IUT Nancy Charlemagne
Université de Lorraine
2 ter Boulevard Charlemagne
54052 Nancy Cedex
Dépt. Informatique

Recherche et développement autour de nouvelles technologies pour la manipulation de grandes masses de données.

Rapport de stage LP CISIIE
INSU CNRS Observatoire astronomique Strasbourg

Alejandro Daniel SERNA FLORES

IUT Nancy Charlemagne

Université de Lorraine
2 ter Boulevard Charlemagne
54052 Nancy Cedex
Dépt. Informatique

Recherche et développement autour de nouvelles technologies pour la manipulation de grandes masses de données.

Rapport de stage LP CISIIE

INSU CNRS Observatoire astronomique Strasbourg
11 rue de l'Université 67000 Strasbourg

Alejandro Daniel SERNA FLORES

André SCHAAFF

TABLE DES MATIÈRES

REMERCIEMENTS

Tout d'abord, je tiens a remercier et à témoigner toute ma reconnaissance à ma famille laquelle m'a supporté depuis que j'ai commencé mes études et qui m'ont toujours motivé pendant cette expérience grande en France.

Je tiens à remercier tout particulièrement à Monsieur Samuel Cruz Lara qui m'a fourni l'opportunité d'étudier la licence professionnelle CISSIE et qui m'a aidé depuis le début de mon séjour en France.

Je tiens à remercier spécialement à mon chère amie Elisa qui m'a encouragé et motivé tout le temps.

Je remercie également Monsieur Andre Schaaff, mon tuteur de stage pour m'avoir accueillir au sein de l'Observatoire, et également Thomas Boch et Gilles Landais qui m'ont co-encadré. apporté leur aide et des conseils lors du déroulement du stage.

Enfin, Je remercie à Monsieur Christophe Bouthier qui répondait mes questions pendant le stage.

INTRODUCTION

À fin de conclure la Licence professionnelle CISIIE, j'ai réalisé mon stage au sein de l'Observatoire Astronomique de Strasbourg du 7 avril au 27 juin 2014.

Ce stage s'adressait à un étudiant intéressé par la recherche et l'apprentissage de nouvelles technologies liées aux grands quantités de données. Donc, cette opportunité m'a permis d'acquérir dans nouvelles connaissances et aussi que l'expérience dans ce domaine.

À l'Observatoire, l'amélioration continue de ses services a une grande importance en ce qui concerne la répartition de données, accès et recherche dans ces données. Ils sont souvent confrontés à plusieurs problématiques liées au traitement de données puisque la quantité a grandi énormément depuis des années.

Ainsi, pour avoir un aperçu d'un futur, le sujet stage s'agit de chercher et tester de différents technologies capables de résoudre, au moins en partie cette problématique.

Ensuite, une présentation de l'Observatoire, ses services et le sujet du stage auront lieu d'une façon plus détaillée au début du rapport.

I. L'OBSERVATOIRE DE STRASBOURG

L'Observatoire astronomique de Strasbourg est un Observatoire des Sciences de l'Univers (OSU), une école interne et UFR de l'Université de Strasbourg, ainsi qu'une Unité Mixte de Recherche entre l'Université et le CNRS.

Il a été construit en 1881 sur le campus principal et historique de l'Université de Strasbourg et il s'agit du troisième observatoire construit à Strasbourg. Il dispose de trois bâtiments: Le bâtiment Nord, nommé La Coupole, abrite la lunette qui est la troisième plus grande lunette de France par rapport à sa taille. Le bâtiment Sud contient le service informatique et les salles serveurs dédiées à l'Observatoire., et le bâtiment Est héberge le Centre de Données astronomiques de Strasbourg et son équipe de travail.

L'établissement accueille trois équipes de recherches:

- l'équipe « **hautes énergies** », étudiant l'astrophysique des hautes énergies, telles que les rayons-X. L'équipe Hautes Énergies est partie prenante du Survey Science Center du satellite XMM*-Newton ;

- l'équipe « **galaxie** » : les activités de l'équipe couvrent des problèmes variés concernant l'histoire et la formation des galaxies ainsi que l'étude des populations stellaires qui constituent ces galaxies;

- Le « **Centre de données astronomiques de Strasbourg** » (CDS) dans lequel j'ai réalise le stage.

a). Le Centre de Données astronomiques de Strasbourg :

Le CDS est un centre de données créé en 1972 par l'Institut National d'Astronomie et de Géophysique, désormais nommé Institut National des Sciences de l'Univers, en accord avec l'ancienne université Louis Pasteur devenue maintenant l'Université de Strasbourg.

Ce centre est destiné à récolter et distribuer dans le monde entier des données astronomiques. Il est l'hôte de la base de référence mondiale pour l'identification d'objets astronomiques. Il a 3 objectifs:

- Réunir les informations utiles concernant les objets astronomiques au format numérique;
- Distribuer ces mêmes informations dans la communauté astronomique mondiale;
- Mener des recherches utilisant ces données collectées.

Le CDS fonctionne grâce à trois différents services, Simbad, Vizier et Aladin. Ces trois outils sont liés et s'utilisent conjointement.

b). Simbad :

Simbad est une base de données de référence dans le monde d'objets astronomiques. Il permet d'accéder rapidement aux propriétés de base (coordonnées, magnitudes, la parallaxe...) de chaque objet recensé dans un catalogue astronomique grâce au nom ou à l'identifiant de cet objet. Ce service propose également un résolveur de noms, ce qui permet d'avoir connaissance de tous les autres noms de l'objet choisi, car de nombreuses normes existent.

c). Vizier :

Le service Vizier collecte les catalogues et les tables publiés dans les journaux académiques, et comptabilise plus de onze mille catalogues et près de douze mille tables à ce jour.

Vizier est utilisé afin de sélectionner et d'extraire puis de formater des valeurs retournées en fonction des critères de re cherche. Pour offrir de meilleures performances lors de l'accès à des catalogues disposant de très gros volumes comme le 2MASS, qui est le catalogue le plus utilisé dans le monde, un accès particulièrement optimisé est intégré à Vizier. Il atteint des pics de plus de 2 millions de requêtes par jour, et fonctionne en moyenne à plus de 370 .000 demandes par jour soit 12 millions par mois.

d). Aladin :

Aladin est la base un serveur d'image associé à sa propre base de données et est aussi un logiciel développé par le Centre de Données astronomiques de Strasbourg.

Ce logiciel a été conçu en Java en 1999, il offre une interface d'accès à la base de données. Depuis, l'application a été considérablement améliorée, devenant par ailleurs un client Aladin, permettant d'avoir un atlas interactif du ciel, en récupérant les images de la base de données Aladin, mais aussi d'autres bases de données d'images astronomiques.

L'atlas permet de visualiser le ciel sous différents moyens d'observation tels que les rayons X ou les infrarouges.
L'interaction avec cet atlas s'effectue en cliquant sur un objet, ce qui permet d'en obtenir les informations.

Il est aussi possible de voir le ciel sous forme de globe, grâce à la technologie HEALPix développée par la National Aeronautics and Space Administration (NASA), qui recrée une sphère par projection en utilisant divers algorithmes appliquées à des images planes.

Ce service est interrogé plus de 18.000 fois par jour représentant plus de 570.000 requêtes par mois.

e). L'Observatoire Virtuel :

Un Observatoire Virtuel, ou VO pour Virtual Observatory, est un ensemble de centres de données qui regroupe les données astronomiques, les logiciels et les capacités de calcul de chacun de ces centres de données.

L’Observatoire Astronomique de Strasbourg fait partie de l’Observatoire Virtuel de France et a une très grande importance dans cet observatoire grâce aux trois services Simbad, Vizier et Aladin qui sont utilisés dans le monde entier. Il y a plusieurs projets d’observatoires virtuels à travers le monde, le plus important étant le projet International Virtual Observatory Alliance (IVOA) initié en 2002 et qui regroupe les observatoires virtuels du monde entier comme ceux des États-Unis, de Chine, de Russie, du Japon et bien sûr de France.

Ce consortium a été créé pour faciliter la coordination internationale des observatoires et pour permettre la collaboration de centres de données afin de développer et de déployer des logiciels, des systèmes et des structures primordiales pour l’utilisation à l’international d’archives de données astronomiques parmi les Observatoires Virtuels de cette communauté.

Pour cet service j'ai effectué la deuxième partie de mon stage

II. OUTILS ET LOGICIELS:

a). Langages de développement mis en œuvre :

Pendant le stage me suis servi de plusieurs langages :

Java : un langage de programmation orienté objet. Son objectif principal est que les logiciels écrits dans ce langage doivent être très facilement portables sur plusieurs systèmes d'exploitation tels que UNIX, Windows, Mac OS ou GNU/Linux.

JavaScript : un langage de programmation de scripts principalement utilisé dans les pages Web interactives mais aussi côté serveur (NodeJS par example). Ce langage est basé sur le standard ECMA.

XML : XML (Extensible Markup Language) est un langage de balisage générique qui dérive du SGML. Cette syntaxe est dite « extensible » car elle permet de définir différents espaces de noms, c'est-à-dire des langages avec chacun leur vocabulaire et leur grammaire, comme XHTML, XSLT, RSS, SVG, etc. Elle est reconnaissable par son usage des chevrons (< >) encadrant les balises. L'objectif initial est de faciliter l'échange automatisé de contenus complexes comme des arbres ou texte riche.

b). Plate formes et services:

Apache Hadoop : Hadoop est un projet libre conçu en Java et basé sur le patron d'architecture de programmation MapReduce, pour les opérations analytiques et à grande échelle sur un grand cluster de serveurs, et aussi le système de fichiers distribués.

Son but est de faciliter la création d'applications distribués et extensibles (scalables), et de leur permettre de travailler avec de milliers de nœuds et des

pétaoctets de données.

Hadoop est devenu synonyme de « Big Data », il est tout à la fois l'un des plus connu pour répondre à certaines problématiques du Big Data et un composant central de son architecture<

Cependant, parfois on se trompe souvent sur sa nature laquelle a été consacrée pour le traitement de grandes quantités de données<
Voici les outils qui composent son architecture :

- Le **HDFS** est un système de fichiers distribués, conçu pour stocker de très gros volumes de données sur un grand nombre de machines équipées de disques durs banalisés, Il permet l'abstraction de l'architecture physique de stockage, afin de manipuler un système de fichiers distribué comme s'il s'agissait d'un disque dur unique.

- **MapReduce** est un algorithme qui fait des calculs parallèles, souvent distribués, de données potentiellement très volumineuses, permettant de manipuler de grandes quantités de données en les distribuant dans le cluster pour être traitées.

Il existe divers sous-projets basés sur Hadoop pour répondre aux différents besoins, et aussi pour simplifier certaines tâches qui peuvent être assez compliquées. Par contre, je ne mentionnerai que celles que nous avons testées :

Apache Hive :
Hive est un logiciel conçu pour faciliter l'interrogation de grands ensembles de données qui se trouvent dans le cluster Hadoop.
Il fournit un langage SQL-like (appelé HiveQL) qui est basé sur le langage SQL, permettant de faciliter la création de programmes MapReduce aux utilisateurs qui ont déjà connaissances sur le langage SQL. Cependant, Il faut garder à l'esprit que les requêtes tendent à avoir une latence élevée et que Hive n'est pas conçu pour les traitements de transactions en ligne, ni pour les requêtes en

temps réel ni pour les mises à jour.

Apache Pig :

Pig ou le langage Pig Latin est un framework de haut niveau destiné à développer des « scripts », ce qui est derrière est MapReduce, à la différence de Hive, Pig tende à être plus analytique et par conséquent, le temps de réponse augment considérablement.

Elasticsearch :

Elasticsearch est un moteur (en cet cas un serveur) de recherche libre basé sur Apache Lucene, conçu pour l'analyse et la recherche de données indexées en temps réel sous un système de machines distribuées, grâce à cela, il est utilisé comme une base de données des documents (NoSQL).

c). Logiciels:

Oracle VM VirtualBox :

VirualBox est un logiciel conçu pour la virtualisation des architectures x86/amd64 dont il est possible d'installer un système d'exploitation virtualisé sur une machine physique.

Ubuntu Server :

Ubuntu est un système d'exploitation libre fondé sur la distribution Linux Debian, ce système d'exploitation est constitué de logiciels libres, et est disponible gratuitement.

Cette version du logiciel, par défaut, n'installe pas un bureau graphique et je m’en suis servi pour mes installations sous VirtualBox.

CURL :

cURL (Client URL Request Library) est une interface en ligne de commande destinée à récupérer le contenu d'une ressource accessible par un réseau informatique. La ressource est désignée à l'aide d'une URL et doit être d'un type supporté par le logiciel. Le logiciel permet de créer ou modifier une ressource, il peut ainsi être utilisé en tant que client REST.

Navigateurs d'Internet :

Les navigateurs Internet m'ont été utiles pour tester les résultats et le temps pris pour les fonctionnalités sur quelques navigateurs: Google Chrome, Mozilla Firefox. En effet, la compatibilité des APIs peut varier selon le navigateur.

Firebug :

Firebug est un module pour le navigateur Firefox. Il est indispensable au développement Web. Il propose une multitude d’outils de développement Web pour modifier, déboguer et contrôler le CSS, HTML et JavaScript sur n’importe quelle page Web.

Eclipse :

Eclipse est un projet libre, décliné et organisé en un ensemble de sous-projets de développements logiciels en s'appuyant principalement sur Java.

SSH :

SSH (Secure Shell) est à la fois un programme et un protocole de communication sécurisé. Le protocole de connexion impose un échange de clés de chiffrement en début de connexion.

II. SUJET DU STAGE

Depuis de longues années l'observatoire consacre une part importante de son travail à l'activité de veille technologique / R&D (Research and development).

Le sujet du stage portait sur le test de nouvelles technologies qui pourraient s'avérer intéressantes pour les services du CDS, au niveau serveur ainsi qu'au niveau client.

Un premier test consistait à mettre en oeuvre Hadoop et à tester son utilisation notamment pour le service VizieR qui est la base de référence mondiale pour les catalogues astronomiques (environ 12000 actuellement).

J’ai commencé par apprendre, dans un premier temps, à mettre en oeuvre Hadoop et à en évaluer les performances avec quelques exemples de données fournis.

L'étape suivante consistait à vérifier le comportement lors d'un passage à l'échelle en injectant cette fois une quantité beaucoup plus significative de données.
Cette étape fut encore une fois l'occasion d'évaluer les performances.

Il existe diverses implémentations d'Hadoop et des frameworks libres. Le choix entre une installation basique (sans recours à un framework) d'Hadoop ou le recours à un framework était également un élément important de cette première partie du stage.
La deuxième partie du stage consistait à développer un parseur coté client conçu en JavaScript.

Le stage était donc très exploratoire mais m'a parmi d'acquérir des connaissances dans des domaines innovants.

III. TRAVAIL RÉALISÉ :

Tout d'abord, il a fallu d'apprendre Apache Hadoop pour avoir une base de travail. Cependant, un défi a été évidement de se plonger dans le domaine du « Big Data », et le fait de n'en avoir aucune expérience est devenu personnellement un souci lors du déroulement du stage.

Une autre contrainte était l'absence de connaissance d'Hadoop au sein de l'Observatoire et quand il y avait de problèmes, la solution était de chercher sur internet et de poser des questions sur les forums.
C'est pour cela que ce stage a été proposé, afin d'avoir une idée de l'utilité de cette plate-forme et de la meilleure façon de la mettre en œuvre.

Donc, j'ai commencé par lire la documentation officielle et aussi quelques articles concernant les grandes quantités de données que M. SCHAAFF m'a proposé et d'autres que j'avais trouvé sur le net.

Dans un premier temps j'ai eu besoin d'apprendre l'architecture, je l'expliquerais tout de suite d'une façon plus simple (Figure 1)

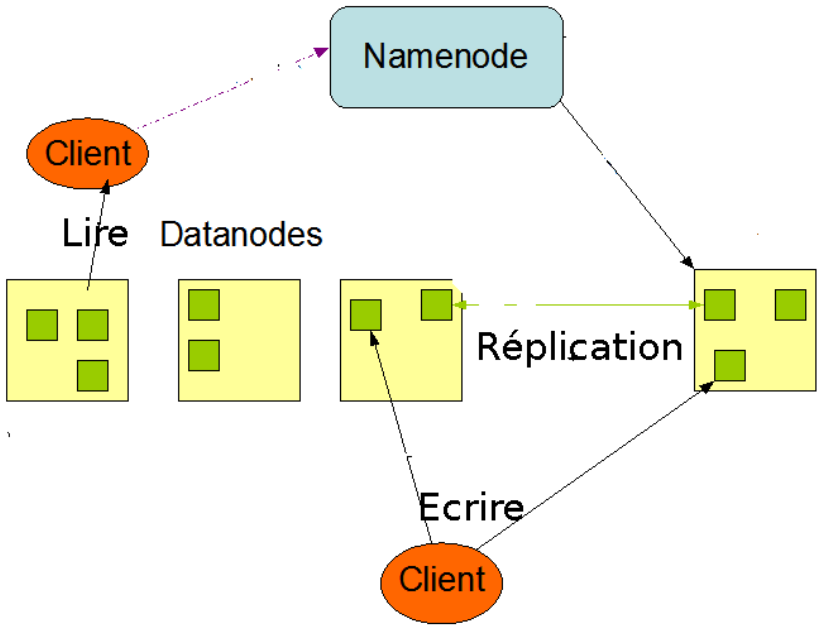


Figure 01, Architecture des fichier distribués dans un ensemble de machines.

En général, le diagramme ci-dessus veut dire que « Namenode » est le nœud maître qui sera chargé de gérer une partie ou tout l'ensemble des machines.
Les « Datanodes » seront les conteneurs des fichiers distribués qui pourront aussi avoir une copie (Réplication) dans n'importe quel endroit du cluster.
Dans cas le client serait quelqu'un qui va lire ou écrire des données dans les machines, ce client pourrait être une API (Java par exemple) capable d'interroger le cluster.

a) La mise en place d'Hadoop :

En ce qui concerne l'environnement de travail, j'ai disposé d'un ordinateur personnel avec suffisamment de mémoire vive (16 GB), d'un disque dur (800GB), d'un bon CPU (Intel Core i5), sous une distribution Ubuntu Linux, avec laquelle je me suis servi pour les test et le développement depuis le début jusqu'à la fin du stage.

Alors, pour mettre en place Hadoop, l'installation de divers logiciels a été nécessaire, telles que Java et la commande SSH comme requis d'Hadoop.

Au début, j'ai été confronté à plusieurs problèmes lors de la mis en place d'Hadoop. principalement à cause de la documentation qui reposait uniquement sur des information concernant la configuration par défaut, et à mon avis, elle n'était pas très bien élaboré, ou au moins il fallait que le lecteur possède déjà un niveau avancé.
C'est pour cela que je me suis appuyé sur de tutoriels qui m'ont permi de mieux comprendre l'architecture.

Je vais expliquer comment la mettre en. Mais avant continuer, il faut comprendre la hiérarchie de ses dossiers et leur organisation (Figure 02) afin d'avoir un aperçu de ce qui a été fait pendant cette étape.

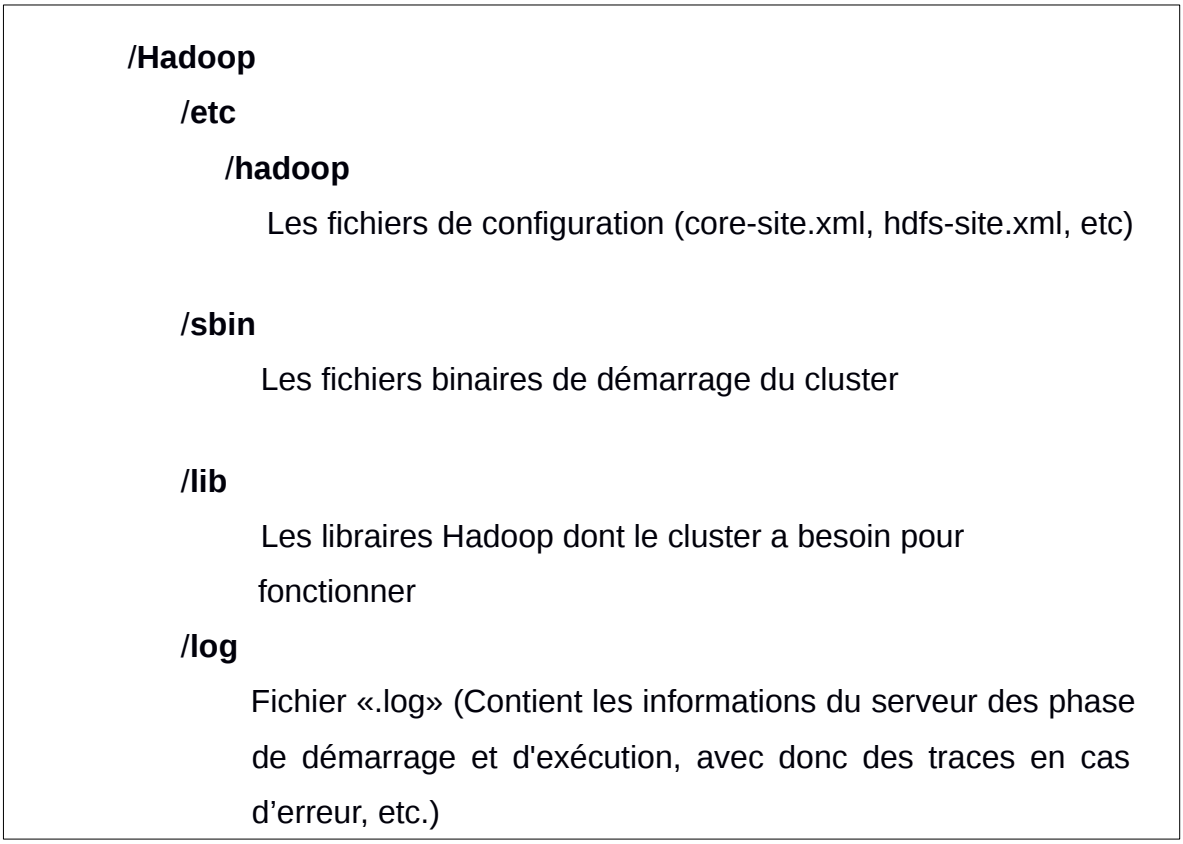


Figure 02: représentation de l'arborescence d'Hadoop

Dans ces fichiers de configuration nous devons spécifier plusieurs indications sur lesquelles notre instance Hadoop pourra agir.
Il est nécessaire qu'Hadoop connaisse ces divers aspects pour réaliser au moins une installation basique (non sécurisée) mais fonctionnelle :

- Le chemin indiquant où seront stocké nos fichiers dans le cluster (voir Annexe A).
- La libraire MapReduce à utiliser.
- Le nœud master et les esclaves (s'il y en a).
- Les ports sur lesquels le nœud va écouter.

Ensuite, j'ai vérifié si tout se passait bien, premièrement, en vérifiant que l'application avait démarré sans aucun problème et sur le navigateur en regardant les information fournis par Hadoop.

D'autre part, quelques exemples sont déjà présents dans l'installation tels que MapReduce et HDFS et pendant le reste de la semaine j'ai consacré mon temps à faire de petits tests en MapReduce grâce à une classe Java, et en regardant les statistiques et les information qu'Hadoop affichait sur une page web.

Dans le prolongement de cette semaine-là, on m'a proposé un démarrage d'un cluster composé cette fois de diverses machines (il s'agissait des machines virtuelles). Je me suis servi de Oracle VM VirtualBox.

Premièrement, j'ai réfléchi avant de faire un choix entre les distributions Linux et finalement Ubuntu Server a attiré mon attention notamment parce que c'est un système d'exploitation basé sur Debian. En outre, cette version du système ne possède pas un bureau et la performance des tests ne serait pas affectée.

D'ailleurs ce système-là est supporté activement par une forte communauté et il est vraiment facile à installer, et en cas de problème lors cette installation, de trouver une solution ne poserait aucun souci.

J'ai débuté une installation du système avec tous les prérequis demandés par Hadoop, puis une instance de celui-ci devait être présente dans chaque nœud (machine), heureusement, cette fois le temps s'est réduis énormément grâce à l'installation précédente.

Il a fallu configurer la machine de façon à ce qu’elle connaisse les futurs nœuds afin d’éviter une édition répétitive des fichiers, puis j'ai cloné la machine virtuelle plusieurs fois, mais il restait encore à mettre en place une connexion entre elles-mêmes et c'est ici que j'ai rencontré quelques inconvénients au niveau du réseaux de VirtualBox.

Auparavant, une connexion par pont devait être employé dans le cluster, mais un problème arrivait lors du démarrage des services Hadoop, la liste de nœuds ne s'affichait pas correctement, c'est a dire que parfois une machine apparaissait dans la liste mais ses caractéristiques étaient inconnues (par exemple, aucune taille de disque dur).
De plus, si un test était réalisé soit avec MapReduce ou HDFS, des exceptions java étaient affichés. Afin d'en avoir une idée, je lisait souvent le fichier « .log » pour réfléchir et chercher la base du conflit.

Finalement le problème a été résolu par une connexion interne, cela veut dire que j'ai mis en place une communication entre les machines dans un réseau interne, indépendant de la machine physique, et à partir de la, l'ensemble de machines a commencé à être fonctionnel (Figure 03):

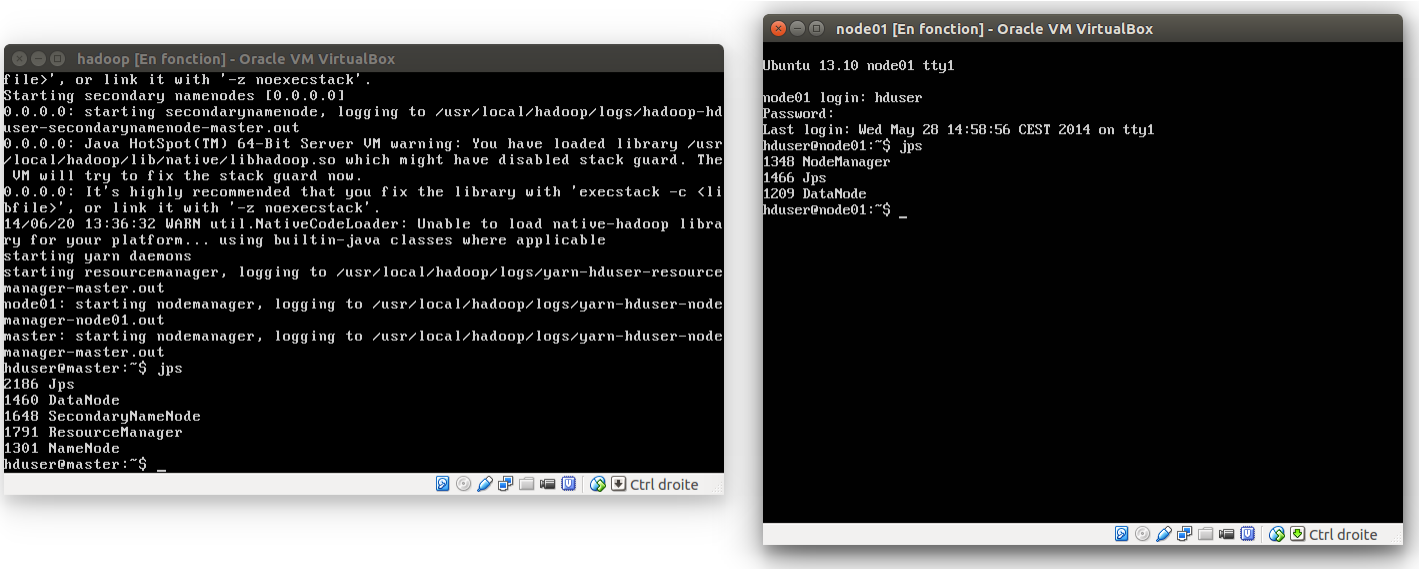


Figure 03 : un cluster à deux machines virtuelles sous hadoop, un maître et un esclave.

En début du stage, M. SCHAFF m'a proposé une réunion avec M. Thomas et M. Gilles pour faire le point. Donc, nous avons discuté tout ce qui avait été fait dans un premier temps, et comment prendre en compte des données du CDS pour

faire des tests avec des données réelles.

Un autre point important à remarquer était de tester une distribution Hadoop (framework), par exemple Hortonworks et de la présenter lors de la prochaine réunion.

Avant de continuer sur les frameworks basés sur Hadoop, j'aimerais faire un point sur Hadoop et ses distributions:

- Si l'on choisit Hadoop lui-même il faut savoir que c'est un choix complexe qui force à ré-implémenter certaines caractéristiques tels que le monitoring et le déploiement et à savoir mélanger les versions de différents projets.

- À la différence des distributions, qui sont déjà prêtes à utiliser et qui offrent un changement de version quasi transparent.

- Dans tous les cas, si l'on choisit une distribution pour la mettre en production, il faut savoir si celle-ci pourra répondre aux besoins de l'entreprise.

Désormais, le choix entre les distributions Hadoop a été très important pour continuer les tests, il a été donc nécessaire de faire une comparaison pour en choisir une pour la suite du stage.

La comparaison portait sur Cloudera et Hortonworks, qui sont les plus significatifs entre les distributions :

L'avantage de Cloudera est qu'il est le distributeur le plus ancien et le plus représenté. En revanche, Hortonworks a fait le choix de rester le plus près des projets libres. Mais, ce qui a attiré mon attention concernant framework a été la qualité des tutoriels et des vidéos qu'il offre, de plus, il est possible de télécharger une machine virtuelle appelée « SandBox » dont avec tous les outils

déjà prêt à l'emploi.
Finalement je me suis dirigé vers Hortonworks. Il existe plusieurs méthodes pour y accéder et interroger soit à partir du navigateur web ou soit via la console du système.
Par ailleurs, elle compte sur un système d'exploitation GNU/Linux appelé CentOS et qui est basé sur RedHat.

Cet outil fournit une interface web simple, ergonomique et intuitive pour effectuer les test seraient plus confortable concernant la visualisation des résultat puisque auparavant la seule façon d'arriver à interroger Hadoop et Hive était sur la console (terminal) du système, et lors des recherches et de l'affichage des données, l'espace de celle-ci n'était pas suffisant. Donc, je me suis servi de ce framework pour les tests suivants pour quelques sous-projets Hadoop.

b). Début avec Hortonworks.

C'est la première fois que j'utilisais cette distribution et il m'a fallu un temps d'adaptation à l'environnement mais l'apprentissage du langage HiveQL m'a pris encore plus de temps, puisque même s'il rassemble à SQL, ils ont des objectifs différents, et le langage de script d'Apache Pig.
Au début, j'ai exploré la plate-forme afin d'injecter plusieurs fichiers afin de savoir comment cela se passe pour l'injection de données.

Avant de commencer les tests avec des données réelles j'ai eu besoin de lire divers tutoriels sur le site d'Hortonworks et sur le net afin d'arriver au moins a faire de requêtes sur Hive.

Une fois que j'étais un peu plus habitué au langage et aussi aux erreurs qu'il peut générer, j'ai pris cette fois des données depuis le site Vizier, sous différents formats (.tsv, .csv). En ce qui concerne son injection, il est vraiment facile de le faire à partir de «SandBox » lequel fournit un module web capable de chercher un fichier sur l'ordinateur et le placer dans le cluster Hadoop (en cet cas il s'agit

d'une seule machine virtuelle).
Ensuite, j'ai envisagé une nouvelle étape de tests durant laquelle je me suis basé sur divers tables crées à partir de fichiers de tailles variées.

Il y a des remarques à faire sur ce logiciel par rapport aux performances des opérations : normalement une requête HiveQL ne prend pas beaucoup de temps si elle est assez simple, mais au fur et à mesure que sa complexité augmente, le temps de réponse est affecté, même si le fichier de données n'a pas une grande taille (voir Annexe B).

Comme je l'ai déjà mentionné avant, Hive est à prêt une interface qui nous facilite la création des algorithmes MapReduce lesquels n'ont pas comme objectif d'être rapides.
Alors, par rapport à Apache pig qui en fait ressemble à Hive (concernant le traitement et l'analyse sur MapReduce) dont la syntaxe est plutôt différente et j'ai encore eu besoin de l'apprendre.

Pendant la deuxième réunion, j'ai présenté Hadoop Hortonworks au niveau des modules et des outils envisagés et les résultats obtenus depuis le début sur Hortonworks et ainsi que les types de fichiers supportés par Apache Hive.
Mais, il fallait garder à l'esprit la nature d'Hadoop et réfléchir à l'intérêt de cette technologie pour l'amélioration du service Vizier.
Alors, ce qui était particulièrement important était de chercher une base de données qui puisse travailler en surcouche d'Hadoop, laquelle était capable de réaliser des requêtes à grande échelle et en temps réel.

À ce moment-là, l'étape de tests d'Hadoop était presque finie, désormais je pouvais apporter mon avis concernant le projet Hadoop et ses distributions et de l'expérience qu'ils m'ont apporté lors de son apprentissage et sa mise en oeuvre.

Afin de conclure cette étape je mentionnerai les avantages et inconvénients que ce logiciel peut nous amener :

Avantages :

- Le premier avantage évidemment est sa capacité d'analyser et traiter de vastes quantités de données.

- Le support concernant les divers types de fichiers possède aussi une grande importance, ce qui permet d'extraire et traiter des données depuis un fichier et les déposer dans une base de données par exemple.

- L'analyse à grande échelle, puisque sa nature est de travailler dans un cluster, et grâce à cela, il suffit d'ajouter un nœud au cluster pour augmenter ses capacités de stockage et de coordination.

- C'est un projet libre et de grandes entreprises contribuent à son amélioration continue.

- Il existe des distributeurs qui consacrent leur temps au support de ses services

- **Quelques inconvénients :**

- Le projet lui-même est vraiment difficile à maintenir si on le met en production.

- Programmer des algorithmes MapReduce n'est vraiment pas facile et cela prend du temps pour préparer les employés.

- La rapidité de l’implémentation de MapReduce est assez faible.

Maintenant nous avons une idée de sa nature, à savoir que Hadoop ne remplace jamais une base de données relationnelle, même si nous pouvons y penser, il est conçu pour l'analyse et le traitement mais pas du tout fait pour la vitesse. Des entreprises et des projets essaient d'accélérer sa vitesse. Et des changements et des améliorations arrivent tous les jours, mais nous ne pouvons pas changer son but de base.

Désormais, le premier objectif était de trouver une nouvelle technologie capable d'être interrogée et de répondre à grande vitesse, et à vrai dire, cette dernière tâche m'a fait encore prendre du temps en ce qui concerne la recherche. À savoir la majorité des bases de données qui fonctionnent ensemble avec Hadoop s'appuient sur MapReduce derrière et comme je l'avais mentionné auparavant, un tel algorithme tend à être assez lent.

C'est pour cela qu'il fallait précisément avoir une technologie consacrée aux applications à grande échelle afin de répondre très rapidement.

c). Un début sur ElasticSearch :

Il existe de plus en plus des outils innovants consacrés à l'indexation de données. Ainsi, une nouvelle technologie qui avait l'air assez intéressante et qui pouvait répondre à une partie de la problématique (la vitesse des requêtes) a été choisie pour quelques tests, cet outil est ElasticSearch.

Comme auparavant, c'était la première fois que j'utilisais Elasticsearch et cela m'a pris encore du temps pour m'habituer à son architecture et surtout son langage.

Par ailleurs, en ce qui concerne la communauté, j'ai vraiment bien aimé puisque même si elle n'est pas si énorme au moins elle compte des collègues très actifs qui normalement répondent tout de suite aux problèmes rencontrés par les utilisateurs.

Par contre, au niveau de la documentation je me suis rendu compte qu'il manquait des exemples claires et concis, de plus, étant donné que c'est une technologie assez récente, trouver des tutoriels complets sur le net n'est pas vraiment simple.
Donc, pour commencer à l'apprendre il a fallu que je me serve d'un livre comme référence pour avoir un bon niveau et maîtriser ce logiciel.

Concernant la manipulation de données, Elasticsearch peut être employé afin d'effectuer plusieurs tâches telles que l'indexation de données, la recherche dans ces données, vérifier et gérer les nœuds et le cluster grâce à des actions CRUD (create-retrieve-update-delete).

Par ailleurs, l'interface principal d'interaction d'Elasticsearch fonctionne au travers du protocole HTTP et ses méthodes (GET, PUT, POST, DELETE). Ainsi, il peut être accédé de plusieurs manières, soit par un client (dans mon cas je me suis servi de l'api client Java) ou à l'aide d'un programme type cURL.

Alors, pour bien comprendre Elasticsearch on doit connaitre quelques concepts liés au sujet :

- **HTTP** (HyperText Transfer Protocol) est un protocole de communication client-serveur développé pour le World Wide Web

- **REST** (REpresentational State Transfer) est un style d'architecture particulièrement bien adapté au World Wide Web, qui peut s'appliquer à d'autres protocoles d'application que HTTP.

- **JSON** (*JavaScript Object Notation*) est un format de données textuelles, générique, dérivé de la notation des objets du langage JavaScript. Il permet de représenter de l'information structurée comme XML par exemple.

En général, on pourrait dire que Elasticsearch est un logiciel basé sur l'architecture client-serveur, qui permet d'être interrogé et géré par quelques méthodes de HTTP en envoyant des requêtes de type REST et en recevant des réponses en format JSON.

Une configuration du serveur a été nécessaire puisque même si les valeurs par défaut étaient déjà un bon début, il a été nécessaire de les modifier afin d'effectuer des tests plus précis.
L'arborescence ressemble à celle de Hadoop sauf que celle-ci est plus simple (Figure 4).

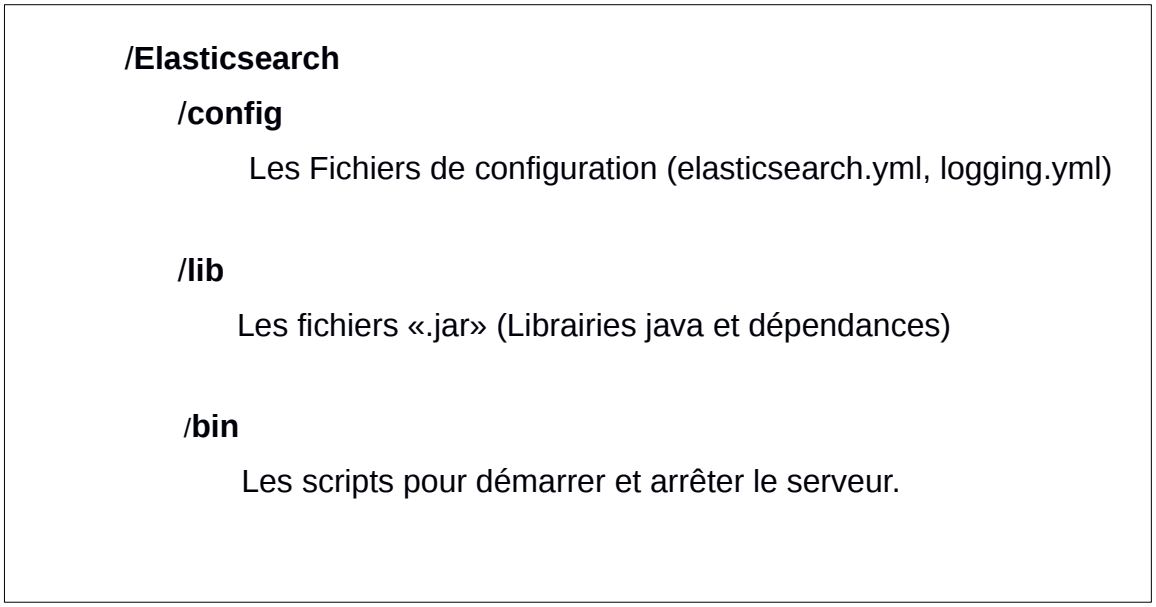


Figure 04. La structure hiérarchique d'Elasticsearch.

Il fallait donc éditer ces fichiers-là pour optimiser le niveau de performance, ce que j'ai fait principalement a été d'enrichir la quantité de mémoire vive consacrée au service Elasticsearch.

J'ai commencé à faire des opérations assez basiques, et au fur et à mesure que je possédais un meilleur niveau il fallait aussi augmenter la complexité des requêtes.

Alors, l'indexation des données Vizier pouvait être mise en place afin d'avoir un avis sur la performance.
Malheureusement ces données avaient un format qui ne s'adaptait pas tout à fait au format désiré, c'est ainsi que j'ai essayer de les injecter à partir de l'API java, puis, j'ai développé un programme en utilisant une librairie appelée Jackson (pour manipuler des objet JSON en java) qui lisait de telles données, les transformait, et les insérait dans Elasticsearch.

À ce moment-là, ce serveur comptait 200,000 lignes contenant des données Vizier lesquelles étaient prêts à être consultés, alors c'était le moment de déterminer ses performances.

J'ai essayé d'élaborer des requêtes type SQL mais dans ce cas le langage n'était pas du tout pareil, cette fois ils s'agissait d'un langage particulier de Elasticsearch et basé sur le format JSON.

Un petit exemple sera expliqué même s'il n'a pas pour but d'apprendre à utiliser Elasticsearch. Il permet de clarifier ce qui a été testé sur cette technologie (Figure 05).

Figure 05, Elasticsearch étant interrogé depuis une requête de son propre langage

Les points ci-dessous expliquent un peu la procédure de la Figure 05.

1.- Une requête en format JSON est envoyée par la méthode GET, ce qu'on souhaite obtenir à partir de cela est sélectionné par les données d'un catalogue suivant des critères portant sur les coordonnées RA/DEC.

2.- On voit la totalité des résultats récupérés (184337 sur 200000).

3.- Le résultat de la réponse fournit par le serveur après la requête.
L'exemple ci-dessus n'est pas vraiment compliqué mais c'est bien pour se rendre compte que chaque fois qu'on apprend une nouvelle technologie cela prend du temps.

Finalement pour conclure sur Elasticsearch, j'aimerais citer quelques avantages et inconvénients que je considère importantes :

Avantages :

- Elasticsearch est un bon outil concernant la recherche du texte.

- La réponse aux requêtes est super rapide.

- L'installation est vraiment simple à faire et en fait, il existe déjà des versions emballées pour quelques distributions Linux telles que Debian ou RedHat.

- Il possède une communauté très active, c'est à dire que la réponse sur les forums se fait normalement tout de suite.

Inconvénients :

- Ce logiciel est assez récent et il se peut qu'il y ait des problèmes difficiles à résoudre.

- Il ne gère absolument pas le niveau de la sécurité, c'est à dire qu'il doit être protégé par une application ou par un proxy inverse (ngnix par exemple).

- Il existe des plugins pour placer une base de données derrière Elasticsearch, mais ils sont si récent qu'ils manquent de maturité.

d). Le dernier test Hadoop / Elasticsearch :

Pendant la dernière étape de tests, je devais mettre ensemble Hadoop et Elasticsearch afin de conclure s'ils pourraient être utiles dans le service VizieR.
Pour mettre à profit le temps mis me servir de Hortonworks qui était installé sur ma machine et qui était prêt a l'emploi, j'ai juste installé une nouvelle instance d'Elasticsearch.

Alors, en ce qui concerne la configuration il a fallu ajouter quelques librairies Elasticsearch/Hadoop (de fichiers .jar) sous Hortonworks afin de m'en servir lors des tests.

Après quelques problèmes auxquels je ne trouvais aucune solution, j'ai réussi à écrire des données sur Hive en utilisant une instance Elasticsearch.
Personnellement, j'ai trouvé une grande différence de performance lors de l'écriture du texte mais par rapport à la lecture, la vitesse n'a absolument pas changé.

D'ailleurs, il est important d'apporter mon avis après d'avoir effectué des tests et aussi après avoir rencontré quelques problèmes depuis le début du stage jusqu'à présent.

Donc, voici ma conclusion aux tests :

Vu que Apache Hadoop ne semble pas répondre à ce dont nous avons besoin, il faudra effectuer des tests complémentaires avant de s’en servir.

e). Développement côté client:

La dernière partie du stage consistait à développer un programme côté client (JavaScript), sur le navigateur.

Le but initial était d'avoir la possibilité de lire et interpréter un fichier sous le format VOTable depuis le navigateur web. L'utilisateur devait pouvoir exploiter les données appartenant au fichier à partir de fonctions simples (Figure 06).

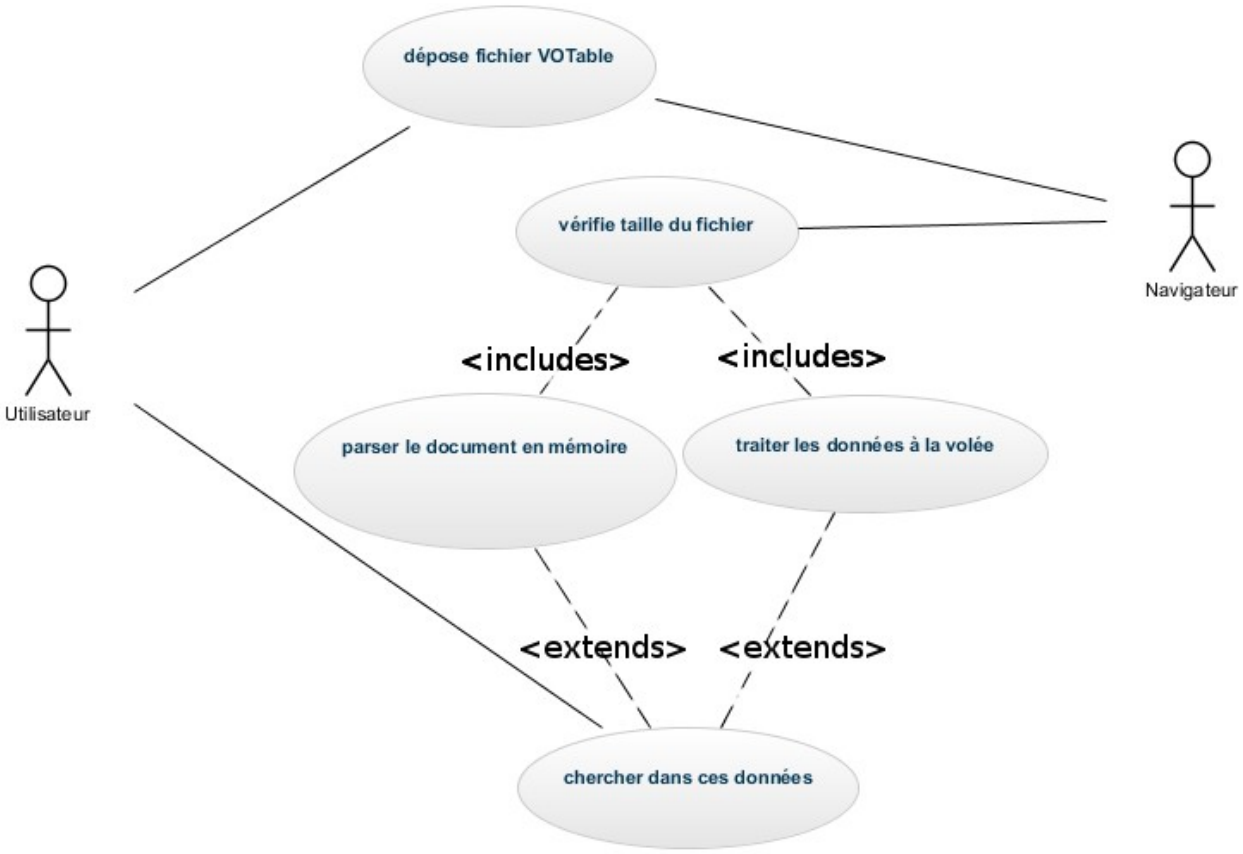


Figure 06. Cas d'utilisation du développement de l'application côté client

Je vais expliquer brièvement ce qu'est un fichier VOTable :

VOTable est un fichier en format XML qui s'appuie sur un standard créé par l'Observatoire virtuel (IVOA) contenant des données astronomiques.
Dans l'entête nous pouvons trouver la description générale et les métadonnées,

c'est à dire, pour simplifier, le nom des colonnes.
Le corps du fichier comporte une table qui est remplie par les données
(voir Annexe C).
Par contre, la taille maximum de ce type de fichier n'est pas fixe, cela dépend de la requête avec laquelle le fichier a été obtenu et aussi de son nombre de lignes.

D'ailleurs, il existe quelques contraintes par rapport à la programmation côté client, c'est à dire que nous sommes limités par mémoire disponible gérée par le navigateur, en outre, chaque navigateur possède ses propres limitations et performances.

Une solution valable serait de stocker de données au fur et à mesure les données.

Au niveau du stockage, il existe des APIs côté client (par exemple pour HTML5) telles que localstorage, indexedDB, etc. qui permettent de manipuler des données côté client même s'il n'a pas une connexion réseaux.
Mais cela peut aussi avoir quelques limitations à cause du navigateur et sa version, à la fois pour le support de l'API et pour la taille de stockage côté client.

Dans un premier temps, je me suis concentré plutôt sur la lecture du document en mémoire en déposant des fichiers pas trop grands (25 MB). En ce qui concerne la transformation j'ai opté pour une mise en place de fonctions natives qui appartiennent au navigateur lui-même (voir Annexe D), car la performance de celles-ci sont intéressantes.

Mais Il fallait chercher une solution afin de lire les gros fichiers, et une possibilité était de traiter ces données à la volée et de ne pas les stocker en mémoire.

CONCLUSION

Ainsi, ce stage de 12 semaines au sein de l'Observatoire de Strasbourg m'a apporté des connaissances innovantes et aussi une expérience professionnelle, à la fois pour avoir travaillé dans un laboratoire de recherche utilisant fortement l'informatique et pour avoir acquis quelques connaissances de l'astronomie.

Je pense que j'ai réussi à répondre aux besoins demandés pendant ce stage.

Par ailleurs, je m'attendais à un peu plus de développement pendant le stage puisque même si j'ai acquis de très bonnes connaissances j'aurais souhaiter dédier plus de temps à la programmation.

Heureusement, grâce à cette expérience j'ai un autre aperçu de l'informatique, en outre, je me suis plongé dans un nouveau domaine appelé « Big Data » lequel m'offrira des opportunités dans un futur proche.

Finalement, Je tiens encore à remercier M. André Schaaff mon responsable de stage de m’avoir offert l’opportunité de travailler à l'Observatoire.

BIBLIOGRAPHIE

- **Tutoriel** :
  - Auteur : DigitalOcean
  - *Titre : How to Install Hadoop on Ubuntu 13.10*
  - Source : https://www.digitalocean.com/community/tutorials/how-to-install-hadoop-on-ubuntu-13-10

- **Tutoriel** :
  - Auteur : Michael G.Noll
  - *Titre : Running Hadoop on Ubuntu Linux (Multi-Node Cluster)*
  - Source : http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/

- **Tutoriel** :
  - Auteur : Hortonworks
  - *Titre : Hello World! – An introduction to Hadoop with Hive and Pig*
  - Source : http://hortonworks.com/hadoop-tutorial/hello-world-an-introduction-to-hadoop-hcatalog-hive-and-pig/

- **E-book** :
  - Auteur : Rafal Cuk, Marek Rogozinski
  - *Titre : Elasticsearch Server*
  - Source : http://books.google.co.in/books?id=PEFK3MuwBsIC&printsec=frontcover&hl=fr&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

- **E-book** :
  - Auteur : Rafal Cuk, Marek Rogozinski
  - *Titre : Mastering Elasticsearch*
  - Source : http://books.google.fr/books?id=NRO8AQAAQBAJ&printsec=frontcover&hl=es#v=onepage&q&f=false

- **Tutoriel** :
  - Auteur : Ravikumar Visweswara
  - *Titre : How To Configure Elasticsearch on Hadoop with HDP*
  - Source : http://hortonworks.com/blog/configure-elastic-search-hadoop-hdp-2-0/

GLOSSAIRE

Cluster : Un cluster est une grappe de serveurs (ou « ferme de calcul ») constituée de deux serveurs au minimum (appelés aussi nœuds) et partageant une baie de disques commune.

Framework : En programmation informatique, un *framework* est un ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel (architecture).

Big Data : Les big data, littéralement les grosses données, parfois appelées données massives, est une expression anglophone utilisée pour désigner des ensembles de données qui deviennent tellement volumineux qu'ils en deviennent difficiles à travailler avec des outils classiques de gestion de base de données ou de gestion de l'information.

SQL : (Structured Query Language, en français langage de requête structurée) est un langage informatique normalisé servant à exploiter des bases de données relationnelles. La partie langage de manipulation des données de SQL permet de rechercher, d'ajouter, de modifier ou de supprimer des données dans les bases de données relationnelles.

Plugin : plug-in, aussi nommé module d'extension, module externe, greffon, plugiciel, ainsi que add-in ou add-on en France, est un paquet qui complète un logiciel hôte pour lui apporter de nouvelles fonctionnalités.

Temps réel :
Les systèmes informatiques temps réel se différencient des autres systèmes informatiques par la prise en compte de contraintes temporelles dont le respect est aussi important que l'exactitude du résultat, autrement dit le système ne doit pas simplement délivrer des résultats exacts, il doit les délivrer dans des délais imposés.

ANNEXES

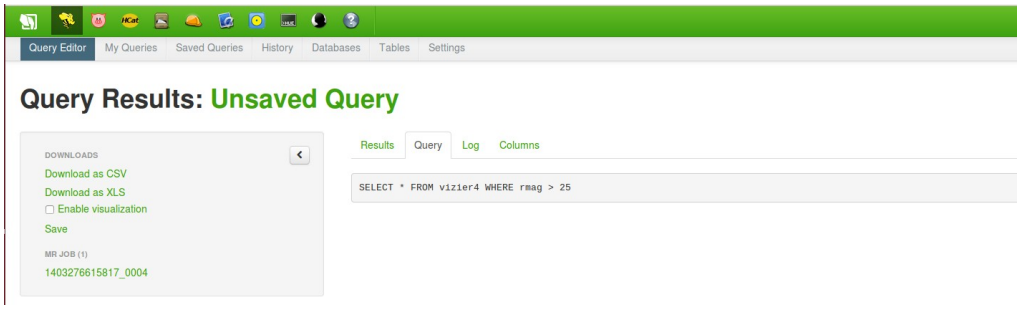
Annexe A

Fichier de configuration HDFS d'Hadoop :

```
<configuration>
        <property>
                <name>dfs.replication</name>
                <value>2</value>
         </property>
         <property>
                 <name>dfs.permissions</name>
                 <value>false</value>
         </property>
         <property>
                <name>dfs.namenode.name.dir</name>
                <value>file:/mnt/data/hadoop_store/hdfs/namenode</value>
        </property>
        <property>
                <name>dfs.datanode.data.dir</name>
                <value>file:/mnt/data/hadoop_store/hdfs/datanode</value>
        </property>
</configuration>
```

Annexe B

Une requête Hive sous Hortonworks:



MapReduce derrière la requête :

```
14/06/20 08:20:49 INFO Configuration.deprecation: mapred.cache.files.timestamps is deprecated. Instead, use mapreduce.job.cache.files.timestamps
14/06/20 08:20:49 INFO Configuration.deprecation: mapred.working.dir is deprecated. Instead, use mapreduce.job.working.dir
14/06/20 08:20:49 INFO Configuration.deprecation: user.name is deprecated. Instead, use mapreduce.job.user.name
14/06/20 08:20:49 INFO Configuration.deprecation: mapred.cache.files.filesizes is deprecated. Instead, use mapreduce.job.cache.files.filesizes
14/06/20 08:20:49 INFO Configuration.deprecation: mapred.reduce.tasks is deprecated. Instead, use mapreduce.job.reduces
14/06/20 08:20:49 INFO Configuration.deprecation: mapred.output.key.class is deprecated. Instead, use mapreduce.job.output.key.class
14/06/20 08:20:49 INFO Configuration.deprecation: mapred.mapoutput.key.class is deprecated. Instead, use mapreduce.map.output.key.class
14/06/20 08:20:49 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1403276615817_0003
14/06/20 08:20:50 INFO impl.YarnClientImpl: Submitted application application_1403276615817_0003 to ResourceManager at sandbox.hortonworks.com/10.0.2.15:8050
14/06/20 08:20:50 INFO mapreduce.Job: The url to track the job: http://sandbox.hortonworks.com:8088/proxy/application_1403276615817_0003/
Starting Job = job_1403276615817_0003, Tracking URL = http://sandbox.hortonworks.com:8088/proxy/application_1403276615817_0003/
14/06/20 08:20:50 INFO exec.Task: Starting Job = job_1403276615817_0003, Tracking URL = http://sandbox.hortonworks.com:8088/proxy/application_1403276615817_0003/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1403276615817_0003
14/06/20 08:20:50 INFO exec.Task: Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1403276615817_0003
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
14/06/20 08:20:57 INFO exec.Task: Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
14/06/20 08:20:57 WARN mapreduce.Counters: Group org.apache.hadoop.mapred.Task$Counter is deprecated. Use org.apache.hadoop.mapreduce.TaskCounter instead
2014-06-20 08:20:57,588 Stage-1 map = 0%,  reduce = 0%
14/06/20 08:20:57 INFO exec.Task: 2014-06-20 08:20:57,588 Stage-1 map = 0%,  reduce = 0%
2014-06-20 08:21:03,815 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.25 sec
14/06/20 08:21:03 INFO exec.Task: 2014-06-20 08:21:03,815 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.25 sec
2014-06-20 08:21:04,843 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.25 sec
14/06/20 08:21:04 INFO exec.Task: 2014-06-20 08:21:04,843 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.25 sec
MapReduce Total cumulative CPU time: 2 seconds 250 msec
14/06/20 08:21:04 INFO exec.Task: MapReduce Total cumulative CPU time: 2 seconds 250 msec
Ended Job = job_1403276615817_0003
14/06/20 08:21:04 INFO exec.Task: Ended Job = job_1403276615817_0003
14/06/20 08:21:04 INFO exec.FileSinkOperator: Moving tmp dir: hdfs://sandbox.hortonworks.com:8020/tmp/hive-beeswax-hue/hive_2014-06-20_08-20-46_113_5347610445962164
985-1/_tmp.-ext-10001 to: hdfs://sandbox.hortonworks.com:8020/tmp/hive-beeswax-hue/hive_2014-06-20_08-20-46_113_5347610445962164985-1/_tmp.-ext-10001.intermediate
14/06/20 08:21:04 INFO exec.FileSinkOperator: Moving tmp dir: hdfs://sandbox.hortonworks.com:8020/tmp/hive-beeswax-hue/hive_2014-06-20_08-20-46_113_5347610445962164
985-1/_tmp.-ext-10001.intermediate to: hdfs://sandbox.hortonworks.com:8020/tmp/hive-beeswax-hue/hive_2014-06-20_08-20-46_113_5347610445962164985-1/-ext-10001
14/06/20 08:21:04 INFO ql.Driver: </PERFLOG method=task.MAPRED.Stage-1 start=1403277647649 end=1403277664891 duration=17242>
14/06/20 08:21:04 INFO ql.Driver: </PERFLOG method=runTasks start=1403277647649 end=1403277664891 duration=17242>
14/06/20 08:21:04 INFO ql.Driver: </PERFLOG method=Driver.execute start=1403277647594 end=1403277664891 duration=17297>
MapReduce Jobs Launched:
14/06/20 08:21:04 INFO ql.Driver: MapReduce Jobs Launched:
Job 0: Map: 1   Cumulative CPU: 2.25 sec   HDFS Read: 28993072 HDFS Write: 14959 SUCCESS
14/06/20 08:21:04 INFO ql.Driver: Job 0: Map: 1   Cumulative CPU: 2.25 sec   HDFS Read: 28993072 HDFS Write: 14959 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 250 msec
14/06/20 08:21:04 INFO ql.Driver: Total MapReduce CPU Time Spent: 2 seconds 250 msec
OK
14/06/20 08:21:04 INFO ql.Driver: OK
```

Résultat obtenu :

Results | Query | Log | Columns

	mode	q_mode	cl	sdss9	m_sdss9	im	raj2000	dej2000	obsdate	q	umag	e_umag	gmag	e_gmag	rmag	e_rmag	imag	e_imag	zr
0	2		6	J170956.26-014812.6		lm	257.484441	-01.803517	2005.4300	3	25.848	0.861	24.800	0.934	25.368	0.921	21.859	0.197	22
1	1		3	J170956.41-014902.3		lm	257.485045	-01.817315	2005.4329	3	24.278	1.233	25.289	0.678	25.049	0.818	22.315	0.265	22
2	1		3	J170956.57-014926.7		lm	257.485724	-01.824095	2005.4329	3	28.793	0.931	29.486	0.436	29.812	0.314	17.427	0.060	27
3	2		6	J170958.73-014835.8		lm	257.494729	-01.809952	2005.4300	3	25.093	1.404	25.382	0.974	25.266	1.009	21.777	0.192	22
4	1		6	J170959.66-015625.6		lm	257.498588	-01.940466	2005.4329	3	25.497	0.877	23.382	0.260	25.341	0.647	23.018	0.403	23
5	1		6	J170959.96-014709.7		lm	257.499872	-01.786027	2005.4329	3	26.743	0.315	25.193	0.621	25.856	0.509	21.354	0.102	20
6	1		3	J171001.43-015620.5		lm	257.505995	-01.939048	2005.4329	3	25.294	1.339	22.747	0.203	26.121	0.560	26.793	0.226	23
7	1		6	J171001.97-015612.5		lm	257.508220	-01.936816	2005.4329	3	25.355	0.944	23.017	0.190	25.352	0.645	26.792	0.152	23
8	2		3	J171003.25-014634.5		lm	257.513546	-01.776257	2005.4301	3	24.098	1.714	24.929	1.013	25.163	1.002	21.652	0.177	21
9	2		3	J171003.91-014828.6		lm	257.516291	-01.807967	2005.4300	3	24.397	1.660	23.833	0.610	25.008	1.160	22.074	0.277	21
10	1		3	J171004.24-015455.6		lm	257.517698	-01.915462	2005.4329	3	21.129	1.505	28.887	0.678	25.889	8.521	18.481	0.130	18
11	2		6	J171004.28-015455.9		lm	257.517838	-01.915538	2005.4300	3	23.034	0.659	25.904	0.733	25.593	0.852	21.727	0.205	21
12	1		6	J171004.56-015516.4		lm	257.518998	-01.921248	2005.4329	3	25.783	0.743	23.121	0.212	25.618	0.582	24.826	0.708	24
13	1		6	J171004.68-015232.3		lm	257.519517	-01.875654	2005.4329	3	26.027	0.591	23.141	0.206	25.661	0.551	23.672	0.620	21
14	1		6	J171005.89-015336.7		lm	257.524565	-01.893535	2005.4329	3	21.828	0.209	25.746	0.572	25.388	0.696	23.502	0.625	21
15	1		3	J171009.34-015314.0		lm	257.538924	-01.887227	2005.4329	3	24.082	1.751	22.488	0.195	25.253	1.166	24.073	1.314	21
16	1		6	J171015.69-014613.0		lm	257.565394	-01.770298	2005.4329	3	23.928	0.991	23.195	0.211	25.068	0.738	23.310	0.521	22
17	1		6	J171020.29-014710.7		lm	257.584549	-01.786324	2005.4329	3	25.691	0.761	23.314	0.231	25.294	0.671	25.538	0.459	23
18	1		6	J171022.49-014551.0		lm	257.593714	-01.764168	2005.4329	3	25.534	0.870	24.512	0.522	25.005	0.728	23.917	0.704	20
19	1		3	J171022.58-015047.3		lm	257.594086	-01.846480	2005.4329	3	25.493	1.308	24.680	0.865	25.870	0.756	21.677	0.211	21
20	2		3	J171024.33-014519.5		lm	257.601394	-01.755420	2005.4300	3	24.274	2.879	22.834	0.468	26.783	0.669	20.866	0.177	23
21	1		3	J171024.47-014525.4		lm	257.601978	-01.757070	2005.4329	3	27.003	0.307	24.960	0.743	25.127	0.897	21.913	0.209	21
22	1		3	J171029.95-014718.9		lm	257.624791	-01.788591	2005.4329	3	25.595	1.369	25.863	0.813	25.421	1.073	21.695	0.235	22
23	1		6	J171031.12-014853.8		lm	257.629704	-01.814966	2005.4329	3	24.057	1.130	25.677	0.550	25.735	0.558	22.063	0.195	22
24	1		3	J171033.24-014612.4		lm	257.638506	-01.770114	2005.4329	3	26.001	0.722	24.782	0.688	25.003	0.860	22.167	0.258	23
25	1		3	J171033.80-014609.6		lm	257.640837	-01.769351	2005.4329	3	24.150	1.409	24.643	0.726	25.067	0.948	22.447	0.393	23

Annexe C

Métadonnées d'un fichier VOTable :

```
<FIELD name="_RAJ2000" ucd="pos.eq.ra" ref="J2000" datatype="double" width="10" precision="6" unit="deg">
        <DESCRIPTION>Right ascension (FK5, Equinox=J2000.0) (computed
                by VizieR, not part of the original data)
        </DESCRIPTION>
</FIELD>
<FIELD name="_DEJ2000" ucd="pos.eq.dec" ref="J2000" datatype="double" width="10" precision="6" unit="deg">
        <DESCRIPTION>Declination (FK5, Equinox=J2000.0) (computed by VizieR, not
                part of the original data)
        </DESCRIPTION>
</FIELD>
```

Annexe D

Extraction de données XML grâce à l'API Xpath de JavaScript :

```
Model.prototype.getValue = function(row, column) {
        row = row || 0;
        column = column || 0;

        // XPath
        var result = this.data.evaluate('//TABLEDATA/TR[' + row + ']/TD[' + column + ']',
        this.data, null, XPathResult.FIRST_ORDERED_NODE_TYPE, null);

        if (result.singleNodeValue)
                return result.singleNodeValue.textContent;
        return null;
}
```

résultat sur le navigateur :