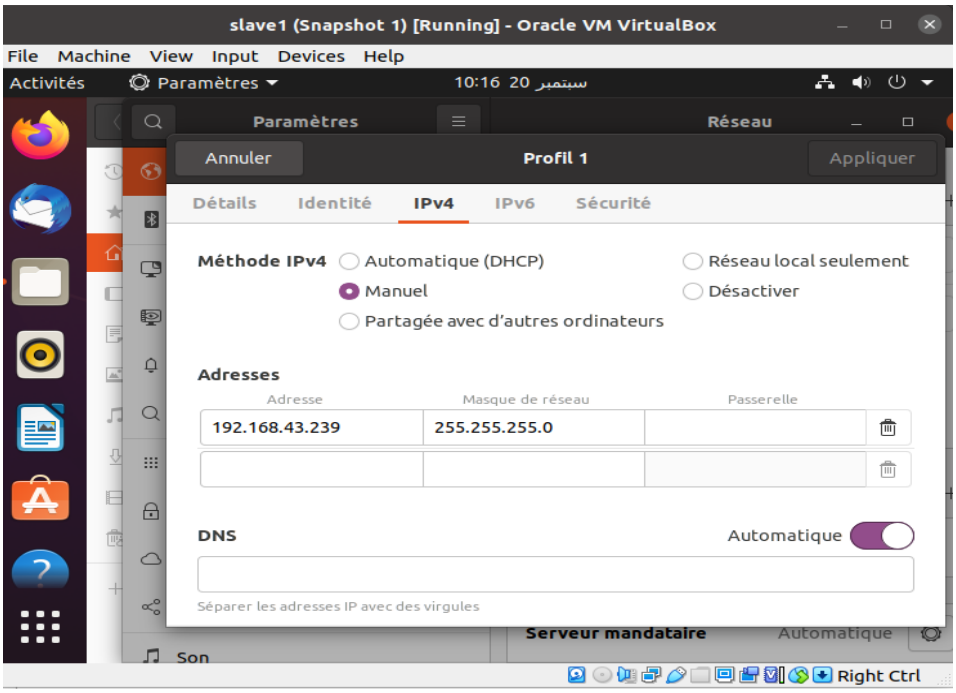
Ouvrez votre navigateur. Vous allez maintenant taper : http://hadoop-master:8088/cluster.

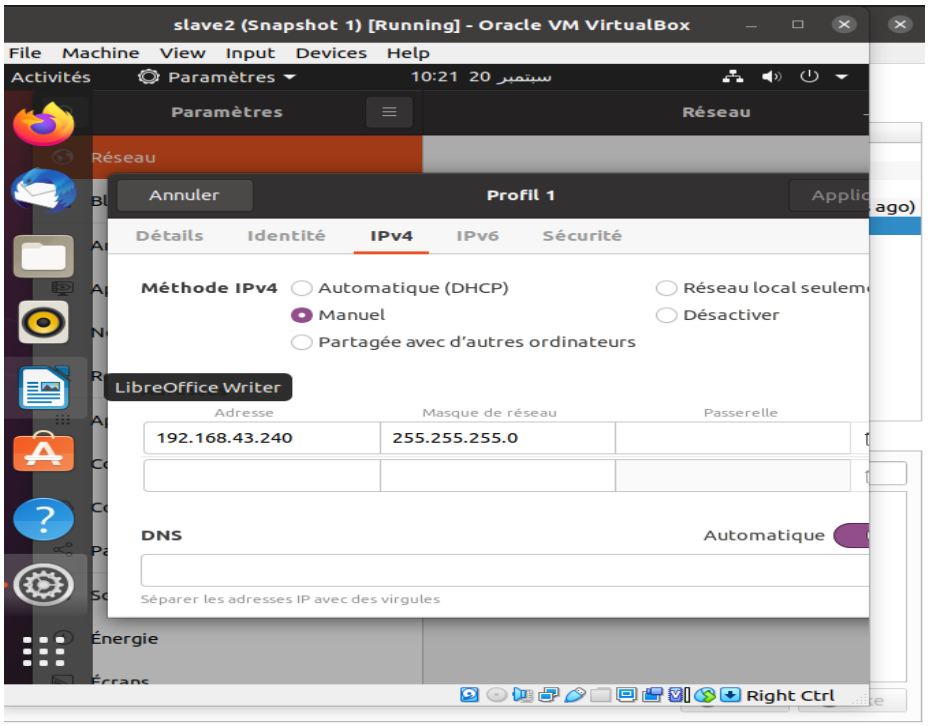


Comme vous pouvez le voir, le cluster affiche 3 nœuds actifs !

Les problèmes rencontrés :

Dans l'étape 23, si vous ne pouvez pas copier la clé SSH pour tous les utilisateurs, vous avez comme solution de vérifier les adresses IP sur les slaves.





Dans l'étape 29 un problème de fonctionnement de démarrage de HDFS, vous pouvez le résolu en définissant les autorisations de tous les fichiers sur 777 :

Utilisez cette commande :

sudo chmod 777 /usr/local/hadoop-3.3.1/* -R.

III.7 Description de la solution proposée par un exemple

Pour tester notre système Hadoop, on a proposé comme exemple d'exécuter un job en s'appuyant sur le modèle MapReduce écrit en java ; il va traiter des données textuelles.

Le stockage des données est effectué dans des fichiers distribués d'Hadoop HDFS.

III.7.1 wordcount.jar

. Présentation du job

Wordcount est un programme simple qui compte le nombre de chaque mot dans un ensemble d'entrée donné.

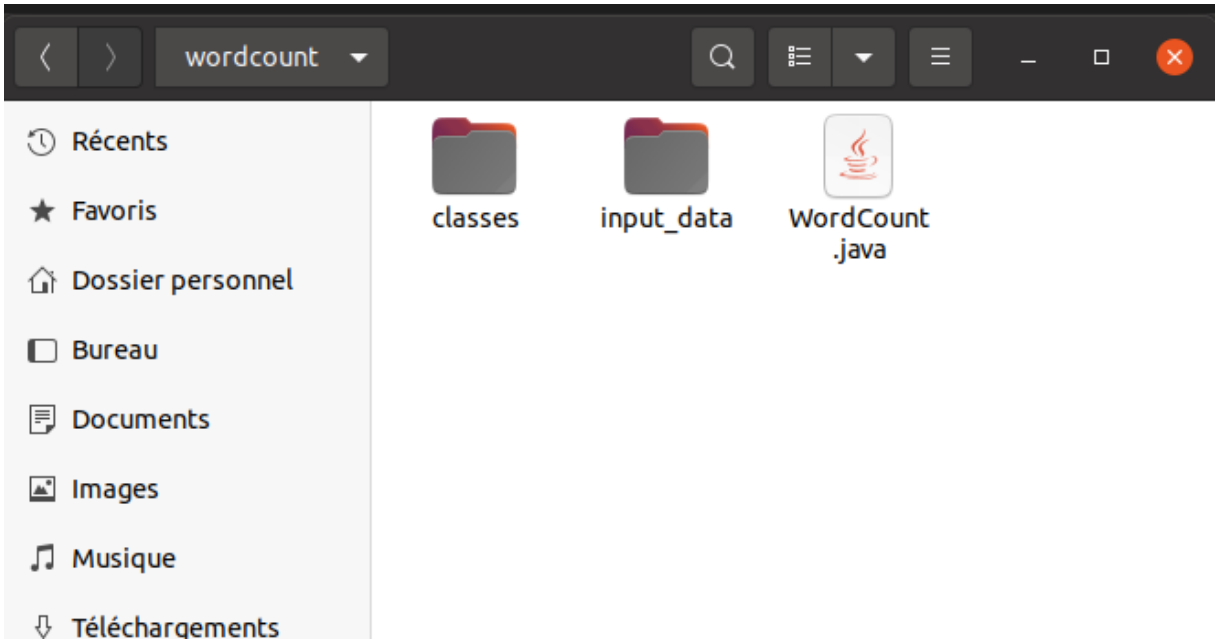
. Données

Pour pouvoir tester le job Wordcount, on a généré un fichier qui contient un texte avec un script Shell dont la fonction est prendre en paramètre une source texte (paramètre1) et d'ajouter son contenu n fois (paramètre3) dans un fichier de sortie (paramètre2).

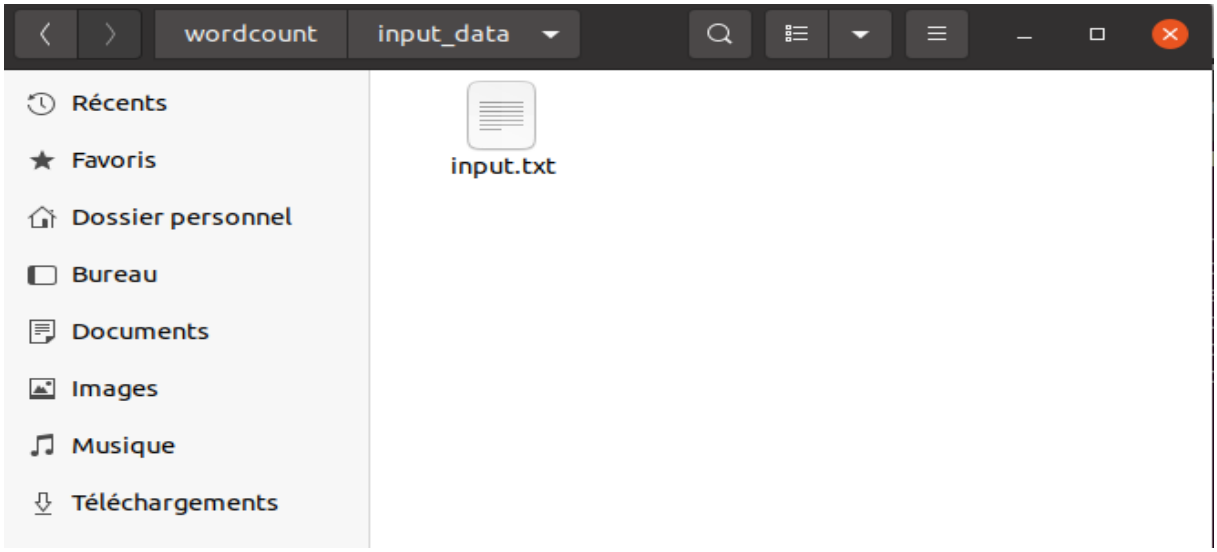
Le code source du script sc.sh est en Annexe C.

III.7.2 Les étapes d'exécution du job wordcount

Créez un nouveau dossier pour les données d'entrée (input data)



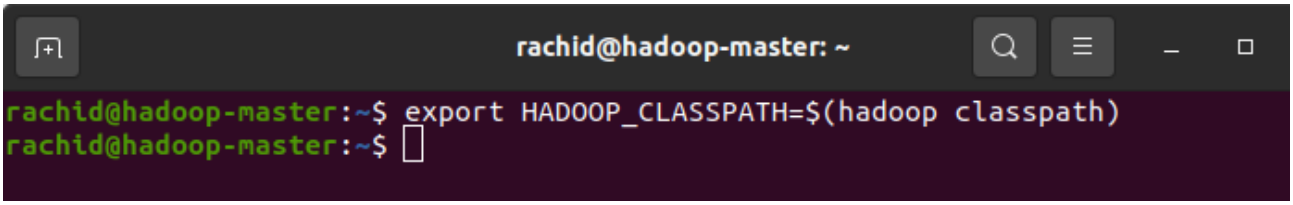
Ajoutez votre propre texte dans ce dossier (input.text).



Maintenant, utilisez cette commande suivante :

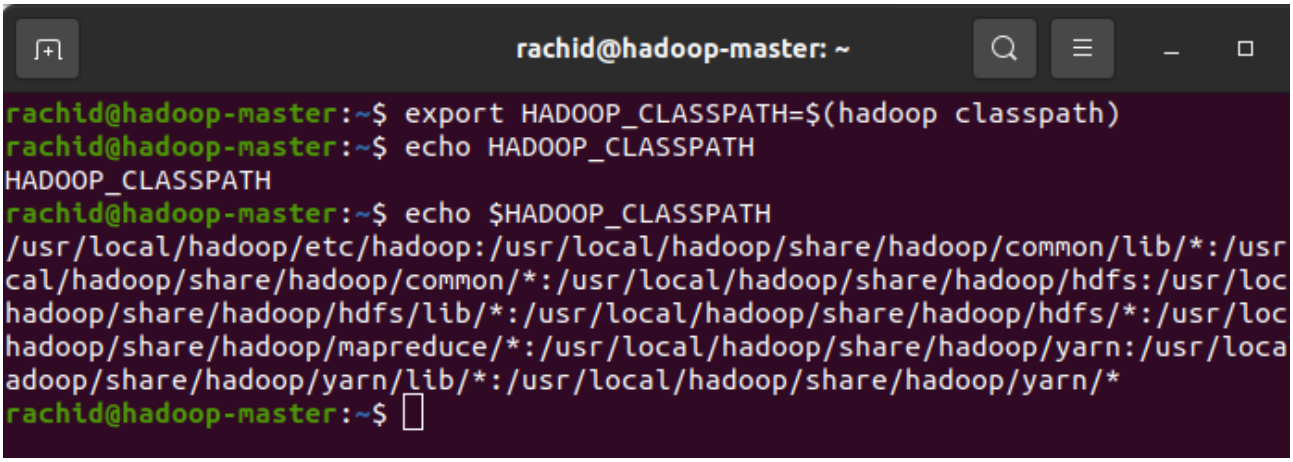
HADOOP_CLASSPATH : pour un variable d'environnement, et la commande :

Export HADOOP_CLASSPATH=$ (hadoop_classpath): pour préciser le chemin du class java.



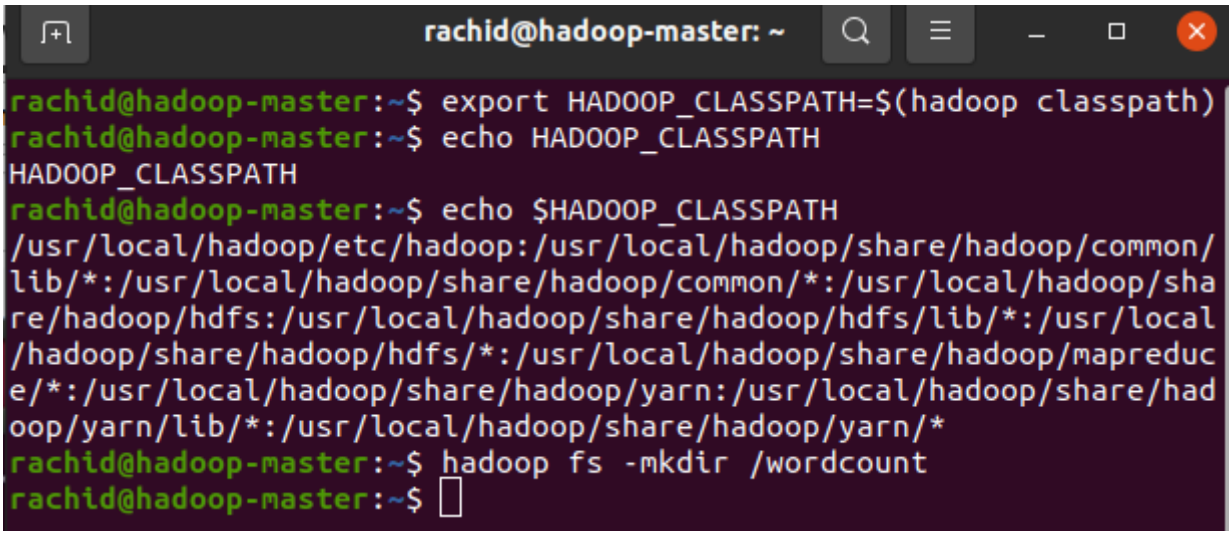
Suivant utilisez cette commande:

echo $HADOOP_CLASSPATH pour assurer qu'il a été réglé correctement



Créer un répertoire sur HDFS par la commande :

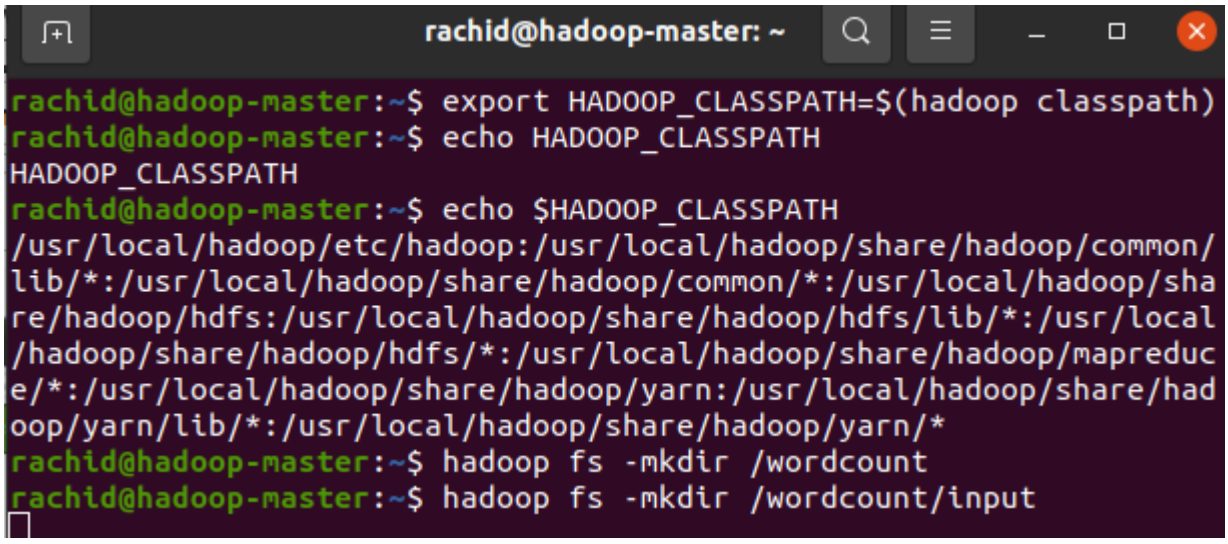
Hadoop fs mkdir/wordcount comme nom de directory.

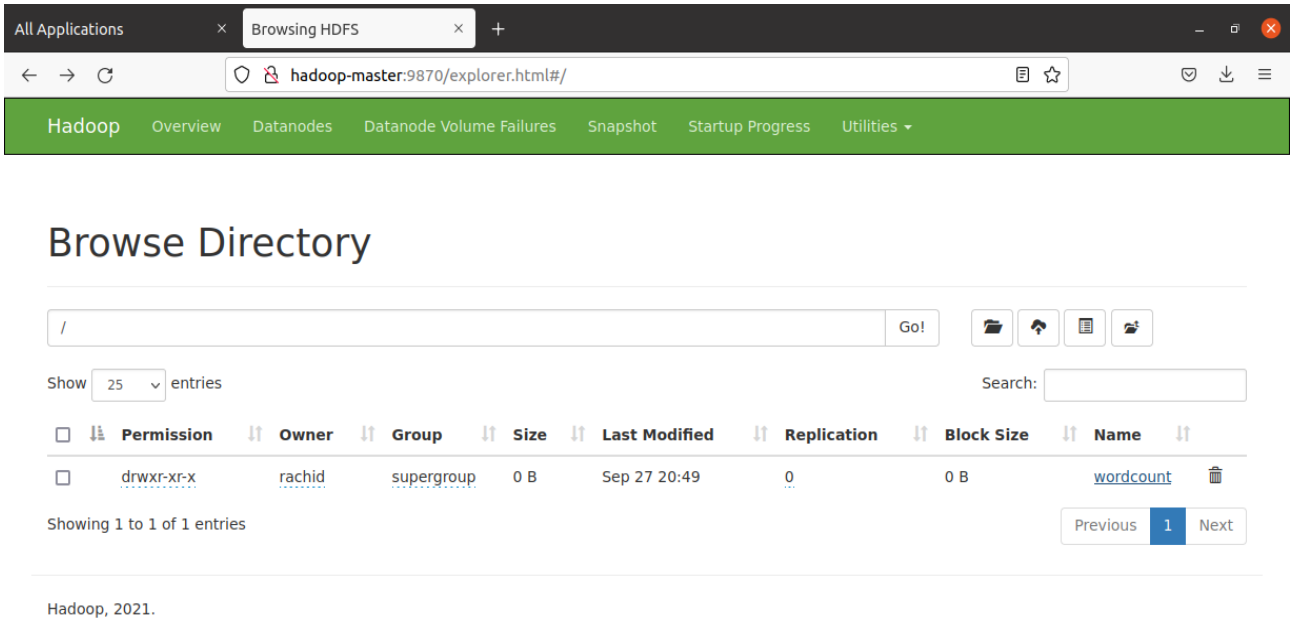


Maintenant, vous devez créer un répertoire à l'intérieur pour l'entrée

Utilisez cette commande :

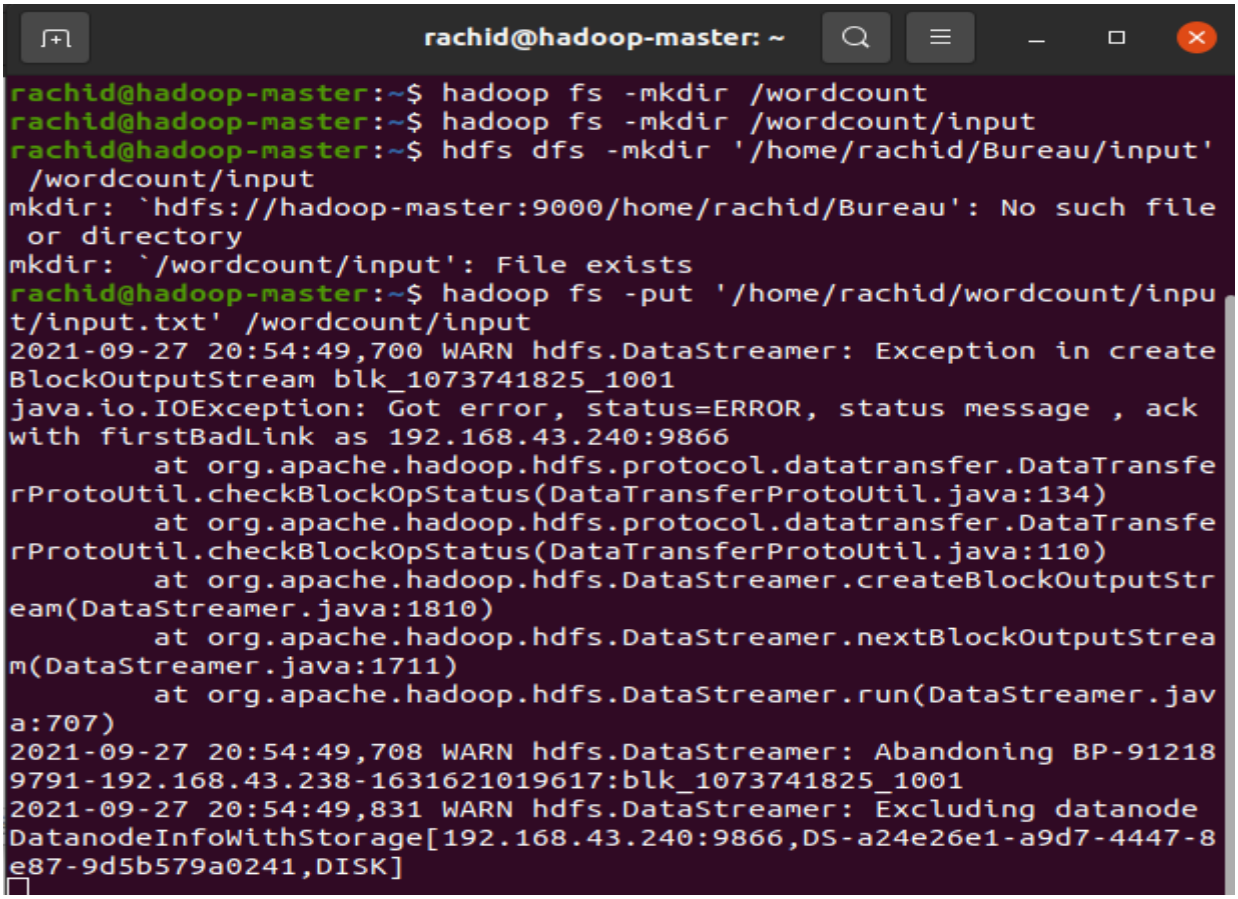
hadoop fs -mkdir /wordcount/input



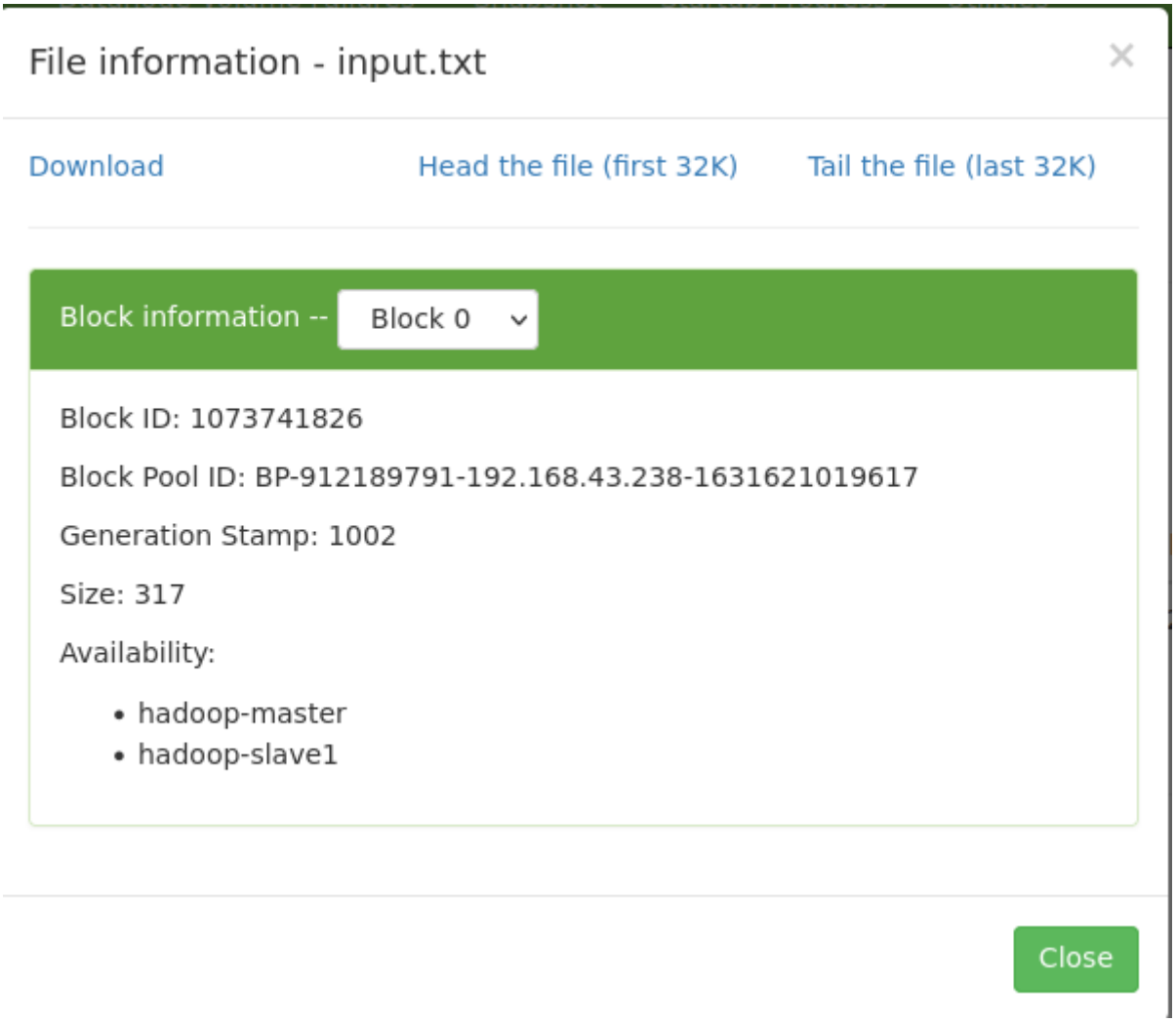


Téléchargez le fichier d'entrée dans ce répertoire par la commande suivante :

hdfs dfs -mkdir '/home/rachid/Bureau/input' /wordcount/input

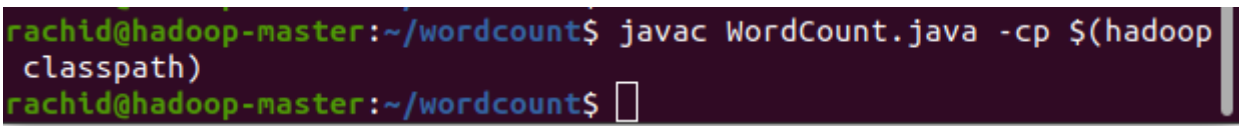


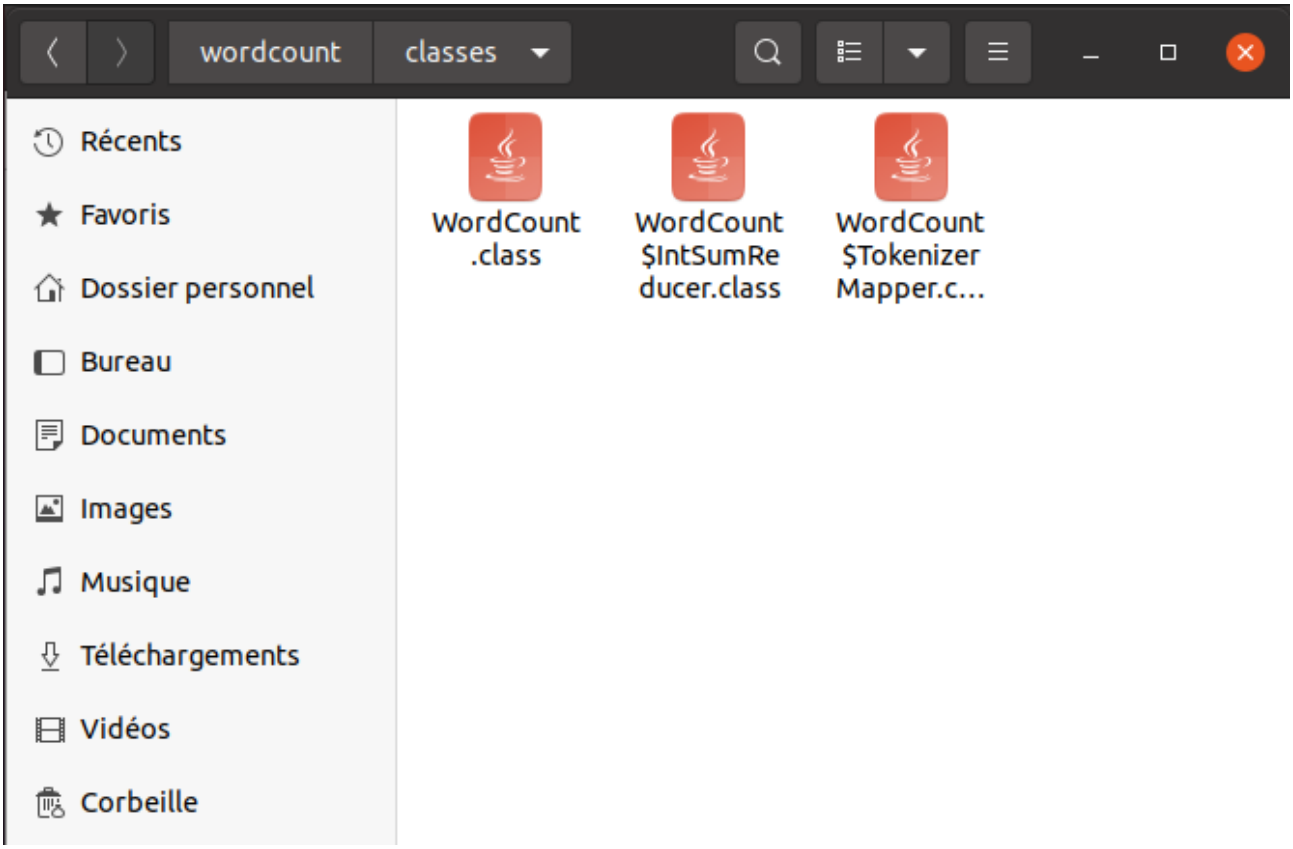
hadoop fs -put '/home/rachid/wordcount/input/input.txt' /wordcount/input



Maintenant, utilisez cette commande:

javac WordCount.java -cp $(hadoop classpath)

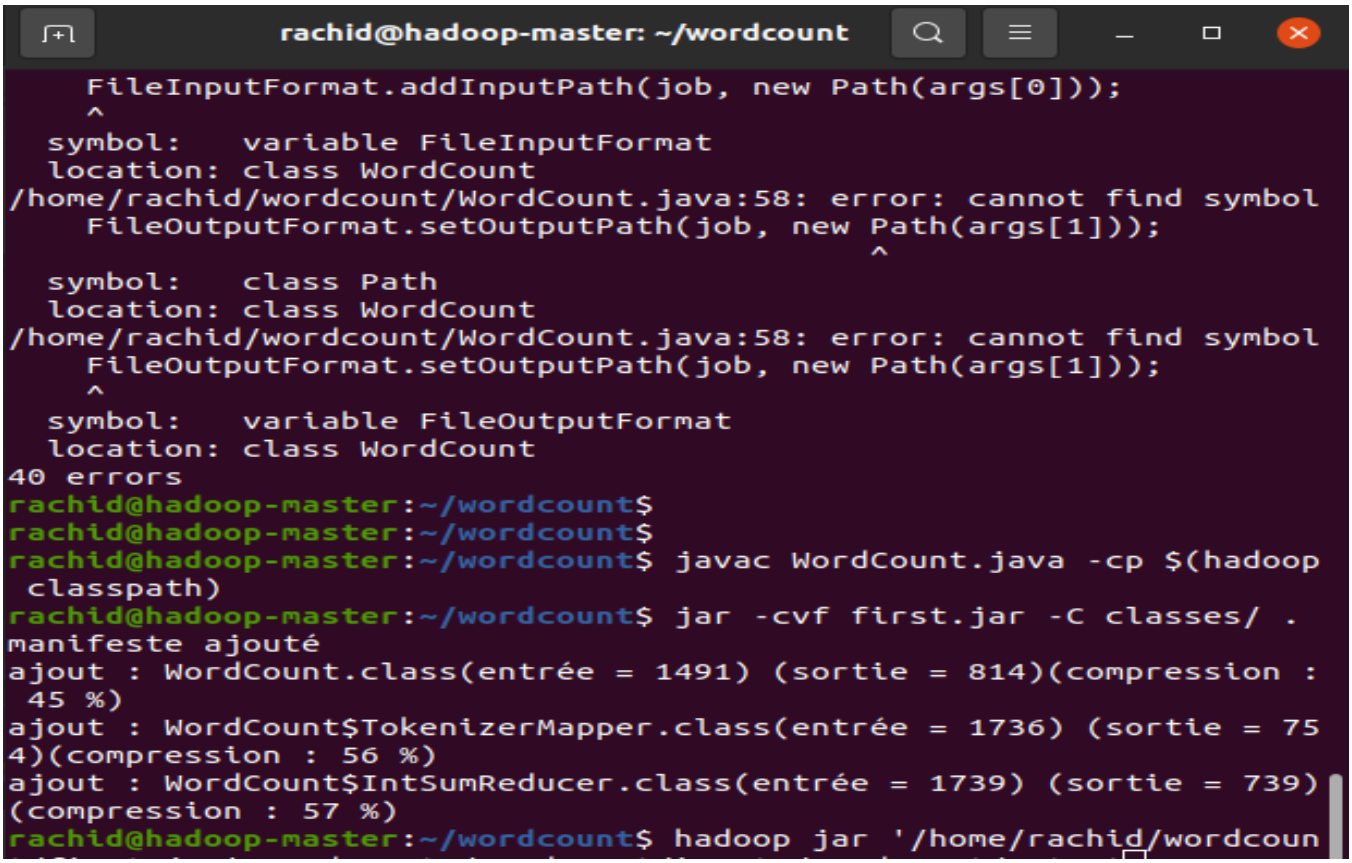


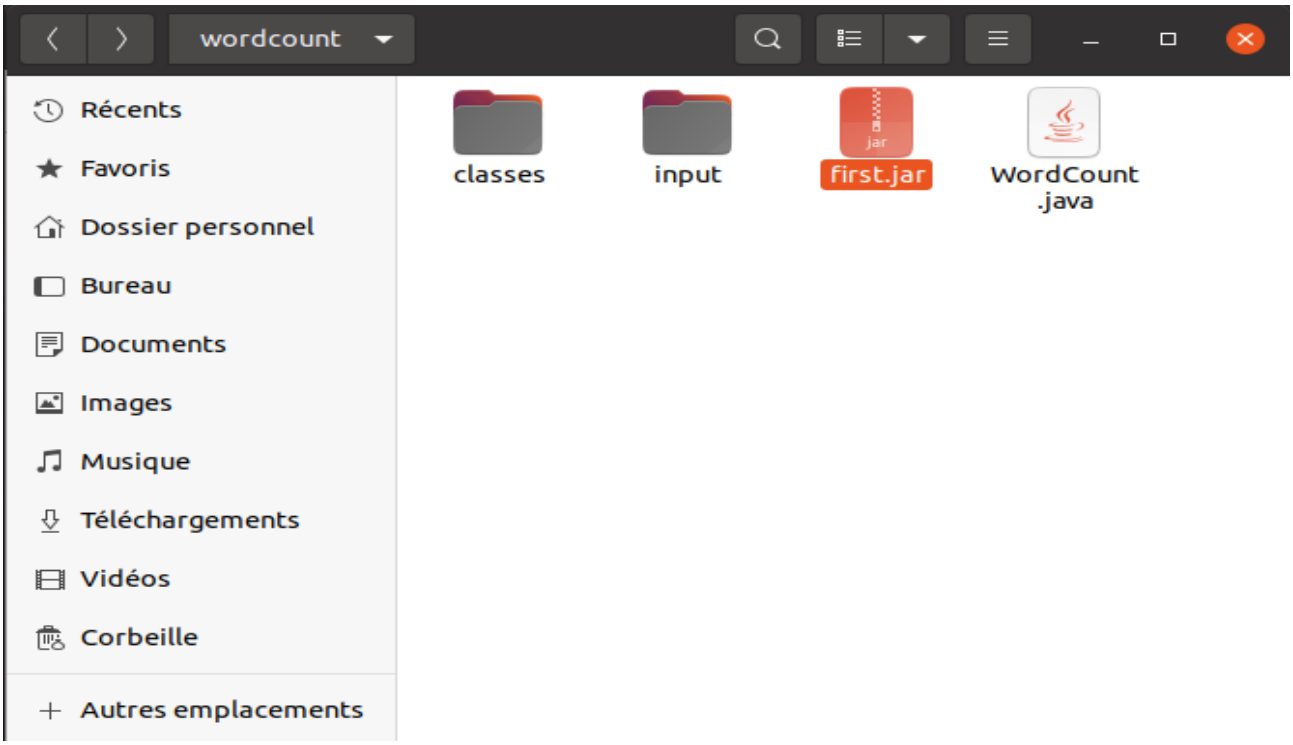


jar -cvf first.jar -C classes/

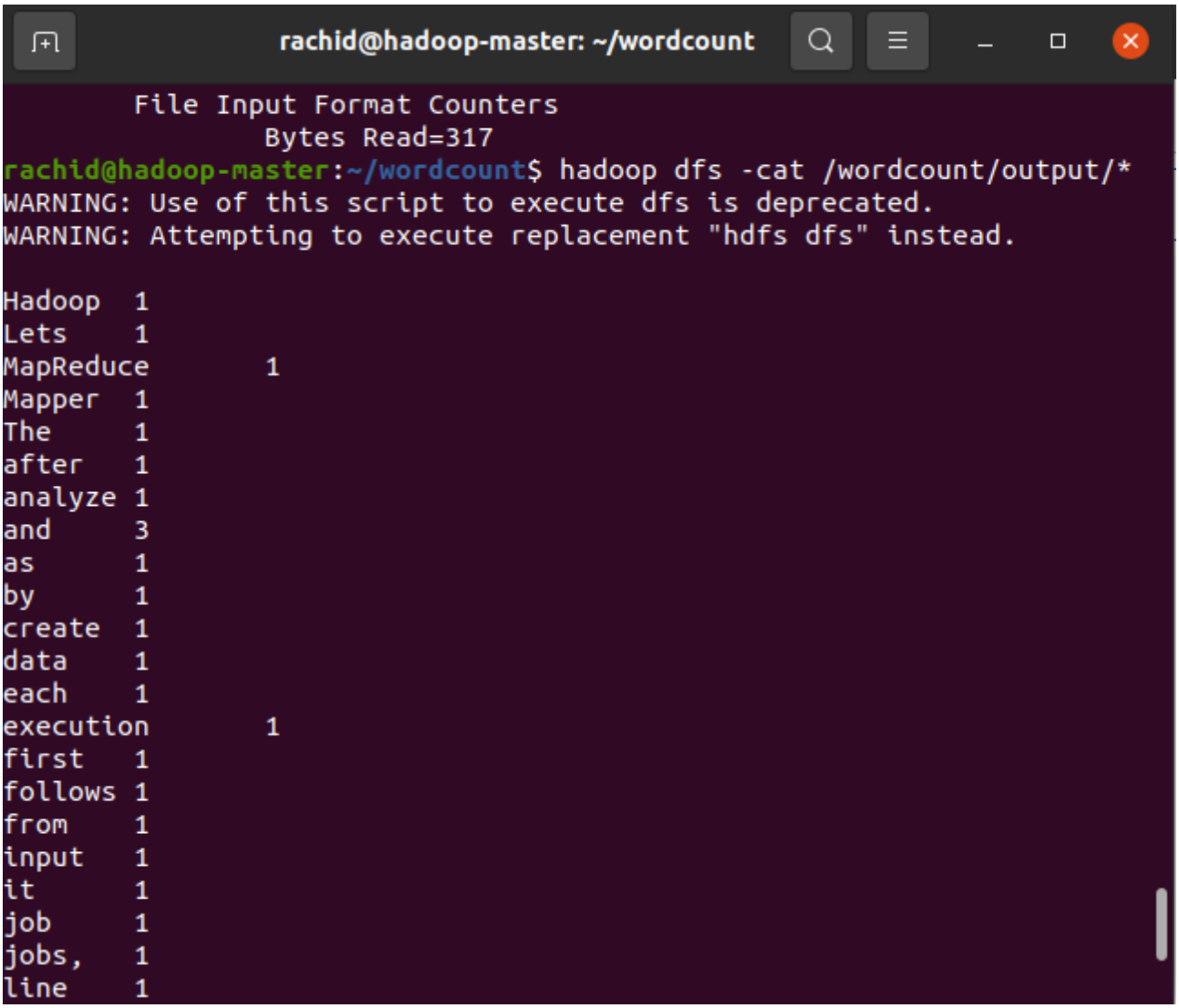
```
rachid@hadoop-master:~/wordcount$ jar -cvf first.jar -C classes/ .
manifeste ajouté
ajout : WordCount.class(entrée = 1491) (sortie = 814)(compression :
 45 %)
ajout : WordCount$TokenizerMapper.class(entrée = 1736) (sortie = 75
4)(compression : 56 %)
ajout : WordCount$IntSumReducer.class(entrée = 1739) (sortie = 739)
(compression : 57 %)
rachid@hadoop-master:~/wordcount$
```

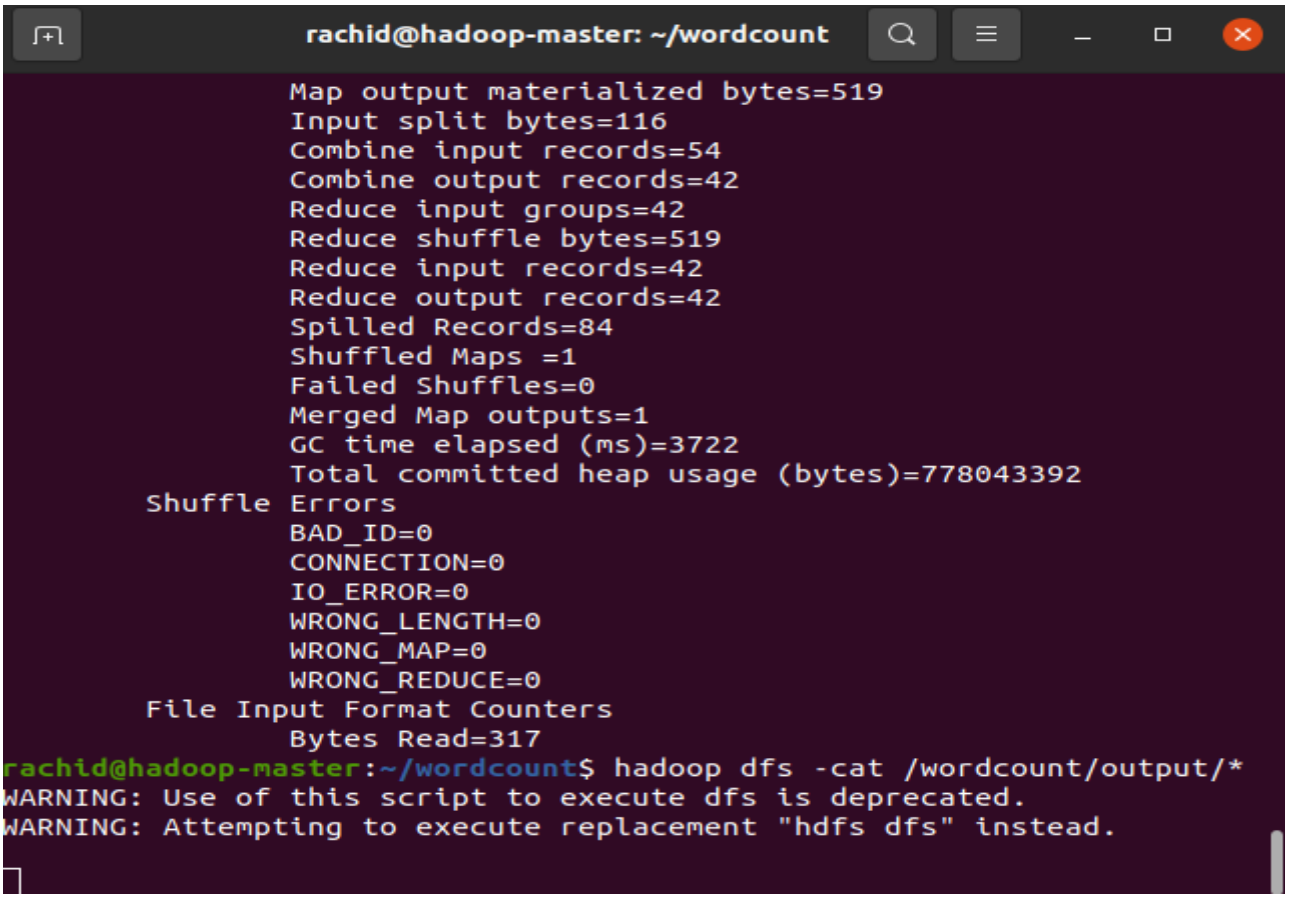
hadoop jar '/home/rachid/wordcount/first.jar' WordCount /wordcount/input /wordcount/output

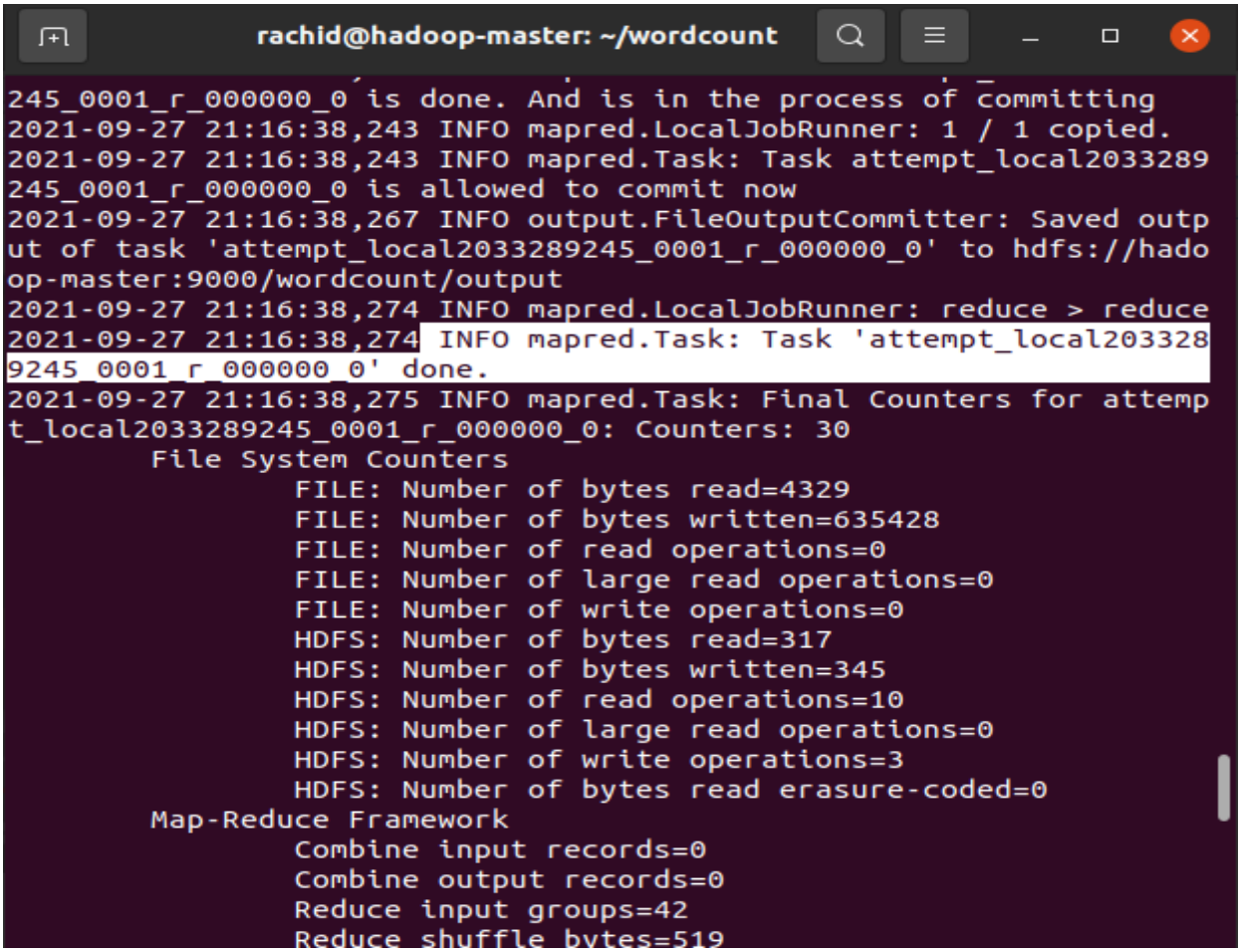


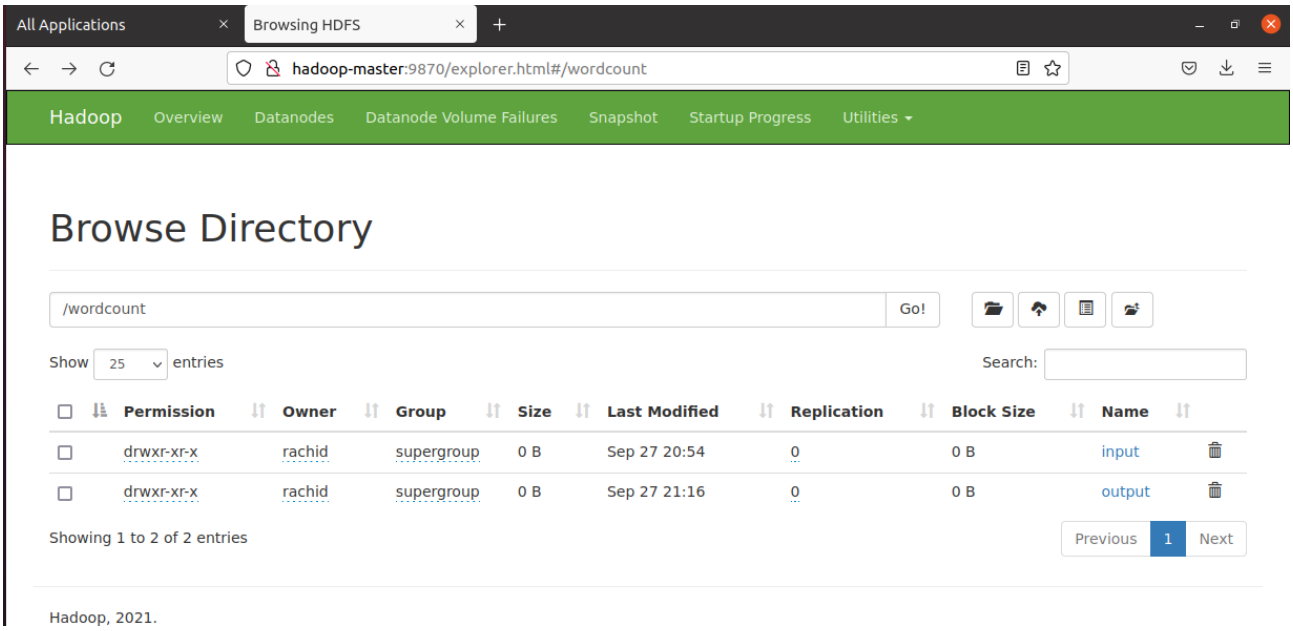


hadoop dfs -cat /wordcount/output/




rachid@hadoop-master: ~/wordcount
Map output materialized bytes=519
Input split bytes=116
Combine input records=54
Combine output records=42
Reduce input groups=42
Reduce shuffle bytes=519
Reduce input records=42
Reduce output records=42
Spilled Records=84
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=3722
Total committed heap usage (bytes)=778043392
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=317
rachid@hadoop-master:~/wordcount$ hadoop dfs -cat /wordcount/output/*
WARNING: Use of this script to execute dfs is deprecated.
WARNING: Attempting to execute replacement "hdfs dfs" instead.


rachid@hadoop-master: ~/wordcount
245_0001_r_000000_0 is done. And is in the process of committing
2021-09-27 21:16:38,243 INFO mapred.LocalJobRunner: 1 / 1 copied.
2021-09-27 21:16:38,243 INFO mapred.Task: Task attempt_local2033289
245_0001_r_000000_0 is allowed to commit now
2021-09-27 21:16:38,267 INFO output.FileOutputCommitter: Saved outp
ut of task 'attempt_local2033289245_0001_r_000000_0' to hdfs://hado
op-master:9000/wordcount/output
2021-09-27 21:16:38,274 INFO mapred.LocalJobRunner: reduce > reduce
2021-09-27 21:16:38,274 INFO mapred.Task: Task 'attempt_local203328
9245_0001_r_000000_0' done.
2021-09-27 21:16:38,275 INFO mapred.Task: Final Counters for attemp
t_local2033289245_0001_r_000000_0: Counters: 30
File System Counters
FILE: Number of bytes read=4329
FILE: Number of bytes written=635428
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=317
HDFS: Number of bytes written=345
HDFS: Number of read operations=10
HDFS: Number of large read operations=0
HDFS: Number of write operations=3
HDFS: Number of bytes read erasure-coded=0
Map-Reduce Framework
Combine input records=0
Combine output records=0
Reduce input groups=42
Reduce shuffle bytes=519



Conclusion

Les tests et les expérimentations effectués dans ce chapitre ont amené à la conclusion des points suivants :

- L'augmentation du nombre de nœuds influe sur l'exécution des programmes MapReduce, réduit son temps d'exécution et en conséquence la **latence** est minimisée.
- La réplication des données garantit un niveau de fiabilité pour le système

Hadoop en cas de défaillance d'un de ses nœuds et en conséquence assure la **disponibilité** du système.

- La flexibilité d'une telle architecture distribuée en augmentant le nombre de nœuds des clusters reflète le passage à l'échelle et assure **l'extensibilité**.

Conclusion générale

Conclusion générale

Notre projet a fait l'objet d'une étude des nouvelles technologies du Big Data, Nous nous sommes intéressés au Framework Hadoop. La partie applicative consistait à un déploiement d'Hadoop avec ses composants MapReduce et HDFS dans un environnement distribué à base de la distribution Apache.

Les différents systèmes de gestion des données conçus ces dernières années, se distinguent à base de trois facteurs principaux : La performance, caractérisée par la latence et le débit, l'extensibilité (élasticité ou scalabilité) et la disponibilité.

La mise en œuvre d'un cluster Hadoop avec ses composants, l'exécution de plusieurs jobs MapReduce sur des fichiers HDFS dans des architectures à plusieurs nœuds et l'évaluation des différents statistiques des opérations effectuées nous ont permis d'approuver l'impact de la distribution et le parallélisme sur l'efficacité des systèmes gérant de grands volumes de données en réduisant la latence et en garantissant et une meilleure extensibilité.

D'un autre côté, on a pu constater, que la réplication des données sur les différents nœuds du réseau garanti un certain niveau de disponibilité du système même en cas de défaillance d'un ou plusieurs nœuds du réseau.

Perspectives

Le domaine de recherche reste très ouvert aux différents travaux et différents tests dans des architectures fortement distribuées, néanmoins le futur projet qui se voit très intéressant, est la mise en œuvre d'un cluster Hadoop sur des machines physiques puissantes dans une plateforme High-Performance-Computing (HPC) pour pallier les problèmes du Hardware qu'on a rencontré. Aussi, la démonstration des points forts d'Hadoop à savoir la performance, la latence et l'extensibilité en appliquant plusieurs tests sur des données très volumineux de plusieurs téraoctets sur des centaines de nœuds voire des milliers de nœuds peut faire l'objet d'autres prochains sujets de recherche.

Références Bibliographiques

Livres électroniques

[1] Olivier BENDAVID, Bi in the Cloud, 18/06/10.

[2] Big data application and archetecteure; de himanshu et soumendra mohanty.

[3] Big Data for Dummies, par Wiley Brand.

[4] Hadoop The Definitive Guide.3rd.Edition, May 2012.Edition OReilly.

[5] Thèse

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

PDF

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

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

[8] Charley CLAIRMONT, Introduction à HDFS Hadoop Distributed File System, HUG France SL2013, 20p, Mai 2013.

[9]Jonathan Lejeune, Hadoop : une plate-forme d'exécution de programme Map-reduce, École des Mines de Nantes, 83p, Janvier 2015.

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

Sites internet

[11] http://www.journaldunet.com/solutions/analytics/big-data, consulté le1/03/2015.

[12]http://webcache.googleusercontent.com/search?q=cache:2DfXDe1Z_mEJ:www.aubay.com/fileadmin/user_upload/Publications_FR/Regard_Aubay__Big_Data_Web.pdf+&cd=1&hl=ar&ct=clnk&gl=dz, consulté le2/03/2015.

[13]http://webcache.googleusercontent.com/search?q=cache:GTE9JlAU6mAJ:www.bigdataparis.com/guide/Guide_du_Big_Data_2013_2014.pdf+&cd=1&hl=ar&ct=clnk&gl= dz, consulté le3/03/2015.

[14] http://blog.ippon.fr/2013/05/14/big-data-la-jungle-des-differentes-distributionsopen-source-hadoop/, consulté le15/04/2015.

[15] http://mbaron.developpez.com/tutoriels/bigdata/hadoop/introduction-hdfs-mapreduce/, consulté le 27/04/2015 .

Références Bibliographiques

[16] http://www.redsen-consulting.com/2013/06/big-data/, consulté le 9/06/2015.

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

[18] http://www.technologies-ebusiness.com/langages/les-bases-no-sql, consulté le. 6/06/2015

[19] http://fr.wikipedia.org/wiki/Hadoop, consulté le 10/04/2015.

[20] http://blog.ippon.fr/2013/05/14/big-data-la-jungle-des-differentes-distributionsopen-source-hadoop/, consulté le 10/04/2015.

[21] http://www.devx.com/opensource/exploring-the-hadoop-distributed-file-systemhdfs.html, consulté le 25/04/2015.

[22] https://doc.ubuntu-fr.org.

[23] https://gandalsmart.com.

Annexes

1. **Annexe A**

.L'algorithme de wordcount

```
import java.io.IOException;
import java.util.StringTokenizer;


import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;


public class WordCount {


public static class TokenizerMapper
extends Mapper<Object, Text, Text, IntWritable>{


private final static IntWritable one = new IntWritable(1);
private Text word = new Text();
```

```
  public void map(Object key, Text value, Context context
```

```
            ) throws IOException, InterruptedException {
    StringTokenizer itr = new StringTokenizer(value.toString());
    while (itr.hasMoreTokens()) {
      word.set(itr.nextToken());
      context.write(word, one);
    }
  }
}

public static class IntSumReducer
     extends Reducer<Text,IntWritable,Text,IntWritable> {
  private IntWritable result = new IntWritable();

  public void reduce(Text key, Iterable<IntWritable> values,
                     Context context
                     ) throws IOException, InterruptedException {
    int sum = 0;
    for (IntWritable val : values) {
      sum += val.get();
    }
    result.set(sum);
    context.write(key, result);
  }
}

public static void main(String[] args) throws Exception {
```

```
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```