الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université de 8 Mai 1945 – Guelma -

Faculté des Mathématiques, d'Informatique et des Sciences de la matière

Département d'Informatique

Mémoire de Fin d'études Master

Filière : Informatique

Option : Systèmes Informatiques

Thème :

Traitement massivement distribué dans un environnement BIG DATA

Sous Hadoop

Encadré Par :

Benhamida Nadjette

Présenté par :

Khelifati Sihem

Octobre 2020

Remerciements

Je tiens tout d'abord à remercier le bon ***Dieu*** *tout puissant, de m'avoir accordé le courage, la patience, la volonté et surtout la santé pour réaliser et mener à bien ce modeste travail.*

Mes remerciements les plus chaleureux et les plus vifs s'adressent à mon encadreur Mme. **Benhamida. N** *pour avoir accepté de m'encadrer, ainsi que pour ses conseils précieux, ses orientations, et le temps qu'elle m'a consacré afin de réaliser ce projet.*

Mes remerciements les plus sincères et les plus profonds sont adressés aux membres du jury pour l'honneur qu'ils m'ont accordé en évaluant ce mémoire.

Enfin un grand merci pour toute personne qui a contribué de près ou de loin à la réalisation de ce modeste travail.

J'espère qu'il sera d'une aide pour les futurs étudiant

Table des matières

Table des figures

Table des acronymes

HDFS
: Hadoop File Système

HP
: Hewlett-Packard

IBM
: International Business Machine

MPP
: Massivey Parallel Processing

NoSQL
: Not Only SQL

SQL
: Strectured Query Langage

Résumé

En conséquence de l'explosion quantitative des données numériques au cours des dernières années, Le traitement intensif des données occupe une place significative dans le quotidien des acteurs du monde scientifique et industriel à savoir : les chercheurs, les ingénieurs, les gestionnaires, les responsables d'entreprises, décideurs, etc. Ce type de traitement peut être parallèle ou distribué comme il peut être basé sur un parallélisme de données, sur un parallélisme de traitements ou les deux. L'objectif de ce travail est de proposer un modèle de traitement des méga données basé sur la communication des nœuds dans un environnement Hadoop.

Mots clés

Système Distribué, Big Data, Hadoop, Communication, MapReduce, HDFS.

Introduction générale

Ces dernières années ont été marqué par une explosion quantitatives croissante des données numériques, cela a créé un nouveau type de données dites les méga donnée ; un terme générique pour désigner un vaste ensemble de données. Ce phénome peut apporter plusieurs avantages dans tous les domaines, à savoir : l'industrie, la commerce, le marketing, etc. comme il peut générer d'autres problèmes qui concerne, particulièrement, le stockage et le traitement de ces données.

Par conséquence, la croissance des volumes de données traitées dépasse largement les limites des solutions traditionnelles à savoir les entrepôts de données et leurs modèles de traitements. Au fait, bien que les capacités de stockage des disques durs aient augmenté massivement au fil des dernières années, la vitesse d'accès aux données ne l'est pas encore. En plus, aucune solution classique ne permet de traiter rapidement et efficacement les grands volumes de données.

Notre projet de fin d'étude se repose, principalement, sur l'étude des méthodes et des technologies du Big Data. Nous nous intéresserons particulièrement aux technologies Hadoop avec ses composantes HDFS et MapReduce afin de réaliser des traitements parallèles et distribués sur de gros volumes de données pour améliorer les performances en termes de disponibilité des informations et d'accélération des traitements dans un tel environnement.

Notre mémoire est structuré comme suite :

Tout d'abord, Nous commencerons par une introduction générale décrivant le contexte de travail, la problématique et l'objectif de notre travail.

— **Chapitre 1 : Données massives**

  Dans ce premier chapitre, nous présentons brièvement quelques généralités sur les données massives.

— **Chapitre 2 : Systèmes distribués**

  Quant au deuxième chapitre, nous abordons les principes de base des systèmes distribués.

— **Chapitre 3 : Hadoop**

Dans ce troisième chapitre, nous détaillons le Hadoop, ses composants et son mode de fonctionnement.

— **Chapitre 4 : Implémentation et réalisation**

Dans le chapitre 4, nous présentons les différentes étapes de l'implémentation et la réalisation de notre travail.

Nous achèverons avec une conclusion générale et quelques perspectives pour les futurs travaux.

Chapitre 1 : Données massives

1.1 Introduction

Ces dernières années ont été marqué par une forte augmentation du développement et d'utilisation des outils informatiques ce qui a provoqué une explosion quantitative des données numériques, c'est ce qu'on appelle aujourd'hui les données massives, les méga données, les grosses données ou encore en anglais Big Data.

En effet, cette explosion de données apporte des bénéfices comme par exemple l'augmentation du nombre de clients aux seins des entreprises, du chiffre d'affaires, ou encore la réduction des coûts. Cependant, elle peut également générer des inconvénients. Particulièrement, les solutions existantes de gestions des données classiques ne sont plus valables dans un tel contexte. Par conséquent, il sera important de chercher de nouvelles solutions relatives à la recherche, le partage, le stockage, l'analyse et la présentation des données massives.

Dans ce chapitre nous présentons quelques généralités sur les données massives. Nous commençons par un peu d'historique et une définition de ce nouveau type des données, ensuite nous discutons leurs caractéristiques sous l'appellation de « les cinq V des données massives ». Également, nous présentons les innovations de technologies des données massives, leurs architecture et acteurs, ainsi que les grands domaines d'utilisation de ce type de données. Nous terminons le chapitre par une conclusion.

1.2 Historique

L'expression données massive serait apparue en octobre 1997, selon les archives de la bibliothèque numérique de l'ACM (Association for Computing Machinery) dans un article scientifique pour la visualisation des grands ensembles de données [1].

Bien que le concept de données massives soit relativement nouveau, les grands ensembles de données remontent aux années 60 et 70, lorsque le monde des données commençait à peine à démarrer avec les premiers datacenters et le développement de la base de données relationnelle [2].

1.3 Définition du données massives

Littéralement, les données massives désignent des ensembles de données qui sont tellement volumineux qu'ils en deviennent difficiles à traiter avec des outils classiques de gestion de base de données ou de gestion de l'information [3]. Cependant, aucune définition exacte ou universelle n'est donnée aux mégas donnés. Etant un objet complexe polymorphe, sa définition varie selon les communautés qui s'y intéressent en tant qu'utilisateur ou fournisseur de services [3].

1.4 Les 5 V des données massives

Les données massives sont souvent caractérisées par (voir la figure 1) [4] :

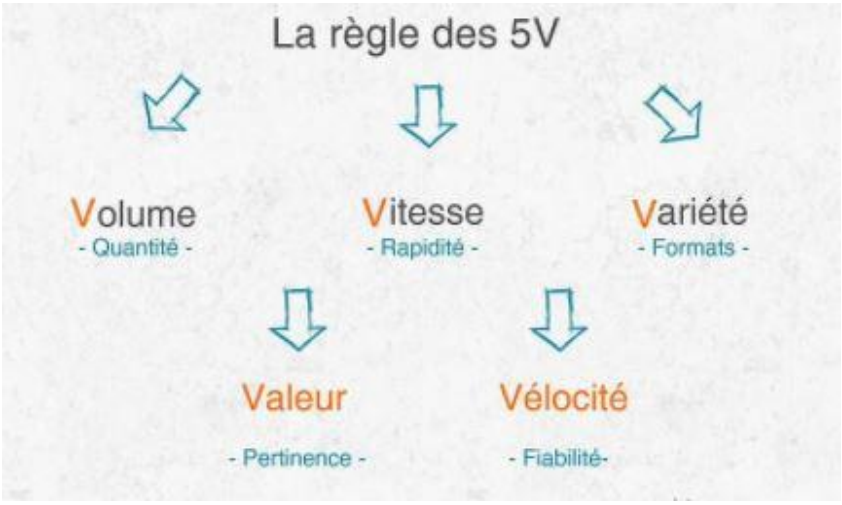

Figure 1: Les 5V des données massives.

Volume : Le volume correspond à la masse d'informations produite chaque unité du temps. Par conséquent, elle désigne le volume considérable de données informatiques à traiter, et qui dépassent les données textuelles et manuscrites. C'est pour cela que les entreprises multisites devront trouver des moyens pour gérer leurs volumes de données énorme créés quotidiennement.

Variété : Il ne s'agit pas de données relationnelles traditionnelles, ces données sont brutes, semi-structurées voire non structurées et de différents types. Elles sont des données complexes provenant spécialement du web, au format texte, images, vidéo, etc. Ce qui les rend difficilement utilisables avec les outils traditionnels. En plus, leurs traitements se varient selon leurs formats. Cette variété est une caractéristique clé des données massives.

Vélocité : la vélocité décrit la fréquence à laquelle les données sont générées, capturées et partagées. Du fait des évolutions technologiques récentes, les consommateurs mais aussi les entreprises génèrent plus de données dans des temps beaucoup plus courts.

Véracité : C'est l'un des enjeux majeurs de l'exploitation des données massives. Il fait référence à la fiabilité et la crédibilité des informations collectées. Cependant, avec autant de types, la qualité et la précision sont moins vérifiables.

Valeur : La notion de valeur correspond au profit qu'on puisse tirer de l'utilisation des données massives. Les technologies de stockage et d'analyse des données massives n'ont de sens que si elles apportent de la valeur à l'ensemble des données traité.

1.5 Innovations des technologies des données massives

Les nouvelles technologies qui nous permettent de stocker et d'accélérer le traitement des gros volumes de données sont classées en trois catégories [5]:

Accélération matérielle : il s'agit des équipements matériels qui utilisent des mémoires dynamiques afin d'améliorer la performance d'accès en lecture /écriture, aussi pour tirer un meilleur bénéfice des processeurs multicœurs.

Base de données d'analyse massivement parallèle : Ces bases de données conformes SQL sont conçues pour répartir le traitement des données sur plusieurs nœuds. Certaines bases de données MPP (Massivey Parallel Processing) utilisent également les architectures sans partage, qui répartissent également le stockage physique des données sur plusieurs machines.

Modèle Map-Reduce, Hadoop et autres approches NoSQL : Ces approches appelées collectivement NoSQL, permettent d'accéder aux données via des langages de programmation sans utiliser les interfaces basées sur SQL et permettent de répartir les données sur plusieurs machines distinctes.

Grâce à ces innovations technologiques, les entreprises peuvent désormais réaliser des analyses qui étaient considérées jusqu'à présent comme infaisables, soit parce qu'il y avait trop de données à traiter, les analyses prenant alors trop de temps, soit parce que le traitement requis était peu approprié pour SQL.

1.6 Architecture des données massives

L'architecture d'un système de gestion des données massives doit être conçue de telle sorte que l'acquisition, la manipulation, la circulation et la restitution des données soient facilitées. La plupart des architectures incluent une partie de ou tous les composants suivants [6]:

Source des données : Tous les systèmes de gestion des données massives reposent sur une ou plusieurs sources de données.

Stockage des données : Les données générées sont généralement stockées dans un magasin des fichiers distribués, ce que on appelle un « lac de données ».

Traitement par lots : Le traitement des mégas donnés doit souvent traiter les fichiers de données à l'aide des traitements par lots à longue durée d'exécution pour filtrer, agréger et préparer les données en vue de l'analyse.

Traitement des messages en temps réel : Vu que la solution inclut des sources en temps réel, l'architecture doit aussi inclure un moyen pour capturer et stocker des messages en temps réel pour le traitement de flux de données. Il peut s'agir d'un simple magasin de données, où les messages entrants sont déposés dans un dossier en vue du traitement.

Traitement de flux : Après l'ingestion des messages en temps réel, la solution doit les traiter en filtrant, en agrégeant et plus généralement, en préparant les données pour l'analyse, ensuite ces données seront écrites dans un récepteur de sortie.

Analyse et rapports : La plupart des solutions de gestion des données massives ont pour but de fournir des informations sur les données par le biais de l'analyse et des rapports.

1.7 Les acteurs et les domaines des données massives

En pleine croissance, le marché des données massives regroupe plusieurs acteurs proposant des services spécifiques [7]:

Les acteurs historiques : Les fournisseurs historiques tels que : HP, Oracle et IBM figurent parmi les principaux acteurs des données massives intervenant chacun à des niveaux différents.

Les grands noms du Web : Les grands acteurs du web, dont les moteurs de recherche Yahoo et Google, ainsi que les réseaux sociaux comme Facebook proposent également des solutions importantes dans ce domaine.

Les spécialises : Parmi lesquels on peut citer les intégrateurs tels que : CapGemini, Accenture, EMC, MapReduce, etc.

Les grands domaines des données massives sont [8]:

L'industrie : La maintenance prédictive, permise par les données massives, permet essentiellement d'anticiper les risques de pannes et de défections de certaines pièces ou composants.

La publicité : Il s'agit probablement du domaine le plus touché. Du fait que ce type de données améliore considérablement la relation entre les clients et les entreprises.

Le marketing : Dans ce domaine aussi, l'émergence des données massives révolutionne le secteur. Grâce aux multiples données établies, la connaissance client sera renforcée et permet d'améliorer grandement le développement d'un produit et d'un service, adapté aux besoins et aux attentes des utilisateurs.

Le commerce : La connaissance de l'historique de consommation d'un client permet aujourd'hui de faire des recommandations personnalisées d'achat très précises.

La banque : Toutes ces évolutions ont fortement amélioré l'automatisation des processus de gestions des sinistres.

1.8 Conclusion

Dans ce chapitre nous avons présenté quelques généralités sur les données massives qui peuvent apporter plusieurs avantages dans tous les domaines. Toutefois, il est difficile de fournir une gestion efficace de gros volumes de données, notamment dans un contexte d'entrepôt de données. C'est pourquoi plusieurs solutions ont été proposées en se basant sur une distribution des données sur des différents disques afin d'assurer une parallélisation de l'exécution des traitements.

Dans le chapitre suivant, nous allons aborder les systèmes distribués qui présente l'une des solutions efficaces pour l'amélioration des performances des systèmes d'informations dans un environnement Big Data.

Chapitre 2 : Système distribué

2.1 Introduction

A nos jours les entreprises multisites connaissent une augmentation des ressources informatiques utilisées, ce qui engendre une masse importante des données à traiter. Cela nécessite une infrastructure informatique haute performances afin de mieux stocker et manipuler ces données. Les systèmes distribués proposent une solution pour améliorer la vélocité des systèmes d'informations des entreprises, ainsi que leur disponibilité.

Dans ce chapitre, nous aborderont quelques généralités sur les systèmes distribués, tout en commençant par une définition d'un système distribué, ensuite son architecture et son mode de communication, ainsi que ses objectifs.

2.2 Définition d'un système distribué

Plusieurs définitions ont étés proposées dans littérature, à savoir :

« Un système distribué est un système réparti, ou plus précisément un ensemble de dispositifs communicants entre elles afin de résoudre une tâche commune, comme c'est un ensemble d'entités autonomes de calcul interconnectées » [9].

« Un ensemble de machines connectées par un réseau, et équipées d'un logiciel dédié à la coordination des activités du système ainsi qu'au partage de ses ressources » [10].

Le fonctionnement collaboratif désigne des traitements coopérants sur des données réparties pour réaliser une tâche commune [10] :

La coopération entre les processus : correspond à la communication entre eux et la synchronisation de leurs évolutions (voir la section 2.6).

La répartition : peut concerner les données comme les traitements (voir la section 2.7).

2.3 Caractéristiques d'un système distribué

Les principales caractéristiques des systèmes distribués peuvent être résumées dans les points suivants [11] :

Absence de mémoire commune : cela peut avoir comme conséquence l'impossibilité de capter instantanément l'état global d'un système réparti au moyen d'un ensemble de variables partagées.

Absence d'une horloge physique commune : A un instant donné, un processus s'exécutant sur une machine ne peut connaître que de manière approchée l'état d'une autre machine.

Variabilité des délais de transmission des messages : De même, deux éléments distincts du système peuvent avoir une perception différente de l'état d'un troisième élément ou sous-système et de l'ordre des événements qui s'y produisent.

Délai de transmission d'un message entre deux processus s'exécutant sur deux machines distinctes est très grand par rapport au temps qui sépare deux points observables consécutifs sur une même machine.

2.4 Architecture d'un système distribué

Actuellement, on distingue deux familles d'architecture distribués [11] [12]:

Architecture répartie : Dans une telle architecture, on trouve souvent une (ou bien plusieurs) machine (s) serveur (s) dont il est implémenté la quasi-totalité des composants logiciels et des fonctionnalités (voir la figure 2).

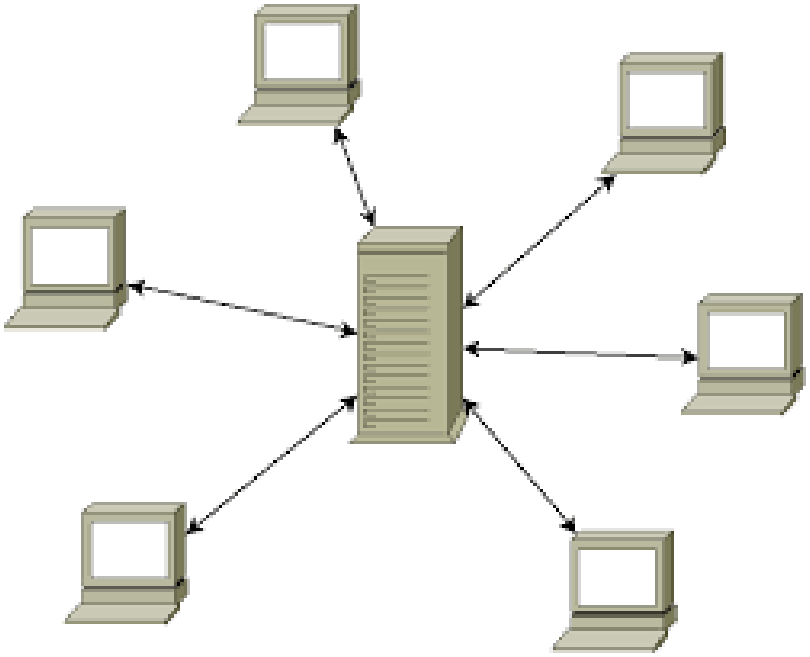

Figure 2: L'architecture répartie.

Architecture distribuée : Dans cette architecture, toutes les machines ont des rôles équivalents (voir la figure 3).

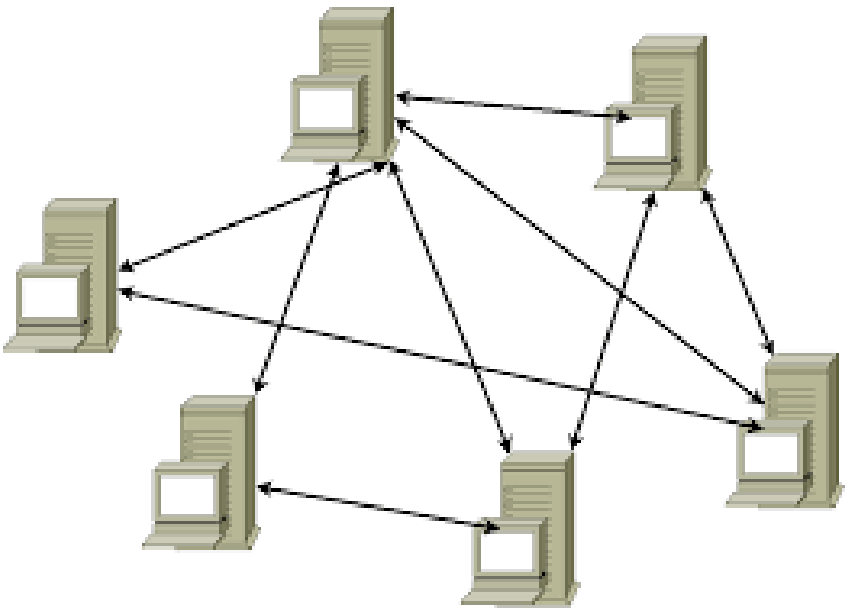

Figure 3: L'architecture distribuée.

2.5 Intérêt et Objectif d'un système distribué

Les systèmes distribués et les systèmes repartis ont plusieurs avantages, tels que [13]:

- Le partage des ressources afin de collaborer le travail.
- Accès distant, qui veut dire que plusieurs acteurs accèdent au même service.
- Confidentialité, les données brutes ne sont pas disponibles partout au même moment.
- Disponibilité des données en raison de l'existence de plusieurs copies.
- Augmentation de la fiabilité grâce à la duplication de machines ou de données, ce qui induit à une réalisation des systèmes à haute disponibilité.
- Réalisation des systèmes à grande capacité d'évolution.

On peut citer également [11]:

- Facilité de communications.
- Concurrence.

Un système distribué fonctionne par la coopération de plusieurs processeurs, ce qui engendre quelques objectifs [14]:

- Coût : la possibilité de réaliser un seul système avec plusieurs processeurs à bas prix.
- Puissance de calcul : aucune machine centralisée ne peut réaliser ces traitements en un tel temps.
- Performances élevées.
- Adaptation : à des classes d'applications réelles naturellement distribuées.
- Fiabilité : résistance aux pannes logicielles ou matérielles.
- Extensibilité : croissance progressive selon le besoin.

2.6 Communication dans un système distribué

La communication dans un système distribué peut être basée sur :

Echangede messages : Un processus doit envoyer un message vers un autre processus qui se trouve sur la même machine ou sur des machines distante [15] (voir la figure 4). Ce modèle de communication est utilisé principalement dans les architectures distribuées et réparties.

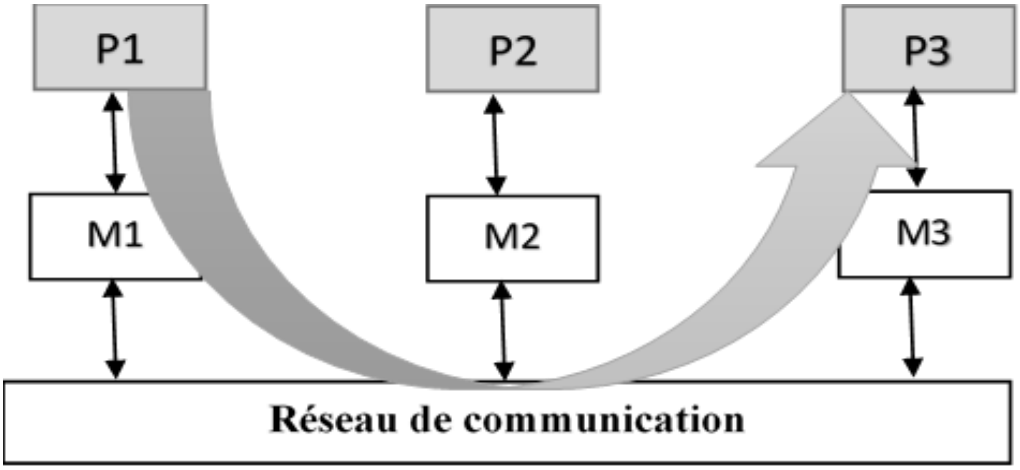

Figure 4: La communication via échange de message [15].

Appel de procédures distantes : Un appel de procédure à distance est en générale un appel entre plusieurs machines ce qui provoque la création d'une nouvelle procédure dans un autre espace d'adressage pour exécuter la tâche appelée [15] (voir la figure 5). Ce modèle de communication peut être considéré comme un cas particulier de la communication par échange de messages.

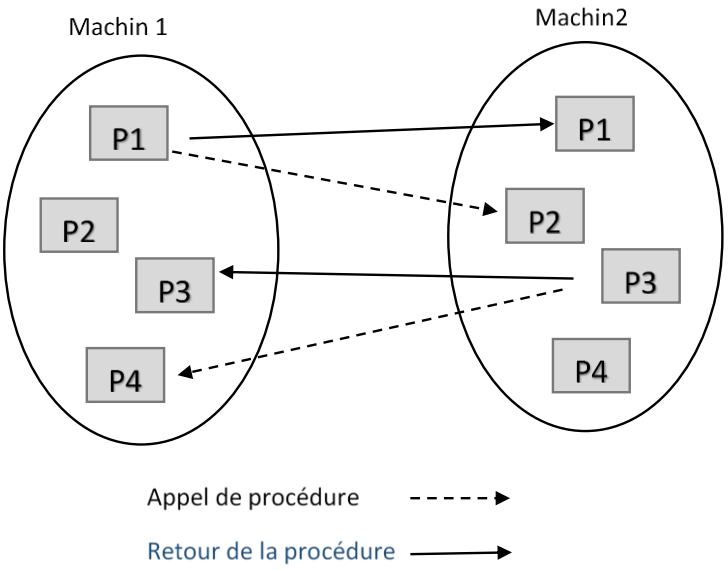

Figure 5: La communication via appel de procédures distantes [15].

Mémoire partagée : Le partage de mémoire est un moyen de partager des données entre différents processus, mais chaque processus ne les voit pas nécessairement à la même adresse [15] (voir la figure 6). Notons que chaque machine ou bien processeur possède son propre espace mémoire local.

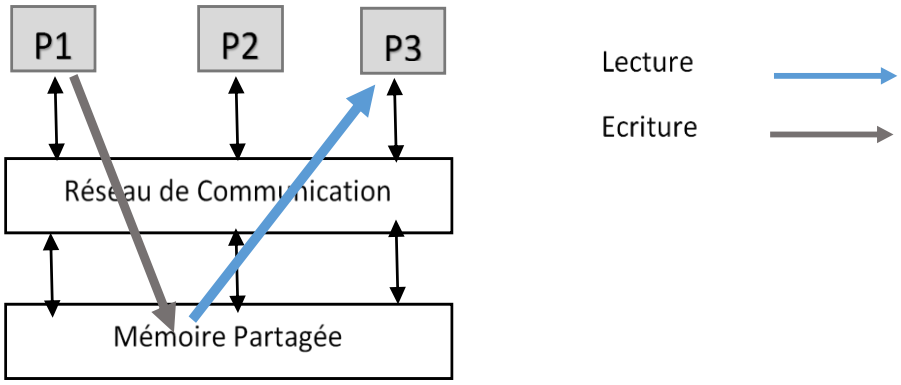

Figure 6: La communication via mémoire partagée [15].

Il existe deux types de communications dans les systèmes distribués [11]:

La communication asynchrone : les opérations d'émission ne sont pas bloquantes alors que les opérations de réception le sont.

La communication synchrone : les opérations d'émission sont-elles aussi bloquantes.

2.7 Traitement distribué

Le calcul ou bien le traitement intensif numérique occupe une place significative dans le quotidien des acteurs du monde scientifique et industriel (chercheurs, ingénieurs, etc) [16]. Il peut être basé sur un parallélisme de données, sur un parallélisme de traitements ou les deux.

Parallélisme de données [17] **:** Les différentes machines effectuent un traitement (un ensemble d'instructions) similaire sur des ensembles de données différents. Le parallélisme concerne alors les données et non plus les traitements (instructions).

Parallélisme de traitements [17]**:** Dans ce cas, les différents nœuds exécutent des traitements différents sur des données similaires.

2.8 Conclusion

Dans ce chapitre, nous avons appris qu'un système distribué peut être considéré comme un ensemble de nœuds communicants entre eux afin de résoudre une tâche commune via un réseau de communication. Ce dernier peut avoir une topologie physique statique ou une topologie logique qui change avec le temps. Ces systèmes ont été proposés initialement pour des environnements classiques mais ils peuvent également fournir de meilleures performances pour le traitement des méga données.

Dans le chapitre suivant, nous allons présenter le Hadoop qui représente l'un des plateformes dédiées au traitement distribué des données massives.

Chapitre 3 : Hadoop

3.1 Introduction

Les systèmes traditionnels ne disposent pas d'outils dédiés aux données massives. Il faut donc se tourner vers les nouveaux acteurs de ce dernier, et accepter leur approche différente de la technologie. Il est logique de dire que l'adaptation est la meilleure solution dans une telle situation. En général, cette adaptation est assurée par l'intégration de nouvelles solutions permettant le déploiement du outils existants à faible coût tout en se basant sur les principes de la distribution des traitements, ainsi que de la duplication des données. Parmi les solutions les plus utilisées, on trouve le une plateforme orientée objet distribuée à haute disponibilité dite le Hadoop. Ce dernier est l'un des systèmes permettant le traitement des fichiers de grands blocs, et cela assure des traitements des données massives plus rapides et surtout plus efficaces.

Dans ce chapitre, nous présentons la définition de Hadoop, ses composantes, ainsi que son mode de fonctionnement au premier lieu. Ensuite, nous détaillons le système de fichier distribué de Hadoop dit HDFS et le modèle de programmation MapReduce. En fin, nous terminons le chapitre par une conclusion.

3.2 Définition de Hadoop

Le Hadoop est un est une plateforme orientée objet de référence libre et open source qui permet de : (1) analyser, (2) stocker et (3) manipuler de très grandes quantités de données. Hadoop fait partie des projets de la fondation logicielle Apache depuis 2009 et son principe de fonctionnement repose sur le traitement distribué entre différents nœuds ce qui permet de traiter de très grande quantité de données en parallèle.

Le noyau de Hadoop est composé principalement de [18]:

1. Une partie stockage consistant en un système de fichiers distribué, extensible et portable appelé HDFS (Hadoop Distributed File System). Ce dernier est un système extrêmement puissant pour la gestion répartie des données.

2. Un modèle de programmation massivement parallèle appelé MapReduce, adapté au traitement de très grandes quantités de données (la partie traitement de Hadoop).

3. Une collection d'outils spécifiques pour la gestion du HDFS et MapReduce.

3.3 MapReduce

MapReduce est un modèle d'architecture de développement informatique, dans lequel sont effectués des calculs parallèles et souvent distribués sur des données pouvant être très volumineuses. Il se base principalement sur deux fonctions ; la fonction « Map » et la fonction « Reduce », empruntées aux langages de programmation fonctionnelle [19].

De façon générale, la fonction « Map » est exécutée par un nœud spécifique dans notre système. Elle analyse un problème, le découpe en sous-problèmes, et ensuite délègue la résolution de ces sous-problèmes à d'autres nœuds de traitements pour être traités en parallèle, ceci à l'aide de la fonction Reduce. Ces nœuds font ensuite renvoyer leurs résultats au nœud qui les avait sollicités (voir la figure 7). Ainsi le modèle MapReduce permet de manipuler de grandes quantités de données en les distribuant dans un cluster de nœuds pour être traitées [19].

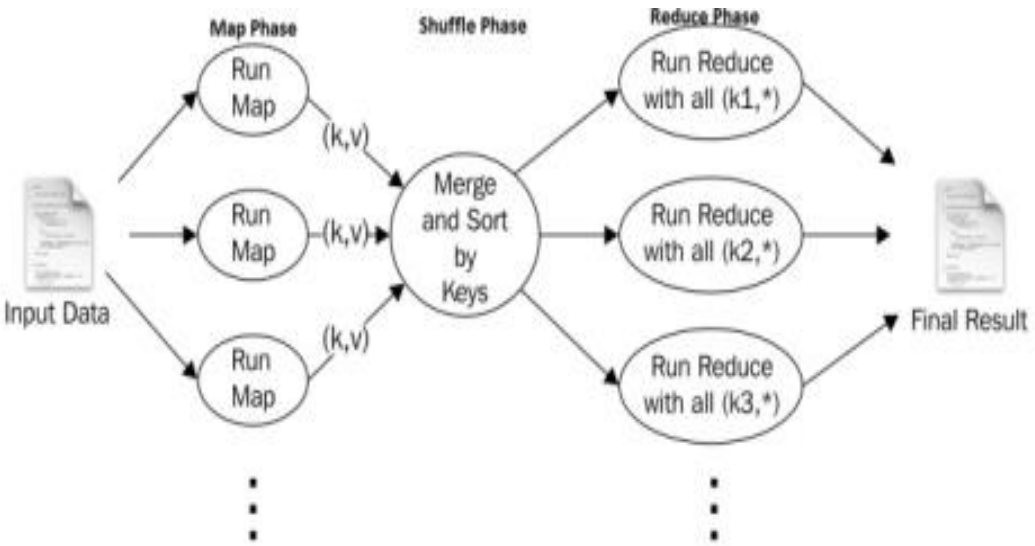

Figure 7: Le principe de fonctionnement de MapReduce [20].

Le modèle MapReduce a été rapidement utilisé par des sociétés intervenant sur le Web et possédant d'importants centres de traitement de données tels que : Amazon et Facebook. Notons que MapReduce est aussi de plus en plus utilisé dans le « Cloud Computing » [19].

De nombreux systèmes implémentant MapReduce ont vu le jour, dont le plus connu est Hadoop. On peut dire que le modèle MapReduce représente l'élément essentiel pour le fonctionnement de Hadoop [21]. Également, il existe deux types de MapReduce : MapReduce 1 (MP 1) et YARN (MP 2) (voir la figure 8 et figure 9).

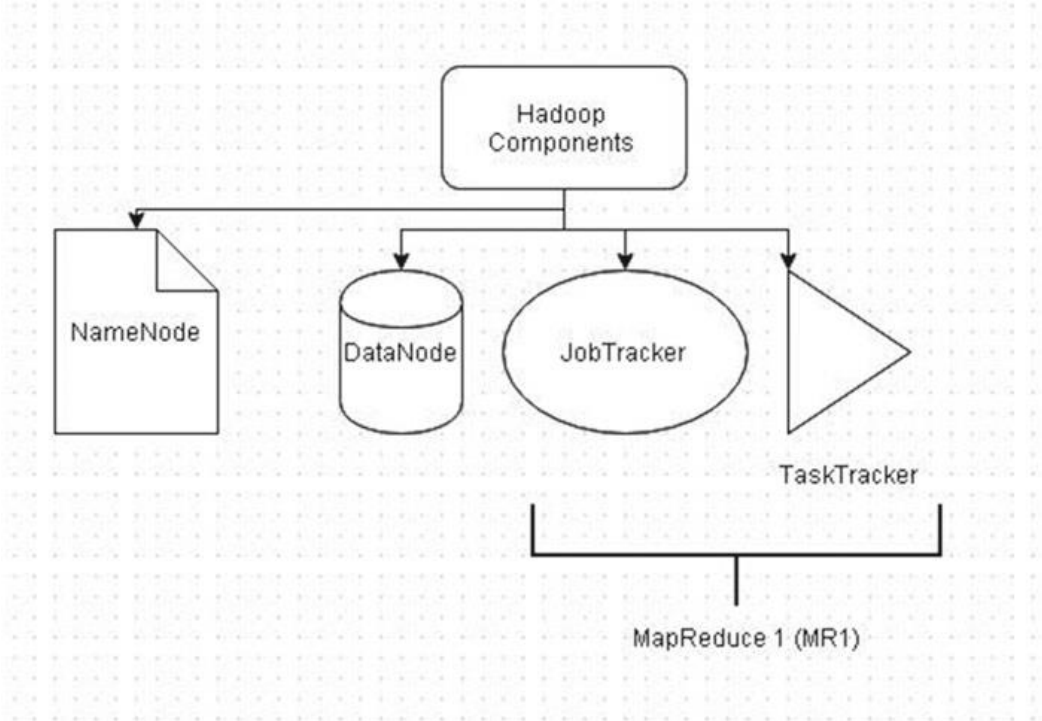

Figure 8 : Les composantes du MapReduce (MP1).

Les composantes du MapReduce sont :

Le gestionnaire de tâches (Job Tracker) [22]**:** Il s'occupe de la coordination des tâches sur les différents clusters. Il attribue les fonctions de MapReduce aux différents « TaskTrackers ». Le « Job Tracker » est un « Daemon » cohabitant avec le « Name Node » et ne possède donc qu'une instance par cluster.

Le moniteur de tâches (Tasktracker) [22] **:** Il permet l'exécution des ordres de MapReduce, ainsi que la lecture des blocs de données en accédant aux différents « Data Nodes ». Par ailleurs, le « TaskTracker » notifie de façon périodique au « Job Tracker » le niveau de progression des tâches qu'il exécute, ou alors d'éventuelles erreurs pour que celui-ci puisse reprogrammer et

assigner une nouvelle tâche Un « TaskTracker est un « Deamon » cohabitant avec un « Data Node », il y a un donc un « TaskTracker » par « Data Node ».

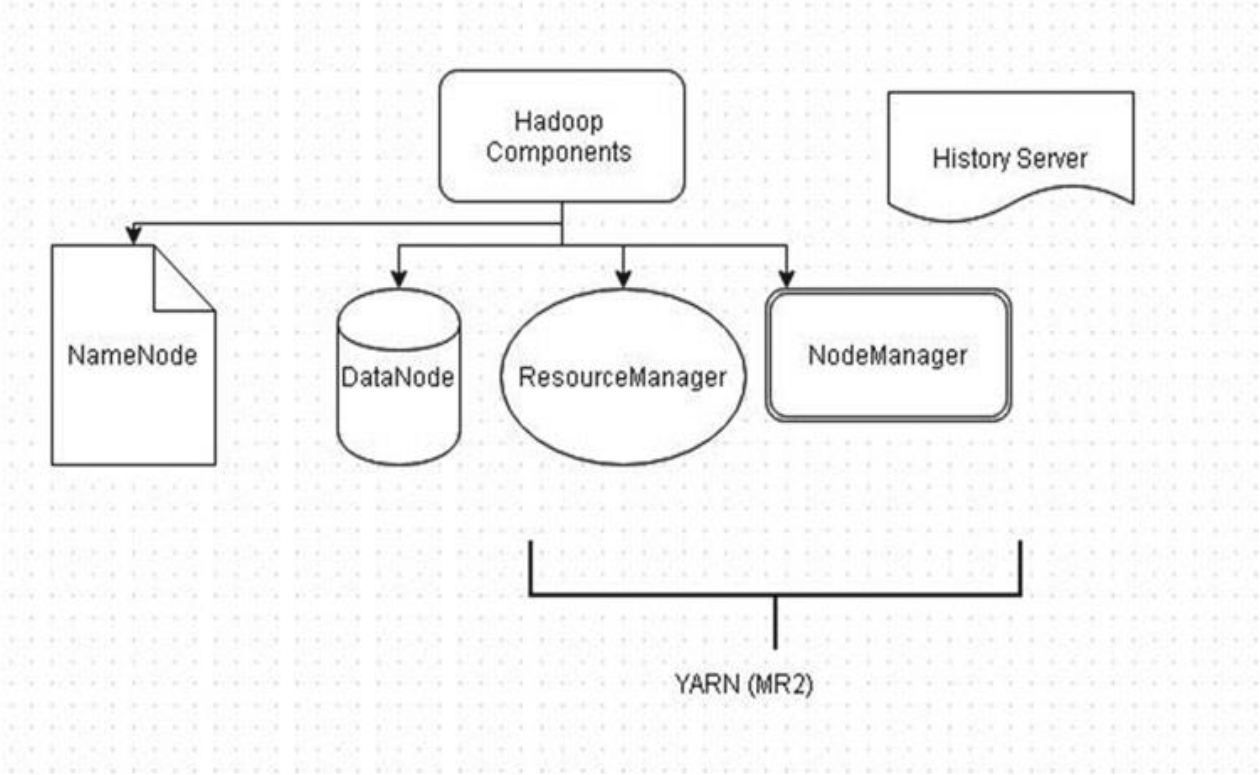

Figure 9 : Les composantes du YARN (MP 2) [23].

Le gestionnaire de ressources (ResourceManager) [23]**:** il est un nœud maître qui gère les tâches des clients. Il affecte également les ressources requises pour chacune des tâches.

Le gestionnaire des nœuds (NodeManager) [23] **:** il est une composante de chaque nœud. Comme son nom l'indique, il s'occupe de la gestion des nœuds du systèmes.

Le serveur d'historique des taches (Job HistoryServer) [23]**:** il est fourni dans MR2 pour conserver les informations sur les tâches MapReduce.

3.4 Système de fichier distribué HDFS

Ce système de fichiers 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 [22].

Toutefois, HDFS se démarque d'un système de fichiers classique pour les principales raisons suivantes [22] :

- 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 méta-données ;
- 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.

Ce système est capable de gérer des milliers de nœuds sans aucune intervention d’un autre opérateur [24].

3.5 Composantes du HDFS

Dans ce qui suit, on présente les composantes du HDFS (voir la figure 8):

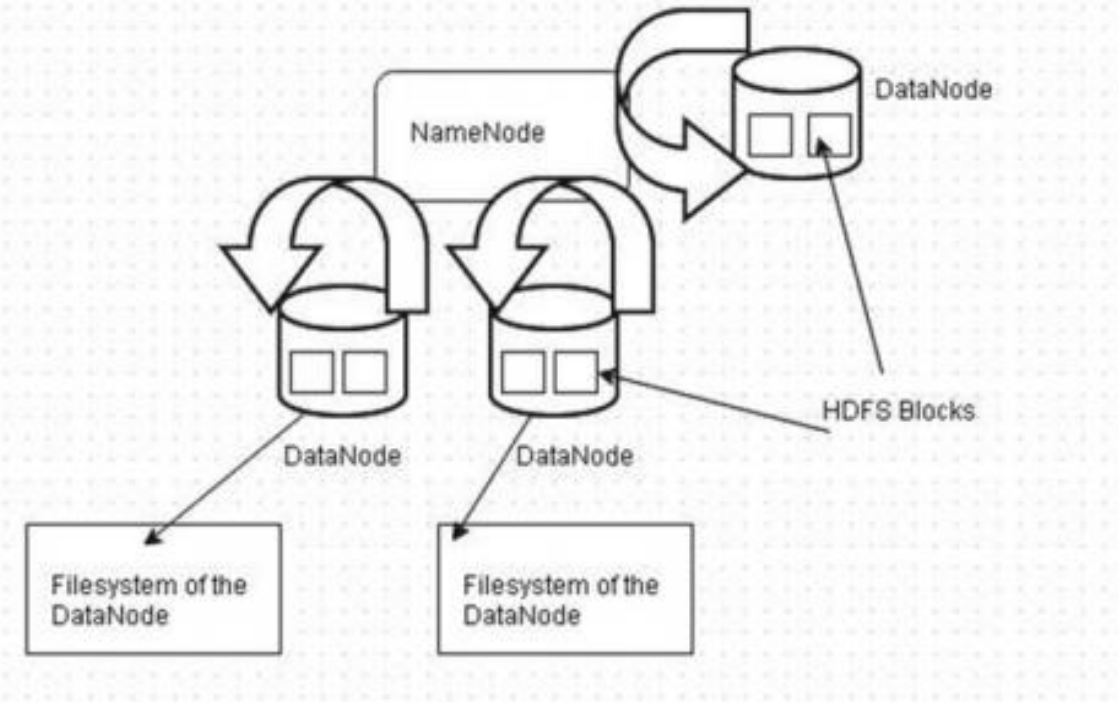

Figure 10 : L'architecture HDFS [23].

Le nœud des noms (Name Node) [22]**:** est un nœud maitre qui stocke les méta-données uniquement. Il maintient une arborescence de tous les fichiers du système et gère l'espace de nommage. Il centralise la localisation des blocs de données répartis sur le système. Sans le « Name Node », les données peuvent être considérées comme perdues car il s'occupe de reconstituer un fichier à partir des différents blocs répartis dans les différents « Data Node ». Il n'y a qu'un « Name Node » par cluster HDFS.

Le nœud secondaire (Secondary Node) [22] **:** La fonction principale de ce nœud consiste à effectuer des vérifications périodiques du journal des modifications pendant que le NameNode est en cours d'exécution. Lorsqu'un journal des modifications a été vérifié, il sera effacé par la suite et, par conséquent, ça permettra de libérer plus d'espace disque sur le NameNode. Le démarrage de NameNode sera plus rapide donc [23]. Le nœud secondaire peut prendre la relève en cas de panne de ce dernier [22].

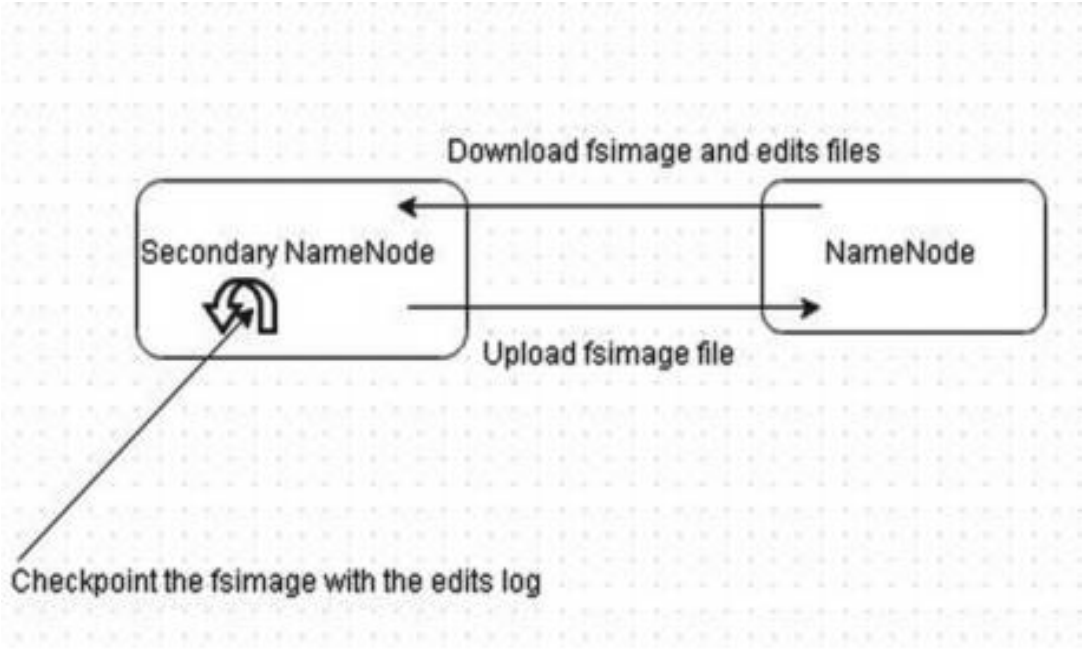

Figure 11 : Le rôle du nœud secondaire [23].

Le nœud de données (Data Node) [22]**:** Il permet le stockage des blocs de données. Il communique périodiquement au « Name Node » une liste des blocs qu'il gère. Un HDFS contient plusieurs noeuds de données ainsi que des réplications d'entre eux. Ce sont les noeuds esclaves.

3.6 Caractéristiques du HDFS

Les principales caractéristiques d'un HDFS sont [25] :

- La haute tolérance aux pannes.
- Le déploiement du matériel à faible coût.
- L'accès haut débit aux données d'application.

3.7 Domaines d'utilisation de Hadoop

Hadoop est recommandé pour l'exécution des traitements itératives tels que les algorithmes de graphes dont chaque itération doit lire (ou écrire) des données à partir du (sur le) disque parce que le coût des opérations d'E / S et de la latence d'une itération est très élevé.

Cependant, Hadoop n'est pas recommandé pour les :

- Applications nécessitant un état partagé ou une coordination car les tâches MapReduce sont indépendantes et ne partagent pas les données.
- Applications à faible latence à savoir les applications temps réel.
- Ensembles de données de petites tailles.
- Recherche des enregistrements individuels.

3.8 Conclusion

Dans ce chapitre, nous avons détaillé le Hadoop, ces composantes, ainsi que son mode de fonctionnement. Notons que de nouveaux systèmes spécifiques ont été créés permettant d'améliorer les performances de Hadoop, notamment dans les milieux hétérogènes, tant en termes de vitesse de traitement, qu'en termes de consommation électrique.

Dans le prochain chapitre, nous allons présenter les différentes étapes de l'implémentation et la réalisation de notre travail.

Chapitre 4 : Implémentation et réalisation

4.1 Introduction

Face aux problèmes de stockage, organisation et manipulation des données massives. L'univers technologique du Big Data s'appuie sur des outils bien identifiés qui constituent la base innovante de ce mode de traitement. Le Hadoop est l'un des outils technologiques qui permettant de traiter un grand nombre de données grâce à son architecture distribuée de bases de données.

Dans ce chapitre nous allons présenter l'environnement de développement de notre application, notre contribution concernant le traitement massivement distribué sous Hadoop puis ses étapes d'installation sous Windows. Ensuite, nous montrons les étapes d'exécution d'un exemple Hadoop, les résultats de l'implémentation de notre travail et enfin une conclusion.

4.2 Environnement de développement

Nous avons utilisé l'IDE Eclipse version 2020.09 (4.17.0) pour l'implémentation de notre système, la version 1.8.0 de java et la version 3.1.0 de Hadoop. Ce choix est basé spécialement sur la stabilité de Hadoop ainsi que la disponibilité des documentations requise pour le développement des objectifs visés.

Nous avons utilisé également une machine DELL i5 avec 8 Go de RAM et qui procède comme système d'exploitation « Windows 10 ».

4.3 Le traitement massivement distribué sous Hadoop

4.3.1 Problématique

Comme nous l'avons déclaré dans le chapitre précédant, le Hadoop peut être considéré comme une solution de parallélisation des traitements. Au fait, une étape d'analyse du problème

est nécessaire afin de pouvoir le découper en sous-problèmes avant d'attribuer la résolution de ces derniers à d'autres nœuds de traitements pour être traités en parallèle.

Après cette phase, aucune communication ou collaboration n'est permise. Les traitements parallèles sont indépendants et le Hadoop n'offre aucune solution efficace pour le partage des données ou autre. Ce qui augmente considérablement les temps des calculs, particulièrement, dans systèmes temps réel dont les quantités des traitements et des données sont beaucoup plus petites. En plus, le renvoie des résultats ne peuvent être effectuer qu'après la fin des traitements parallèles. Soulignons que dans un système distribué, la communication joue un rôle très important pour l'amélioration des performances des systèmes informatiques.

L'objectif de ce travail est de proposer un nouveau modèle de traitement des méga données sous Hadoop basé sur le principe de communication inter-nœuds.

4.3.2 Contribution

Notre solution consiste à proposer un nouveau type de « processus Hadoop » qui peuvent servir à la communication inter-nœuds dont la fonction « Reduce » est en cours d'exécution sur la totalité ou une partie des nœuds participant à l'exécution de la tâche courante.

Les processus proposés doivent être plus prioritaires que les processus utilisateurs et ont la possibilité d'accéder aux résultats partiels des calculs distribués. La gestion de ces processus doit être transparente à l'utilisateur, c'est-à-dire sans l'intervention de ce dernier ou dans le cas échéant avec une intervention minime.

Les processus proposés peuvent servir également à :

- L'accélération des cycles de traitement de Hadoop,
- La synchronisation des traitements,
- La détection de terminaison des traitements,
- La détection de l'interblocage,
- Le partage d'état,
- La détection des pannes,
- Etc.

Les nœuds doivent disposer d'une file d'attente pour le stockage des processus dans le cas de réception multiple de ce type de ces derniers. A la réception d'un processus de ce type, le nœuds en question doit :

- Enregistrer le processus dans la file d'attente,
- Enregistrer l'état (registres, variables, etc.) du traitement courant s'il n'est pas encore fait,
- Servir le premier processus de la fille d'attente en appliquant la politique FIFO (Firt In First Out) par exemple.

4.3.3 Etapes de réalisation

Les étapes de réalisation de notre solution sont :

1. Préparation de l'environnement,
2. Choix du domaine d'application,
3. Développement de l'application,
4. Implémentation de notre solution,
5. Evaluation des performances.

4.3.4 Choix de l'application

Le domaine d'application choisi doit favoriser : (1) les communications entre les différents nœuds et aussi (2) le traitement itératif nécessaire pour Hadoop. Après discussion, nous avons opté pour une application de classification de la population.

4.4 Installation de Hadoop sous Windows

Comme nous l'avons mentionné dans le chapitre précédant, Hadoop est un Framework Java open source d'Apache dédié aux traitements intensif sur des volumes de données massifs en cluster. Sa mise en marche passe par quelques étapes qu'on va détailler dans ce qui suit **Source spécifiée non valide.**:

4.4.1 Etape 1 : Téléchargement.

Pour Installer Hadoop sur une machine qui utilise Windows 10 comme système d'exploitation, on a besoin de télécharger certains fichiers : le Hadoop, les fichiers de configuration ainsi que Java.

4.4.2 Etape 2 : Configurer les variables d'environnement système

Après le téléchargement, on doit extraire les différents packages puis on passe à la phase de configuration des variables d'environnement pour Hadoop et Java (voir la figure 12).

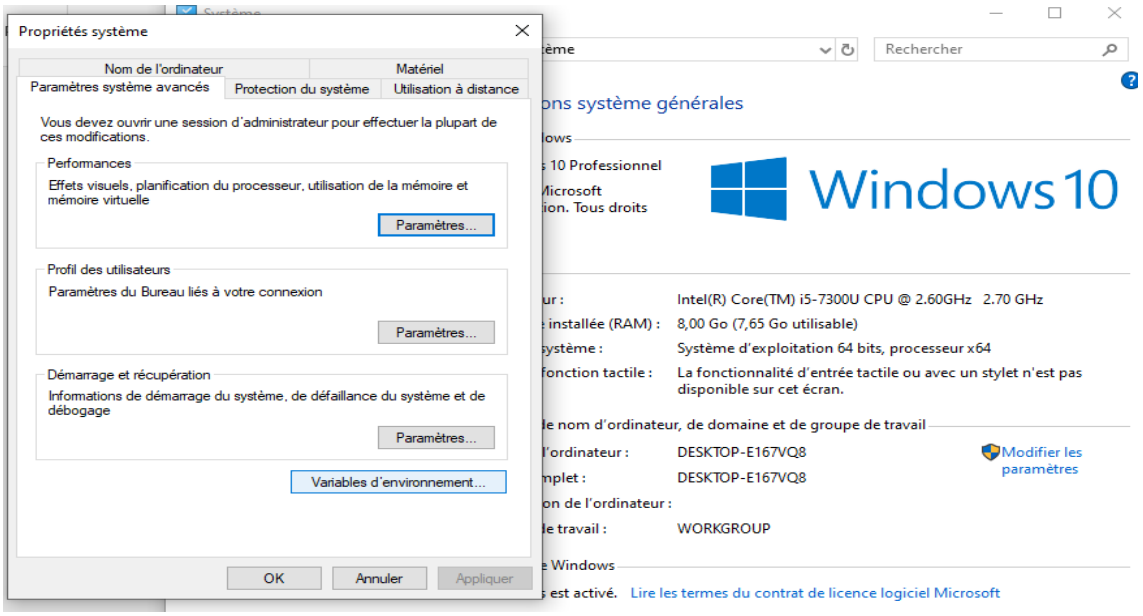

Figure 12 : La configuration des variables d'environnement système.

Ensuite, il faut choisir l'option Variable d'environnement et créer une nouvelle variable utilisateur et mettre dans le nom de la variable « HADOOP_HOME » et dans la valeur de la variable le chemin du dossier bin dans lequel vous avez extrait Hadoop (voir la figure 13).

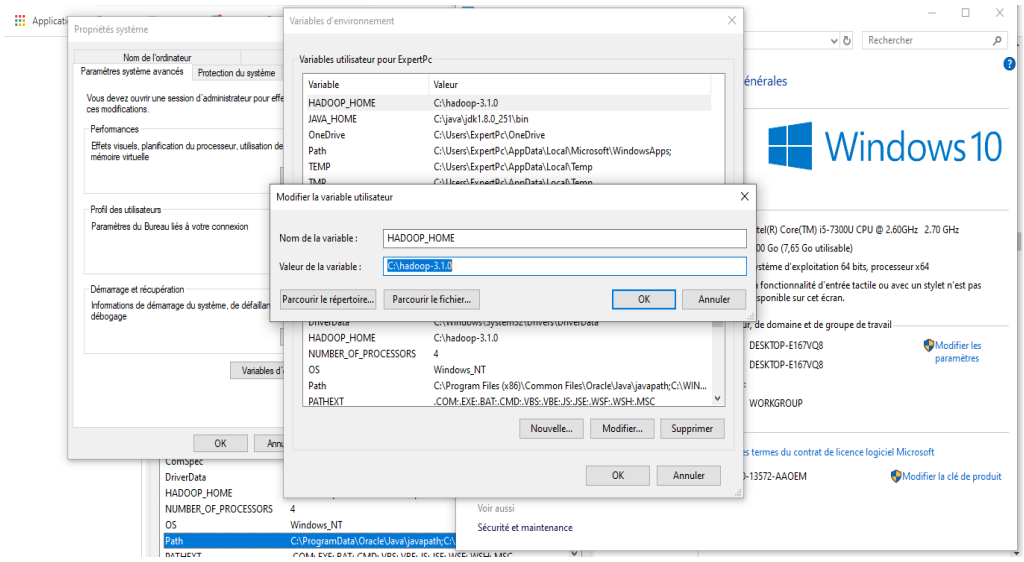

Figure 13 : La modification des variables d'environnement pour Hadoop.

De même, On crée une nouvelle variable utilisateur avec le nom de la variable JAVA_HOME et la valeur de la variable comme chemin du dossier bin dans le répertoire Java (voir la figure 14).

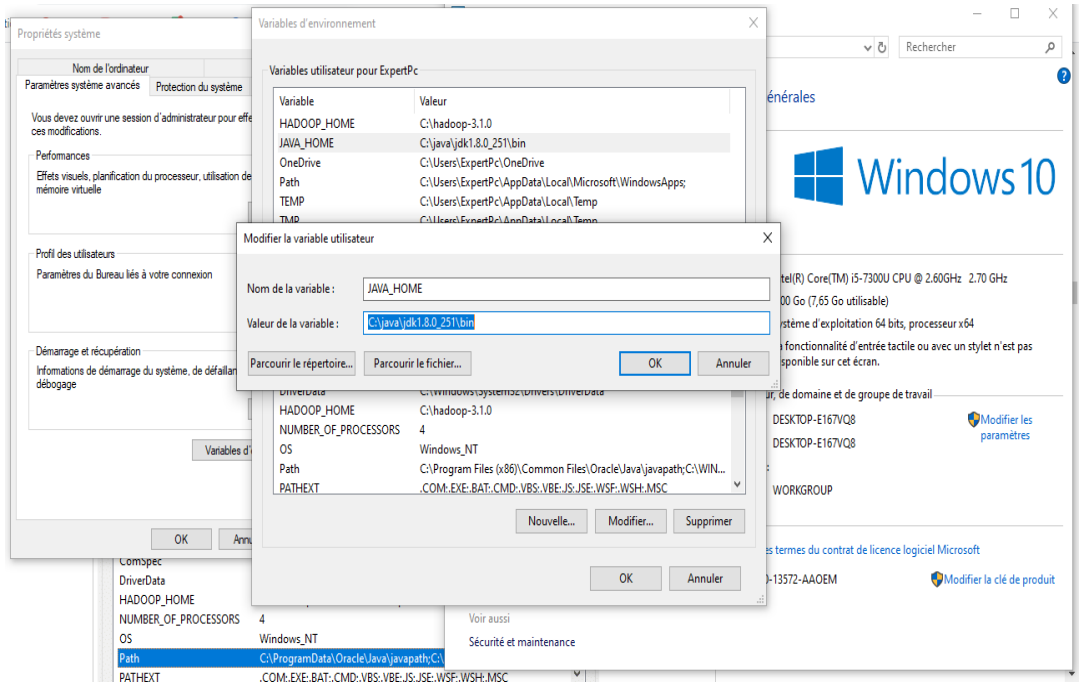

Figure 14 : La modification des variables d'environnement pour Java.

Simultanément, on doit définir le répertoire bin Hadoop et le chemin du répertoire bin Java dans le chemin de la variable système (PATH) tout en spécifiant le chemin dans la variable système (voir la figure la 15).

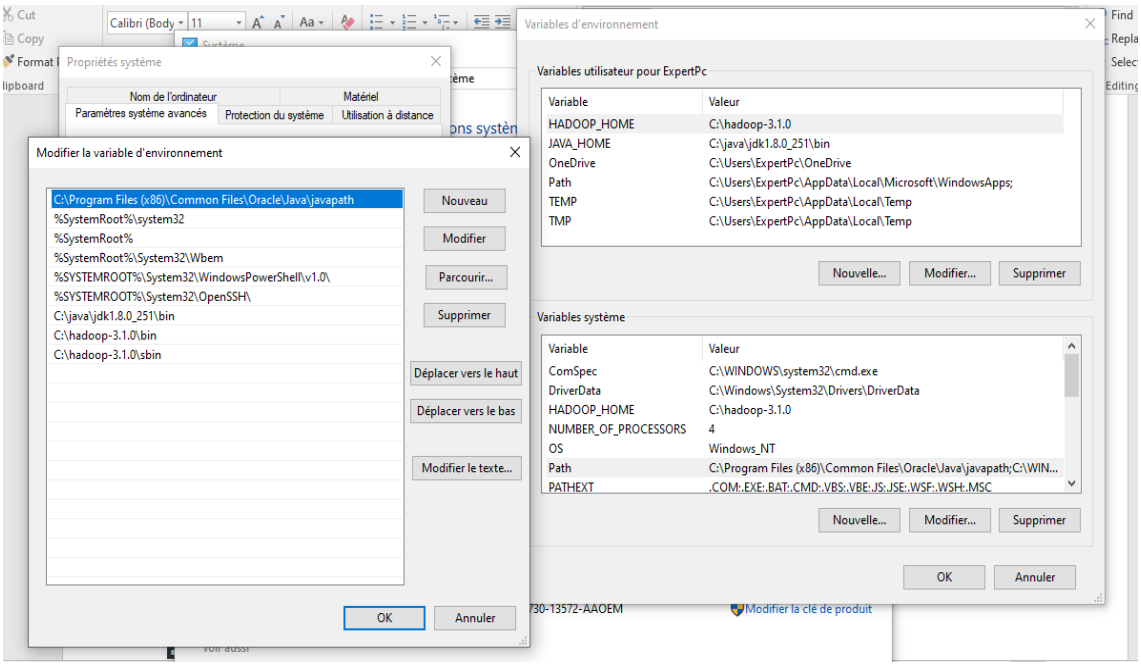

Figure 15 : La modification du chemin des variables systèmes.

4.4.3 Etape 3 : Configuration de Hadoop

Maintenant, nous devons éditer certains fichiers situés dans le répertoire « hadoop » du dossier appelé « etc » où nous avons installé le Hadoop. Les fichiers à modifier sont :

- **Le fichier core-site.xml**

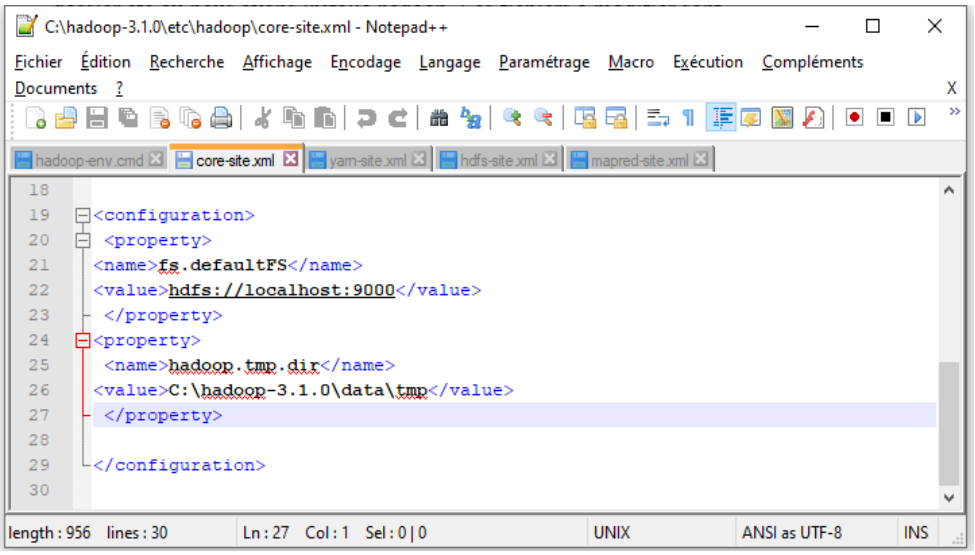

Figure 16 : Le fichier core-site.xml après modification.

Après avoir spécifie le nom du système de fichier, tous les répertoires et fichiers HDFS seront donc préfixés par hdfs://localhost :9000 (voir la figure 16).

- **Le fichier mapred-site.xml**

Ensuite, il faut configurer les paramètres spécifiques à MapReduce qui sont dans les fichiers nommé mapred-site.xml (voir la figure 17).

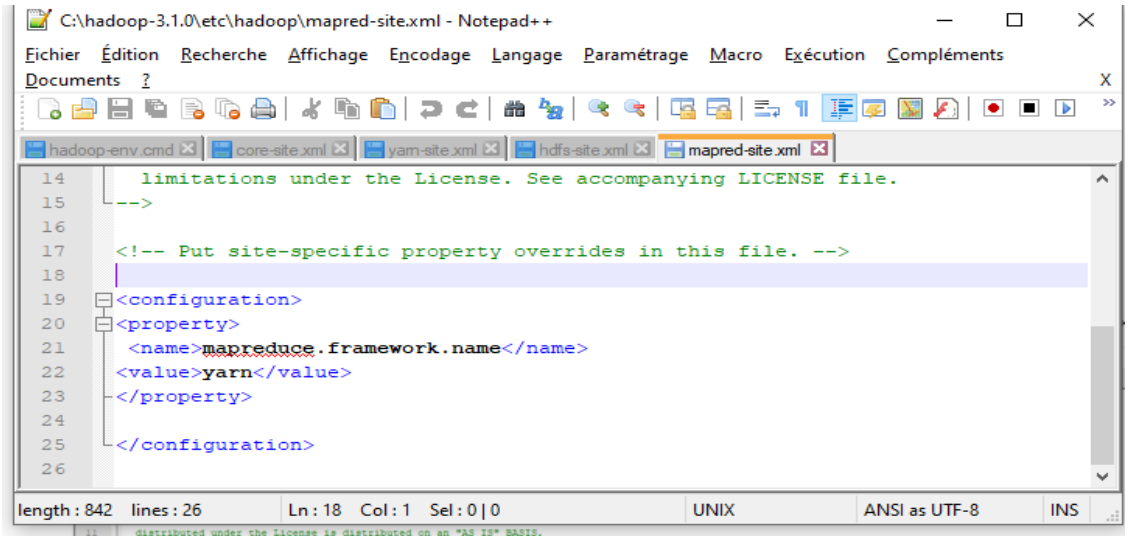

Figure 17 : Le fichier mapred-site.xml après modification

- **Le fichier hdfs-site.xml**

Nous aurons besoin de créer un dossier nommé « data » et deux sous-dossiers dedans « datanode » et « namenode » puis nous procédons à la modification du fichier hdfs-site.xml (voir la figure 18).

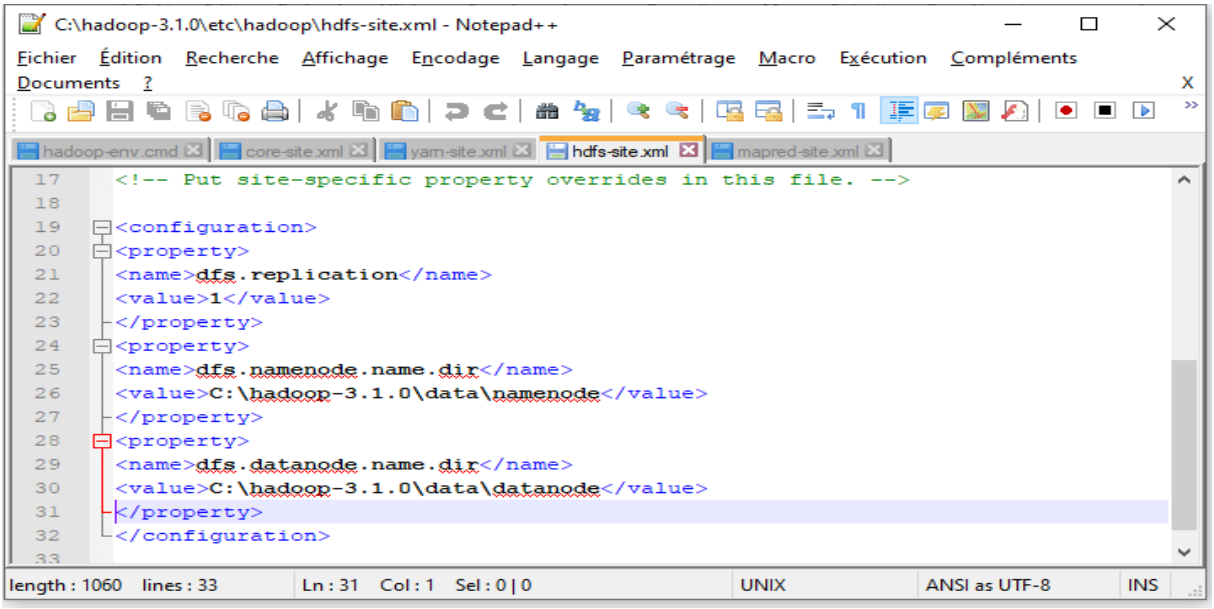

Figure 18 : Le fichier hdfs-site.xml après modification.

- **Le fichier yarn-site.xml**

Pareillement, le fichier yarn-site.xml doit être mis à jour (voir la figure 19).

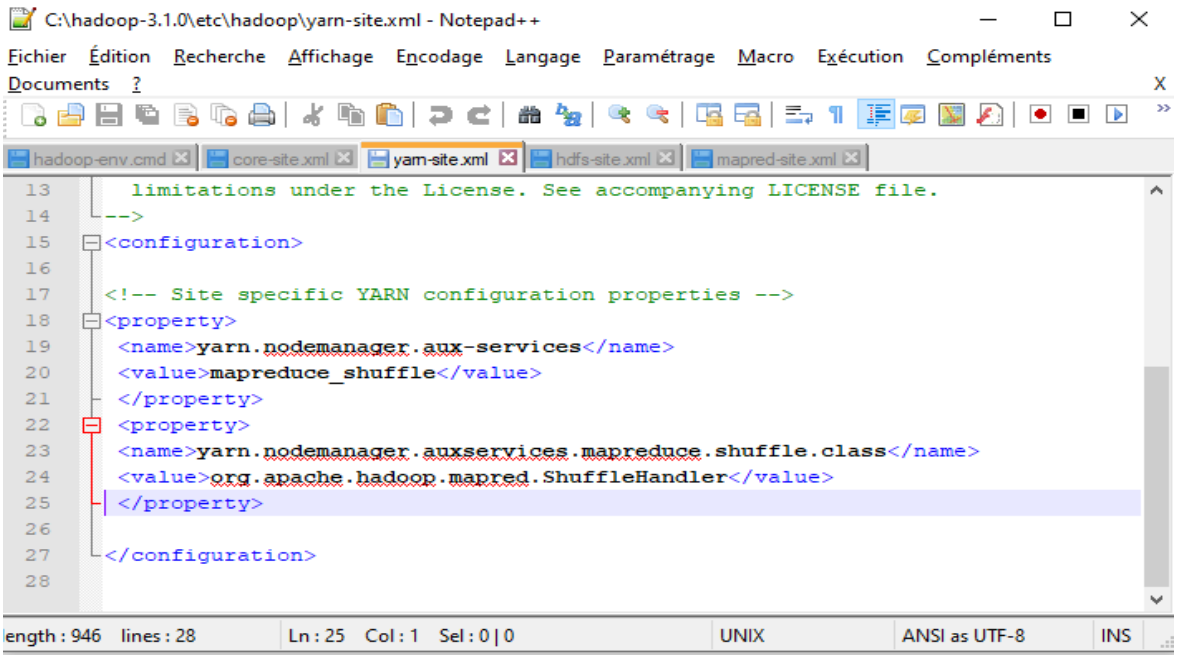

Figure 19 : Le fichier yarn-site après modification.

- **Le fichier hadoop-env.cmd**

Nous devrons, également, modifier le fichier hadoop-env.cmd et remplacer la ligne « % JAVA_HOME% » par le chemin de notre dossier d'installation du java (voir la figure 20).

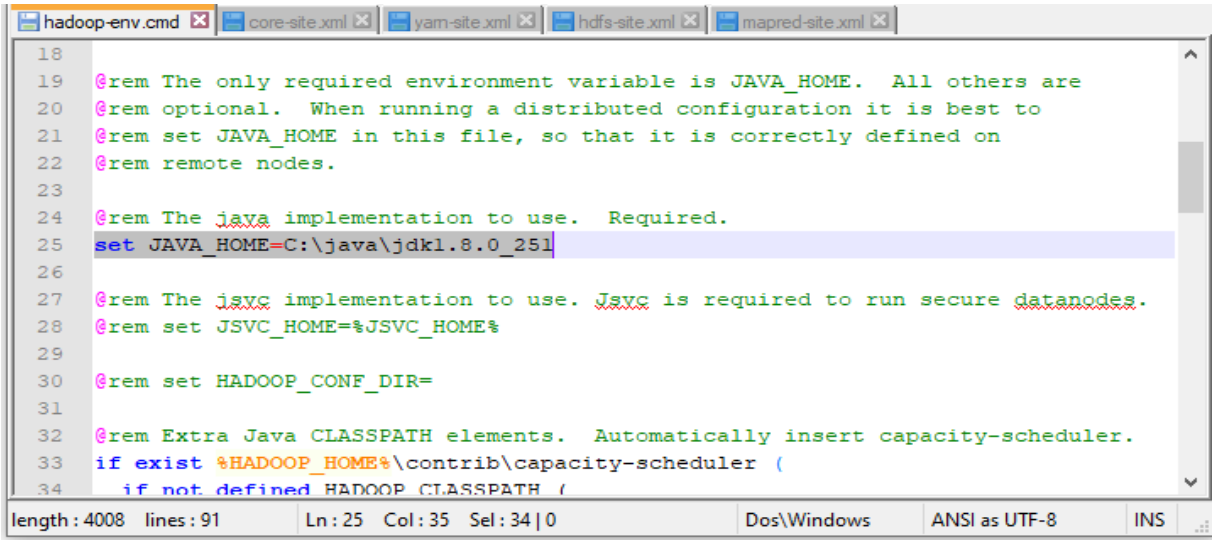

Figure 20 : Le fichier hadoop-env.cmd après modification

Enfin, nous décompressons le fichier « configuration.zip » déjà téléchargé, et nous remplaçons le dossier « bin » sous le dossier d'installation de Hadoop par celui le résultat de la décompression. Nous pouvons vérifier si le Hadoop est bien installé en tapent la commande « hadoop version » dans un nouveau terminal de commande (voir la figure 21).

Figure 21 : La vérification de l'installation de Hadoop.

4.4.4 Etape 4 : Exécution

Tout d'abord, nous ouvrons une fenêtre d'invite de commande en mode administrateur et nous tapons la commande suivante « **hdfs namenode –format »**, qui permet de formater le système de fichiers HDFS local (voir la figure 22).

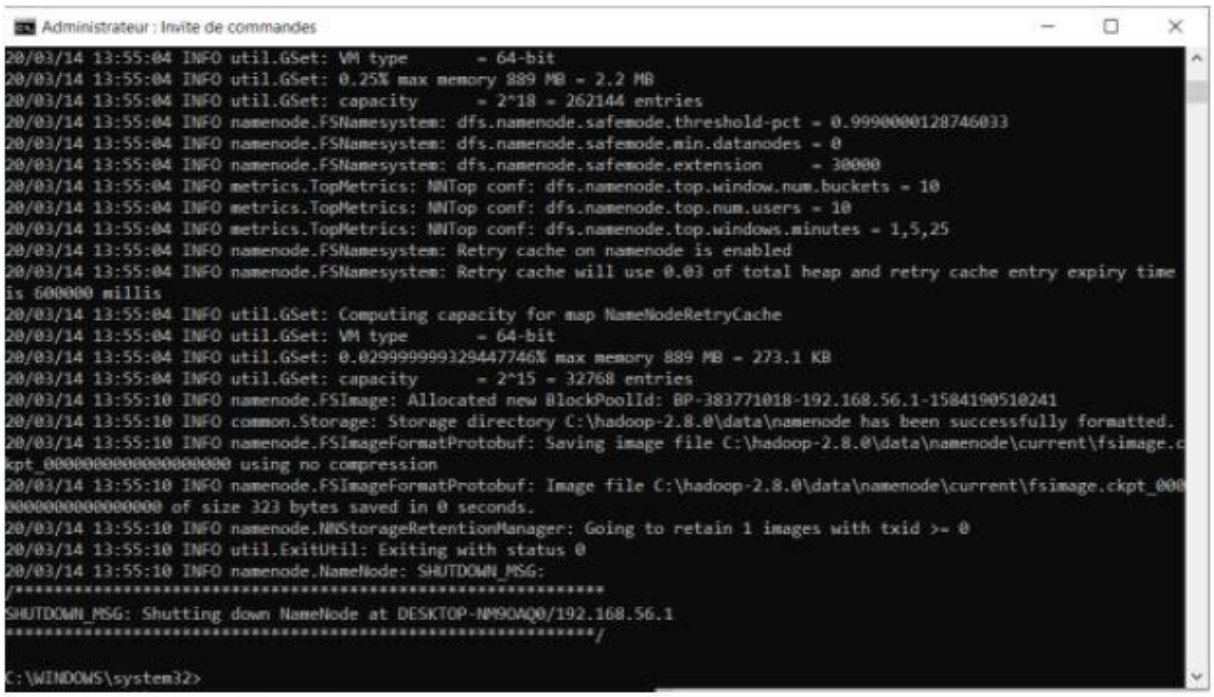

Figure 22 : le formatage du HDFS.

Enfin, nous démarrons Hadoop en tapant la commande « **start-all.cmd »** (voir la figure 23)**.**

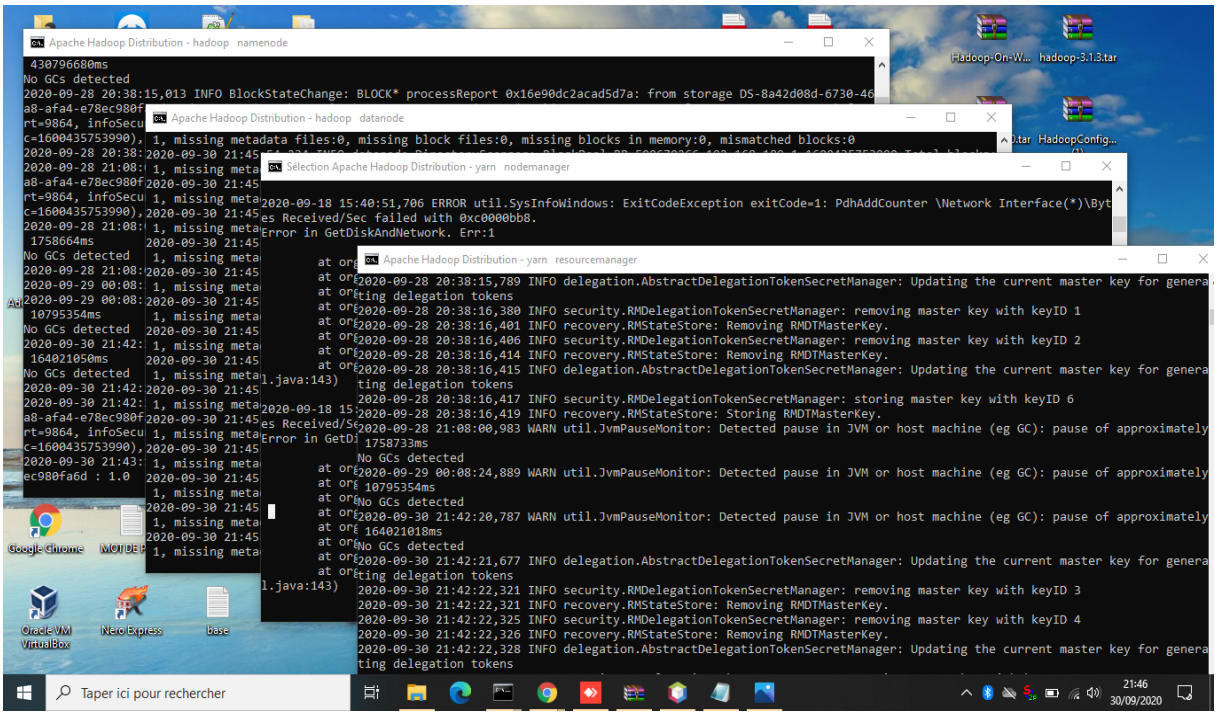

Figure 23 : Le démarrage de Hadoop.

Pour accéder aux informations sur les travaux actuels du gestionnaire de ressources, les travaux réussis et échoués, nous accédons à la page « http: // localhost: 8088 / cluster » (voir la figure 24).

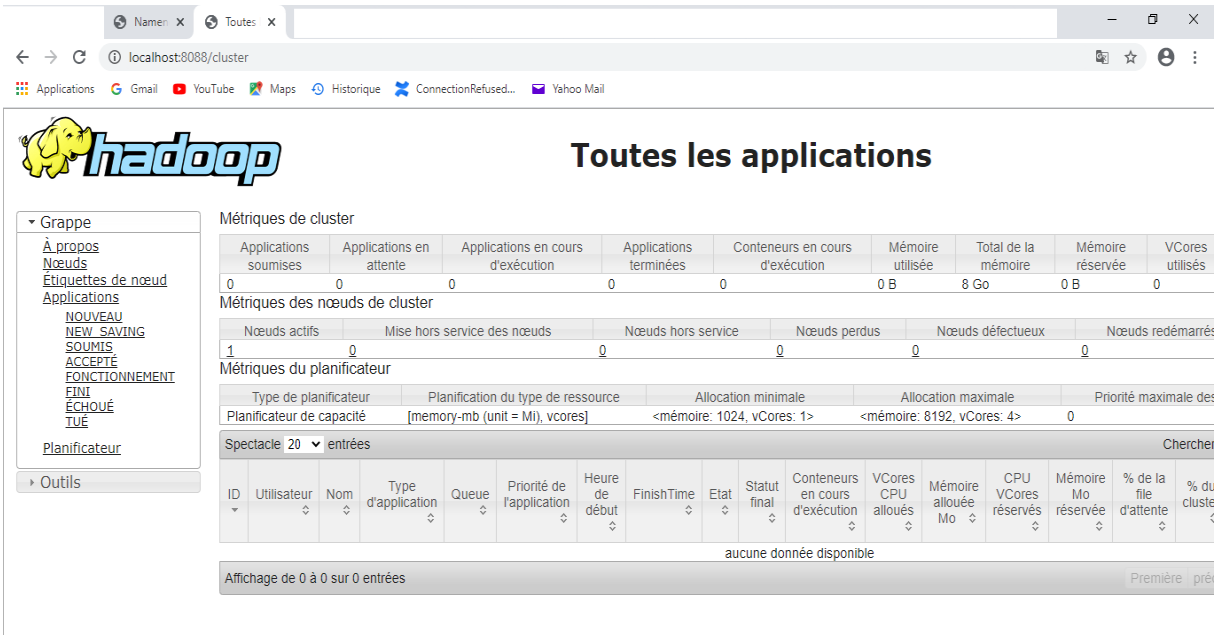

Figure 24 : Les informations de Hadoop.

Pour vérifier les détails du HDFS (voir la figure 25), nous devrons ouvrir :

http: // localhost: 9870 /

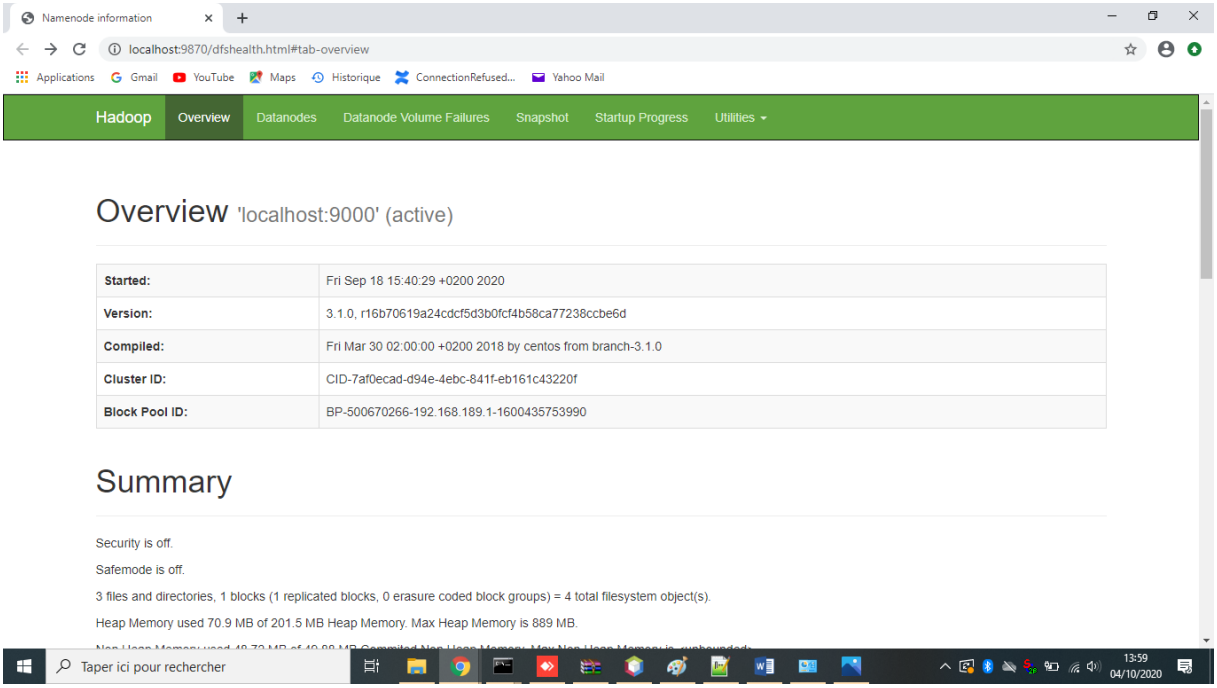

Figure 25 : Les informations du HDFS.

4.5 Exemples

Nous présentons, maintenant, quelques commandes HDFS avec les résultats de leurs exécutions sur notre machine :

- L'affichage du contenu d'un dossier à l'aide de l'instruction « **hdfs dfs –ls »** (voir la figure 26) .

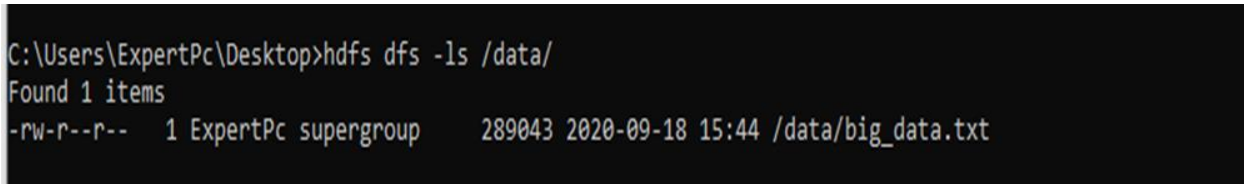

Figure 26 : l'exécution d'une commande HDFS 1.

- L'enregistrement d'un fichier texte par exemple dans un dossier au choix (dossier data dans notre cas) à l'aide de l'instruction « **hdfs dfs -copyFromLocal big-data.txt /data/** » (voir la figure 27).

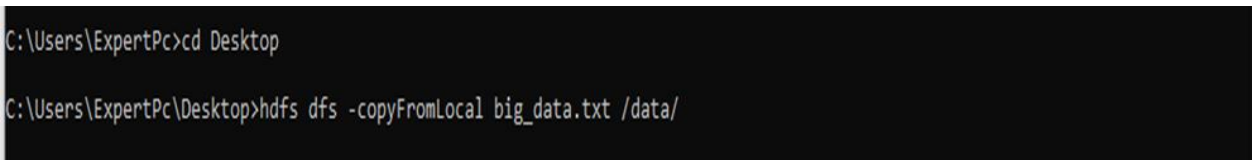

Figure 27 : l'exécution d'une commande HDFS 2.

- L'affichage du contenu d'un fichier à l'aide de l'instruction « **hdfs dfs –cat /data/big_data.txt** » (voir la figure 28).

Figure 28 : l'exécution d'une commande HDFS 3.

4.6 Implémentation

4.6.1 Description générale de l'application réalisée

La partie de l'application réalisés permet d'exécuter plusieurs traitements distribués sous Hadoop. Elle nous offre la possibilité de classifier le contenu d'un fichier (Data Set) qui contient des informations des personnes et qui est considéré comme la source de notre système. La classification peut être faite selon : (1) l'âge, (2) le gendre, ou bien par (3) région (voir la figures 29, la figure 30 et la figure 31). Les étapes d'exécution de notre application sont les suivantes :

1. L'initialisation de l'environnement,
2. L'élaboration du Data Set,
3. L'extraction des schémas,
4. Le choix du critère de classification,
5. L'élimination des valeurs nulles,
6. L'extraction des classes de classification,
7. Le calcul des occurrences dans chaque classe,
8. Le retour des résultats partiels.
9. Le calcul du résultat final.

4.6.2 Quelques interfaces de l'application

- **Classification selon l'âge :**

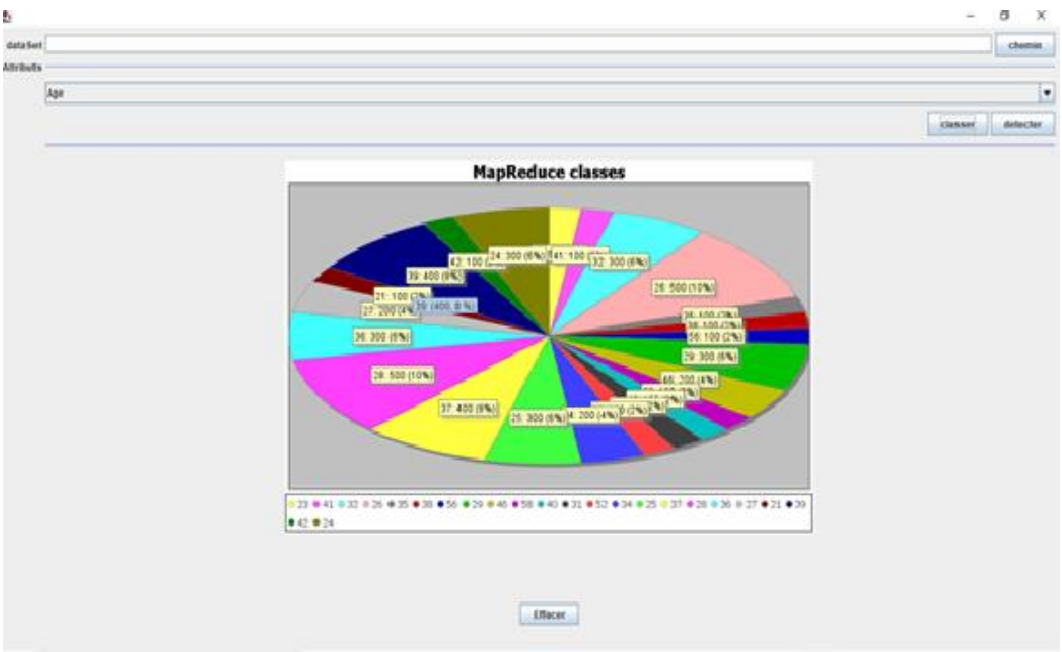

Figure 29 : classification selon l'âge des personnes.

- **Classification selon le gendre :**

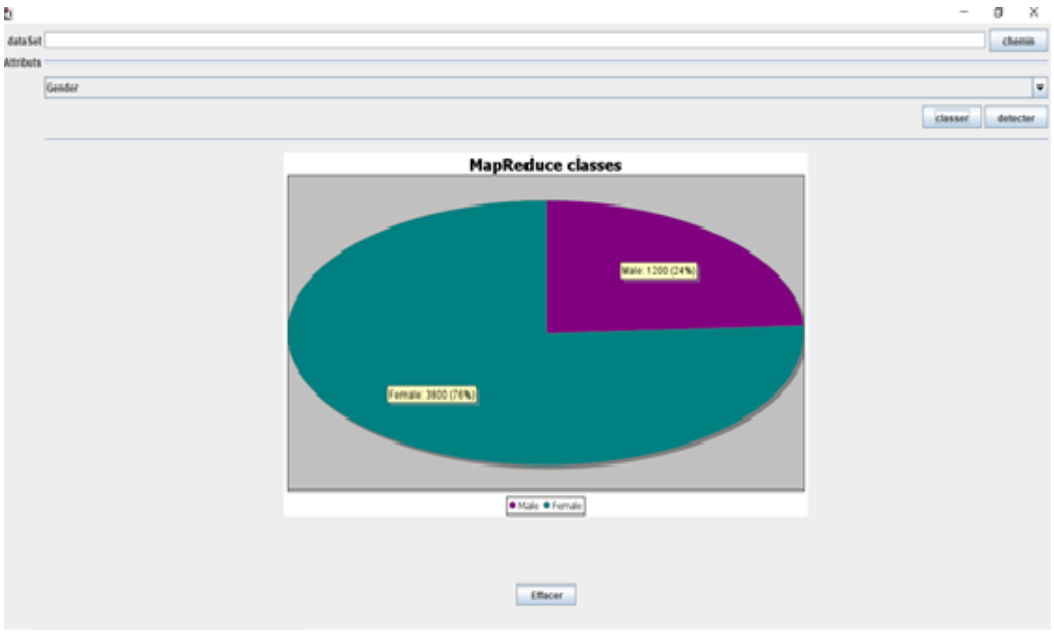

Figure 30 : classification selon le gendre.

- **Classification selon la région :**

Figure 31 : classification selon la région.

4.7 Discussion

L'application réalisée nous permet de lancer plusieurs types de traitements distribués sous Hadoop. Pour chaque cycle de traitement, le Hadoop envoie le même traitement à

l'ensemble des nœuds participants à cette exécution à fin d'être exécutés sur des ensembles de données différents. C'est le Hadoop qui s'occupe de la distribution des données ; c'est-à-dire la division de notre Data set en plusieurs blocs et ensuite l'envoie de ces derniers aux nœuds en question. Bien sûr, cette distribution est transparente à l'utilisateur. Soulignons que, nous n'avons pas pu atteindre tous les objectifs visés par ce travail. Ce retard est dû aux nombreuses causes dont la cause principale est la difficulté de réalisation de notre proposition sous un tel outil qui est encore sous-développement.

4.8 Conclusion

Dans ce chapitre, nous avons présenté les principales étapes de réalisation et d'implémentation de notre travail. Tout en commençant par la description de l'environnement de développements, les étapes d'installation et l'exécution de Hadoop, l'exécution de quelques instructions HDFS et les résultats obtenus.

Conclusion générale et perspectives

Notre projet a fait l'objet d'une étude des nouvelles technologies du Big Data. Dans un autre coté Nous avons abordé les principes des méga données, ces caractéristiques, les différents domaines dans lesquels elles sont utilisées, ainsi que les innovations des technologies de données massives.

Cependant, parmi les systèmes permettant un traitement rapide et efficace de ce type des données, nous trouvons les systèmes distribués. Ces derniers proposent une solution pour améliorer la vélocité des systèmes d'informations des entreprises, ainsi que leur disponibilité. Nous avons choisi de travailler avec le Hadoop, qui est une plateforme orientée objet distribuée haute disponibilité et qui traite les fichiers de grands blocs. Notons que de nouveaux systèmes spécifiques ont été créés permettant d'améliorer les performances de Hadoop, notamment dans les milieux hétérogènes, tant en termes de vitesse de traitement, qu'en termes de consommation électrique.

Dans ce travail, nous avons proposé un nouveau modèle de traitement des méga données dans un environnement Hadoop. Ce modèle est basé sur le principe de communication des différents nœuds. La partie applicative nous a permet de se familiariser avec l'environnement Hadoop et la maitrise de nouvelles technologies à savoir le MapReduce et le HDFS.

Perspectives :

Concernant les futurs travaux, nous nous intéressons à l'implémentation de notre proposition et la comparaison de ses performances à celles de l'application réalisé.

Bibliographie

[1] 26Academy, «BigData:un peu d'histoire,» 2020.

[2] «Qu'est-ce que le Big Data ?,» [En ligne]. Available: www.oracle.com . [Accès le 30 09 2020].

[3] B. L, «Le BgData,» Le magazine I.A, 12 avril 2017.

[4] Y. grandmontagne, «Les5V du Big Data,» 13/03/2014.

[5] J.-S. Vayre, «Les tableaux de bord sur données massives, pour un nouveau management de l'innovation,» HAL, 4 Jan 2016.

[6] Microsoft, «Architectures de Big Data,» 12/02/2018.

[7] «Les acteurs du marché Big Data,» 2020. [En ligne].

[8] *LIVRE BLANC du big data au big business*, 2014.

[9] J. T. MUDIKOLELE, «Mise en oeuvre d'un système distribué pour l'identification et le suivi du casier judiciaire,» 2016. [En ligne].

[10] B. M. Chaouki, *Introduction aux systèmes répartis*, Université de Biskra.

[11] N. BENHAMIDA, *cours système distribué*, Guelma, Algerie: université 08 Mai 1945 de Guelma., 2019.

[12] A. DHRAIEF, «Architectures des systèmes répartis,» 2019. [En ligne].

[13] J. T. MUDIKOLELE, «Memoire en ligne,» 2016. [En ligne].

[14] N. L. Thanh, «Principe des systèmes d'exploitation,» Faculté de sciences -UNSA, février 1995.

[15] N. Abdennadher, chez *Systèmes distribués* , Genève, HEPIA, 2015/2016.

[16] J. CLET-ORTEGA, *Exploitation efficace des architectures parallèles de type grappes de NUMA à l'aide de modèles hybrides de programmation,* École doctorale de Mathématiques et Informatique, l'Université de Bordeaux 1, 2012.

[17] N. BENHAMIDA, *Architectures Parallèles*, Guelma, Algérie.: Université 08 Mai 1945 de Guelma.

[18] T. Wbite, Hadoop the definitive guide, O'REILLY.

[19] [En ligne]. Available: www.lebigdata.fr. [Accès le 10 12 2019].

[20] T. Gunarathne, Hadoop MapReduce v2 Cookbook Second Edition, Packt Publishing Ltd, 2015.

[21] BastienL, «MapReduce – Présentation générale,» 13 Octobre 2017. [En ligne].

[22] O. Mebark et B. O. Mohammed, *L'utilisation d'une base NoSQL (HBASE) dans un milieu distribué (Hadoop), Mémoire de fin d'études,* Université Ahmed Draia - Adrar, 2017.

[23] D. Vohra, Practical Hadoop Ecosystem: A Definitive Guide to Hadoop-Related Frameworks and Tools, Apress Media, 2016.

[24] BAstienL, «Le big data,» 3 Octobre 2017. [En ligne]. Available: https://www.lebigdata.fr/hdfs-fonctionnement-avantages.

[25] M. COOPMANS, *SAS & HADOOP*, 2014.

[26] H. bouzidi, «Les 5 grands enjeux du Big Data,» *Big Data*, 12/07/2016.

[27] M. ANF, «Structures de données complexes,» Univ. Grenoble Alpes, 2018.

[28] L. JOLIA-FERRIER, Big Data-Concepts et mise en oeuvre de Hadoop, ENI, 2014.

[29] [En ligne]. Available: https://www.26academy.com/le-big-data-un-peu-dhistoire/.