Université Alger 1
Faculté des mathématiques et d'informatique

Support de cours

Master 2 informatque

Spécialité : Informatique

"Big Data Analytics."

Présentée Par
Dr. MEZZOUDJ Saliha

Année universitaire 2021-2022

Table des matières

Table des figures

0.1 Introduction

Actuellement, des trillions d'octets de données (Big Data) se génèrent par jour. Ces données proviennent de différentes sources telles que : les capteurs dédiés à collecter les informations climatiques, les messages sur les réseaux sociaux, les images numériques et des vidéos publiées en ligne.

Les Big Data se caractérisent par leur volumétrie (données massives) ; ils sont connus aussi par leur variété en termes de formats et de nouvelles structures, ainsi, qu'une exigence en termes de rapidité dans le traitement. À ce jour, il n'existe aucun logiciel capable de gérer plusieurs types et formes de données qui n'arrête pas d'accroitre très rapidement. De ce fait , la problématique du Big Data fait partie de notre vie quotidienne, et nécessite des solutions plus avancées pour gérer cette masse de données dans des délais très réduits.

Le calcul distribué concerne le traitement de grandes quantités de données. Ce traitement ne peut être réalisé par les paradigmes classiques de traitement de données, mais il nécessite l'utilisation des plateformes distribuées. Dans la littérature, il existe plusieurs solutions, pour l'implémentation de ce paradigme. Parmi ces solutions on trouve l'exemple Google qui a développé un modèle de programmation très fiable pour le traitement de Big Data : c'est le modèle MapReduce. Ce modèle est implémenté sur plusieurs plateformes comme la plateforme Hadoop. Cependant, cette dernière souffre de problèmes de la latence considérée comme le principal motif de développement d'une nouvelle alternative pour améliorer les performances du traitement, c'est la plateforme Spark qui est plus puissante, plus souple et rapide que Hadoop MapReduce.

0.2 Les notions de base des big Data

0.2.1 Définition

Le Big Data désigne un très grand volume de données souvent hétérogènes qui ont plusieurs formes et formats (texte, données de capteurs, son, vidéo, données sur le parcours, fichiers journaux, etc.), et comprenant des formats hétérogènes : données structurées, non structurées et semi-structurées. Le Big Data a une nature complexe qui nécessite des technologies puissantes et des algorithmes avancés pour son traitement et stockage. Ainsi, il ne peut être traité en utilisant des outils tels que les SGBD traditionnels [4]. La plupart des scientifiques et experts des données définissent le Big Data avec le concept des 3V comme suit [4] :

— Vélocité : Les données sont générées rapidement et doivent être traitées rapidement pour extraire des informations utiles et des informations pertinentes.

Par exemple, Wallmart (une chaîne internationale de détaillants à prix réduits) génère plus de 2,5 petabyte(PB) de données toutes les heures à partir des transactions de ses clients. YouTube est un autre bon exemple qui illustre la vitesse rapide du Big Data.

— Variété : Les données volumineuses sont générées à partir de diverses sources distribuées dans plusieurs formats (vidéos, documents, commentaires, journaux, par exemple). Les grands ensembles de données comprennent des données structurées et non structurées, publiques ou privées, locales ou distantes, partagées ou confidentielles, complètes ou incomplètes, etc.

— Volume : il représente la quantité de données générées, stockées et exploitées. Le volume des données stockées aujourd'hui est en pleine explosion il est presque de 800.000 Péta-octets, Twitter générer plus de 7 téraoctets chaque jour de données, Facebook génère plus de 10 téraoctets et le volume de données dans 2020 peut atteindre 40 zêta-octets [5].

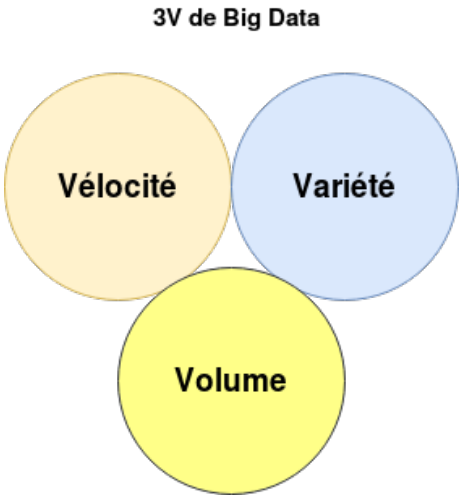

FIGURE 1 – Le modèle de Big Data 3V

Par la suite, les trois dimensions initiales sont élargies par deux autres dimensions des données Big Data (on parle aussi des « 5 V du Big Data») :

— Véracité : La véracité (ou validité) des données correspond à la fiabilité et l'exactitude des données, et la confiance que ces Big Data inspirent aux décideurs. Si les utilisateurs de ces données doutent de leur qualité ou de leur pertinence, il devient difficile d'y investir davantage.

— Valeur : Ce dernier V joue un rôle primordial dans les Big Data, la démarche Big Data n'a de sens que pour atteindre des objectifs stratégiques de création de valeurs pour les clients et pour les entreprises dans tous les domaines.

Une des raisons de l'apparition du concept de Big Data est le besoin de réaliser le défi technique qui consiste à traiter de grands volumes d'information de plusieurs types

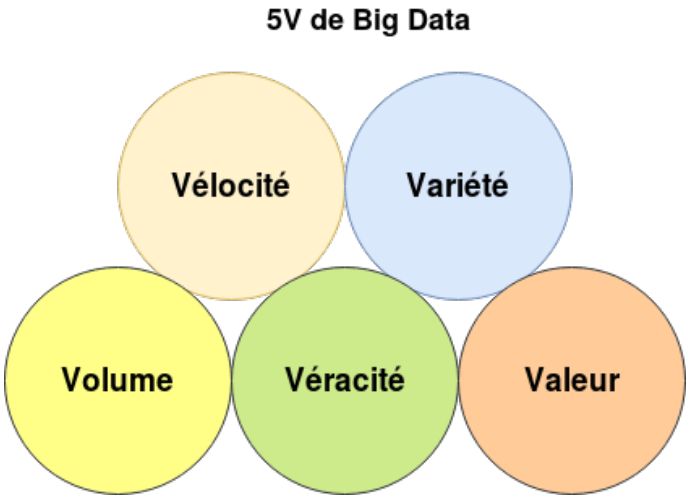

FIGURE 2 – Le modèle de Big Data 5V.

(structurée, semi structuré et non structurée) générée à grande vitesse. Le Big Data s'appuie sur quatre sources de données [6] :

— Les logs (journaux de connexion) issus du trafic sur le site officiel de l'entreprise : Ces sources de données, sont les chemins pris par les visiteurs pour parvenir sur le site : moteurs de recherche, annuaires, rebonds depuis d'autres sites, etc. Les entreprises d'aujourd'hui disposent d'une vitrine sur le Web au travers de son site officiel. Ce dernier génère du trafic qu'il est indispensable d'analyser, ainsi ces entreprises disposent des trackers sur les différentes pages afin de mesurer les chemins de navigation, ou encore les temps passés sur chaque page,.etc. Citons parmi les solutions d'analyse les plus connues : Google Analytics, Adobe Omniture, Coremetics.

— Les issus des médias sociaux «insights» : Une approche complémentaire, consiste à recueillir les commentaires aux publications et à y appliquer des algorithmes d'analyse de sentiment. Citons quelques pistes pour suivre nos différents comptes : Hootsuite, Radian6 ou encore les API mises à disposition et interrogées avec le complément Power Query pour Excel, IRaMuTeQ pour l'analyse de données textuelles.

— Les données comportementales (third party data) sont toutes des données récoltées sur les internautes via des formulaires ou des cookies. Au-delà des classiques informations d'identité (sexe, âge, CSP, etc), il est maintenant beaucoup plus efficace de mesurer les comportements (navigation, configuration matérielle, temps passé sur les pages, etc). Pour cela, il existe des acteurs spécialisés du Web qui nous aident à collecter de l'information sur nos clients ou prospects et à améliorer ainsi les campagnes de communication. Quelques acteurs du domaine de la third party data : Bluekai, Exelate, Weborama, Datalogix, etc.

— Les données ouvertes et réutilisables «L'open data» sont toutes les données ou-

vertes et réutilisables, L'open data permet de mettre en ligne les données ouvertes, de fiabiliser les données et de les rendre réutilisables et exploitables, où l'ouverture consiste à rendre une donnée publique : libre de droits, téléchargeable, réutilisable et gratuite. L'ouverture ne concerne pas les données à caractère privé, les informations sensibles et de sécurité, les documents protégés par des droits d'auteur, etc. Les données ouvertes et réutilisables ne sont pas encore légion même si une mission gouvernementale et très active sur le sujet manque de complétude, niveau de détail insuffisant, relative ancienneté sont les défauts actuels de nombreux jeux de données. Toutefois, c'est un champ d'investigation qu'il ne faut pas négliger, ne serait-ce que par son faible coût (celui du temps passé à chercher!) et son développement inéluctable.

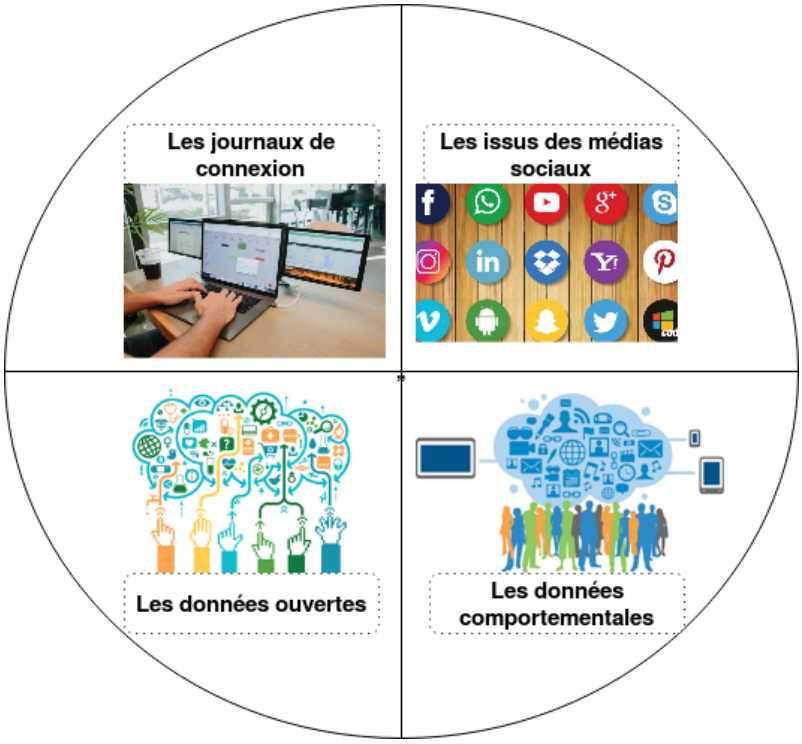

FIGURE 3 – Les quatre sources de Big Data.

0.2.2 Les technologies de traitement et de stockage de Big Data

Les Big Data requièrent de redéfinir les systèmes de stockage et de traitement de données, qui peuvent supporter ce volume de données. En effet, plusieurs technologies ont été proposées afin de représenter ces données, ces technologies prennent au moins un axe parmi les deux soit l'amélioration des capacités de stockage, soit l'amélioration de la puissance de calcul [4] :

— Amélioration de la puissance de calcul : le but de ces techniques est de permettre de faire le traitement sur un grand ensemble de données, avec un coût considérable et d'améliorer les performances de l'exécution comme le temps de traitement et la tolérance aux pannes. Avant l'‘apparition de la plateforme Hadoop on trouve plusieurs technologies comme le Could Computing, les architectures massivement parallèles MPP et les technologies In-Memory.

— Amélioration des capacités de stockage : l'amélioration du stockage vers des systèmes distribués, où un même fichier peut être réparti sur plusieurs disques durs, cela permet d'augmenter les volumes de stockage en utilisant un matériel de base. Ces technologies de stockage évoluent toujours pour offrir des accès plus rapides aux données comme le NoSQL, HDFS de la plateforme Hadoop, HBase, le Cloud Computing,etc.

0.3 MapReduce

0.3.1 Pourquoi MapReduce ?

Les systèmes d'entreprise traditionnels disposent normalement d'un serveur centralisé pour stocker et traiter les données. Le modèle traditionnel n'est certainement pas adapté au traitement de gros volumes de données évolutives et ne peut pas être géré par des serveurs de base de données standard. De plus, le système centralisé crée trop de goulot d'étranglement lors du traitement simultané de plusieurs fichiers. Google a résolu ce problème du goulot d'étranglement à l'aide de modèle MapReduce.

0.3.2 Définition de modèle MapReduce :

Il a été conçu dans les années 2000 par les ingénieurs de Google. C'est un modèle de programmation conçu pour traiter plusieurs téraoctets de données sur des milliers de nœuds de calcul dans un cluster [7]. MapReduce peut traiter des téraoctets et des pétaoctets de données plus rapidement et efficacement. Par conséquent, sa popularité a connu une croissance rapide pour diverses marques d'entreprises beaucoup de champs. Il fournit une plateforme très efficace pour l'exécution parallèle des applications, l'allocation de données dans des systèmes de bases de données distribuées, et communications réseau à tolérance de pannes [8]. L'objectif principal de MapReduce est de faciliter la parallélisation des données, leur distribution et l'équilibrage de la charge dans une bibliothèque simple [7].

0.3.3 L'architecture de modèle MapReduce

Google a créé MapReduce pour traiter de grandes quantités des données non structurées ou semi-structurées, tels que les documents et journaux de demandes de pages Web, sur de grands clusters de noeuds. Il a produit différents types de données telles que des indices inversés ou fréquences d'accès aux URL [9]. Le MapReduce comporte trois parties principales, y compris Master, la fonction de Map et de reduce. Un exemple de ce flux de données est présenté à la Fig. 4.

Le Master est responsable de la gestion des fonctions Map et Reduce et de la mise à leur disposition des données et des procédures, il organise la communication entre les mappeurs et les réducteurs. La fonction map est appliquée à chaque enregistrement d'entrée et produit une liste d'enregistrements intermédiaires. La fonction Réduire (également appelée Réducteur) est appliquée à chaque groupe d'enregistrements intermédiaires avec la même clé et génère une valeur. Par conséquent, le processus de MapReduce inclut les étapes suivantes :

— Les données d'entrée sont divisées en enregistrements.
— Les fonctions de Map traitent ces données et produisent des paires clé/valeur pour chaque enregistrement.
— Toutes les paires clé/valeur issues par la fonction Map sont fusionnées ensemble et regroupées par une clé, puis elles sont triées.
— Les résultats intermédiaires sont transmis à la fonction de réduction (Reduce), qui va produire le résultat final [10].

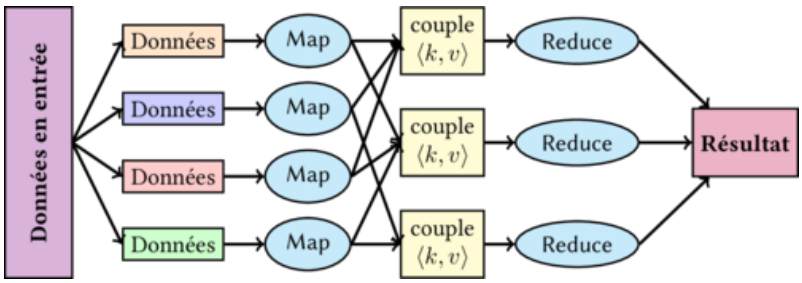

FIGURE 4 – Un exemple de flux de données dans l'architecture MapReduce [1]

0.3.4 Les applications de MapReduce

L'application MapReduce facilite l'exécution de nombreuses applications parallèles de données . MapReduce est le facteur principal dans de nombreuses applications importantes et peut améliorer le parallélisme du système. Il reçoit une attention considérable, pour les applications gourmandes en données et en temps de calculs sur

des clusters de machines. Il est utilisé comme un outil de calcul distribué, efficace pour résoudre des problèmes variables, tels que la recherche, le regroupement, l'analyse de journaux, différents types d'opérations de jointure, la multiplication de matrices, la correspondance de modèles et l'analyse de réseaux sociaux. Il permet aux chercheurs d'étudier dans différents domaines [9].

MapReduce est utilisé dans de nombreuses applications de Big Data tels que [9] : l'exploration de messages courts, les algorithmes génétiques, k-means, algorithme de clustering, fragment d'ADN, système de transport intelligent, applications scientifiques dans le domaine de la santé, systèmes de classification à base de règles floues, environnements hétérogènes, la machine d'apprentissage extrême, forêt aléatoire, l'énergie proportionnelle, capteur mobile de données, web sémantique, etc.

0.4 Les plateForme de traitement des Big Data

0.4.1 La plateforme hadoop pour le calcul distribué de Big Data

Tout d'abord, Hadoop est un Framework libre, écrit en java, créé et distribué par la fondation Apache, et destiné au traitement de données volumineuses (de l'ordre des péta-octets et plus) ainsi qu'à leur gestion intensive. Inspirée de plusieurs publications techniques rédigées par le géant Google, son objectif est de fournir un système de stockage et de traitement de données distribué, évolutif et extensible. Il permet de traiter un grand nombre de types de données (y compris les données non structurées). On dit qu'il est organisé sur un mode non-relationnel, il est plus général que NoSQL, on peut par exemple stocker les données avec deux types de systèmes HDFS (Hadoop Distributed File System) et HBase qui forment un système de gestion de bases de données orientées, colonnes projetées sur des serveurs distribués en clusters [3].

Hadoop parallélise le traitement des données à travers de nombreux nœuds faisant partie d'une grappe d'ordinateurs (clusters), ce qui permet d'accélérer les calculs et masquer la latence des opérations d'entrées et de sorties. Hadoop contient un système de fichiers distribué fiable qui garantit la tolérance aux pannes grâce à la réplication des données.

Hadoop contient deux composants de base, HDFS et MapReduce. Les deux sont liés au calcul distribué comme suit :

— HDFS est le responsable de stockage des données et MapReduce est le cœur de Hadoop qui effectue le traitement parallèle grâce aux deux fonctions Map et Reduce.

— Au niveau architectural, Hadoop contient deux types de nœuds (fig.5).

— Les serveurs ou bien les maitres : le NameNode, le NameNode secondaire et le JobTracker.

— Les esclaves qui sont distribués dans le cluster : le DataNode et le TaskTracker.

— L'exécution des tâches se fait par les TaskTrackers qui sont déployés sur chaque machine selon les instructions du JobTracker.

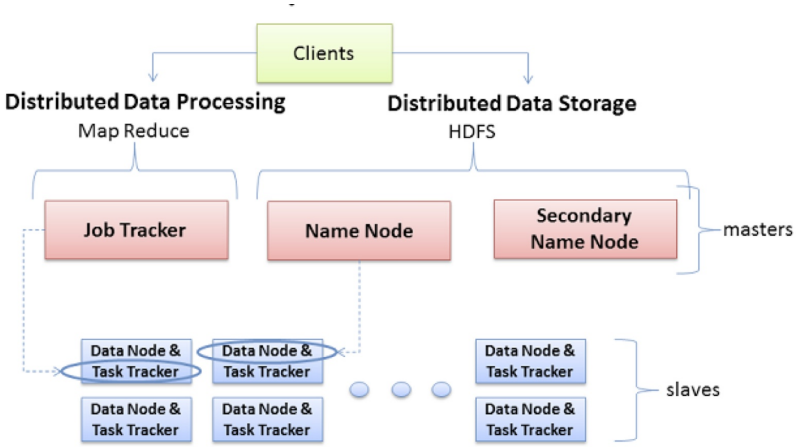

FIGURE 5 – L'architecture maître/esclave de la plateforme Hadoop [1]

0.4.2 La plateforme Spark pour le calcul distribué de Big Data

0.4.3 Motivation de Spark

Depuis sa création, Hadoop est devenu une technologie importante pour le Big Data. Une des principales raisons de ce succès est sa capacité à gérer d'énormes quantités de données quel que soit leur type (structuré, semi structuré, non structuré). Toutefois, les utilisateurs ont été systématiquement plaignants du problème de la latence élevée avec Hadoop MapReduce indiquant que la réponse en mode batch pour toutes ces applications en temps réel est très douloureuse quand il s'agit de données de traitement et d'analyse.

0.4.4 Historique de Spark

Spark est un cluster de calcul rapide développé par les contributions de près de 250 développeurs de 50 entreprises AMPLab de l'Université de Berkeley. Spark a commencé en 2009 comme un projet de recherche dans le Berkeley Lab RAD, qui deviendra plus tard l'AMPLab. Les chercheurs en laboratoire avaient déjà travaillé sur Hadoop MapReduce, et ont observé que MapReduce était inefficace pour des emplois informatiques itératifs et interactifs. Ainsi, depuis le début, Spark a été conçu pour être rapide pour les requêtes interactives et les algorithmes itératifs, apportant des idées comme le support de stockage en mémoire et la récupération efficace de fautes.

Les documents de recherche ont été publiés à propos de Spark à des conférences universitaires et peu de temps après sa création en 2009, il était déjà de 10 à 100 fois plus vite que MapReduce pour certains emplois.

Certains des premiers utilisateurs de Spark étaient d'autres groupes à l'intérieur de l'Université de Berkeley, y compris des chercheurs, tel que le projet mobile du Millénaire, qui a utilisé Spark pour surveiller et prévoir les embouteillages dans la baie de San Francisco Machine Learning. Dans un temps très court, cependant, de nombreuses organisations externes ont commencé à utiliser Spark, et aujourd'hui, plus de 50 organisations listent eux-mêmes sur la Page Spark PoweredBy, et des dizaines parlent de leurs cas d'utilisation lors d'événements.

En 2011, l'AMPLab a commencé à développer des composants de haut niveau sur Spark, comme Shark et Spark streaming. Ceux-ci et d'autres composants sont parfois appelés comme Berkeley Data Analytics Stack (ODB). Spark a été en open source en Mars 2010, et il a été transféré à Apache Software Fondation en juin 2013, où il est maintenant un projet de haut niveau [11].

0.4.5 Définition

Apache Spark est un Framework open source de traitement, il est construit autour de la vitesse, la facilité d'utilisation et capacité de gérer des grands ensembles de données qui sont de nature diverse (données de texte, données de graphes, etc.), Spark étend le modèle MapReduce pour soutenir efficacement plusieurs types de calculs, y compris le traitement itératif, les requêtes interactives et le traitement de flux [11].

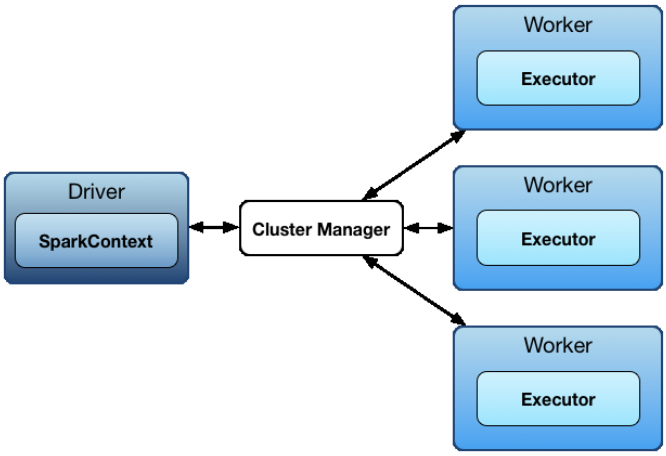

FIGURE 6 – Architecture de Spark [2]

0.4.6 Avantages de Spark par rapport à Hadoop MapReduce

Spark est un fort Framework pour les futures grandes applications de données qui peuvent nécessiter des requêtes de faible latence, calcul itératif et traitement en temps réel. Spark a beaucoup d'avantages par rapport au Framework Hadoop MapReduce parmi eux on trouve [12],[11] :

0.4.6.1 La rapidité

Spark est un environnement de calcul open source similaire à Hadoop, mais il a quelques différences utiles qui le rendent supérieur dans certaines charges de travail, il permet de charger l'ensemble de données en mémoire distribuée pour optimiser la charge de travail itératif et les requêtes interactives.

Spark peut exécuter les traitements de 10 à 100 fois plus rapidement que Hadoop MapReduce tout simplement en réduisant le nombre de lectures et écritures sur le disque.

0.4.6.2 Traitement itératif

Il existe beaucoup d'algorithme qui appliquent la même fonction à plusieurs étapes. Comme les algorithmes d'apprentissage, Hadoop MapReduce est basée sur un modèle de flux de données acyclique, c'est-à-dire que la sortie d'une tâche MapReduce précédente est l'entrée de la prochaine tâche MapReduce. Dans ce cas on perd beaucoup de temps dans l'opération d'E/S, alors dans Hadoop MapReduce entre deux opérations MapReduce, il existe une barrière de synchronisation et on a besoin de conserver les données sur le disque à chaque fois [12].

Mais avec Spark, le concept de RDD (Resilient Distributed Datasets) permet d'enregistrer les données sur la mémoire et préserver le disque seulement pour les opérations de résultats. Ainsi il n'a pas tout une barrière de synchronisation qui éventuellement pourrait ralentir le processus. Alors Spark permet de réduire le nombre de lecture/écriture sur le disque, donc le temps principal est lié au traitement des données car il n'existe pas un temps de E/S.

0.4.6.3 Requêtes interactives

Pour le traitement dans les algorithmes d'extraction de données interactive où un utilisateur a besoin d'exécuter de multiples requêtes sur un même sous ensemble de données, Hadoop charge les mêmes données plusieurs fois à partir de disque selon le nombre de requêtes c'est-à-dire chaque requête à son propre lecteur de disque et son propre traitement MapReduce.

Mais Spark charge les données une seule fois, il stocke ces données dans la mémoire distribuée, ensuite il applique le traitement adéquat. Pour le traitement dans les algorithmes d'extraction de données interactive où un utilisateur a besoin d'exécuter de multiples requêtes sur un même sous ensemble de données, Hadoop charge les mêmes données plusieurs fois à partir de disque selon le nombre de requêtes c'est-à-dire chaque requête à son propre lecteur de disque et son propre traitement MapReduce.

0.4.6.4 Plus riche

Spark fournit des API concises et cohérentes à Scala, Java et Python et Prend en charge plusieurs fonctions (actions et transformations), contrairement à Hadoop, on trouve seulement les deux fonctions Map et Reduce et un seul langage Java.

0.4.6.5 Facilité d'utilisation

Spark vous permet d'écrire rapidement des applications en Java, Scala, ou Python avec des instructions simples et lisibles.

0.4.6.6 Généralité

Du côté de la généralité, Spark est conçu pour couvrir un large éventail de charges de travail qui nécessitent auparavent des systèmes distribués distincts, y compris les applications de traitement en temps réel, les algorithmes itératifs, les requêtes interactives et le streaming. En soutenant ces charges de travail dans le même moteur, Spark est facile et peu coûteux de combiner les types de traitement différents, ce qui est souvent nécessaire dans les pipelines d'analyse de données de production.

0.4.6.7 Méthode streaming Real-Time de Spark à traiter des flux

En cas de Hadoop MapReduce il est juste possible de traiter un flot de données stockées, mais avec Apache Spark il est ainsi possible de modifier les données en temps réel grâce à Spark streaming [11].

0.4.6.8 Traitement graphique

Les développeurs peuvent maintenant aussi bien faire usage de Apache Spark pour le traitement graphique qui mappe les relations dans les données entre les diverses entités telles que les personnes et les objets [11].

0.4.6.9 Les algorithmes d'apprentissage

Spark est livré avec une bibliothèque d'apprentissage appelé MLlib, elle fournit plusieurs types d'algorithmes d'apprentissage, notamment la classification, la régression, le regroupement et le filtrage collaboratif, ainsi que l'appui des fonctionnalités telles que l'évaluation du modèle et l'importation de données [11]. Mais dans Hadoop il faut intégrer une bibliothèque d'apprentissage appelé Mahout.

0.4.6.10 Gestion rapide des données structurées

Spark SQL est le module de Spark pour travailler avec des données structurées. Spark SQL permet d'interroger des données structurées comme un ensemble de données distribuées (RDD) dans Spark, avec des API intégrées en Python, Scala et Java [3].

0.4.6.11 Généralité de stockage

Spark utilise le système de fichier HDFS pour le stockage de données. Il fonctionne aussi avec n'importe quelle source de données compatible avec Hadoop y compris, HBase, Cassandra, etc.

0.4.6.12 Interactive

Offre une console interactive pour Scala et Python. Ce n'est pas encore disponible en Java.

0.4.7 Déploiement

Exécuter des traitements lourds sur un cluster, piloter les noeuds esclaves, leur distribuer les tâches équitablement, et arbitrer la quantité de CPU et de mémoire qui sera allouée à chacun des traitements, tel est le rôle d'un gestionnaire de cluster. Spark offre pour l'instant trois solutions pour cela : Spark standalone, YARN et Mesos. Livré avec Spark, Spark Standalone est le moyen le plus simple à mettre en place. Ce gestionnaire de cluster s'appuie sur Akka pour les échanges et sur Zookeeper pour garantir la haute disponibilité du noeud maître. Il dispose d'une console pour superviser les traitements, et d'un mécanisme pour collecter les logs des esclaves.

Autre possibilité, YARN le gestionnaire de cluster Hadoop, Spark peut s'exécuter dessus, et aux côtés de jobs Hadoop. Enfin, plus sophistiqué et plus généraliste, Mesos

3. Spark Programming Guide - Spark 1.2.0 Documentation. [Online]. Available : http ://spark.apache.org/docs/1.2.0/programming-guide.html

permet de configurer plus finement l'allocation des ressources (mémoire, CPU) aux différentes applications.

0.4.8 Composants de Spark

Parce que le moteur de base de Spark est à la fois rapide et polyvalent, il alimente de multiples composants de haut niveau spécialisés pour diverses charges de travail, tels que SQL ou l'apprentissage automatique. Ces composants sont conçus pour inter opérer étroitement, vous permettant de les combiner comme les bibliothèques dans un projet logiciel.

Spark Core : contient les fonctionnalités de base de Spark, y compris les composants pour la planification des tâches, gestion de la mémoire, la reprise après incident, interaction avec les systèmes de stockage, et plus. Spark Core est également l'API qui définit les ensembles de données distribués élastiques (RDD), qui sont les principales abstractions de programmation de Spark. RDD représentent une collection d'objets répartis sur plusieurs nœuds de calcul qui peuvent être manipulés en parallèle. Spark Core offre de nombreuses API pour la construction et la manipulation de ces collections.

Autre que Spark core API, il ya des bibliothèques supplémentaires qui font partie de l'écosystème de Spark et fournissent des capacités supplémentaires dans l'analyse des Big Data 7. Ces bibliothèques sont : Spark streaming, Spark SQL, Spark MLlib, Spark GraphX [11].

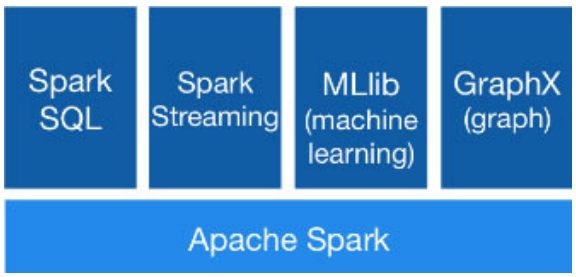

FIGURE 7 – Les composants de Spark

0.4.9 RDD Dataset Resilent Distribue

0.4.9.1 Définition

Un RDD est une collection d'objets partitionnés à travers un ensemble de machines, permettant aux programmeurs d'effectuer des calculs en mémoire sur grosses

grappes d'une manière qui assure la tolérance aux pannes [4].

0.4.9.2 Caractéristiques

1. RDD atteindre la tolérance aux pannes à travers une notion de lignée : si une partition d'un RDD est perdue, le RDD a suffisamment d'informations pour reconstruire simplement cette partition. Cela supprime la nécessité de la réplication pour atteindre la tolérance aux pannes.

2. Il y a deux possibilités pour créer un RDD soit de référencer des données externes ou bien de paralléliser une collection existante. Spark permet de créer un RDD à partir de toute source de données acceptée par Hadoop (fichier local, HDFS, HBase, etc.).

3. Vous pouvez modifier un RDD avec une transformation, mais la transformation vous retourne une nouvelle RDD alors que le RDD original reste le même.

4. RDD prend en charge deux types d'opérations les transformations et les actions (voir figure 8) :

Transformation : les transformations ne renvoient pas une seule valeur, elles renvoient une nouvelle RDD. Rien n'est évalué lorsque vous appelez une fonction de transformation. L'évaluation des transformations est paresseuse, les opérations sont effectuées seulement quand un résultat doit être retourné.

Action : une opération qui évalue et renvoie une nouvelle valeur. Lorsqu'une fonction d'action est appelée sur un objet RDD, toutes les requêtes de traitement de données sont calculées à ce moment et la valeur de résultat est renvoyée [12].

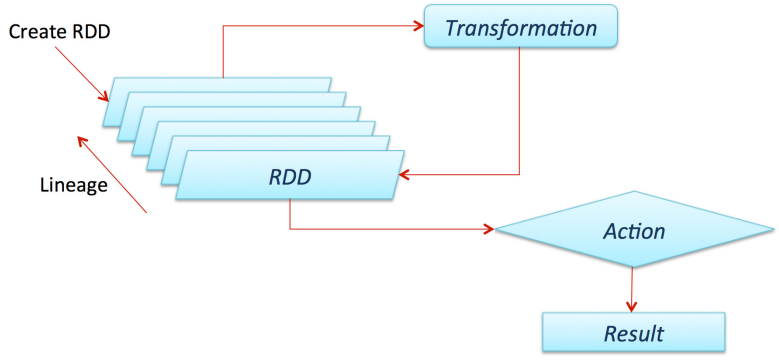

FIGURE 8 – Les actions et les transformations de Spark

4. Spark Programming Guide - Spark 1.2.0 Documentation. [Online]. Available : http ://spark.apache.org/docs/1.2.0/programming-guide.html.

0.5 Le stockage

Lors du traitement d'une grande quantité de données, les données d'entrée et les résultats doivent être stockés. En outre, la performance des applications à forte intensité de données dépend généralement de l'infrastructure matérielle et logicielle utilisée pour le stockage.

0.5.1 Le stockage classique

Ce type de système de stockage est parfois appelé stockage direct (DAS). Sur ces disques, les données sont stockées à l'aide d'un système de fichiers hiérarchique classique comme ext3 ou ReiserFS. Ces systèmes de fichiers sont généralement implémenté par un pilote du système d'exploitation comme une pièce sensible pour la sécurité, la performance et la fiabilité.

Ce type de stockage permet des opérations de lecture et d'écriture rapides puisque tout est effectué localement. Il est également simple à utiliser car il est utilisé avec n'importe quel système d'exploitation. Cependant, il n'existe aucun moyen simple d'échanger des données entre plusieurs nœuds.

0.5.2 Le stockage centralisé réseau

Une deuxième façon de stocker les données, c'est le stockage centralisé dans le réseau, habituellement appelé stockage attaché au réseau etworkAttached Storage (NAS). Dans ce cas, un nœud a un ou plusieurs disques connectés et permet à d'autres noeuds de lire et d'écrire des fichiers via une interface standard et les servir par le biais du réseau. Le système de fichiers réseau (NFS) est principalement un protocole d'accès aux fichiers via le réseau. Bien que le serveur soit libre de mettre en œuvre n'importe quel moyen d'accès aux données réelles à fournir via le réseau, la plupart des implémentations dépendent simplement du fait que les données sont directement accessibles sur le serveur.

L'un des principaux avantages de ce type d'architecture est la facilité de partager des données entre plusieurs nœuds de calcul. Étant donné que les données sont stockées sur un serveur, elles sont facilement maintenues.

Cependant, l'un des principaux inconvénients est que les opérations simultanées de lecture et d'écriture doivent être supportées par un seul serveur. Il a également l'inconvénient de ne pas tolérer aux fautes par lui-même. Si le serveur de stockage tombe en panne, toutes les données ne sont pas disponibles.

0.5.3 Le stockage parallèle et distribué

Afin de surmonter les limites de stockage centralisé réseau, les données peuvent être réparties sur plusieurs nœuds de stockage. L'utilisation de plusieurs nœuds permet d'accéder à plusieurs fichiers en même temps sans conflit. Il permet également un meilleur débit pour lire le même fichier lorsqu'il existe des répliques sur plusieurs nœuds. Un système de fichiers distribués est habituellement conçu pour être meilleur que le stockage centralisé. De plus, en théorie, les systèmes de stockage distribués peuvent éviter le seul point de problème de tolérance à la panne.

Le système de stockage distribué possède souvent un noeud de point d'entrée unique qui reçoit toutes les demandes de lecture ou d'écriture de données. Comme son rôle est central et critique, son travail doit être réduit au minimum. Ce nœud maître ne permet généralement qu'une collaboration globale parmi tous les noeuds impliqués dans le système de stockage et peut stocker les méta-données (nom de fichier, taille de fichier, attributs d'accès, ...). Ainsi, lorsqu'une demande de lecture ou d'écriture d'un fichier est reçue, le client est redirigé vers un autre nœud qui va réellement traiter sa demande. Cependant, si les métadonnées peuvent être stockées sur le noeud maître, les données réelles sont toujours stockées sur d'autres nœuds et peuvent être reproduites.

L'inconvénient d'un stockage distribué est que pour obtenir les meilleures performances, l'application devrait tenir compte de la localité. En effet, même si l'on pensait que le comportement par défaut du système de stockage pourrait être assez bon, il est généralement préférable de lire ou d'écrire des données de / vers le nœud le plus proche du stockage du système à partir d'un noeud avec un coût de réseau élevé.

Un exemple de ce système de stockage est le système de fichiers distribués Hadoop (HDFS) et Tachyon.

0.5.3.1 Architecture de HDFS

HDFS a une architecture du type maitre/esclave. Un cluster HDFS est constitué d'un maitre unique appelé NameNode qui gère l'espace de noms du système de fichiers et règle l'accès aux fichiers par les clients (ouvrir, fermer, renommer, etc.), ainsi qu'un ensemble de DataNodes pour gérer le stockage réel de données [10].

0.5.3.2 Les composants de HDFS

HDFS est géré avec l'architecture maître-esclave, il inclut les composants suivants :

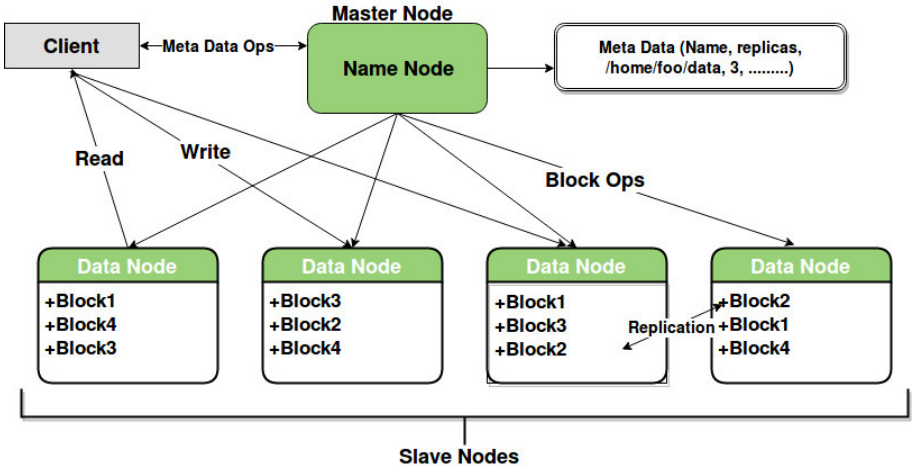

FIGURE 9 – L'architecture HDFS

0.5.3.3 NameNode

Il n'y a qu'une seule instance de NameNode par cluster HDFS, c'est le maître du système HDFS. Il gère les blocs qui sont présents sur les DataNodes [10], cette tâche nécessite plusieurs fonctionnalités comme la gestion de l'espace de nommage, la gestion de l'arborescence du système de fichiers et le stockage des modifications des métadonnées (noms, permissions, etc.) des fichiers et répertoires [13], sans NameNode tous les fichiers peuvent être considérés comme perdus. Chaque fois qu'un client veut lire ou écrire une donnée, il communique avec le NameNode lequel répond à la demande en retournant l'emplacement des blocs de données ainsi que les informations du DataNode qui contient ces blocs. Les fonctions principales du NameNode sont :

— Contrôler l'exécution des opérations d'E/S de bas niveau des DataNodes.

— Choisir de nouveaux DataNodes pour de nouvelles réplications de blocs de données, en cas de défaillance d'un DataNode dans le processus d'écriture .

— Enregistrer les métadonnées de tous les fichiers enregistrés dans le cluster, comme par exemple l'emplacement, la taille des fichiers, les permissions, la hiérarchie, etc.

— Gérer le facteur de réplication de tous les blocs.

0.5.3.4 Name Node secondaire

Afin d' assurer la fiabilité de HDFS, on trouve le NameNode secondaire. Ce dernier, c'est le responsable de l'exécution périodique des points de contrôle. Donc si le NameNode est indisponible, peut-être remplaçait par une image instantanée stockée par les points de contrôle de NameNode secondaires [10]. Alors sa fonction principale est de lire en permanence tous les fichiers et métadonnées à partir de la RAM du NameNode

et les écrits sur son disque pour rendre ces informations accessibles en cas d'échec dans le fonctionnement du NameNode.

0.5.3.5 Data Nodes

Ce sont des esclaves qui sont déployés sur chaque machine et fournissent le stockage réel. Ils sont chargés de servir les demandes de données de lecture et d'écriture pour les clients. Les DataNodes sont sous les ordres du NameNode et sont surnommés les Workers. Ils sont donc sollicités par les Name Nodes lors des opérations de lecture et d'écriture. En lecture, les Data Nodes vont transmettre au client les blocs correspondant au fichier demandé. En écriture, les Data Nodes vont retourner les données de chaque bloc, pour cela ils ont plusieurs fonctionnalités comme [13] :

— Envoyer régulièrement un rapport sur tous les blocs présents dans le cluster au NameNode.
— Un DataNode effectuer les opérations de lecture et d'écriture de bas-niveau.
— Il est responsable de la création de blocs et un canal de communication avec les autres DataNodes.
— Stocker chaque bloc de données HDFS dans des fichiers séparés dans son système de fichiers local.

0.5.3.6 Les opérations de HDFS

Comme la plupart des systèmes de fichiers classiques, HDFS contient un nombre important d'opérations comme la lecture, l'écriture, la suppression des fichiers, et d'autres opérations pour créer et supprimer des répertoires. Le NameNode est le responsable de la gestion des métadonnées pour les fichiers et les répertoires, il expose un espace de noms qui permet de stocker les données sur un cluster de nœuds. Le NameNode gère toutes les opérations telles que (ouvrir, fermer, renommer ou déplacer un fichier ou un répertoire). Les DataNodes sont les responsables de servir les données de fichiers réels.

Une caractéristique importante dans l'HDFS : lorsqu'un client demande ou envoie des données, ces dernières ne passent pas physiquement à travers NameNode, ce serait un énorme goulot d'étranglement. Au lieu de cela, le client obtient tout simplement les métadonnées sur le fichier du NameNode et récupère les blocs de fichiers directement à partir des nœuds.

0.5.3.7 Le processus de lecture d'un fichier sur HDFS

Ce processus commence par l'interrogation entre le client et le NameNode. L'identité du client est d'abord validée en utilisant un mécanisme d'authentification ; ensuite le NameNode confirme si le client a le droit d'accès pour la lecture de ce fichier ; enfin pour chaque bloc de ficher le NameNode renvoie l'adresse du DataNode le plus proche possédant une copie du bloc (Figure 10) [2].

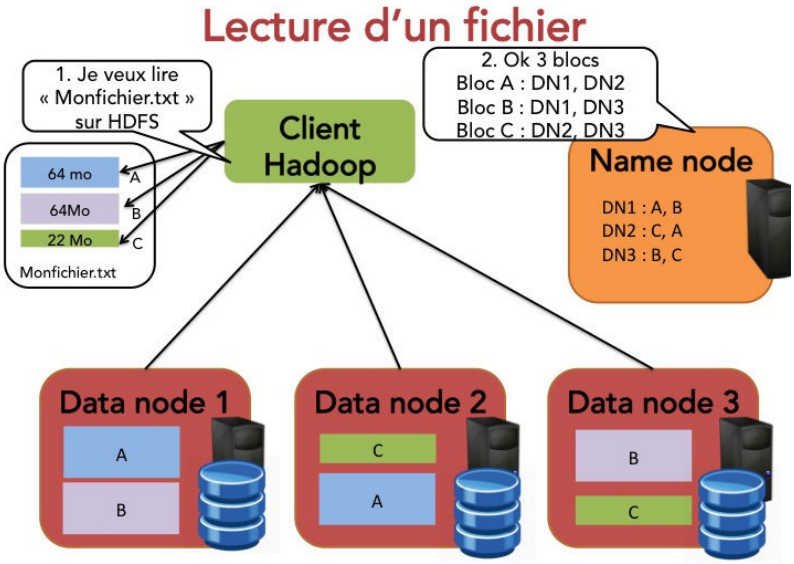

FIGURE 10 – Le processus de lecture d'un fichier sur HDFS [2]

0.5.3.8 Le processus d'écriture d'un fichier sur HDFS

Dans ce processus le client contacte le NameNode, ce dernier crée un fichier dans la hiérarchie du système de fichiers HDFS et informe le client sur l'identificateur de bloc et l'emplacement des DataNodes. Le client utilise ces informations pour faire une écriture réelle dans les DataNodes concernés (le processus de réplication est fait) ; chaque DataNode informe le NameNode par la création d'un fichier réel sur son système de stockage local si l'opération est réussie (Figure 12) [2].

0.5.3.9 La gestion des blocs HDFS

Il faut comprendre comment les fichiers sont stockés physiquement dans le cluster. Dans Hadoop, chaque fichier est divisé en multiples blocs. Une taille typique de bloc est de 64 Mo, mais il n'est pas typique pour configurer la taille des blocs de 32 Mo ou 128 Mo. Les tailles de bloc peuvent être configurées par fichier dans le HDFS [13]. Si le

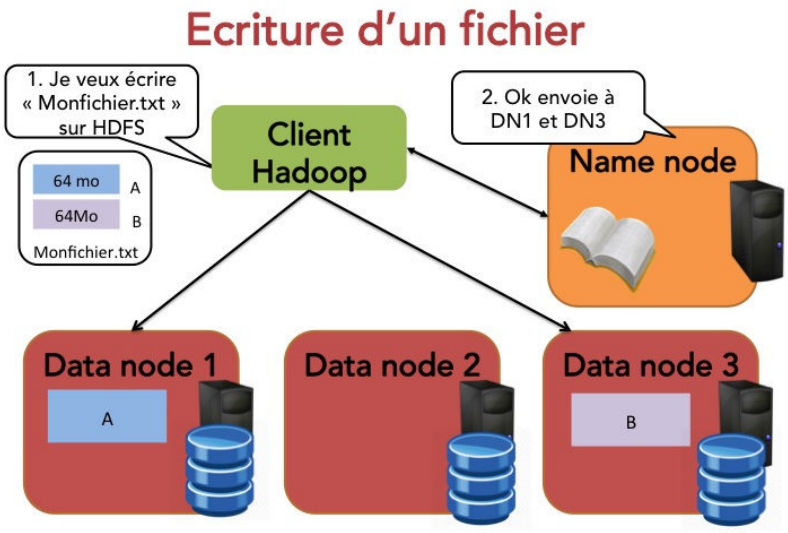

FIGURE 11 – Le processus d'écriture d'un fichier sur HDFS [5]

fichier n'est pas un multiple exact de la taille du bloc, l'espace n'est pas gaspillée et le dernier bloc est juste inférieure à la taille totale du bloc.

— Un grand fichier sera divisé en plusieurs blocs. Chaque bloc est stocké sur un DataNode. Il est également répliqué pour s'assurer contre l'échec.

— HDFS permet également de définir le facteur de réplication d'un fichier. Par défaut le facteur de réplication d'un fichier est de trois, il améliore la tolérance aux pannes et réduit les défauts système et augmente la bande passante de lecture

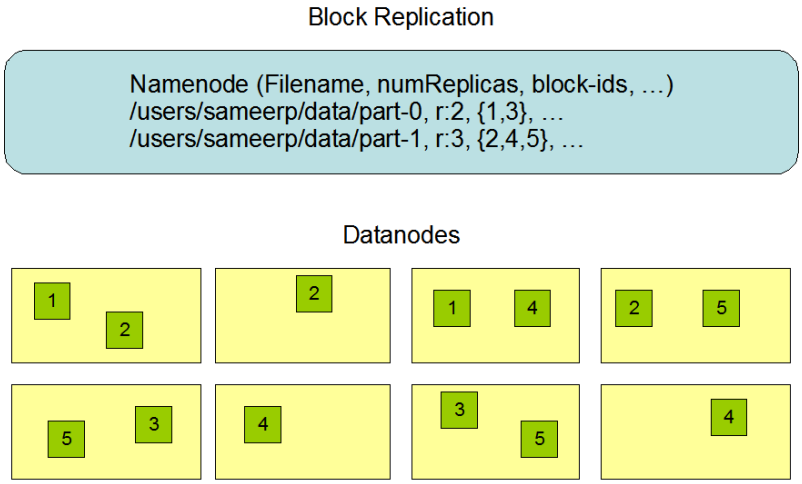

FIGURE 12 – Le processus de réplication de blocs [3]

0.5.4 Définition de Tachyon

Tachyon est un système de stockage distribué centré sur la mémoire, qui permet aux utilisateurs de partager des données à travers les plateformes et d'exécuter des actions de lecture/écriture à la vitesse de la mémoire à travers les plateformes de traitement en Cluster. Il permet aussi d'obtenir un débit d'écriture de 110x supérieur que dans la mémoire HDFS [14]. Afin d'assurer la tolérance aux pannes, la sortie perdue est récupérée en exécutant à nouveau les opérations qui ont créé la sortie, appelée lignage (lineage) [14]. Ainsi, l'option de lignage de Tachyon est considérée comme un défi majeur à Tachyon, et la couche de lignage fournit des E/S à haut débit et suit la séquence de travaux et le lignage de données dans la couche de stockage.

0.5.4.1 Architecture de Tachyon

Tachyon utilise une architecture master-slave standard similaire à HDFS (voir la figure 13), cette architecture s'appelle master-worker.

Le master gère les métadonnées et contient un gestionnaire de flux de travail, ce dernier interagit avec un gestionnaire de ressources de cluster pour allouer des ressources et recalculer. Tandis que, les workers gèrent les ressources locales et rapportent l'état au maître, et chaque worker utilise un RAMdisk pour stocker des fichiers mappés en mémoire.

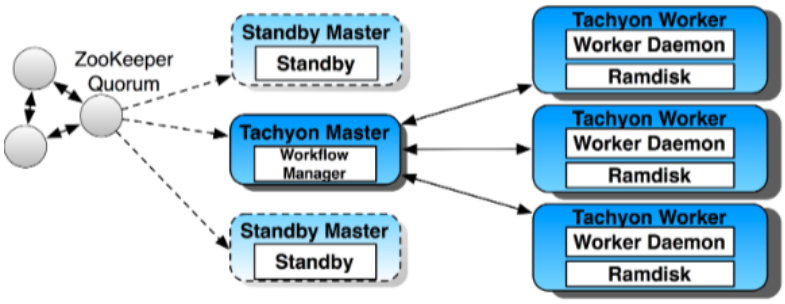

FIGURE 13 – L'archieture de Tachyon [6]

0.5.4.2 Les composants de tachyon

La conception de Tachyon utilise un seul maître et plusieurs travailleurs. Tachyon peut être divisé en trois composantes, le maître, les travailleurs et les clients. Le maître et les travailleurs constituent ensemble les serveurs Tachyon, qui sont les composants qu'un administrateur système maintiendrait et gérerait. Les clients sont généralement

les applications, telles que Spark ou MapReduce, ou les utilisateurs de la ligne de commande Tachyon. Donc, les utilisateurs de Tachyon ne doivent habituellement qu'interagir avec la partie client de Tachyon [14].

i. Maître (Master) : Tachyon peut être déployé dans l'un des deux modes principaux, un maître unique ou plusieurs maîtres. Le maître est principalement responsable de la gestion des métadonnées globale du système, par exemple, l'arborescence du système de fichiers. Les clients peuvent interagir avec le maître pour lire ou modifier ces métadonnées. En outre, tous les travailleurs interrogent périodiquement le maître pour maintenir leur participation au cluster. Le maître ne lance pas de communication avec les composants; il interagit uniquement avec les composants en répondant aux demandes.

ii. Travailleurs (worker) : Les travailleurs de Tachyon sont responsables de la gestion des ressources locales allouées à Tachyon. Ces ressources pourraient être la mémoire locale, le SSD ou le disque dur et ils sont configurables par l'utilisateur. Les travailleurs de Tachyon stockent les données en tant que blocs et répondent aux demandes des clients de lire ou d'écrire des données en lisant ou en créant de nouveaux blocs. Toutefois, le travailleur n'est responsable que des données dans ces blocs.

iii. Client : Le client Tachyon offre aux utilisateurs une passerelle pour interagir avec les serveurs Tachyon. Il lance la communication avec le maître pour mener à bien les opérations de métadonnées et avec les travailleurs pour la lecture et l'écriture des données.

0.6 L'infrastructure distribuée

Toutes les plates-formes présentées précédemment ont besoin d'un logiciel pour les rendre faciles à utiliser comme un ensemble de noeuds cohérent. Cela peut se faire soit comme un système d'exploitation, soit comme un logiciel de gestion de niveau supérieur [15].

0.6.1 Les grappes Cluster

On peut définir un cluster comme un ensemble de noeuds connectés entre eux, où l'ensemble homogène de noeuds qui font le cluster peut-être utilisé de deux manières différentes. L'une des façons est d'utiliser un système d'exploitation de cluster qui fait apparaître le cluster en tant que système unique. L'autre façon est d'utiliser un gestionnaire de ressources qui permet d'attribuer un ensemble de noyaux (cores) ou de nœuds pendant un certain temps [15].

Une grappe de machine (Cluster en Anglais) désigne un ensemble d'ordinateurs,

appelés nœuds, tous inter-connectés, dans le but de partager des ressources informatiques. Une grappe peut être constituée d'ordinateurs de bureau, de "racks" de machines constituées de composants standards ou de "lames" également constituées de composants standards afin d'optimiser l'espace physique. Une grappe est généralement composée de machines homogènes en termes d'architecture et de système d'exploitation. Elle ne regroupe que des machines appartenant au même domaine d'administration réseau et les nœuds communiquent entre eux en utilisant un réseau de communication rapide. Les différents nœuds d'une grappe possèdent souvent une configuration logicielle semblable [16].

0.6.2 Les grilles

Le terme "grille" désigne un ensemble beaucoup plus important de machines, hétérogènes, réparties sur différents domaines d'administration réseau . Les différentes entités composant une grille peuvent être réparties sur l'ensemble de la planète et communiquent entre elles en utilisant une grande diversité de réseaux allant de l'Internet à des réseaux privés très haut débit.

Une grille informatique est une infrastructure matérielle et logicielle qui fournit un accès consistant à des ressources informatiques [17]. Le but est ainsi de fédérer des ressources provenant de diverses organisations désirant collaborer en vue de faire bénéficier aux utilisateurs une capacité de calcul et de stockage qu'une seule machine ne peut fournir. Cependant, tout système informatique distribué ne peut posséder l'appellation de grille [18].

En effet, une grille est un système qui coordonne des ressources non soumises à un contrôle centralisé, qui utilise des protocoles et interfaces standards dans le but de délivrer une certaine qualité de service (en termes de temps de réponse ou bien de fiabilité par exemple). Plusieurs types de grilles peuvent être discernées selon l'utilisation recherchée :

— Grille d'information : la ressource partagée est la connaissance. L'Internet en est le meilleur exemple : un grand nombre de machines hétérogènes réparties sur toute la surface du globe autorisant un accès transparent à l'information.
— Grille de stockage : l'objectif de ces grilles est de mettre à disposition un grand nombre de ressources de stockage d'information afin de réaliser l'équivalent d'un "super disque dur" de plusieurs PetaBytes. Le projet DataGrid ou les réseaux x Kaaza ou Gnutella sont un bon exemple de grille de stockage.
— Grille de calcul : l'objectif de ces grilles est clairement d'agréger la puissance de traitement de chaque nœud de la grille afin d'offrir une puissance de calcul "illimitée" (exemple : grid5000).

0.6.3 Les nuages

Le concept de nuage ou "cloud" [19] est une évolution de la notion de grille. ll se focalise plus sur les fournisseurs des ressources qui mettront à disposition la puissance de calcul et de stockage avec leur propre technologie. La définition d'utilisateurs et de fournisseurs permet de créer une relation claire entre ces deux types d'entités. On pourra alors associer des garanties de puissance fournie ou de quantité de stockage par exemple. Trois modèles peuvent être utilisés sur un nuage [15] :

— IaaS (Infrastructure as a service) : le fournisseur met à disposition uniquement des ressources matérielles (machine, réseau, disque), l'utilisateur gère le système, les intergiciels et les applications.

— PaaS (Platform as a service) : l'utilisateur ne gère et ne contrôle ici que ses applications métier, le reste est délégué au fournisseur de service.

— SaaS (Software as a service) : c'est l'ultime niveau de transfert vers le fournisseur, l'utilisateur ne gère plus que ses données métier, il utilise les applications fournies par le fournisseur. Il s'agit d'utiliser le logiciel à la demande.

0.7 Conclusion

Dans ce rapport, nous avons introduit les concepts de base de Big Data et ses technologies, permettant de dépasser des outils classiques, qui ne parviennent pas à traiter ces énormes volumes de données, et aller vers des outils modernes, qui font face à la grande taille de données.

Plusieurs outils ont été cités et introduits dans ce rapport, à savoir les plateformes de stockage comme HDFS et Tachyon et les plateformes de traitement comme Hadoop MapReduce et Spark. La plateforme Spark est une alternative à MapReduce permet aux développeurs de simplifier la complexité du code MapReduce et écrire des programmes et des requêtes d'analyse de données en Java, Scala ou Python en utilisant le même système de stockage Hadoop HDFS ou Tachyon pour traiter d'énormes quantités de données « Big Data » et d'accélérer leurs temps de traitement, car elle supprime le temps de lecture/écriture de disque grâce à l'utilisation de structure de données distribué RDD, permettant aux programmes d'effectuer des calculs en mémoire.

Bibliographie

[1] Jeffrey Dean and Sanjay Ghemawat. Mapreduce : simplified data processing on large clusters. *Communications of the ACM*, 51(1) :107–113, 2008.

[2] Eric Sammer. *Hadoop operations*. " O'Reilly Media, Inc.", 2012.

[3] Aditya B Patel, Manashvi Birla, and Ushma Nair. Addressing big data problem using hadoop and map reduce. In *2012 Nirma University International Conference on Engineering (NUiCONE)*, pages 1–5. IEEE, 2012.

[4] Ahmed Oussous, Fatima-Zahra Benjelloun, Ayoub Ait Lahcen, and Samir Belfkih. Big data technologies : A survey. *Journal of King Saud University-Computer and Information Sciences*, 30(4) :431–448, 2018.

[5] Raghavendra Kune, Pramod Kumar Konugurthi, Arun Agarwal, Raghavendra Rao Chillarige, and Rajkumar Buyya. The anatomy of big data computing. *Software : Practice and Experience*, 46(1) :79–105, 2016.

[6] Desamparados Blazquez and Josep Domenech. Big data sources and methods for social and economic analyses. *Technological Forecasting and Social Change*, 130 :99–113, 2018.

[7] Alberto Fernández, Sara del Río, Abdullah Bawakid, and Francisco Herrera. Fuzzy rule based classification systems for big data with mapreduce : granularity analysis. *Advances in Data Analysis and Classification*, 11(4) :711–730, 2017.

[8] Lu Lu, Xuanhua Shi, Hai Jin, Qiuyue Wang, Daxing Yuan, and Song Wu. Morpho : a decoupled mapreduce framework for elastic cloud computing. *Future Generation Computer Systems*, 36 :80–90, 2014.

[9] Seyed Nima Khezr and Nima Jafari Navimipour. Mapreduce and its applications, challenges, and architecture : a comprehensive review and directions for future research. *Journal of Grid Computing*, 15(3) :295–321, 2017.

[10] Vignesh Prajapati. *Big data analytics with R and Hadoop*. Packt Publishing Ltd, 2013.

[11] Holden Karau, Andy Konwinski, Patrick Wendell, and Matei Zaharia. *Learning spark : lightning-fast big data analysis*. " O'Reilly Media, Inc.", 2015.

[12] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark : Cluster computing with working sets. *HotCloud*, 10(10-10) :95, 2010.

[13] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. The hadoop distributed file system. In *2010 IEEE 26th symposium on mass storage systems and technologies (MSST)*, pages 1–10. Ieee, 2010.

[14] Haoyuan Li, Ali Ghodsi, Matei Zaharia, Scott Shenker, and Ion Stoica. Tachyon : Reliable, memory speed storage for cluster computing frameworks. In *Proceedings of the ACM Symposium on Cloud Computing*, pages 1–15. ACM, 2014.

[15] Sylvain Gault. *Improving MapReduce Performance on Clusters*. PhD thesis, Ecole normale supérieure de lyon-ENS LYON, 2015.

[16] Thierry Monteil. *Du cluster à la grille sous l'angle de la performance*. PhD thesis, Institut National Polytechnique de Toulouse-INPT, 2010.

[17] Ian Foster and Carl Kesselman. The grid : Blueprint for a future computing inf, 1999.

[18] Ian Foster. What is the grid?-a three point checklist. *GRIDtoday*, 1(6), 2002.

[19] Peter Mell, Tim Grance, et al. The nist definition of cloud computing. 2011.