Table des matières

1 INTRODUCTION **1**

1.1 Contexte 1

1.2 Travail de stage 3

1.3 Le stage 4

1.4 Le laboratoire LIAS 5

1.5 Organisation du manuscrit 5

2 ÉTAT DE L'ART **7**

2.1 Introduction 7

2.2 Histoire de l'utilisation du terme Big Data 7

2.2.1 Caractéristique du big data 9

2.3 Histoire de la gestion de données 10

2.4 Hadoop 13

2.4.1 HDFS 13

2.4.2 MapReduce 14

2.5 Apache Spark 16

2.5.1 Caractéristiques de spark 16

2.5.2 Architecture de spark 16

2.6 Apache Spark vs Hadoop 17

2.7 QDAG 18

2.8 Conclusion 20

3 CONCEPTION ET RÉALISATION **21**

3.1 Introduction 21

3.2 Orchestra 21

3.2.1 Broadcast 22

3.2.2 Cornet VS bittorrent 22
3.2.3 Shuffle 23
3.2.4 la gestion des transferts de données 24
3.3 Présentation de la solution 25
3.4 Mécanismes de Transferts de données 25
3.4.1 Chain Broadcast 25
3.4.2 TreeBroadcast 26
3.4.3 Broadcast selon le bit Torrent 28
3.4.4 Évaluation des différentes stratégies de broadcast 30
3.4.5 Shuffle 31
3.5 Mécanismes de vérification du réseau 33
3.5.1 Mécanisme de battement du coeur 33
3.5.2 Accusé de réception (Acknowlegement) 34
3.6 Scénario d'utilisation du framework 34
3.7 Outils et technologies utilisés 36
3.7.1 java 36
3.7.2 L'environnement de test 37
3.7.3 PuTTY 37
3.7.4 SuperPuTTY 38
3.7.5 FileZilla 39
3.7.6 Parallel-scp 40
3.7.7 Git 40
3.7.8 Git LIAS 41
3.7.9 Gestion de projet avec Trello 41
3.8 Les étapes à suivre pour effectuer les tests 42
3.8.1 Conclusion 43

4 CONCLUSION GÉNÉRALE **45**
4.1 Conclusion 45
4.2 Perspectives 46

Bibliographie **47**

Table des figures

1.1 Le logo du laboratoire **LIAS** 5

2.1 Les 4 Vs du Big Data [8] 9
2.2 L'architecture de Hadoop[23] 13
2.3 L'architecture de Hadoop HDFS[23] 14
2.4 L'architecture de Hadoop MapReduce[23] 15
2.5 L'architecture de Spark[25] 17
2.6 Architecture du système QDAG 19

3.1 L'architecture d'Ochestra[26] 24
3.2 Illustration de ChainBroadcast 26
3.3 Illustration de TreeBoadcast 27
3.4 Illustration de broadcast au bitTorrent 29
3.5 Broadcast au bitTorrent(étape avancée) 30
3.6 Illustration du Shuffle 32
3.7 Illustration du mécanisme battement de coeur 33
3.8 Diagramme de classe "accusé de réception" 34
3.9 Diagramme de séquence d'un scénario d'utilisation de framework 36
3.10 Inteface PuTTY 38
3.11 Inteface SuperPuTTY 39
3.12 Inteface fileZilla 40
3.13 Inteface Git LIAS 41
3.14 Inteface Trello 42

Liste des Algorithmes

1 Master side (chain broadcast) 26
2 Worker side (chain broadcast) 26
3 Master side (tree broacast) 28
4 Worker side (tree broacast) 28
5 Master side (bitTorrent like broadcast) 29
6 Worker side (bitTorrent like broadcast) 30
7 Master side (shuffle) 32
8 sender side (shuffle) 32
9 receiver side (shuffle) 32

Table des abréviations

API	**Application Programming Interface**
CNRS	**Centre National de la Reherche Scientifique**
EDI	**Environnement de Développement Intégré**
ENSMA	**École Nationale Supérieure de Mécanique et d'Aérotechnique**
ETL	**Extract, Transform and Load**
FIFO	**First In First Out**
HDFS	**Hadoop Distributed File System**
IDD	**Ingénierie des Données et des moDèles**
IP	**Internet Protocol**
ITC	**Inter Transfer Controller**
LSST	**Large Synoptic Survey Telescope**
LIAS	**Laboratoire d'Informatique et d'Automatique pour les Systèmes**
QDAG	**Querying Data As Graphs**
RDD	**Resilient Distributed Datasets**
SFTP	**Secure File Transfer Protocol**
SGBDR	**Système de Gestion de Base de Données**
SQL	**Structured Query Language**
SSH	**Secure SHell**
TC	**Transfer Controller**
WSS	**Weighted Shuffle Scheduling**

Chapitre 1

INTRODUCTION

1.1 Contexte

Le Big Data représente un défi non seulement pour le monde socio-économique mais aussi pour la recherche scientifique [1]. En effet, comme il a été souligné dans plusieurs articles scientifiques [2] et rapports stratégiques [3], les applications informatiques modernes sont confrontées à de nouveaux problèmes qui sont liés essentiellement au stockage et à l'exploitation de données générées par les instruments d'observation et de simulation. La gestion de telles données représente un véritable goulot d'étranglement qui a pour effet de ralentir la valorisation des différentes données collectées non seulement dans le cadre de programmes scientifiques internationaux mais aussi par des entreprises, ces dernières s'appuyant de plus en plus sur l'analyse de données massives. La recherche scientifique, à l'ère des Big Data, est devenue multidisciplinaire. En effet, il est nécessaire de combiner des techniques issues de plusieurs disciplines (informatique, physique, mathématique, ...) afin de faire avancer la science. D'ailleurs, à titre d'exemple, le projet LSST [1] ambitionne la construction du plus grand télescope au monde. Le défi ultime de LSST est de mettre à disposition des scientifiques une base de données commune à partir de laquelle seront conduites des recherches scientifiques qui s'intéressent, entre autres, à la recherche de petits objets dans le système solaire, à l'astrométrie de précision des régions extérieures à la Voie Lactée, à la surveillance des effets

1. http://www.lsst.org

Rapport-Gratuit.com

transitoires dans le ciel optique et à l'étude de l'univers lointain. La communauté française utilisera ces données pour mener des études sur l'énergie noire responsable de l'accélération de l'expansion de l'univers, incomprise à ce jour. Le goulot d'étranglement lié à ces analyses repose en grande partie sur la méthodologie d'accès et de traitement des données retenues. LSST produira des images CDD de 3,2 Giga pixel toutes les 17 secondes (la nuit), pendant 10 ans. Il permettra à terme de générer 15 à 30 Téraoctets de données par nuit pour arriver à un volume d'environ 140 Péta octets d'images en fin de programme. Le catalogue de données est constitué de tables relationnelles ayant des tailles allant jusqu'à 5 Péta octets [4]. Par conséquent, de telles applications sont orientées par des questions telles que : comment stocker, organiser, indexer et distribuer des milliers de Péta Octets de données ? Comment combiner l'indexation et la gestion de mémoire pour des bases de données extrêmement volumineuses, distribuées et multidimensionnelles ? Comment évaluer des jointures entres des objets ayant plus de 100 milliards d'éléments, ce qui induit un problème de passage à l'échelle.

Afin de répondre à ces questions, le CNRS a décidé de financer le projet *Petasky* dans le cadre du défi MASTODONS. Ce dernier vise à répondre à plusieurs défis liés à l'analyse et la gestion des données scientifiques dans le domaine de la cosmologie. Par conséquent, le projet réside clairement dans la perspective de l'e-science. *Petasky* repose sur le projet LSST comme contexte d'application. Le groupe *gestion de données* de *Petasky* se concentre sur la conception et la mise en oeuvre de techniques d'évaluation et d'optimisation de requête garantissant le passage à l'échelle. En effet, les techniques permettant une utilisation efficace de nouvelles plateformes matérielles et logicielles représentent une étape importante pour le développement du "Big Data" dont les challenges scientifiques sont reconnus. Pour cela, trois principaux défis ont été identifiés : (i) la conception de nouvelles abstractions et des modèles pour capturer les propriétés des plateformes technologiques modernes, (ii) la conception de systèmes qui supportent la parallélisation massive des traitements sur des grandes masses de données, et (iii) la définition formelle des modèles de coûts pour évaluer l'efficacité des algorithmes utilisés dans les plateformes technologiques

modernes. Les contributions scientifiques attendues sont liées principalement à 1) l'identification des bonnes abstractions pour capturer les nouveaux environnements d'exécution, 2) le développement de structures appropriées au stockage et 3) le développement des modèles de coût et des algorithmes d'optimisation qui permettent d'exécuter les requêtes parallèles et distribuées sur de très grandes masses de données.

Les responsables de LSST ont déjà identifié quelques contraintes liées au futur système permettant de stocker et requêter les données de LSST. En effet, ils recommandent l'utilisation d'un système open source, basé sur l'architecture "Shared-Nothing". Deux facteurs principaux motivent une telle recommandation : (i) faciliter l'intégration et l'optimisation des fonctions ad hoc (c-à-d., des fonctions propres au calcul astronomique) et (ii) supporter des requêtes déclaratives.

Tenant compte de ces recommandations, et après analyse approfondie des capacités de certains systèmes existants pour gérer les données LSST, les membres de Petasky ont proposé un nouveau système permettant de gérer d'une manière efficace et transparente les données de LSST. Ce système, baptisé QDAG (Querying Data As Graphs), a pour objectif de garantir à la fois le passage à l'échelle et les performances lors du traitement des Big Data. Il est d'ailleurs modulaire et permet d'injecter de nouveaux composants selon le besoin métier.

1.2 Travail de stage

Mon travail de stage s'inscrit dans la continuité des travaux menés par les membres de l'équipe ingénierie des modèles et de données (IDD) du LIAS et qui ont pour objectif de booster le développement de nouveaux composants pour le système QDAG.

D'ailleurs, à travers deux sujets de thèse de doctorat, l'équipe IDD vise à proposer un nouveau moteur d'interrogation de requêtes de nouvelle génération et un module de chargement de données intelligent.

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

L'objectif de ce stage était de concevoir et implémenter un framework permettant de gérer tous les transferts de données engendrés par les différents composants (e.g., Partitionnement, évaluation de requêtes, optimiseur,...) de QDAG et ce d'une manière transparente et efficace. Le transfert de données représente en effet une tâche indispensable qui devrait survenir pendant le traitement d'une requête, le chargement de données et l'optimisation du fonctionnement du système. D'ailleurs, ces transferts ont un impact significatif sur les performances du traitement, représentant plus de 50% de temps d'achèvement du traitement.

Durant mon stage, j'ai commencé par étudier le fonctionnement du système QDAG. J'ai analysé d'une manière minutieuse chaque composant nécessitant un mécanisme de transfert de données. J'ai pu constater que les composants de QDAG font appel à deux (2) types de transfert, à savoir le Shuffle (avec ses deux cas spéciaux le Collector et le distributor) et le BroadCast. Après cette étape d'analyse du besoin, j'ai fait un état de l'art sur les mécanismes de transfert de données dans le cadre du cluster computing. Je me suis intéressé plus particulièrement au framework Orchestra [5]. J'ai par la suite proposé non seulement une nouvelle conception et implémentation de ce framework mais aussi des extensions nécessaires au système QDAG. Avec les doctorants du LIAS, j'ai réussi à intégrer mon travail dans le moteur d'exécution de QDAG.

1.3 Le stage

Le stage s'est déroulé au sein du LIAS qui se trouve au niveau des locaux de L'école nationale supérieure de mécanique et d'aérotechnique (ISAE-ENSMA) située à Poitiers - France, depuis le 02 Mars jusqu'au 18 Juillet 2018. Le stage est le résultat d'une convention entre l'université de Tlemcen, l'université de Poiters, l'ENSMA et le LIAS.

1.4 Le laboratoire LIAS

Le LIAS [2] (**L**aboratoire d'**I**nformatique et d'**A**utomatique pour les **S**ystèmes) représente 35 enseignants chercheurs dans les disciplines de l'Automatique, du Génie électrique et de l'Informatique. Il a été créé le 1er janvier 2012, suite à la fusion des laboratoires du LAII (Laboratoire d'Automatique et d'Informatique Industrielle) et du LISI (Laboratoire d'Informatique Scientifique et Industrielle).

Le laboratoire LIAS est composé de trois équipes de recherche : l'équipe Ingénierie des Données et des Modèle, l'équipe Systèmes embarqués Temps Réel et l'équipe Automatique et Système.

Mon stage de fin d'étude s'est effectué sous l'encadrement de l'équipe Ingénierie des données et des Modèles (IDD). L'équipe IDD s'intéresse aux différentes problématiques liées à la construction de systèmes de gestion de données permettant la collecte, l'intégration, la persistance des données et des modèles et l'exploitation des données d'une manière efficace, flexible et intelligente.

FIGURE 1.1 – Le logo du laboratoire **LIAS**

1.5 Organisation du manuscrit

Le reste de ce manuscrit est organisé de la manière suivante : Nous aborderons les Big Data et les technologies utilisées pour le traitement des données massives dans le chapitre 2. Nous présentons par ailleurs le système QDAG. Le chapitre 3 est consacré à la présentation de notre travail de conception et de réalisation. Enfin, une conclusion générale sera discutée dans le chapitre 4. Nous

2. https ://www.lias-lab.fr/

donnerons un bilan personnel et professionnel lié à ce stage ainsi que quelques pistes qui permettent d'améliorer les différentes propositions.

Chapitre 2

ÉTAT DE L'ART

2.1 Introduction

Le terme Big Data est utilisé largement de nos jour afin de désigner les nouvelles opportunités liées à l'exploitation des données massives. Il est d'ailleurs l'objet de plusieurs débats dans le monde scientifique et industriel. En effet, la question principale qui anime ce genre de débat est la suivante : Est-ce que le Big Data est une réalité ou seulement un Buzzword ? Afin de répondre à cette question, il est nécessaire de revenir en arrière pour voir le contexte d'utilisation de ce terme dans le passé. Dans ce chapitre, nous commençons par donner un petit historique sur l'utilisation du terme Big Data. Nous discuterons par la suite les caractéristiques du Big Data et pour quoi cela représente un défi. Nous présentons un état de l'art sur la gestion de données massives. Cela permettra de justifier la proposition du système QDAG. Nous finirons ce chapitre par une présentation des différents composants du système QDAG.

2.2 Histoire de l'utilisation du terme Big Data

Dans un article scientifique, le terme Big Data a été utilisé pour la première fois pour désigner les données collectées pour une étude météorologique sur l'île de la Barbade. L'étude en question date du milieu des années 60. Le volume de ces données ne dépassait pas les quelques méga-octet. Malgré cela, traiter ces données représentait un vrai défi à cause de la technologie qui n'était

pas au point. Entre temps, et avec le développement des bases de données relationnelles[6] [7] début des années 70, la gestion de données est devenue plus facile et efficace. En effet, les bases de données relationnelles sont devenues l'outil numéro 1 pour gérer les données transactionnelles. Les SGBDR sont utilisés jusqu'à maintenant pour différents types de données, e.g., assurances, banques, etc.

Le besoin en terme de nouvelles techniques d'exploitation de données s'est évolué fin des années 80. En effet, Avec l'augmentation des volumes de données et la complexité des taches d'analyse dont ces données font partie, les SGBDR sont devenus incapable de satisfaire le besoin des utilisateurs en termes de performances. On a réutilisé le terme Big Data afin de désigner ce genre de données et de tâches d'exploitation. Heureusement, les acteurs des bases de données ont su s'adapter à ce besoin grandissant en offrant de nouvelles technologies basées sur l'indexation et la matérialisation des informations. OLAP est en effet la principale technologie développée pendant cette époque. Avec le développement du Web fin des années 90 et par conséquence les données semi-structurées, les SGBDR sont devenus encore une fois dans l'incapacité de fonctionner correctement. On a en effet réutilisé le terme Big Data pour désigner ce besoin émergeant. Jusqu'à maintenant, on s'est basé sur des architectures standards de systèmes. Le NoSQL a été donc la solution à ce nouveau besoin. Ce genre de système permet de gérer les données d'une manière personnalisée. A l'heure actuelle, le terme Big data est utilisé pour désigner les données massives collectés par les instruments de simulation et d'observation modernes. Ces données concernent des domaines différents et variés. Dans le domaine de l'aéronautique par exemple, un avion Airbus génère 20 To par heure de vol. Dans le domaine de la physique des particules, l'accélérateur du CERN génère 500 To par jour. Facebook d'un autre côté avec son réseau social génère plus de 300 Po par an. Avec ce genre de données, il était nécessaire de penser à de nouvelles solutions. côté matériel, le cluster computing est devenu une solution incontournable à cause de la facilité de mise en place et le passage à l'échelle. côté logiciel, de nouveau framework (e.g., Hadoop, Spark et Flink, ...) ont vu le jour. Ces frameworks supportent le cluster computing et offrent des interfaces de

programmation qui font abstraction sur la gestion de la partie physique du cluster. Malheureusement, même avec ces technologies, on est loin d'avoir ce que les SGBDR offraient dans les années 90. Il manque en effet, tous ce qui est traitement déclarative et performances. Si on revient maintenant à notre question initiale, c.-à-d. Est ce que le Big Data est une réalité ou seulement un Buzzword ? la réponse à notre avis est que l'utilisation du Big Data est une manifestation de notre retard technologique face à un problème de traitement de données.

2.2.1 Caractéristique du big data

Il y a quatre caractéristiques pour le Big Data : le volume, la vélocité, la variété et la véracité.

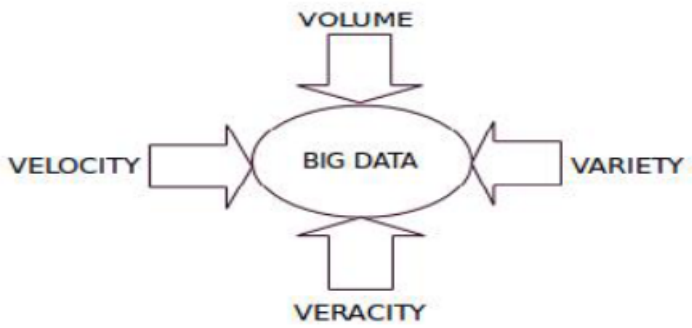

FIGURE 2.1 – Les 4 Vs du Big Data [8]

— **le volume**

le volume signifie le passage à l'échelle de données ou la grande quantité de données générées chaque seconde. Les données générées par les machines sont des exemples pour ces caractéristiques. De nos jours, le volume de données augmente de gigaoctets à pétaoctets [9]. 40 zettaoctets des données seront créées d'ici 2020, soit 300 fois à partir de 2005[9].

— **la vélocité**

La vélocité est la vitesse à laquelle les données sont générées et traitées. Par exemple, les messages générés par les réseaux sociaux [9].

— **la variété**

la variété est une autre caractéristique importante du big data. elle fait référence au type de données. Les données peuvent prendre différentes formes, telles que du texte, des images numériques, de l'audio, de la vidéo et des données qui viennent des réseaux sociaux [9]. Sur Twitter, 400 millions de tweets sont envoyés par jour et il y a 200 millions d'utilisateurs actifs [10].

— **la véracité**

la véracité signifie l'incertitude ou l'exactitude des données. Les données sont incertaines en raison de l'incohérence et de l'incomplétude [9].

2.3 Histoire de la gestion de données

En 1970 jusqu'à la fin des années 90, beaucoup de travaux de normalisation ont été effectués. Cela a permis l'adoption de SQL comme langage de requêtes. Dans cette période, les chercheurs ont cru au concept de "one size fits all", c.-à-d., les SGBDR pourraient être utilisés pour gérer efficacement la plupart des bases de données. D'ailleurs, les efforts des concepteurs des SGBDR ont principalement porté sur l'intégration des nouvelles extensions pour SQL et la façon de traiter efficacement les extensions proposées. L'architecture de tels systèmes n'a pas beaucoup changé depuis la première implémentation par IBM du système[11].

En général, le goulot d'étranglement dans les SGBD centralisés est dû au taux de transfert du disque [12]. L'utilisation de plusieurs machines, avec plusieurs disques, pourrait être une solution intéressante. En effet, comme les ressources d'une machine sont limitées, il serait intéressant de créer un cluster de plusieurs machines pour fournir d'avantage de ressources. Nous appelons ce type de système "parallèle" lorsque ses noeuds fonctionnent comme une seule entité. Un tel système peut être considéré comme un ordinateur très puissant, construit en utilisant plusieurs ordinateurs, chacun avec un très bon rapport coût/performance. Dans le cas où ces ordinateurs ne partagent aucune

ressource (c.à-d., RAM, disque, CPU), cette architecture est appelée shared-nothing. Cette architecture matérielle permet dans plusieurs cas de supporter le passage à l'échelle.

Dans le domaine des bases des données, l'architecture shared-nothing a été d'abord utilisée par Teradata [13], Tandem [14], Bubba [15], Arbre, et nCUBE. Par la suite, elle a été adoptée par la quasi-totalité des SGBDR, y compris Teradata, Netezza, Greenplum, ParAccel, DB2 et Vertica. Elle est également utilisée par la plupart des plateformes d'e-commerce, y compris Amazon, Akamai, Yahoo, Google et Facebook.

A la fin des années 90, les SGBD sont devenus la pièce la plus importante des systèmes d'information. Ces derniers sont largement utilisés dans de nombreuses applications variant de l'entrepôt de données jusqu'au traitement des flux et du texte. Malheureusement, en expérimentant les performances des SGBDR et les outils spécialisés, on peut constater que le SGBDR peut être dépassé de 1 à 2 ordres de grandeur par les outils spécialisés dans l'entreposage de données, le traitement des flux, du texte et des bases de données scientifiques[16]. Cela est dû à l'architecture des SGBDR qui ne peut pas s'adapter aux besoins métiers de l'utilisateur. L'une des propositions intéressantes qui permet aux SGBDR de garantir de meilleures performances [16][17] consiste à offrir aux concepteurs de SGBD un ensemble de composants séparés avec un fonctionnement et des performances prévisibles. Ces composants peuvent ensuite être assemblés pour répondre aux besoins de l'utilisateur. Deux exemples entrent dans cette catégorie : RDF-3X [18] conçu pour gérer les données RDF, et C-Store [19] conçu pour gérer les données analytiques. RDF-3X utilise des composants déjà existants dans la littérature tels que le Buffer, les B+Tree et les histogrammes. Sa contribution majeure consiste en la façon d'organiser et de récupérer les données. Par ailleurs, C-Store adopte un mécanisme de stockage par colonnes sans affecter les autres parties du SGBD. Les technologies liées à C-Store sont désormais commercialisés par HP dans la HP Vertica Analytics Platform[20].

Les SGBD traditionnels se révèlent être moins efficaces quand il s'agit de gérer des grandes masses de données. De ce fait, il y a eu récemment une tendance

Rapport-Gratuit.com

croissante pour utiliser des outils ad hoc au lieu des SGBDR dans plusieurs applications. Par conséquent, un nouvel effort est mené par les concepteurs des SGBD afin d'offrir un ensemble d'outils permettant de supporter la gestion efficace des grandes masses de données. Ces derniers sont destinés à être incorporés dans les prochaines générations des SGBD. C'est ainsi que le modèle MapReduce a été conçu. Ce dernier est sensé supporter le traitement efficace d'une grande masse de données via l'utilisation de centaines, voire des milliers de machines dans un cluster de type shared nothing. D'autres paradigmes (e.g., Flink et Spark [21]) ont permis d'améliorer [22] les performances de MapReduce. Le modèle MapReduce est particulièrement efficace pour mettre en oeuvre les algorithmes nécessitant seulement un seul scan des données. Plusieurs cadres ont été proposés ensuite pour étendre ces paradigmes afin de donner la possibilité de traiter déclarativement des requêtes SQL. Ceci dit, ces outils ne s'adapte pas bien aux besoins métiers des utilisateurs. Un effort très important est toujours nécessaire afin de prendre en charge de nouveaux besoins.

Il est clair que les applications modernes se basent sur la gestion et la manipulation de données comme modèle économique. Ceci a permis le développement d'une nouvelle génération de systèmes (généralement adhoc) de gestion de données. D'ailleurs, le défi commun entre ces applications, d'un point de vu base de données, réside dans la manipulation des grandes masses de données. Le choix de système adhoc est naturel. En effet, de tels systèmes ont tendance de garantir des meilleures performances par rapport aux systèmes classiques. Cela est dû au fait que les concepteurs des systèmes Adhoc arrivent à exploiter les connaissances métiers du domaine d'application. Ces derniers donnent des résultats impressionnants pour certaines requêtes. Ceci dit, après quelques années de développement, de tels systèmes sont loin d'être adaptable pour d'autres besoins. En effet, le développement d'un nouveau système de gestion de données reste une tâche compliquée à cause de la philosophie adoptée par la plupart des concepteurs des systèmes (libres et commerciaux) de gestion de données et qui consiste à offrir un système figé avec des composants fortement connectés.

Avant de présenter le systeme QDAG, sur lequel nous avons travaillé, nous allons donner quelques détails sur deux solution, à savoir Hadoop et Spark, qui

sont largement utilisées à l'heure actuelle dans le monde industriel.

2.4 Hadoop

Hadoop est un framework de programmation utilisé pour supporter le traitement de grands ensembles de données dans un environnement distribué de type cluster. Hadoop a été développé par Yahoo à la base et offre l'interface de programmation MapReduce (developpée par Google). Hadoop est maintenant un projet de la fondation Appache. Le Ce framework offre un moteur d'exécution de taches MapReduce, une interface de programmation permettant de faciliter la spécification des tâches de programmation et le HDFS, un système de fichiers distribués[23].

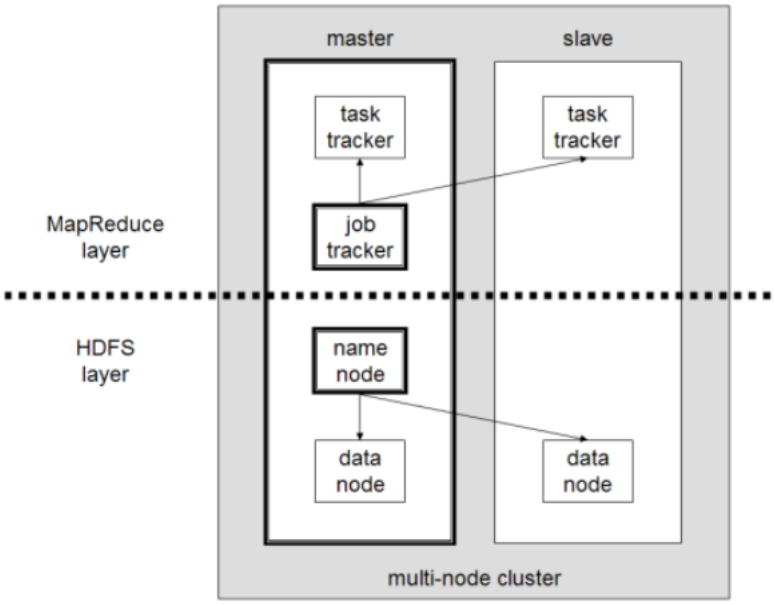

FIGURE 2.2 – L'architecture de Hadoop[23]

2.4.1 HDFS

Hadoop inclut un système de stockage à tolérance de pannes appelé Hadoop Distributed File System ou HDFS. HDFS est capable de stocker de grandes quantités d'informations, d'augmenter progressivement et de survivre à la défaillance de parties importantes de l'infrastructure de stockage sans perdre de

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

données. Hadoop crée des clusters de machines et coordonne le travail entre elles. Les clusters peuvent être construits avec des ordinateurs peu coûteux. En cas d'échec, Hadoop continue à faire fonctionner le cluster sans perdre de données ni interrompre le travail, en déplaçant le travail vers les autres machines. HDFS gère le stockage sur le cluster en décomposant les fichiers entrants en morceaux, appelés "blocs", et en stockant chaque bloc de manière redondante dans le pool de serveurs. Dans le cas habituel, HDFS stocke trois copies complètes de chaque fichier en copiant chaque pièce sur trois serveurs différents[23].

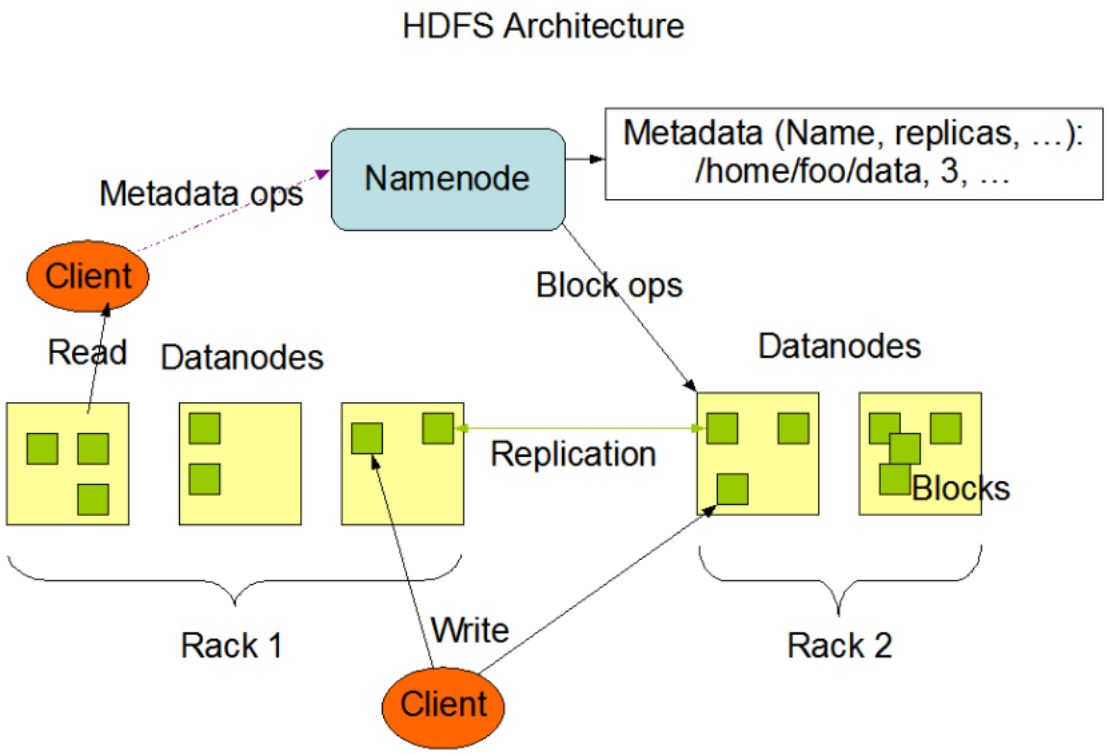

FIGURE 2.3 – L'architecture de Hadoop HDFS[23]

2.4.2 MapReduce

le framework MapReduce est le pilier de traitement de l'écosystème Hadoop. Le framework permet d'appliquer la spécification d'une opération à un énorme ensemble de données, de diviser le problème et les données et de l'exécuter en parallèle. Du point de vue d'un analyste, cela peut se produire sur plusieurs dimensions. Par exemple, un ensemble de données très volumineux

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

peut être réduit à un sous-ensemble plus petit où les analyses peuvent être appliquées.

Dans un entrepôt de données traditionnel, cela peut impliquer l'application d'une opération ETL sur les données pour produire quelque chose utilisable par l'analyste. Dans Hadoop, ces types d'opérations sont écrits en tant que jobs MapReduce en Java. Il existe un certain nombre de langages de niveau supérieur, tels que Hive et Pig, qui facilitent l'écriture de ces programmes. Les résultats de ces travaux peuvent être réécrits sur HDFS ou placés dans un entrepôt de données traditionnel. MapReduce a deux fonctions :

Map : la fonction prend des paires clé / valeur en entrée et génère un ensemble intermédiaire de paires clé / valeur.

Reduce : la fonction qui fusionne toutes les valeurs intermédiaires associées à la même clé. intermédiaire[23].

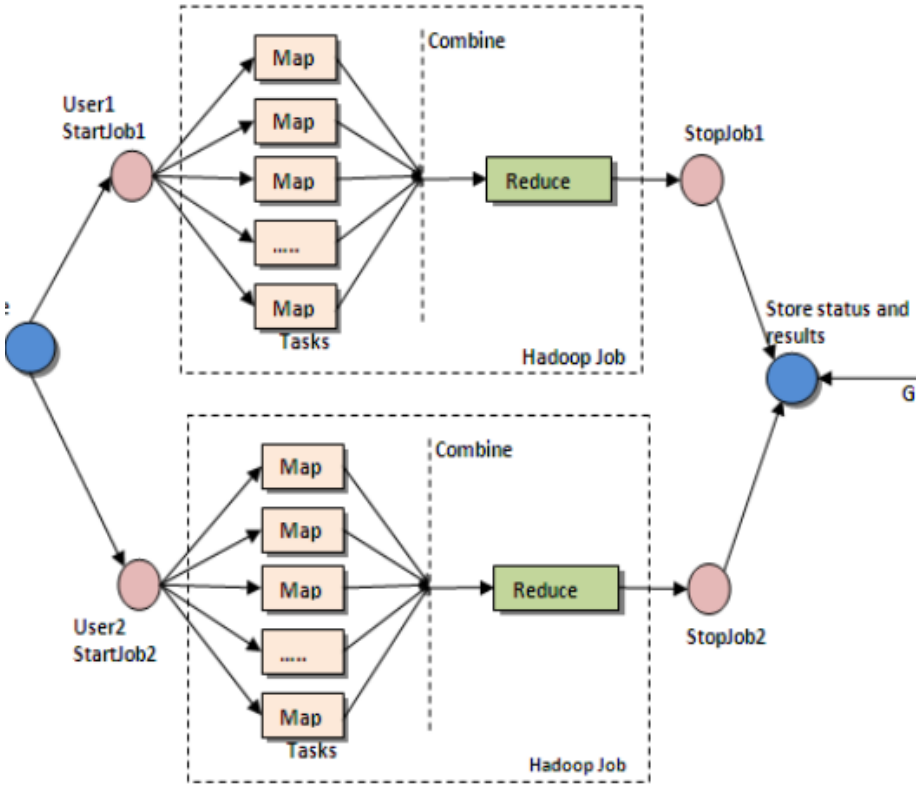

FIGURE 2.4 – L'architecture de Hadoop MapReduce[23]

2.5 Apache Spark

Spark offre un moteur de traitement parallèle de données open source permettant d'effectuer des analyses de grande envergure par le biais de machines en clusters. Ses principaux avantages sont sa vitesse, sa simplicité d'usage, et sa polyvalence. Codé en Scala, Spark permet notamment de traiter des données issues de référentiels de données comme Hadoop Distributed File System, les bases de données NoSQL, ou les data stores de données relationnels comme Apache Hive. Ce moteur prend également en charge le traitement In-memory, ce qui permet d'augmenter les performances des applications analytiques du Big Data. Il peut aussi être utilisé pour un traitement conventionnel sur disque, si les ensembles de données sont trop volumineux pour la mémoire système[24].

2.5.1 Caractéristiques de spark

Ce qui caractérise Spark est sa rapidité. C'était l'un des objectifs de son développement : corriger les lenteurs d'Hadoop qui son due à son architecture et notamment au fait que l'implémentation d'Hadoop MapReduce qui travaillait directement avec les fichiers sur le disque en HDFS. Rien n'est prévu dans MapReduce pour maintenir un cache en mémoire. Spark utilise la mémoire vive de façon à pouvoir optimiser de multiples traitements. Il est typiquement de 10 fois à 100 fois plus rapide que MapReduce[25].

2.5.2 Architecture de spark

Le coeur de Spark est appelé Spark Core. Il offre les fonctionnalités de base, à savoir la distribution planification des tâches, ainsi que les primitives d'entrées-sorties. Spark Core utilise pour ses entrées-sorties une abstraction nommée RDD, pour Résilient Distributed Datasets. Le nom évoque le fait que les données sont naturellement distribuées, et répliquées de façon à pouvoir survivre aux défaillances matérielles des noeuds. Cette abstraction permet à Spark Core de travailler de manière identique avec n'importe quel stockage physique sous-jacent. Cette couche offre donc un niveau intermédiaire qui permet d'allier au

niveau bas n'importe quel type de stockage et différents environnement de distribution des tâches comme Hadoop YARN or Mesos, et au niveau haut, plusieurs algorithmes ou environnements de traitement qui basent sur Core pour l'accès à la planification et aux données. Sont actuellement disponibles avec Spark quatre formes de traitement : des moteurs d'exécution SQL pour le requêtage interactif, le module Spark Streaming pour le traitement en temps réel des flux, MLib pour les algorithmes d'apprentissage sur les données[25].

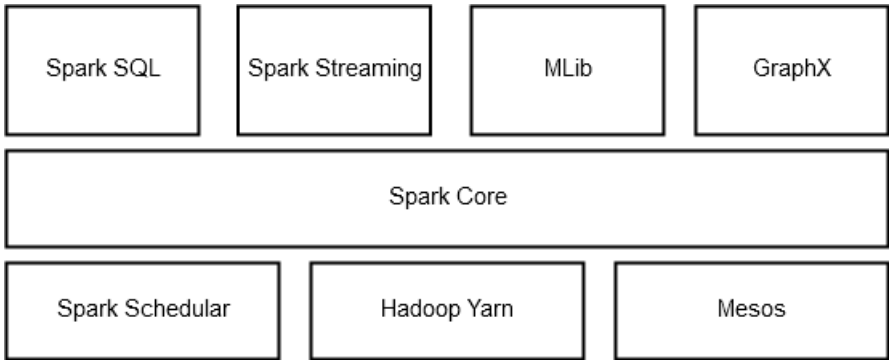

FIGURE 2.5 – L'architecture de Spark[25]

2.6 Apache Spark vs Hadoop

Depuis plus de 10 ans, Hadoop est considéré comme la principale technologie de traitement de données Big Data. Il s'agit effectivement d'une solution de choix pour le traitement de larges ensembles de données. Pour les calculs " one-pass ", MapReduce est effectivement très efficace, mais se retrouve moins pratique pour les cas d'usage nécessitant des calculs et des algorithmes multi-pass. Pour cause, chaque étape du traitement de données est décomposée entre une phase Map et une phase Reduce. Entre chaque étape, les données doivent être stockées avec le système de Fichiers Distribués avant que la prochaine étape ne puisse débuter. Dans la pratique, cette approche se révèle très lente. De plus, les solutions Hadoop incluent généralement des clusters difficiles à configurer et à gérer. Plusieurs outils doivent également être intégrés pour les différents cas d'usage Big Data. Pour le Machine Learning, il faudra par exemple utiliser Mahout. Pour le traitement de flux de données, il sera nécessaire d'intégrer

Storm. De son côté, Apache Spark permet aux programmeurs de développer des pipelines de données multi-step complexes en utilisant des patterns DAG. Spark prend également en charge le partage de données in-memory à travers les DAGs, permettant d'effectuer différentes tâches avec les mêmes données. Il est exécuté à partir d'une infrastructure HDFS existante pour fournir des fonctionnalités améliorées et additionnelles. Il permet de déployer des applications sur un cluster Hadoop V1 avec SIMR, un cluster Hadoop V2 YARN ou sur Apache Mesos. Plutôt qu'un remplacement d'Hadoop, il peut être considéré comme une alternative Spark à Hadoop MapReduce. Spark n'a pas pour vocation de remplacer Hadoop, mais de fournir une solution unifiée et compréhensible pour gérer différents cas d'usage du Big Data[24].

2.7 QDAG

Afin de répondre aux besoins du projet LSST, les chercheurs de Petasky ont proposé un nouveau système, baptisé QDAG (Querying Data As Graphs). Ce système permet d'évaluer les requêtes d'une manière déclarative, c.-à-d. un utilisateur spécifie seulement les contraintes sur les résultats sans se préoccuper de comment les obtenir. QDAG a pour objectif de garantir à la fois le passage à l'échelle et les performances. Deux types de passage à l'échelle devraient être supportés, celui en termes de volume de données et celui en termes de nombre de machines. En effet, QDAG devrait non seulement être en mesure de supporter plusieurs Peta-octet de données mais aussi de supporter une plateforme matérielle constituée de plusieurs centaines de machines. Concernant les performances, QDAG devrait être en mesure d'évaluer les requêtes simples qui nécessitent quelques secondes de calcul, mais aussi les requêtes qui nécessitent plusieurs jours de calcul. QDAG est complètement modulaire et basé sur la philosophie RISC (Reduced Instruction Set Computing). Il supporte la gestion de plusieurs types de données. Principalement QDAG comporte trois composants :

— Le module de chargement de données : ce module permet de préparer les

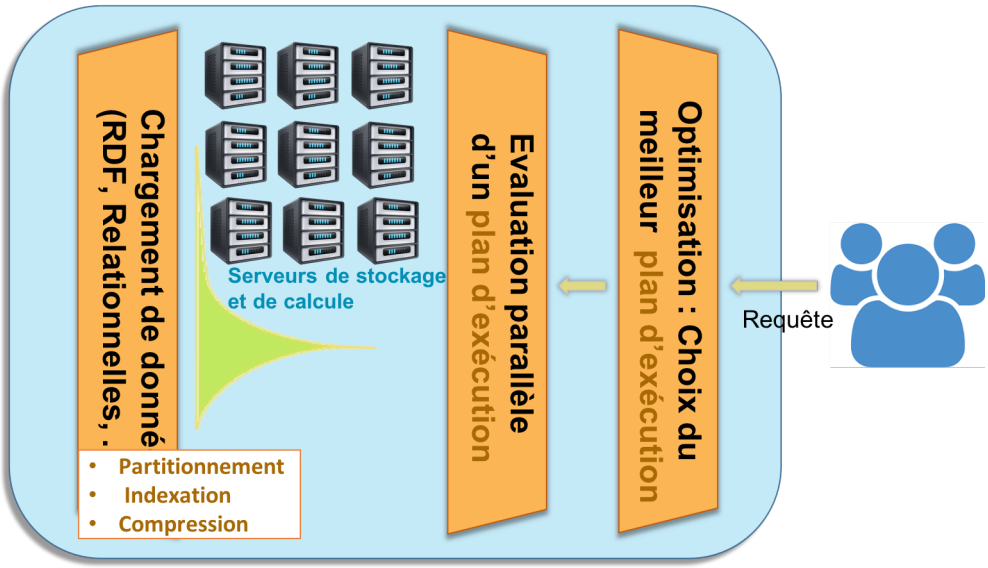

FIGURE 2.6 – Architecture du système QDAG

données afin de les charger dans le système. Il permet aussi de partitionner et répliquer les données. Il prend en entrée un ensemble de fichiers brutes et propose une fragmentation, réplication et allocation de ces données.

— Le moteur d'évaluation parallèle de requêtes : ce module prend en entrée un plan d'exécution afin de l'évaluer. Il se repose sur le modèle de calcul parallèle BSP (Bulk synchronous Parallel). Chaque machine évalue le maximum de partie de la requête en local et fera une étape de transfert de résultats intermédiaires afin de communiquer ces résultats aux autres machines qui peuvent continuer le traitement.

— Le module d'optimisation : ce module permet de choisir le meilleur plan d'exécution pour une requête. En effet, plusieurs plans d'exécution sont possibles pour la même requête. L'optimiseur collecte des statistiques sur les données afin d'estimer le coût d'un plan d'exécution et en choisir le meilleur.

Il est clair que, pendant son fonctionnement, QDAG fait appel à plusieurs types de transfert de données. Par exemple, le module de chargement de données a besoin d'envoyer les fragments de données aux différentes machines. Le moteur d'exécution a besoin de communiquer le plan d'exécution et la requête

à toutes les machines. Les résultats intermédiaires devraient être échangés entre les machines pendant l'évaluation de la requête. Le client a besoin de récupérer les résultats finaux à partir des machines du cluster. L'optimiseur a besoin de collecter des statistiques afin d'établir le meilleur plan d'exécution.

Principalement, nous avons besoin de deux types de mécanismes de transfert de données :

— Le broadcast : qui permet d'envoyer la même donnée à toutes les machines du cluster.

— Le shuffle : qui permet d'échanger des données en pair à pair. Lorsqu'il y a plusieurs transferts sont nécessaires en même temps, il est nécessaire d'avoir une stratégie d'orchestration afin d'éviter les goulots d'étranglement et booster les performances. Nous avons identifié deux cas spéciaux de Shuffle : 1) data collector : lorsqu'une seule machine reçoit les données et 2) data distribution : lorsqu'une seule machine envoie les données.

Notre objectif est d'offrir un cadre applicatif permettant de faciliter l'intégration de ces mécanismes de transfert de données non seulement au sein du système QDAG mais aussi au sein de n'importe quelle application nécessitant un mécanisme de transfert de données. Il est à noter que ces mêmes types de transferts (c.-à-d. Broadcast et Shuffle) qui sont utilisés par Hadoop et Spark.

2.8 Conclusion

Dans ce chapitre, nous avons présenté le big data et ces caractéristiques. Nous avons donné quelques pistes sur les solutions actuelles utilisées pour traiter les Big Data. Nous avons aussi présenté le système QDAG et ses différents composants. QDAG est en effet le système sur lequel nous avons travaillé.

Pour finir nous avons présenté le besoin d'un cadre applicatif transparent offrant un ensemble de mécanismes, fiables et efficaces, de transfert de données.

Chapitre 3

CONCEPTION ET RÉALISATION

3.1 Introduction

Dans ce chapitre, nous abordons la partie conception et réalisation de notre proposition. Nous présentons une architecture permettant un contrôle globale de la gestion de transferts de données dans un environnement distribué. Cette architecture est basée sur le framework Orchstra[26], qui sera présentée dans la prochaine section. Il est à noter qu'il était nécessaire de faire appel à un certain nombre d'outils et technologies existants afin de booster le développement.

3.2 Orchestra

Récemment, les frameworks de type cluster computing ont connu une croissance rapide. Ces Frameworks sont utilisés pour analyser la quantité énorme des données collectée et/ou générée par les instruments de simulation et d'observation modernes. Ces Frameworks (ex. MapReduce, Dryad, Spark) implémentent des modèles computationnels pour la gestion de flux de données. En effet, l'ensemble de données passent par plusieurs étapes de traitement.

Les tâches considérées dans ces Frameworks permettent de manipuler un volume massif de données et qui s'exécutent sur des clusters formés de plusieurs dizaine de millier de machines. Afin de maximiser l'utilisation de ces clusters, la gestion des activités réseau devrait été considéré d'une manière prioritaire [26].

Rapport-Gratuit.com

Orchestra est une architecture de contrôle globale permettant de gérer les activités intra et inter-transferts. Cette architecture est proposée par Mosharaf Chowdhury [26] et utilisée en partie par les fondateurs du Framework Apache spark. Orchestra propose aussi les deux principaux types de transfert, à savoir le Broadcast et le Shuffle.

3.2.1 Broadcast

L'architecture orchestra présente un mécanisme de transfert de données de type broadcast appelé Cornet. Cornet est un mécanisme de transfert de données qui émule le protocole bitTorrent. Cornet est conçu pour les datacenter et peut surpasser l'implémentation Hadoop jusqu'a 4.5 fois.

3.2.2 Cornet VS bittorrent

Cornet diffère de BitTorrent en trois aspects principaux :

— Contrairement au BitTorrent, qui divise les fichiers en blocs et subdivise les blocs en petits blocs de 256 Ko maximum, Cornet ne divise les données que en gros blocs (4 Mo par défaut).

— Alors que dans BitTorrent certains peers [1] (leechers) ne contribuent pas au transfert et quittent dès qu'ils finissent le téléchargement, dans Cornet, chaque noeud apporte sa pleine capacité pendant toute la durée du transfert.

— Cornet n'utilise pas d'opérations SHA1 coûteuses sur chaque bloc de données pour garantir l'intégrité des données, mais il effectue une vérification d'intégrité unique sur l'ensemble des données[26].

Il existe d'autres mécanismes pour faire le broadcast, L'une des solutions de broadcast les plus courantes dans les infrastructures en cluster existantes consiste à écrire les données sur un système de fichiers partagé (ex. HDFS, NFS) et le lire plus tard à partir de ce stockage centralisé. Le problème majeur avec ce type de broadcast est que le système de stockage centralisé peut rapidement devenir un goulot d'étranglement (BottleNeck).

1. un peer est un noeud qui reçoit et envoie les données en même temps

Pour éliminer le goulot d'étranglement centralisé, certains systèmes utilisent des arborescences de distribution d-ary basées sur le noeud source. Les données sont divisées en blocs transmis le long de l'arbre. Dès qu'un noeud a fini de recevoir les données complètes, il peut devenir la racine d'un arbre séparé. d est parfois défini sur 1 pour former une chaîne à la place d'un arbre (par exemple, dans LANTorrent [27] et dans le protocole d'écriture des blocs dans HDFS [28]). Malheureusement, les schémas d'arbres et de chaînes souffrent de deux limitations. Tout d'abord, dans un arbre avec d> 1, la capacité d'envoi des noeuds feuilles (qui sont au moins la moitié des noeuds) n'est pas utilisée. Deuxièmement, un noeud ou un lien lent ralentira l'ensemble de son sous-arbre, ce qui pose problème à grande échelle en raison de la prévalence des retardataires [29]. Cet effet est particulièrement apparent dans les chaînes.

Cornet utilise également un nombre maximal de connexions simultanées pour améliorer les performances. Lorsqu'un peer envoie au nombre maximal de destinataires, il place d'autres requêtes dans une file d'attente jusqu'à ce que l'un des emplacements d'envoi devient disponible. Cela permet de garantir des temps de service plus rapides pour le petit nombre de peers connectés et leur permet de terminer rapidement pour rejoindre la session en tant que sources les plus récentes pour les blocs qu'ils viennent de recevoir[26].

3.2.3 Shuffle

Shuffle est un type de transfert qui consiste à redistribuer des données à partir de plusieurs noeuds expéditeurs. Les destinataires reçoivent les données afin de les réutiliser dans les étapes avenir (ex. transférer un résultat intermédiaire afin de le compléter). Pour les transferts de type shuffle, Mosharaf Chowdhury et al proposent un algorithme optimal appelé Weighted Shuffle Scheduling (WSS). WSS consiste à attribuer un poids à chaque flux en utilisant un partage équitable pondéré, de sorte que le poids du flux entre le récepteur r_i et l'expéditeur s_j soit proportionnel à d_{ij}.

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

3.2.4 la gestion des transferts de données

En plus des différents mécanismes de transferts de données, orchestra gère les transferts en deux différents niveaux, inter et intra-transferts. Orchestra implémente des politiques d'ordonnancement au niveau du transfert via le ITC (Inter-Transfer Controller) y compris priority et FIFO.

Lorsqu'une application souhaite effectuer un transfert dans Orchestra, elle appelle une API qui lance un TC (Transfer controller) pour ce transfert. Le TC s'enregistre auprès de l'ITC pour obtenir sa part du réseau. L'ITC consulte périodiquement une politique de planification (ex. FIFO, priority) pour attribuer des parts aux transferts actifs et les envoie aux TC. Chaque TC peut diviser sa part entre ses paires source-destination comme il le souhaite. L'ITC met également à jour périodiquement les parts des transferts au fur et à mesure de la modification de l'ensemble des transferts actifs. Enfin, chaque TC se désenregistre à la fin de son transfert.

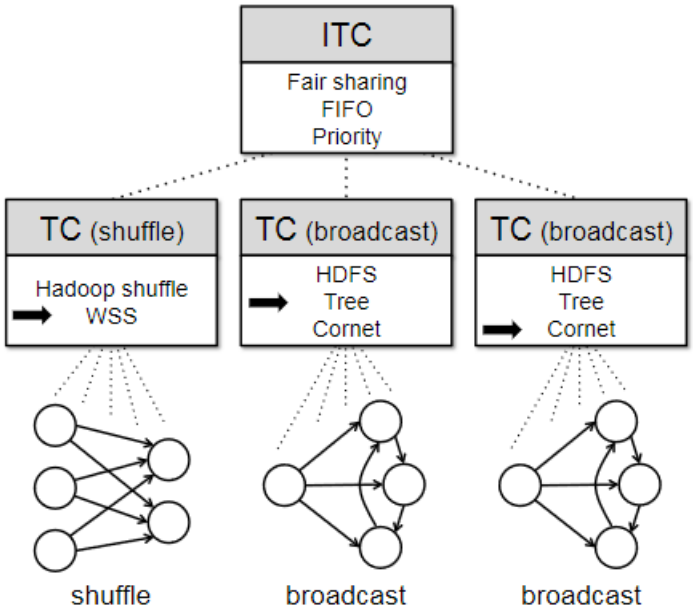

FIGURE 3.1 – L'architecture d'Ochestra[26]

3.3 Présentation de la solution

Cette partie est dédiée à la présentation de l'application développée "Network Management". Il s'agit d'un cadre général (Framework) qui s'occupe de la partie réseau du système existant et qui répond aux besoins exprimés. Le Framework a été développé afin qu'il soit extensible, facile à configurer et à maintenir, il se compose essentiellement de modules qui représentent les différents mécanismes de sorte que chaque module répond à un besoin pertinent.

On distingue deux catégories de mécanisme :1) mécanismes pour le transfert de données ,2) mécanismes pour s'assurer de l'état du réseau.

3.4 Mécanismes de Transferts de données

Cette catégorie de mécanismes, regroupe deux types transfert de données, le broadcast et le Shuffle.

Le broadcast de données (diffusion) permet de transférer la même donnée à partir d'un noeud vers tous les noeuds concernés (one to many). On distingue trois différents types de broadcast fixés au moment de la conception, à savoir Chain, treeBroadcast et le broadcast au bitTorrent.

3.4.1 Chain Broadcast

Ce type de broadcast est trop simple. Il consiste à envoyer la donnée (un objet, requête, plan d'exécution etc.) d'un noeud vers tous les autres noeuds impliqués d'une manière séquentiel (un par un).

Le problème fréquent avec ce type de transfert, est d'avoir des noeuds qui attendent l'arrivée de la donnée sans rien faire, ce qui est considéré comme utilisation non optimal des ressources et qui peut prolonger le temps du traitement.

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

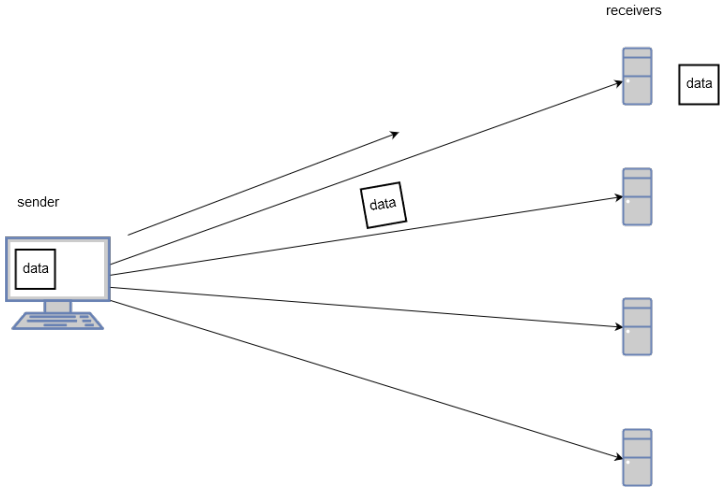

FIGURE 3.2 – Illustration de ChainBroadcast

Algorithme 1 : Master side (chain broadcast)

```
1 input :configuration,data
  /* workers list belongs to configuration input                    */
2 foreach worker ∈ workersList do
3 |   sendData(worker,data);
4 end
5
```

Algorithme 2 : Worker side (chain broadcast)

```
1 input :configuration while true do
2 |   receiveData();
3 end
```

3.4.2 TreeBroadcast

TreeBroadcast est considéré comme une amélioration de Chain, ce type de transfert est basé sur une structure d'arborescence a trois niveaux appelé d-ary. La structure d'arbre est construite en fonction du paramètre d (d = 1 est un cas particulier, il représente le premier type de transfert mentionné préalablement qui est le ChainBroadcast). d > 1 donne un arbre dont la machine de 1er niveau représente la machine qui veut envoyer la donnée aux autres machines.

Les machines de niveau 2 sont des machines intermédiaires (juste pendant ce type de broadcast) qui s'occupent de la réception et le renvoie de la donnée aux machines de niveau 3 qui représentent les feuilles dans la structure d'arbre.

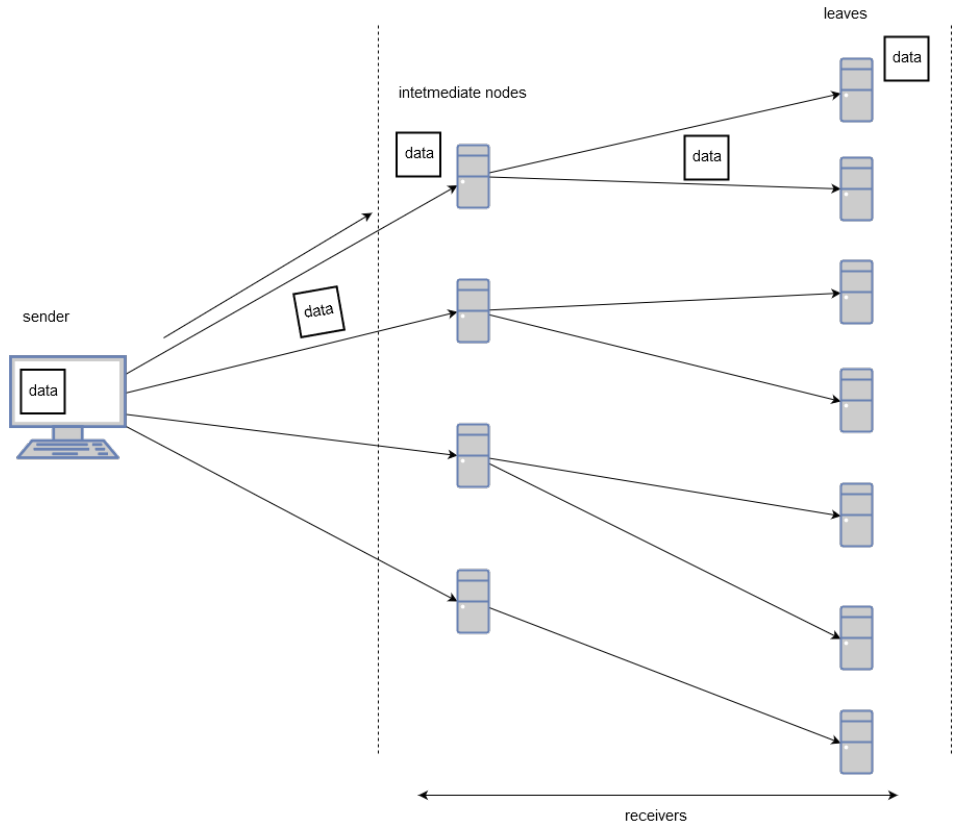

FIGURE 3.3 – Illustration de TreeBoadcast

Le transfert selon le treeBroadcast nécessite d'abord la construction de la structure d'arborescence à partir d'un fichier, appelé Workers (fichier de configuration), disponible sur chacun des noeuds. Après cela chaque noeud reconnaît sa position dans l'arbre c.-à-d. soit un noeud intermédiaire sinon un noeud feuille. La diffusion des données aux noeuds intermédiaires à partir du noeud de 1er niveau est faite en premiere puis chaque noeud intermédiaire et après la réception des données renvoie à l'ensemble de ses feuilles selon la politique Chain.

Algorithme 3 : Master side (tree broacast)

```
1 input :configuration,data
2 extractTree();
3 foreach intermediateNode ∈ treeNodes do
4 |   sendData(intermediateNode,data);
5 end
6
```

Algorithme 4 : Worker side (tree broacast)

```
1  input :configuration
2  extractTree();
3  if isIntermediateNode then
4  |   receiveData();
5  |   foreach leaf ∈ leavesNodes do
6  |   |   sendData(leaf,data);
7  |   end
8  else
   |   /* leaf node                                    */
9  |   receiveData();
10 end
```

3.4.3 Broadcast selon le bit Torrent

Ce type de broadcast est un peu plus complexe que les autres types de broadcast. Il s'agit d'un mécanisme qui simule le protocole bit Torrent. La particularité de ce type de broadcast est qu'il ne se limite pas sur un seul noeud pour diffuser les données, mais chaque noeud contribue au transfert avec toutes ses capacités au long du transfert contrairement au bit Torrent version internet. Il existe des noeuds qui ne contribuent pas dans le transfert, et quittent lorsqu'ils finissent le téléchargement.

Premièrement, l'objet à transférer est divisé en blocs de taille fixe. La taille est précisée dans les fichiers de configuration. Le noeud duquel le transfert se déclenche (le noeud Master généralement) envoie l'ensemble des blocs aux autres noeuds en utilisant la politique roundRobin (tourniquet) de sorte que les blocs sont distribués sur les noeuds.

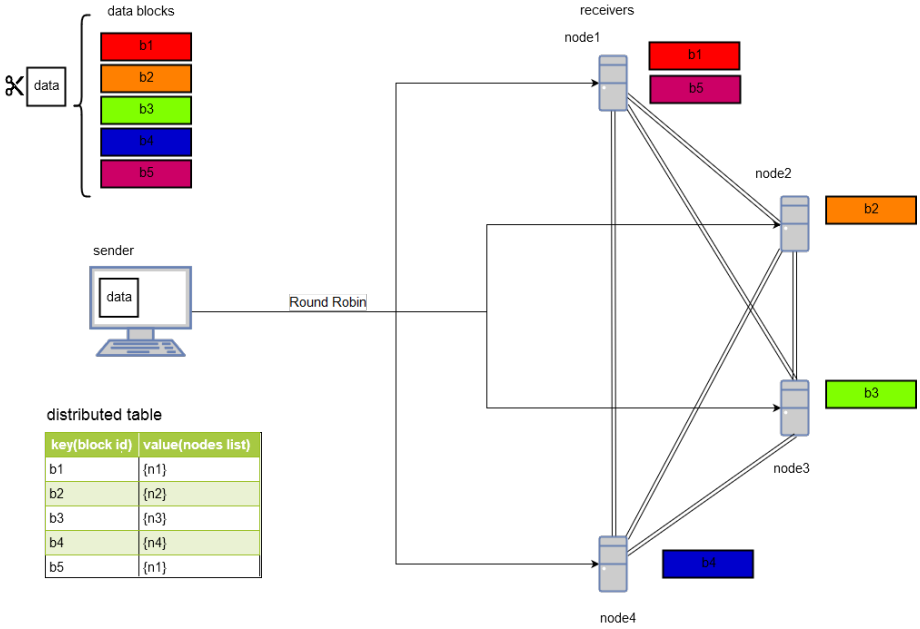

FIGURE 3.4 – Illustration de broadcast au bitTorrent

Algorithme 5 : Master side (bitTorrent like broadcast)

1 **input :**configuration,data
2 map blocksMap $\longleftarrow devideIntoBlocks(data)$;
3 roundRobinBroadcastData(workersList,blocksMap);

A la réception de(s) bloc(s), les noeuds s'échangent les blocs entre eux jusqu'à ce que tous les noeuds reçoivent la totalité des blocs.

Enfin, chaque noeud reconstitue l'objet à partir des blocs récupérés.

Nous utilisons dans ce type de broadcast la notion de tracker, de sorte que sur chaque noeud on aura un tracker pour avoir une idée globale sur l'emplacement des blocs. L'implémentation du tracker se fait à travers une table de hachage distribuée DHT[30](Distriubted Hash Table) , de cette façon la tache devient plus facile. The rarest first [31](le morceau le plus rare), est une stratégie utilisée dans ce type de broadcast, pour donner avantage aux blocs qui sont moins possédés par les noeuds afin d'équilibrer la multiplicité des blocs. Chaque noeud trouvera le bloc le plus rare depuis la DHT, ensuite il vérifiera l'existence de ce bloc parmi les blocs qu'il possède, enverra le bloc à l'un des

noeuds qui appartient à l'ensemble complémentaire des noeuds qui possèdent déjà ce bloc.

Algorithme 6 : Worker side (bitTorrent like broadcast)

```
1 input :configuration
2 map blocksMap ⟵ roundRobinReceiveBlocks();
3 while ! allBlocksReceived(blocksMap) do
4     startSendingAndReceiving(blocksMap);
5     updateDHT();
6 end
  /* get the data from blocksMap(deserialization)          */
7 retrieveData(blocsMap);
```

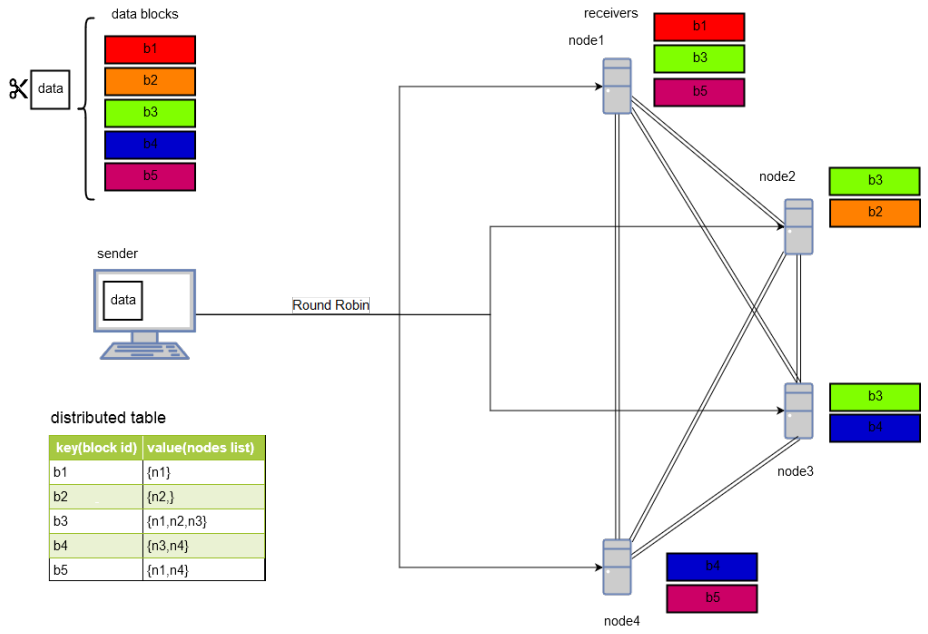

FIGURE 3.5 – Broadcast au bitTorrent(étape avancée)

3.4.4 Évaluation des différentes stratégies de broadcast

Le tableau suivant montre la dissemblance entre les trois stratégies de broadcast selon deux critères essentiels le volume de données à transférer et le nombre de noeuds impliqués dans le transfert, Chain broadcast est meilleur quand le volume de données et le nombre de noeuds sont limités, par contre, le Tree

critère / broadcast	Volume de données	Nombre de noeuds
Chain	limité	limité
Tree	limité	plus important
bitTorrent	données volumineuses	plus large

TABLE 3.1 – evaluation des stratégies de broadcast

broadcast est meilleur que le Chain quand le nombre de noeuds est plus important. Lorsque les données transférées sont volumineuses et le nombre de noeuds est plus large le broadcast au bit torrent donne de meilleurs résultats.

3.4.5 Shuffle

Contrairement au broadcast, le shuffle est un transfert de données de type plusieurs vers plusieurs (many to many). Le shuffle est utilisé dans un contexte où on veut faire une redistribution de données à partir d'un groupe de noeuds vers d'autre noeuds afin de continuer le traitement en se basant sur l'ensemble de données (des résultats) reçues. WSS est une implémentation du shuffle. Il s'agit d'un shuffle équilibré et pondéré, dont le but est de minimiser le temps d'achèvement c.-à-d. le moment ou le dernier récepteur se termine avec la récupération des données. WSS consiste à allouer des taux à chaque flux du transfert en utilisant un partage équitable selon le poids du flux de sorte que le poids du flux entre l'expéditeur et le récepteur est proportionnel à la quantité des données récupérée par le récepteur à partir de l'ensemble des expéditeurs.

Rapport-Gratuit.com

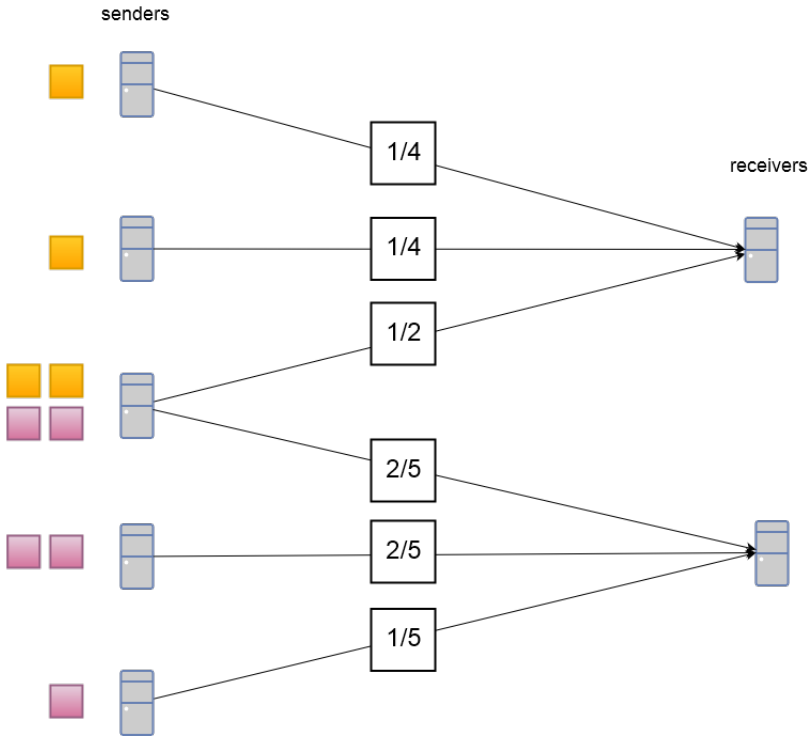

FIGURE 3.6 – Illustration du Shuffle

Algorithme 7 : Master side (shuffle)

```
1 input :configuration
2 receiveShuffleNeed();
3 sendNetworkRatios();
```

Algorithme 8 : sender side (shuffle)

```
1 input :configuration,data
2 sendData(data);
```

Algorithme 9 : receiver side (shuffle)

```
1 input :configuration
2 receiveNetworkRatios();
3 receiveData();
```

3.5 Mécanismes de vérification du réseau

Ce sont des mécanismes conçus pour suivre l'état des noeuds sur le réseau et faire au long du traitement des vérifications après chaque transfert de données. On a donc le mécanisme de battement de coeur et l'accusé de réception.

3.5.1 Mécanisme de battement du coeur

Heartbeat en anglais, est un mécanisme conçu afin qu'il tourne tout au long du traitement mais en arrière-plan. Il permet de vérifier à partir du noeud master l'état des noeuds (active ou passive). La vérification se fait périodiquement.

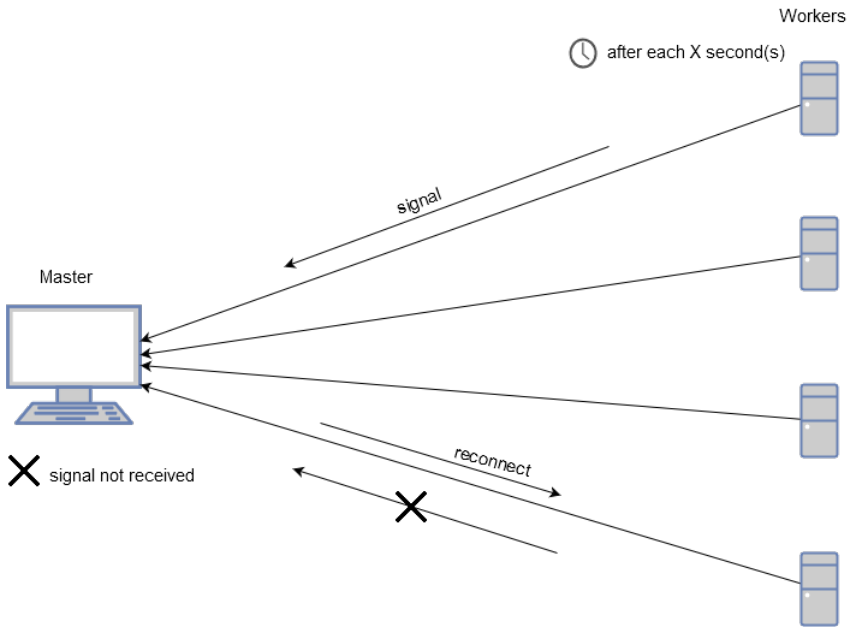

FIGURE 3.7 – Illustration du mécanisme battement de coeur

Toute machine concernée par le traitement envoie un signal au master après une période de temps, précisée dans le fichier de configuration. Le master reste à l'écoute des signaux, tout signal reçu d'un noeud est enregistré, il manifeste lors d'absence du signal d'un noeud et voir si c'est possible de rétablir la connexion à nouveau avec le noeud inactif.

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

3.5.2 Accusé de réception (Acknowlegement)

Pendant le traitement d'une requête, de nombreux objets importants sont envoyés par le master comme la requête, plan d'exécution ainsi que des commandes. A la réception il est primordial d'informer l'expéditeur de la bonne réception de l'objet afin de continuer le traitement. Le mécanisme accusé de réception, est un module conçu pour répondre à ce besoin.

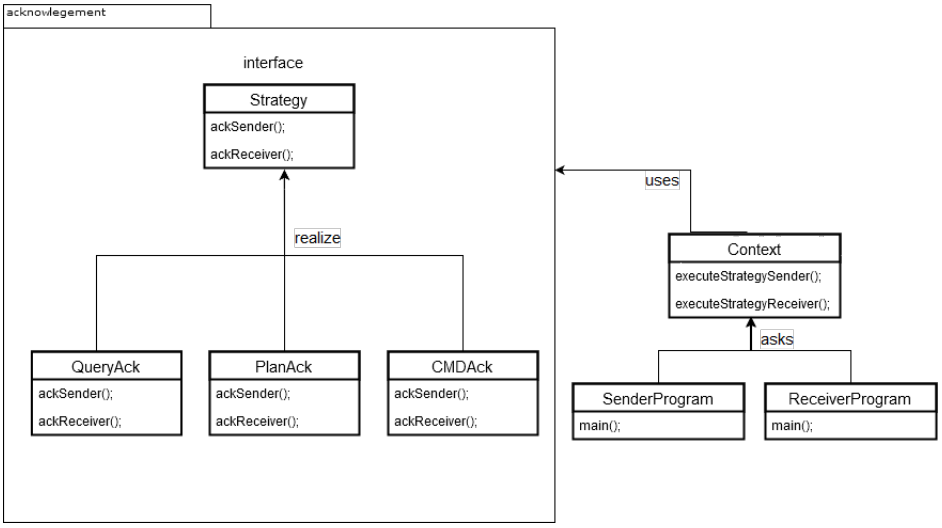

FIGURE 3.8 – Diagramme de classe "accusé de réception"

Ce module est implémenté selon le patron de conception stratégie (strategy design pattern) en considérant chaque accusé de réception comme une stratégie (ex. accusé de réception de la bonne réception de la requête, le plan d'exécution, les commandes).

Stratégie permet d'avoir des extensions sans faire des changements sur le corps du module. La figure ci-dessus illustre à travers un diagramme de classe le mécanisme accusé de réception.

3.6 Scénario d'utilisation du framework

La figure ci-dessous illustre un scénario d'exploitation de framework. Deux tâches qui s'exécutent en parallèle, le mécanisme battement du coeur fonctionne

en arrière plan pour s'assurer de l'état des workers. Chaque noeud worker envoie périodiquement un signal au noeud master. L'absence de signal déclenche un traitement de vérification au niveau du noeud master.

En premier plan, l'utilisation de broadcast pour diffuser la requête, le plan d'exécution et les commandes. A la réception des éléments diffusés, le noeud worker envoie un accusé de réception au master pour confirmer la bonne réception avec le master afin de continuer.

Afin de déclencher une tâche de shffule entre les noeuds workers, le noeud master communique les parties nécessaires de la bande passante avec les noeuds workers.

Enfin, les noeuds workers communiquent le résultat final avec l'application cliente.

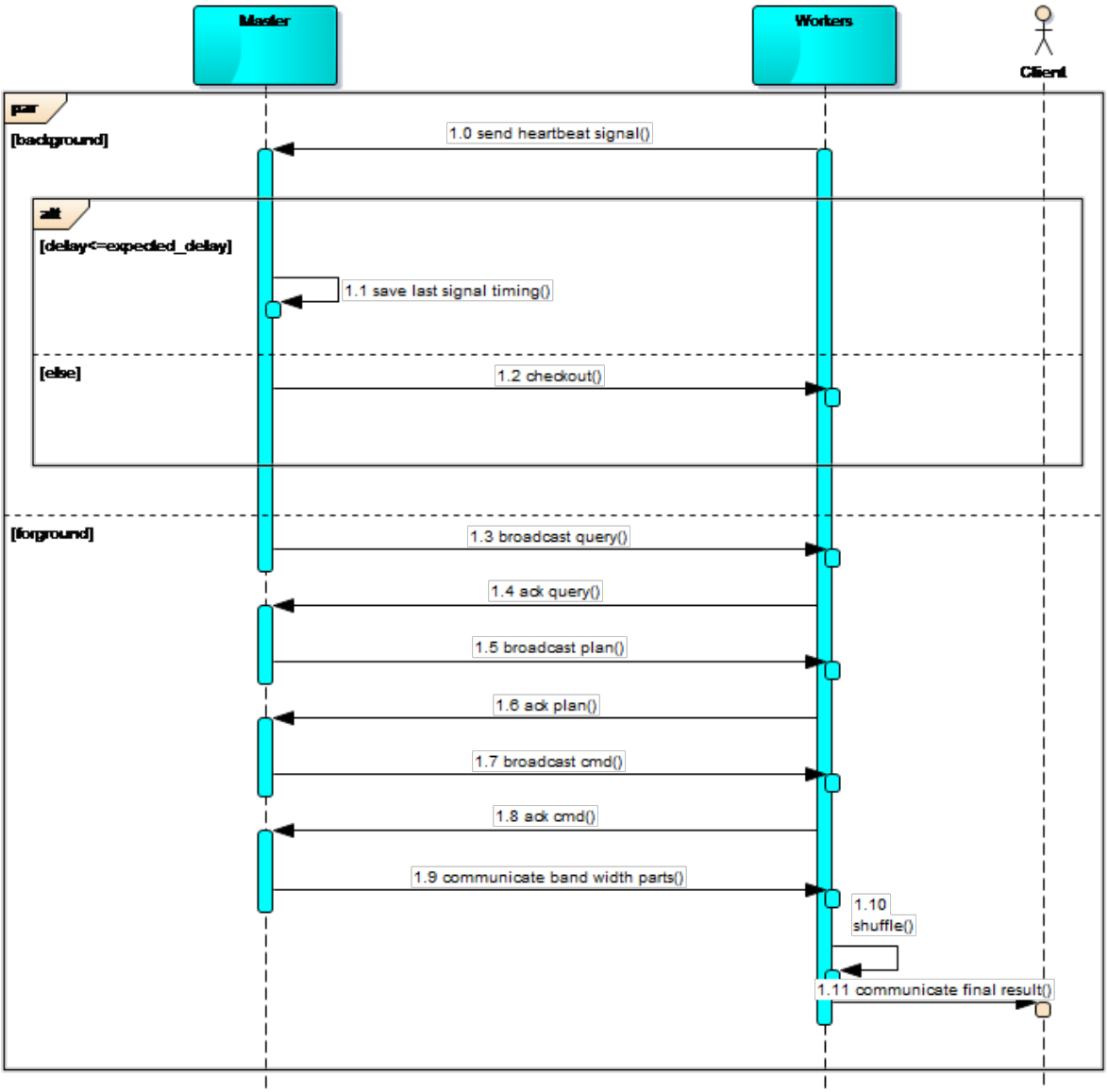

FIGURE 3.9 – Diagramme de séquence d'un scénario d'utilisation de framework

3.7 Outils et technologies utilisés

3.7.1 java

La technologie Java est à la fois un langage et une plate-forme de programmation. Java est un langage de programmation haut niveau orienté objet. Une grande partie de système existant est implémentée en java, ce qui nous a encouragés à continuer le développement avec ce langage. Le but derrière cette

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

décision est tout simplement de faciliter les taches avenir, comme la partie intégration du nouveau système dans le système existant. Java et un langage distribué, sécurisé et supporte le multithreading ce qui rend la tâche plus facile, cela nous permettait de se focaliser plutôt sur les aspects applicatifs.[32] Il est caractérisé par sa robustesse, la simplicité, dynamité et portabilité.

3.7.2 L'environnement de test

Nous avons pu nous appuyer sur un environnement cloud afin de d'assurer les taches de tests et d'expérimentations. Cet environnement est offert par l'université de Poitiers. Nous avons pu exploiter quinze machines virtuelles Ubuntu déployées avec Openstack. Ces machines sont dotées de 8 Go de RAM et 350 Go de disque dur.

3.7.3 PuTTY

PuTTY est un émulateur de terminal. Il permet aussi de détourener les obstacles réseaux en s'appuiyant sur le protocole SSH. L'usage de PuTTY au moment de développement et de test se résume dans la partie création des tunnels pour accéder à l'environnement de test, établir la première connexion pour permettre l'utilisation de superPuTTY et fileZilla par la suite.

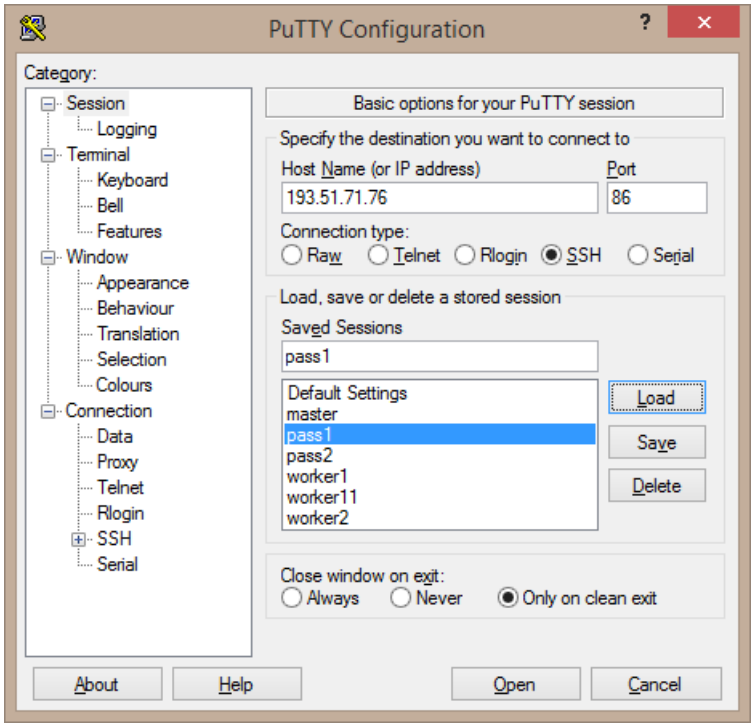

FIGURE 3.10 – Inteface PuTTY

3.7.4 SuperPuTTY

SuperPuTTY est une application Windows utilisée principalement comme un gestionnaire de fenêtre pour le client PuTTY SSH. Elle Permet d'intégrer plusieurs instances de terminal PuTTY dans une seule fenêtre en fournissant une interface avec des onglets, ou chaque onglet représente une connexion établie. SuperPuTTY nous a permis de gérer plusieurs machines en même temps en lançant des commandes simultanément sur ces machines. L'utilisation de cet outil nous a considérablement fait gagner plus de temps et nous a aidés à accélérer le développement.

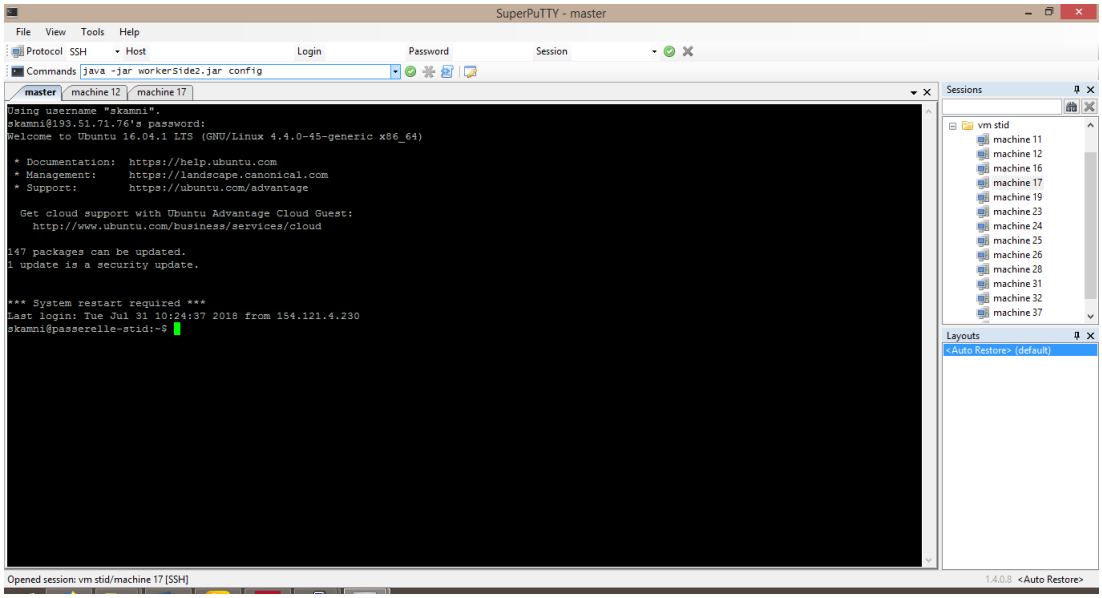

FIGURE 3.11 – Inteface SuperPuTTY

3.7.5 FileZilla

FileZilla est une application gratuite et multiplateforme , offrant un Client FTP. Filezilla permet d'accéder, transférer et gérer les fichiers à travers le protocole SFTP. Nous avons exploité cet outil pour charger les programmes exécutables sous forme de fichiers jar, les fichiers de configuration, ainsi que les fichiers contenant les données nécessaires pour effectuer des tests sur l'environnement d'expirementation.

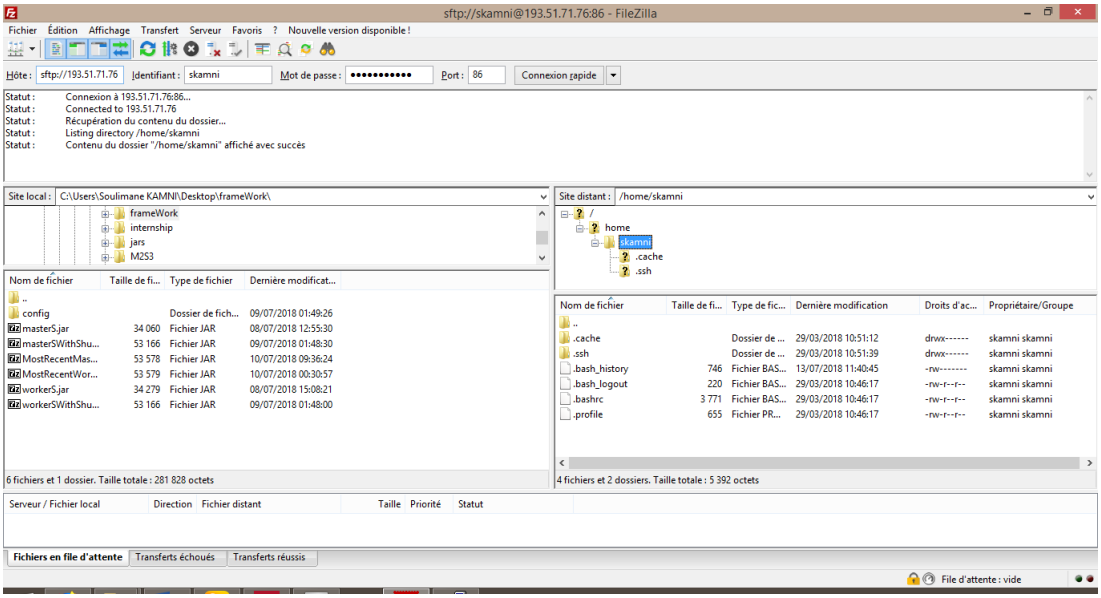

FIGURE 3.12 – Inteface fileZilla

3.7.6 Parallel-scp

Parallel-scp est un outil permettant de transférer / copier des fichiers sur plusieurs serveurs Linux distants en utilisant un seul terminal avec une seule commande.

```
$ parallel-scp -h workers jars/master.jar /home/ubuntu/jars
```

la commande parallel-scp prend comme arguments un fichier contenant les adresses IP des machines destinataires, le fichier ou bien le répertoire à transférer et l'emplacement dans lequel on veut placer le fichier sur les machines cibles.

3.7.7 Git

Git est un système de gestion de version pour suivre la trace des changements de fichiers informatiques et coordonner le travail sur ces fichiers. Il est principalement utilisé pour la gestion de code source dans le développement logiciel, mais il peut être utilisé pour garder la trace de changements de n'importe quel ensemble de fichiers. Comme un système de commande de révision distribué, il vise la vitesse et l'intégrité de données.

3.7.8 Git LIAS

Git LIAS est un dépôt de code source pour héberger les développements des logiciels du laboratoire LIAS. Il propose les même services et fonctionnalités que le fameux dépôt de code source Github. Il permet entre autre de cloner un projet sur la machine local afin de l'améliorer, aussi d'effectuer des commandes comme push et pull sur le dépôt hébergé à travers le terminal git ainsi que la version Git intégrée dans l'EDI éclipse, partager les projets avec les membres de l'équipe et gérer les accès au code source. Il permet aussi de visualiser chaque commit.

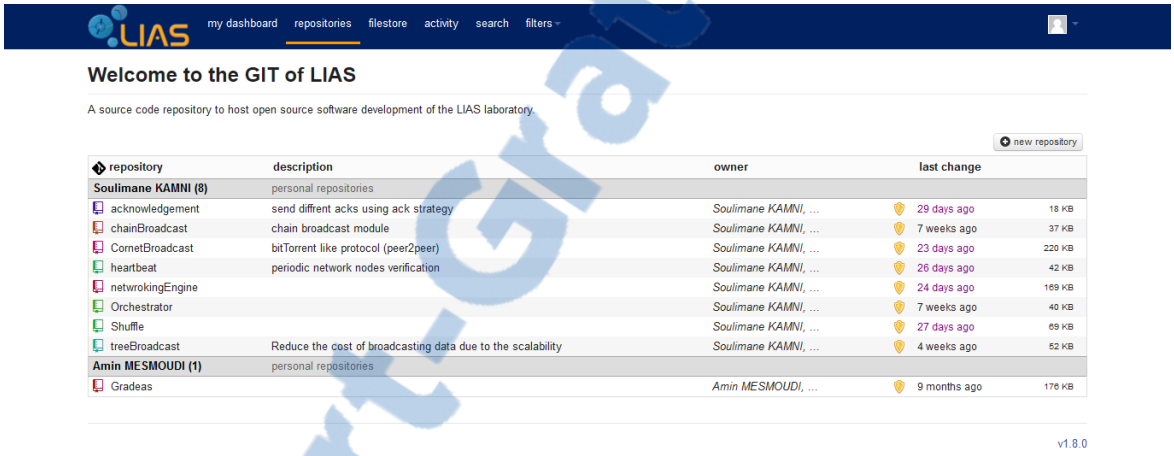

FIGURE 3.13 – Inteface Git LIAS

3.7.9 Gestion de projet avec Trello

Trello est un outil en ligne, ergonomique et gratuit. Nous utilisons cet outil pour organiser nos tâches, consigner les informations essentielles et tenir un planning avec tous les membres de notre équipe. Grâce à cet outil nous simplifions le suivi de notre projet. Ce service en ligne nous aide à mieux organiser nos activités. Nous consignons nos tâches sur des post-it " cards " que nous accrochons sur un panneau " board " en les rangeant dans des listes de gauche à droite :

Rapport-Gratuit.com

— Les tâches à faire " To Do ".

— Les tâches en cours de réalisation " Doing ".

— Les taches réalisées " Done ".

Les colonnes contiennent autant de post-it que nous le souhaitons. Sur chaque post-it, en plus de renseigner son contenu, nous pouvons :

— Indiquer la liste des points à vérifier pour cette tâche.

— Epingler des documents bureautique et multimédia.

— préciser une date limite.

— affecter une tâche à un membre de l'équipe.

Au fur et à mesure de l'avancement de nos tâches, nous déplaçons les post-it d'une colonne à l'autre. Le principe de base est très intuitif et le service regroupe plusieurs options qui peuvent contribuer à l'amélioration de la productivité de l'équipe.

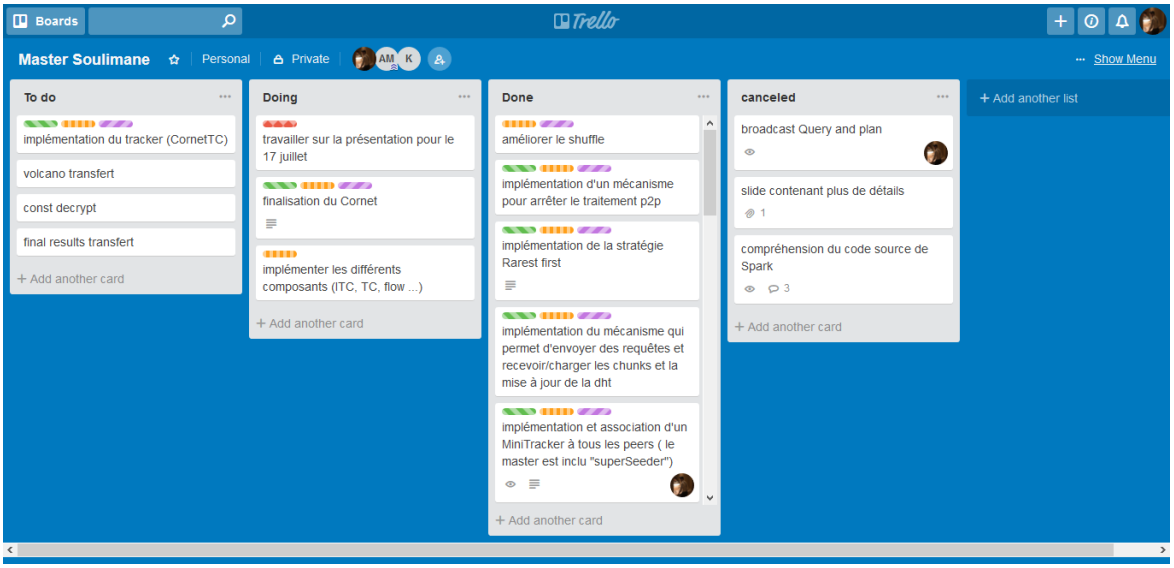

FIGURE 3.14 – Inteface Trello

3.8 Les étapes à suivre pour effectuer les tests

A chaque fois qu'on finisse le développement d'une fonctionnalité, on est obligé à procéder à des tests sur l'environnement dédié car parfois ce n'est pas possible de tester la nouvelle fonctionnalité en local. C'est pour cela que nous

avons suivi une procédure bien précise pendant la phase de test, cette procédure se déroule en cinq étapes indispensables :

1. Accéder à la plateforme de test : c'est une étape de connexion. Dans cette étape, on utilise l'outil PuTTY pour ouvrir la passerelle entre la machine local et l'environnement de test, ensuite on utilise SuperPuTTY pour se connecter aux différentes machines distantes et enfin se connecter avec FileZilla à une machine parmi l'ensemble de machines (généralement celle qui est considérer comme Master) afin de charger les fichiers nécessaires pour le test. Cette étape n'est demandée qu'une seule fois.
2. Exportation des programmes exécutables en fichiers jar à l'aide de l'IDE eclipse.
3. Charger les fichiers exécutables (.jar) ainsi que les fichiers de configuration et les fichiers contenant des données pour le test (s'il est nécessaire) en utilisant l'outil fileZilla.
4. Dupliquer les fichiers sur l'ensemble des machines à partir de la machine contenante des fichiers à l'aide de la commande parallel-scp.
5. Lancer les programmes en spécifiant les fichiers de configuration en utilisant SuperPuTTY.
6. Repasser par les étapes de 2 à 5 après correction de problème quand le résultat et différent du résultat attendu.

3.8.1 Conclusion

En conclusion, le Framework " network management " et un outil permettant de transférer des données de volumes divers sur le réseau entre plusieurs noeuds. Il suffit juste de paramétrer l'outil en bénéficiant de la partie configuration qui facilite la tâche aux utilisateurs, qui peut l'exploiter sans avoir besoin de penser à l'implémentation bas niveau qui est déjà traitée et qui fonctionne en arrière-plan pendant l'invocation des fonctionnalités du Framework.

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

Chapitre 4

CONCLUSION GÉNÉRALE

4.1 Conclusion

En conclusion, J'ai pu réalisé mon projet de fin d'études du Master Génie Logiciel en France. C'était un stage de recherche au sein de laboratoire LIAS. Pendant ce stage de 5 mois, j'ai pu mettre en pratique mes connaissances théoriques et pratiques acquises durant ma formation à l'université de Tlemcen, tout en étant confronté aux difficultés réelles du monde du travail et de la recherche scientifique.

Après ma rapide intégration dans l'environnement du laboratoire, j'ai eu l'occasion de réaliser plusieurs tâches non seulement qui ont été affecté par mon maître de stage mais aussi des tâches introduites par moi-même.

Ce stage était très enrichissant pour moi, car il m'a permis de découvrir le domaine de la recherche scientifique en big data, ses technologies et contraintes. Il m'a permis d'avoir ma propre contribution à travers mes compétences en conception et développement logiciel. Ce stage m'a aussi permis de comprendre que le travail dans l'entreprise n'était pas le plus adapté à mon profil et que la recherche scientifique me passionne plus. Je préfère ainsi m'orienter vers un poste lié au secteur académique.

Côté personnel, et grâce à cette opportunité, j'ai rencontré des gens d'un peu partout dans le monde. Ainsi j'ai pu créer des relations et des contacts avec des doctorants et des stagiaires, ce qui m'a permet d'apprendre autant de choses ainsi communiqué des sujets divers et faire des échanges culturels. Nous avons pratiqué autant d'activités sportives et culturelles ensemble.

Je suis très content de l'ensemble des aspects de ce stage. Faute de temps je n'ai pu résoudre tous les problèmes posés ni approfondir suffisamment certains points restés en suspens, mais l'éventuelle situation au cours de l'année me laisse l'espoir de continuer à contribuer à l'aboutissement du projet.

4.2 Perspectives

Je peux considérer qu'on a avancé sur une grande partie de nos objectifs fixés afin de satisfaire le besoin de l'équipe (IDD). Il reste toujours un espace pour les améliorations future y compris :

— Travailler beaucoup plus sur la tolérance au pane de notre solution.
— Ajouter d'autres mécanismes émergés récemment.
— La gestion inter et intra-transfert.
— Réalisation des APIs pour permettre l'utilisation de Framework en accédant aux fonctionnalités à l'aide des interfaces uniquement.

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

Bibliographie

[1] R SUGANYA, S RAJARAM, A SHEIK ABDULLAH, K. ABD-ELMONIEM, A. YOUSSEF, Y. KADAH, J ACZEL, Z DAROCZY, A AHMADIAN, B AIAZZI et al., « Big Data : Challenges and Opportunities, Big Data Computing », in *Big Data in Medical Image Processing*, 9, t. 49, McGraw-Hill New York, 2018, p. 1-45.

[2] X. WU, X. ZHU, G. Q. WU et W. DING, « Data mining with big data », *IEEE Transactions on Knowledge and Data Engineering*, 2014, ISSN : 10414347. DOI : 10.1109/TKDE.2013.109. arXiv : PAI.

[3] S WANG, H. J. WANG, X. P. QIN et X ZHOU, « Architecting big data : Challenges, studies and forecasts », *Jisuanji Xuebao/Chinese Journal of Computers*, 2011, ISSN : 0254-4164. DOI : 10.3724/SP.J.1016.2011.01741.

[4] Ž. IVEZIĆ, A. J. CONNOLLY et M. JURIĆ, « Everything we'd like to do with LSST data, but we don't know (yet) how », in *Proceedings of the International Astronomical Union*, 2016, ISBN : 0000000000000. DOI : 10.1017/S1743921316013156. arXiv : 1612.04772.

[5] M. CHOWDHURY, M. ZAHARIA, J. MA, M. I. JORDAN et I. STOICA, « Managing Data Transfers in Computer Clusters with Orchestra », *SIGCOMM Comput. Commun. Rev.*, t. 41, n° 4, p. 98-109, août 2011, ISSN : 0146-4833. adresse : http://doi.acm.org/10.1145/2043164.2018448.

[6] E. F. CODD, « A relational model of data for large shared data banks », *Communications of the ACM*, t. 13, n° 6, p. 377-387, 1970.

[7] D. D. CHAMBERLIN, M. M. ASTRAHAN, M. W. BLASGEN, J. N. GRAY, W. F. KING, B. G. LINDSAY, R. LORIE, J. W. MEHL, T. G. PRICE, F. PUTZOLU et al., « A history and evaluation of System R », *Communications of the ACM*, t. 24, n° 10, p. 632-646, 1981.

[8] R. BEAKTA, « Big Data And Hadoop : A Review Paper », t. 2, jan. 2015.

[9] S. T, « Application of Big Data in Data Mining », *in International Journal of Emerging Technology and Advanced Engineering*, t. 3, n° 7, 2013.

[10] IBM, *IBM Big Data analytics HUB*. adresse : www.ibmbigdatahub.com/infographic/four-vs-big-data.

[11] E. F. CODD, « A relational model of data for large shared data banks », *Communications of the ACM*, 1970, ISSN : 00010782. DOI : 10.1145/362384.362685. arXiv : arXiv:1011.1669v3.

[12] D. BITTON, H. BORAL, D. J. DEWITT et W. K. WILKINSON, « Parallel Algorithms for the Execution of Relational Database Operations », *ACM Trans. Database Syst.*, t. 8, n° 3, p. 324-353, sept. 1983, ISSN : 0362-5915. adresse : http://doi.acm.org/10.1145/319989.319991.

[13] D. J. DEWITT, R. H. GERBER, G. GRAEFE, M. L. HEYTENS, K. B. KUMAR et M MURALIKRISHNA, « GAMMA - A High Performance Dataflow Database Machine », *Proc 12th International Conference on VLDB*, 1986, ISSN : 978-1-60558-551-2.

[14] T. T. P. GROUP, « A Benchmark of NonStop SQL on the Debit Credit Transaction (Invited Paper) », in *SIGMOD Conference*, 1988.

[15] W. ALEXANDER et G. COPELAND, « Process and Dataflow Control in Distributed Data-intensive Systems », *SIGMOD Rec.*, t. 17, n° 3, p. 90-98, juin 1988, ISSN : 0163-5808. DOI : 10.1145/971701.50212. adresse : http://doi.acm.org/10.1145/971701.50212.

[16] M. STONEBRAKER, S. MADDEN, D. J. ABADI, S. HARIZOPOULOS, N. HACHEM et P. HELLAND, « The End of an Architectural Era (It's Time for a Complete Rewrite) », *Vldb*, 2007, ISSN : 1595936491. DOI : 10.1080/13264820701730900. arXiv : arXiv:1011.1669v3.

[17] S. CHAUDHURI et G. WEIKUM, « Rethinking Database System Architecture : Towards a Self-Tuning RISC-Style Database System », *Proceedings of the International Conference on Very Large Data Bases*, 2000.

[18] T. NEUMANN et G. WEIKUM, « RDF-3X : a RISC-style engine for RDF », *Proceedings of the VLDB Endowment*, 2008, ISSN : 21508097. DOI : 10.1145/1453856.1453927.

[19] M. STONEBRAKER, D. J. ABADI, A. BATKIN, X. CHEN, M. CHERNIACK, M. FERREIRA, E. LAU, A. LIN, S. MADDEN, E. O'NEIL, P. O'NEIL, A. RASIN, N. TRAN et S. ZDONIK, « C-store : a column-oriented DBMS », *VLDB Conference*, 2005, ISSN : 09295666. DOI : 10.1007/BF02443652.

[20] A. LAMB, M. FULLER, R. VARADARAJAN, N. TRAN, B. VANDIVER, L. DOSHI et C. BEAR, « The vertica analytic database », *Proceedings of the VLDB Endowment*, 2012, ISSN : 21508097. DOI : 10.14778/2367502.2367518. arXiv : 1208.4173.

[21] M. ZAHARIA, M. J. FRANKLIN, A. GHODSI, J. GONZALEZ, S. SHENKER, I. STOICA, R. S. XIN, P. WENDELL, T. DAS, M. ARMBRUST, A. DAVE, X. MENG, J. ROSEN et S. VENKATARAMAN, « Apache Spark : a unified engine for big data processing », *Communications of the ACM*, 2016, ISSN : 00010782. DOI : 10.1145/2934664.

[22] A. ALEXANDROV, R. BERGMANN, S. EWEN, J. C. FREYTAG, F. HUESKE, A. HEISE, O. KAO, M. LEICH, U. LESER, V. MARKL, F. NAUMANN, M. PETERS, A. RHEINLÄNDER, M. J. SAX, S. SCHELTER, M. HÖGER, K. TZOUMAS et D. WARNEKE, « The Stratosphere platform for big data analytics », *VLDB Journal*, 2014, ISSN : 0949877X. DOI : 10.1007/s00778-014-0357-y.

[23] H. S. BHOSALE et D. P. GADEKAR, « A Review Paper on Big Data and Hadoop », *International Journal of Scientific and Research Publications*, 2014, ISSN : 2250-3153.

[24] B. L, *Apache Spark : histoire et avantages du moteur Big Data*, 2018. adresse : https://www.lebigdata.fr/apache-spark-tout-savoir.

[25] R. BRUCHEZ, *Les bases de donnees NoSQL et le big data : comprendre et mettre en oeuvre*. Eyrolles, 2015.

[26] M. CHOWDHURY, M. ZAHARIA, J. MA, M. I. JORDAN et I. STOICA, « Managing data transfers in computer clusters with orchestra », *ACM SIGCOMM Computer Communication Review*, 2011, ISSN : 01464833. DOI : 10.1145/2043164.2018448.

[27] *lanTorrent*. adresse : http://www.nimbusproject.org.

[28] *Apache Hadoop*. adresse : http://hadoop.apache.org.

[29] J. DEAN et S. GHEMAWAT, « MapReduce : Simplified Data Processing on Large Clusters », *Proceedings of 6th Symposium on Operating Systems Design and Implementation*, 2004, ISSN : 00010782. DOI : 10.1145/1327452.1327492. arXiv : 10.1.1.163.5292.

[30] S. SARMADY, « A Survey on Peer-to-Peer and DHT », *Computing*, 2010. arXiv : 1006.4708.

[31] A. LEGOUT, G. URVOY-KELLER et P. MICHIARDI, « Rarest first and choke algorithms are enough », *ACM Internet Measurement Conference (IMC)*, 2006. DOI : 10.1145/1177080.1177106. arXiv : 0609026 [cs].

[32] *About the Java Technology*. adresse : https://docs.oracle.com/javase/tutorial/getStarted/intro/definition.

Résumé:

Afin de valider mon Master génie Logiciel, j'ai eu l'opportunité d'effectuer un stage dans le laboratoire de recherche LIAS. J'ai pu travailler sur un sujet en relation avec le Big Data, qui représente un défi non seulement pour le monde socio-économique mais aussi pour la recherche scientifique. L'objectif de mon stage était de proposer un cadre applicatif permettant de gérer les différents transferts réseaux de données d'une manière transparente et efficace. Les outils proposés devraient être intégrés au sein du système de gestion de données parallèle QDAG, qui est en cours de développement au sein de l'équipe IDD du LIAS.

Durant mon stage, j'ai commencé par étudier le fonctionnement du système QDAG. J'ai analysé d'une manière minutieuse chaque composant nécessitant un mécanisme de transfert de données. J'ai pu constater que les composants de QDAG font appel à deux (2) types de transfert, à savoir le Shuffle (avec ses deux cas spéciaux le Collector et le distributor) et le BroadCast. Après cette étape d'analyse du besoin, j'ai fait un état de l'art sur les mécanismes de transfert de données dans le cadre du cluster computing. Je me suis intéressé plus particulièrement au framework Orchestra [25].J'ai par la suite proposé non seulement une nouvelle conception et implémentation de ce framework mais aussi des extensions nécessaires au système QDAG. Avec les doctorants du LIAS, j'ai réussi à intégrer mon travail dans le moteur d'exécution de QDAG.

Mot-clé: Big data, Système de gestion de données, environnement parallèle, transfert de données, optimisation des traitements, QDAG.

Abstract:

In order to validate my Masters in Software Engineering, I had the opportunity to perform an internship in the LIAS research laboratory. I was able to work on a topic related to Big Data, which represents a challenge not only for the socio-economic world but also for scientific research. The objective of my internship was to propose an application framework to manage the various data network transfers in a transparent and efficient way. The proposed tools should be integrated into the QDAG parallel data management system, which is being developed within the LIAS IDD team.

During my internship, I started by studying how the QDAG system works. I have carefully analyzed each component requiring a data transfer mechanism. I could notice that the components of QDAG use two (2) types of transfer, namely the Shuffle (with its two special cases the collector and distributor) and Broadcast. After this step of requirement analysis, I made a state of the art on the data transfer mechanisms as part of the cluster computing. I was particularly interested in the orchestra framework [25]. I then proposed not only a new design and implementation of this framework but also the necessary extensions for QDAG system. With the LIAS PhD students, I could integrate my work into QDAG's execution engine.

Keywords: big data, Data Management System, parallel environment, data transfer, processing optimization, QDAG.

ملخص:

من أجل الحصول على شهادة الماستر في هندسة البرمجيات ، أتيحت لي الفرصة لإجراء تدريب داخلي في مختبر الأبحاث LIAS . استطعت العمل على موضوع متعلق بالبيانات الضخمة ، وهو ما يمثل تحديًا ليس للعالم الاجتماعي-الاقتصادي فحسب ، بل أيضًا للبحث العلمي . كان الهدف من التدريب الداخلي اقتراح إطار عمل لإدارة عمليات النقل الشبكي للبيانات المختلف بطريقة شفافة وفعالة . يجب دمج الأدوات المقترحة في نظام إدارة البيانات الموازية QDAG ، والذي يتم تطويره من قبل فريق LIAS IDD.

خلال تدريبي، بدأت بدراسة كيفية عمل النظام QDAG . قمت بتحليل دقيق لكل مكون يتطلب آلية نقل البيانات لقد لاحظت أن مكونات QDAG تستخدم نوعين (2) من آليات النقل shuffle، (مع حالتيه الخاصتين المجمع والموزع) وbroadcast.

بعد تحليل المتطلبات قمت بعمل بحث حول آليات نقل البيانات في الحوسبة العنقودية حيث صببت اهتمامي على اطار العمل Orchestra[25].

بعد ذلك ، اقترحت ليس فقط تصميمًا جديدًا وتطبيقًا لهذا الإطار بل أيضًا الإضافات اللازمة لنظام QDAG. كما تمكنت برفقة طلاب الدكتوراه LIAS من دمج عملي في محرك التنفيذ QDAG .

الكلمات المفتاحية: البيانات الضخمة، نظام إدارة البيانات، بيئة متوازية، تحسين المعالجة، QDAG.

Rapport-Gratuit.com