UNIVERSITÉ DE TECHNOLOGIE DE BELFORT-MONTBÉLIARD

Big Data en astronomie

Étude et implémentation de Hadoop et Spark, application au Cross-Match du CDS

Rapport de stage ST40 - A2015

WALI Noémie

Département Informatique

Observatoire astronomique de Strasbourg

11 rue de l'Université
67 000 Strasbourg

Tuteurs en entreprise
André SCHAAFF
François-Xavier PINEAU

Suiveur UTBM
Nicolas GAUD

Observatoire astronomique de Strasbourg

utbm
université de technologie Belfort-Montbéliard

Remerciements

Ce stage à l'Observatoire astronomique de Strasbourg n'aurait pas eu lieu sans la volonté et la participation des personnes suivantes.

Je tiens en premier lieu à remercier mes deux tuteurs de stage, M. André SCHAAFF et M. François-Xavier PINEAU, ingénieurs de recherche en informatique au Centre de Données astronomiques de Strasbourg (CDS), pour m'avoir encadrée, guidée et aidée dans mon travail ainsi que pour leur disponibilité et leurs conseils tout au long des vingt-quatre semaines de stage.

J'exprime également ma reconnaissance envers M. Hervé WOZNIAK et M. Mark ALLEN, respectivement directeurs de l'Observatoire et du CDS qui ont permis mon intégration au sein de l'organisation.

Enfin j'adresse mes remerciements aux différentes équipes de l'Observatoire dont le CDS pour leur accueil et leur présence.

Introduction

La formation d'ingénieur en informatique à l'Université de Technologie de Belfort-Montbéliard (UTBM) comprend un stage d'assistant ingénieur de six mois au cours de la quatrième année. Dans ce contexte, j'ai effectué mon stage au sein de l'Observatoire astronomique de Strasbourg, un établissement de recherche et d'enseignement, qui héberge également le Centre de Données astronomiques de Strasbourg (CDS).

Le CDS est un centre de données dont les activités sont la recherche en astrophysique et le développement de services en ligne pour la communauté. Ses objectifs sont de collecter, d'enrichir et de distribuer les données astronomiques dans le monde entier.

Dans un contexte de Big Data, un domaine qui désigne des ensembles de données volumineux, les données astronomiques, en raison de leur quantité, ne peuvent plus être traitées avec des outils traditionnels de traitement de données.

Dans le cadre de l'évolution importante des quantités de données que le CDS sera amené à gérer dans les années à venir et dans le cadre d'une amélioration continue des services, le stage qui a pour sujet « Etude et implémentation de Hadoop et Spark, application au cross-match du CDS » a pour but de tester et prototyper des applications dans le domaine du Big Data.

Le CDS propose notamment un service de cross-match (identification croisée) de catalogues de données astronomiques. Ce type de traitement est potentiellement lourd et demande la mise en place de techniques appropriées pour assurer de bonnes performances et l'utilisation dans des services en ligne. Le stage a pour objectif de mettre en œuvre une application de type cross-match et de l'implémenter dans un environnement technique approprié tel que Spark et Hadoop pour tester des jeux de données étoffés, impliquant des tests comparatifs à la fois avec Spark et avec le service de cross-match du CDS.

Le rapport s'articule autour de trois grandes parties : la présentation de l'Observatoire astronomique de Strasbourg et en particulier du Centre de Données astronomiques de Strasbourg (CDS), le déroulement et l'organisation du stage, et enfin le travail réalisé et les résultats obtenus.

Table des matières

I. Présentation de l'Observatoire astronomique de Strasbourg

A. Observatoire

1. Historique

Ce haut lieu scientifique prend ses origines dans la seconde moitié du XIXème siècle, à la fin de la guerre franco-prussienne. L'Allemagne victorieuse de ce combat face aux Français, récupère les territoires de l'Alsace-Lorraine ainsi que la ville de Metz. Le gouvernement veut faire de la ville de Strasbourg une vitrine de l'empire et du savoir-faire allemand. C'est dans son vaste plan d'urbanisation de Strasbourg que l'empereur Guillaume Ier décide d'y installer une université avec un jardin botanique et un observatoire astronomique.

Crédits : topic-topos.com

L'édifice construit entre 1876 et 1880 est inauguré en 1881. Il est composé d'une Grande Coupole, d'un bâtiment des salles méridiennes avec deux plus petites coupoles et un bâtiment accueillant des bureaux et des résidences. La Grande Coupole en fer de plus de 9 mètres de diamètre abrite le Grand Réfracteur, une lunette de 7 mètres de long, la plus grande d'Europe à l'époque, aujourd'hui la troisième plus grande de France.

Un siècle plus tard en 1981, l'Observatoire se dote d'un planétarium, dont il a eu la responsabilité de 1986 à 2008, et désormais intégré au Jardin des Sciences de l'Université de Strasbourg. Le Planétarium de Strasbourg est un nouveau lieu de loisir, de diffusion de la connaissance et un outil pédagogique puissant qui permet de comprendre l'architecture et l'évolution de l'Univers.

L'Observatoire dispose d'un riche patrimoine d'instruments et d'ouvrage anciens.

2. Description

L'Observatoire astronomique de Strasbourg est aujourd'hui un Observatoire des Sciences de l'Univers (OSU) et une Unité de Formation et de Recherche (UFR) de l'Université de Strasbourg, ainsi qu'une Unité Mixte de Recherche (UMR 7550) entre l'Université de Strasbourg et le CNRS.

Il est structuré en trois équipes de recherche et deux Services d'Observation de l'Institut National des Sciences de l'Univers (INSU), le Survey Science Centre d'XMM-Newton (SSC-XMM) et le Centre de Données astronomiques de Strasbourg (CDS).

Son statut d'OSU place l'Observatoire astronomique au cœur du dispositif national mis en œuvre par l'INSU.

L'Observatoire astronomique de Strasbourg a pour mission de contribuer aux progrès de la connaissance par :

- l'acquisition de données d'observation
- le développement et l'exploitation de moyens appropriés
- l'élaboration des outils théoriques nécessaires

Il est par conséquent également chargé :

- de fournir des services liés à son activité de recherche
- d'assurer la formation des étudiants et des personnels de recherche
- d'assurer la diffusion des connaissances
- des activités de coopération internationale

3. Activités de recherche

La vocation initiale des activités de recherche concernait l'astronomie de position et l'observation des comètes, de météorites et d'étoiles variables. Par la suite les activités se sont étendues à la photométrie de nébuleuses et à l'observation d'étoiles doubles. Durant les années 1970, l'Observatoire développe l'archivage informatique, qui contribuera à la naissance du centre de données stellaires et qui deviendra en 1972 le Centre de Données astronomiques de Strasbourg (CDS).

Actuellement les activités de recherche s'articulent autour de trois grandes équipes scientifiques : « Astrophysique des hautes énergies » étudie la physique des astres compacts en fin d'évolution. « Galaxies » se charge des populations stellaires, propriétés chimiques et dynamiques de la Galaxie et des galaxies proches, milieu intergalactique, grandes structures et dynamique gravitationnelle. Enfin le CDS s'occupe des méthodes de gestion de l'information et de l'exploitation scientifique des grands relevés. Ce dernier est labellisé depuis 2008 "Très Grande Infrastructure de Recherche" (TGIR) par le Ministère de l'Enseignement Supérieur et de la Recherche.

4. Un rayonnement international

L'Observatoire est également membre du consortium Survey Science Center de la mission XMM-Newton. Cette dernière a pour objectif l'étude des rayons-X afin de mieux comprendre le fonctionnement de l'Univers et des événements violents qui s'y déroulent comme la vie des trous noirs ou les explosions d'étoiles. Mais c'est probablement le CDS qui contribue le plus à la renommée internationale de l'établissement. En effet il est membre fondateur de l'International Virtual Observatory Alliance (IVOA) dont nous parlerons ci-dessous, et est à l'initiative de services fortement utilisés par les astronomes et amateurs du monde entier.

5. Les Observatoires Virtuels

L'Observatoire Virtuel français est une collection d'archives de données interactives et des projets similaires d'autres pays comme la Chine, l'Australie, le Canada, le Japon, l'Inde, la Russie et la Corée, se sont associés en 2002 afin de coordonner leurs efforts au sein de l'alliance internationale IVOA avec la mission suivante : « Faciliter la

coordination et les collaborations internationales nécessaires au développement et au déploiement d'outils, de systèmes et de structures rendant possible l'utilisation des archives astronomiques comme s'il s'agissait d'un Observatoire Virtuel unique ».

L'IVOA regroupe ainsi 20 projets nationaux. Son action principale consiste à favoriser l'établissement de standards à travers des groupes de travail. Deux conférences par an permettent de coordonner les actions des différents groupes (IVOA Interoperability Conferences).

Les groupes de travail de l'IVOA couvrent l'ensemble des besoins de standardisation de la discipline. Des données jusqu'aux applications, ils définissent les formats des données, des métadonnées, les langages d'interrogation, les protocoles d'échange, etc.

Le Centre de Données astronomiques de Strasbourg, fortement impliqué dans l'Observatoire Virtuel, participe à de nombreux groupes de travail. Le logiciel Aladin a été régulièrement utilisé pour tester et valider certains de ces standards. De ce fait, Aladin est devenu l'exemple type de « l'outil compatible OV » et un portail reconnu d'accès à l'Observatoire Virtuel.

Il s'agit donc de construire des standards d'échange, des outils d'interrogation, des systèmes d'extraction de l'information, de manière à globaliser les données et plus généralement l'information en astronomie au niveau international. De nombreuses bases de données sont déjà interconnectées dans la communauté astronomique et de multiples applications client/serveur diffusent des données hétérogènes (articles de journaux, catalogues d'observations, images, spectres, etc.). Le Centre de Données astronomiques de Strasbourg (CDS) y prend une part importante au travers de ses services Aladin, Simbad ou VizieR qui sont utilisés par les astronomes du monde entier. Nous les présenterons dans la partie qui suit.

B. Centre de Données astronomiques de Strasbourg

J'ai intégré l'équipe de recherche du CDS, qui en plus d'être une équipe de recherche est un centre de données astronomiques, proposant notamment plusieurs services, décrits ci-dessous. Mon stage s'inscrit dans le cadre des activités de R&D à visée opérationnelle pour l'amélioration continue des services.

1. L'équipe du CDS

Le Centre de Données astronomiques de Strasbourg est l'une des principales composantes de l'Observatoire. Il offre un accès à des données astronomiques à forte valeur ajoutée au travers de services en ligne et d'applications grâce au travail de ses astronomes, informaticiens et documentalistes. Ils représentent un effectif d'une trentaine de personnes. Ces trois professions sont absolument complémentaires et sont indispensables au bon fonctionnement du centre. En effet, ils forment une chaîne entraînant le processus de collecte et d'enrichissement des données par les documentalistes, de vérification et d'exploitation des données par les astronomes et de développement d'outils et services pour la sauvegarde et l'accès aux données par les informaticiens.

La mission principale de ces derniers consiste à développer et maintenir les services et outils du CDS mais aussi d'assurer la veille technologique. En effet, la technique évolue très rapidement et le CDS tient à identifier les nouveautés prometteuses et à évaluer leur intérêt pour les services en ligne qu'il propose. Il a donc une activité significative de recherche et de développement.

2. Les services du CDS :

a) VizieR

VizieR est une base de données qui regroupe plus de 14 000 catalogues d'objets astronomiques. Ces catalogues sont en fait des tables relevées durant des missions d'observation et ajoutées à VizieR par les documentalistes. Ce service permet à un utilisateur d'accéder de manière homogène à des données hétérogènes de catalogues, de les croiser et de les exporter sous différents formats sur la base de catalogues. Les interrogations peuvent porter sur de multiples critères comme la longueur d'onde avec laquelle les objets ont été observés ou encore le nom de la mission correspondante.

b) Aladin

Aladin est un logiciel d'astronomie développé par le CDS pour la communauté internationale. Véritable atlas interactif du ciel, il permet aux scientifiques de localiser, accéder, comparer et analyser les données, images et catalogues issus de la plupart des serveurs astronomiques (images d'archives, relevés du ciel, catalogues, bases bibliographiques). Ses caractéristiques techniques lui ont permis de devenir l'un des outils clés de la discipline aussi bien pour la préparation de missions d'observations (télescopes spatiaux Hubble et James Webb), que pour la visualisation de données des plus importants centres de données astronomiques européens (ESO, ESAC, CDS), américains (MAST/NASA, NED/NASA), canadiens (CADC)… Il est un véritable intégrateur de données hétérogènes issues des sites et des projets répartis sur toute la planète.

c) Simbad

Simbad est la base de données de référence pour la nomenclature et la bibliographie des objets astronomiques. C'est un service qui permet aux astronomes à partir du nom, des coordonnées ou des références d'un objet d'accéder facilement aux informations comme les coordonnées, mesures physiques, données bibliographiques, etc. de plus de huit millions de références astronomiques en dehors du système solaire. Simbad dispose également d'un résolveur de noms qui permet de connaître toutes les nomenclatures d'un objet considéré.

II. Organisation du stage

A. Objectifs du stage

Dans le but d'améliorer en permanence la qualité des services hébergés, présentés ci-dessus, le CDS mène une activité soutenue de recherche et développement. C'est dans ce cadre que s'inscrit mon stage.

Articulé autour du domaine du Big Data (littéralement « grosses données ») en astronomie, le stage a initialement eu pour sujet « Big Data en astronomie (Hadoop, MapReduce, Spark, etc.) ».

Le domaine de l'astronomie est un domaine de Big Data par nature, étant données les quantités de données observées et traitées. En plus de cela, les quantités de données que le CDS est amené à traiter dans les années à venir vont évoluer de manière importante, ce qui accentue le besoin et l'intérêt de nouvelles technologies pour le traitement de grosses données.

L'objectif était en effet d'étudier et de mettre en place des technologies du Big Data dans le but de tester et prototyper des applications d'analyse et de traitement de données astronomiques.

Suite à une première étude de quelques technologies du Big Data et au fur et à mesure de l'avancement du stage, nous nous sommes orientés vers des technologies spécifiques, et le sujet du stage a été précisé en : « Etude et implémentation de Hadoop et Spark, application au cross-match du CDS ».

Le CDS propose notamment un service de cross-match (identification croisée) de catalogues de données astronomiques. Ce type de traitement est potentiellement « lourd » et demande la mise en place de techniques appropriées pour assurer de bonnes performances et l'utilisation dans des services en ligne. Le but est donc d'implémenter une application, réalisant un traitement de type cross-match, dans des technologies du Big Data comme Spark et Hadoop, présentées par la suite.

Les problématiques techniques du stage concernent le choix des outils du Big Data utilisés, les configurations et installations adoptées, l'environnement technique

approprié comme le nombre de nœuds (ou machines) nécessaires aux résultats attendus. Ces problématiques techniques mettent en jeu les performances en termes de temps et d'efficacité des traitements et des calculs sur les jeux de données de taille importante.

L'enjeu final du stage est de voir, après des tests comparatifs entre l'application du cross-match du CDS et celle écrite et implémentée au cœur des nouvelles technologies du Big Data, si l'on obtient des temps comparables.

B. Planning du stage

Les vingt-quatre semaines de stage ont été divisées en quatre grandes étapes :

- Etat de l'art, étude et prise en main des technologies du Big Data comme Hadoop, Spark et d'autres, et des langages de programmation utilisés par ces outils

- Implémentation et configuration de Spark et Hadoop sur les machines de l'observatoire

- Ecriture et implémentation d'une application de cross-match au sein de la structure Spark / Hadoop configurée auparavant

- Réalisation des tests

C. Méthodologie

Ce stage étant orienté vers la recherche en informatique, la méthodologie utilisée est de type R&D. On ne sait pas à l'avance les résultats que l'on va obtenir et si les choix réalisés sont les mieux adaptés à notre situation et à nos objectifs.

La méthode et le processus de développement suivis peuvent donc être différents de ceux pour un stage en entreprise étant donné que le stage s'est déroulé dans le cadre

d'activités R&D.

Après chaque test réalisé, une étape d'adaptation et de changement suivait. Au fur et à mesure que le stage avançait, il a fallu être réactif face aux problèmes rencontrés et faire des choix pour la suite. Ce stage a demandé de l'autonomie car il n'y avait pas d'experts dans le domaine des technologies utilisées telles que Spark et Hadoop, ce qui impliquait de nombreuses recherches et des choix à faire.

Des réunions régulières permettaient une mise au point sur l'avancement du stage, les problèmes rencontrés, les objectifs à atteindre et sur les résultats obtenus.
Le stage permettait une certaine flexibilité dans le travail puisqu'il n'y avait pas de date butoir hormis le fait de finir le travail au bout des vingt-quatre semaines de stage et d'atteindre les objectifs fixés initialement.

D. Description de l'existant

Concernant les technologies Spark et Hadoop utilisées durant le stage, le CDS avait une première expérience avec Hadoop mais dans un environnement très restreint d'une seule machine.

A mon arrivée à l'Observatoire, il a fallu repartir de zéro puisque la situation était différente. Rien n'était donc installé sur les machines. Après l'étude des technologies possibles et le choix de celles-ci, j'ai donc installé et configuré Spark et Hadoop sur les différentes machines de l'Observatoire mises à disposition.

En ce qui concerne l'application de cross-match à écrire et implémenter dans la deuxième partie du stage, celle-ci est écrite sur la base du cross-match « maison » du CDS développé par François-Xavier Pineau.

Ce cross-match développé au CDS et dénommé également « identification croisée » ou « corrélation croisée » est une jointure floue entre deux tables de plusieurs centaines de millions de données. Le terme de jointure floue désigne le fait que la jointure ne soit pas exacte pour toutes les données de la table.

Le programme de cross-match du CDS n'est exécuté que sur un seul serveur mais dont les performances sont assez importantes pour réaliser le traitement sur des tables de plusieurs millions de données.

L'application est mise à disposition de tous et actuellement utilisée par des astronomes et des centres de données astronomiques du monde entier pour effectuer des jointures entre différentes tables de données.

E. Aspect technique

Dans le contexte du processus de développement et de la méthodologie utilisée, je n'ai pas eu recours à des outils ou matériels spécifiques. En effet le code écrit en Java pour l'application de cross-match a demandé beaucoup de connaissance du fonctionnement de Spark et Hadoop mais pas un nombre de lignes de codes très important. Les erreurs et problèmes rencontrés venaient principalement de la structure Spark / Hadoop, ce qui ne demandait pas l'utilisation d'outils spécifiques.

III. Travail réalisé

Tout au long de mon stage à l'Observatoire, j'ai été amené à travailler, en collaboration avec mes deux tuteurs, sur un projet unique de son début jusqu'à la fin avec l'obtention des résultats. Cette dernière partie décrit donc de manière approfondie mon travail au sein du CDS, les tâches effectuées, les difficultés rencontrées et les résultats obtenus.

A. Analyse

Pour rappel, le stage s'articule autour du Big Data en astronomie, et comprend la connaissance et la manipulation des nouvelles technologies liées à ce domaine telles que Spark et Hadoop.
Le stage a pour objectifs d'implémenter et de tester des applications sur de larges catalogues de données et de réaliser des tests comparatifs avec Spark/Hadoop et avec le service d'identification croisée (cross-match) du CDS.

L'objectif de la première partie de mon travail a été de mettre en place les outils nécessaires pour l'implémentation de l'application de cross-match réalisée par la suite. Ces outils n'étaient pas fixés de manière définitive au départ, c'est pourquoi cette première partie d'étude des différents outils était importante pour la suite. Elle permettait de considérer les avantages et inconvénients de chaque technologie, de voir celle qui était la plus adaptée à notre situation et enfin de faire un choix quant à la configuration à adopter.
Cette section a eu une place importante au cours de mon stage puisque j'y ai appris à connaître et à manipuler les outils tels que Spark et Hadoop utilisés par la suite. Cette partie est également essentielle pour comprendre les choix réalisés, les problèmes rencontrés et les solutions apportées. Elle permet également de comprendre les tests réalisés et les résultats obtenus, c'est la raison pour laquelle cette partie tient également une place importante dans mon rapport.

Dans un premier temps, le travail a donc consisté à apprendre à connaître quelques outils et notions liés au domaine du Big Data.

1. Big Data

La notion de Big Data ou volumes massifs de données désigne des ensembles de données volumineux.
Mais l'idée du Big Data représente bien plus que la notion de volume, le Big Data c'est aussi une notion de variété de contenu et une vélocité de la collecte et du traitement en temps-réel.

Avec des quantités de données échangées et traitées de plus en plus volumineuses, le domaine du Big Data est considéré comme l'un des grands défis informatiques de la période 2010-2020 et est devenu un des axes de recherche et développement.

Les outils classiques de traitement de données qui jusque-là ont permis la manipulation des données, ne sont aujourd'hui plus adaptés aux données de type Big Data, d'où l'émergence de nouveaux outils, tels que Apache Hadoop.

2. Apache Hadoop

Apache Hadoop est un Framework open source permettant la création et l'exécution d'applications de calcul distribué sur de larges volumes de données dans des clusters de machines – réseaux de serveurs – allant jusqu'à des milliers de nœuds.
Les deux principales fonctions de Hadoop sont le système de fichiers distribué HDFS (Hadoop Distributed File System) et l'implémentation de l'architecture MapReduce.

MapReduce est un modèle de programmation parallèle et distribuée pour des traitements sur de très grandes quantités de données. Principalement utilisé dans des clusters d'ordinateurs, le modèle comprend deux fonctions map() et reduce() à implémenter.

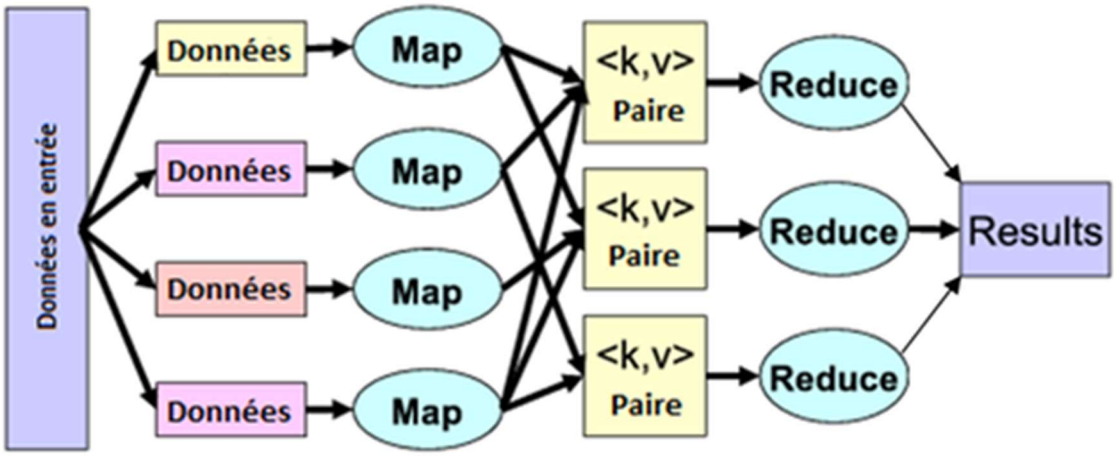
- Chaque nœud du cluster applique la fonction map() sur les données locales. Dans un nœud donné, la fonction map() associe aux objets de la collection de données d'entrée des couples (clé, valeur). Pour un même objet, une ligne d'un fichier d'entrée par exemple, on peut avoir par la fonction map() plusieurs paires (clé, valeur). La fonction map peut s'écrire de la façon suivante : map(clé1, valeur1) → List(clé2, valeur2).

- A chaque nœud, la fonction reduce regroupe ensuite les valeurs associées à une

même clé sous forme d'une unique paire (clé, valeur) et fait remonter l'information au nœud parent et ainsi de suite de manière récursive. Finalement le nœud d'origine peut générer des paires uniques (clé, valeur) où la valeur regroupe l'ensemble des valeurs associées à la clé dans le fichier d'entrée.

La fonction s'écrit : reduce(clé2, list(valeur2))→ valeur2.

Fonctionnement de MapReduce

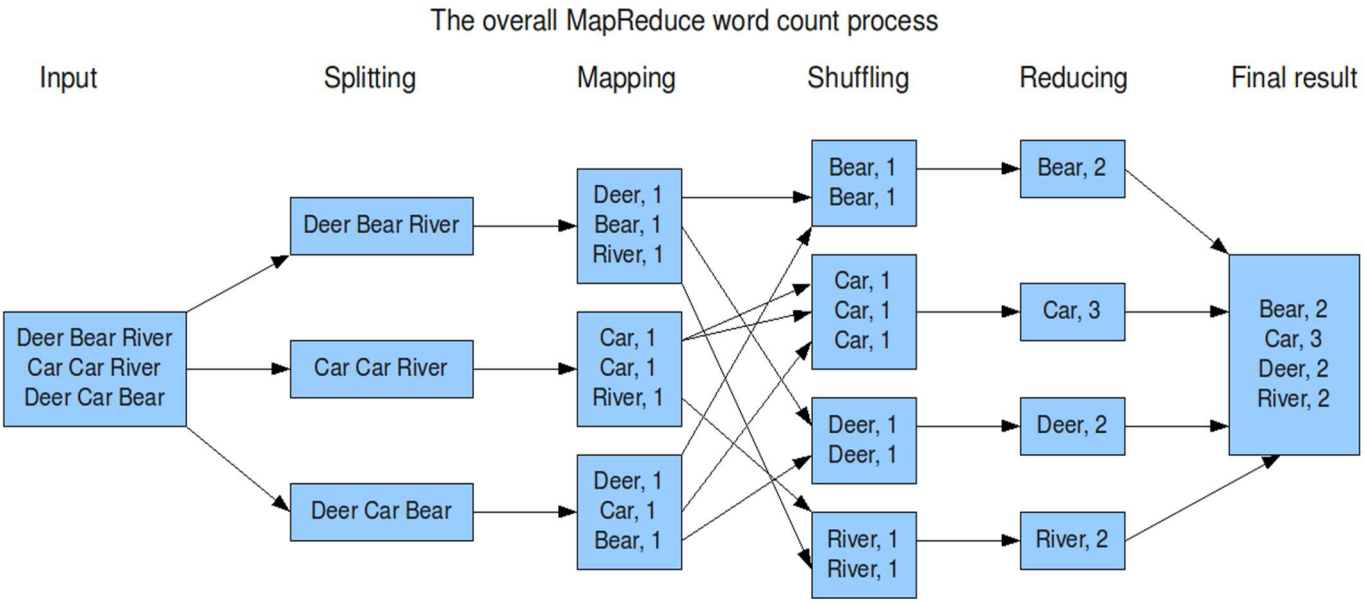


Crédits : Wikipédia

A ce schéma s'ajoute une étape entre les étapes de map et reduce : le shuffle. A cette étape, les résultats des fonctions map() - les couples (clé, valeur) – sont redistribués à travers le cluster par les différents nœuds en fonction des clés, tel que les couples ayant la même clé se retrouvent sur le même nœud.

Le schéma suivant présente les différentes étapes d'un modèle MapReduce qui compte le nombre d'occurrence des mots d'un fichier.

Exemple des différentes étapes du MapReduce



Crédits : Grégory PAUL

J'ai voulu commencer par travailler avec le Framework Apache Hadoop pour mieux comprendre le fonctionnement. L'installation de Hadoop peut se faire directement depuis le site d'Apache en téléchargeant une des différentes versions proposées ou bien par l'intermédiaire d'une distribution Hadoop. Contrairement à la version packagée d'Apache, les distributions Hadoop proposent des services de formation mais également un support avec la licence commerciale. Autrement le mode gratuit est possible.

Elles offrent en outre, en plus de Hadoop lui-même, des outils de la fondation Apache construits autour de Hadoop tels que Hive, Pig, Cassandra (outils de stockage et/ou de traitement des données). Ces distributions mettent également à disposition entre autres une console de management et des outils intégrés de monitoring, des fonctionnalités manquantes à la distribution Apache.

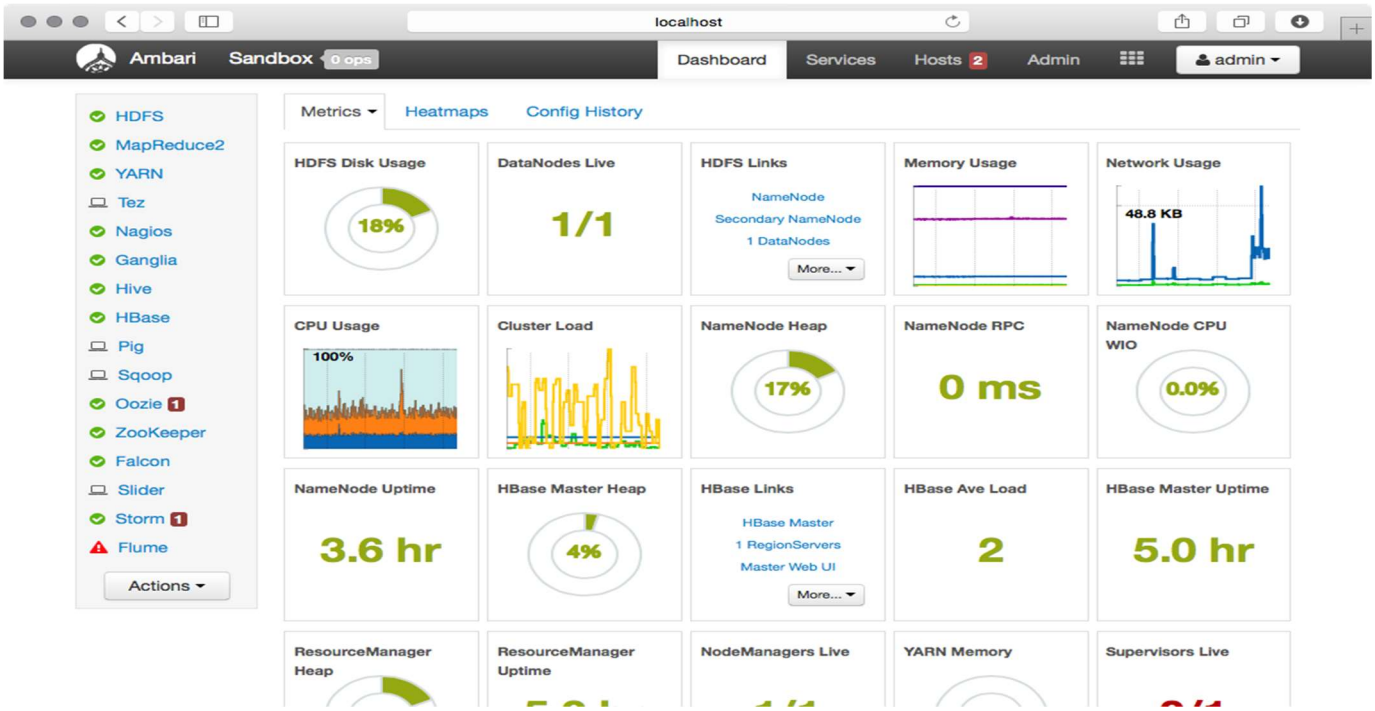
Parmi les principales distributions, on compte :

- Cloudera, avec la Cloudera Hadoop Distribution
- HortonWorks
- MapR

Après quelques difficultés rencontrées avec l'installation de la distribution Cloudera

CDH 5 (Cloudera's Distribution Including Apache Hadoop), je suis passée à la distribution HortonWorks avec un cluster simple nœud (une seule machine).
La plateforme de données HortonWorks (HortonWorks Data Platform – HDP) est une distribution Hadoop facilitant le déploiement et le management des clusters Hadoop. Pour un déploiement sur une seule machine, on peut opter pour l'installation via HortonWorks Sandbox, un environnement Hadoop installé sur une machine virtuelle. Sandbox comprend plusieurs outils dont la distribution HDP. L'accès et le contrôle des outils et applications de la distribution sont facilités via l'interface utilisateur web Ambari.

Aperçu de l'interface utilisateur Ambari



Crédits : HortonWorks

La distribution regroupe plusieurs outils tels que HDFS le système de fichiers distribué de Hadoop, YARN l'outil de management des ressources du cluster et de l'ordonnancement des tâches, qui est un élément essentiel de Hadoop, ainsi que d'autres applications de traitement et d'analyse de données comme MapReduce, Hive et Pig.
Après quelques lectures et recherches sur l'intérêt, le rôle et l'utilisation de ces différents outils, j'ai commencé à manipuler quelques fonctionnalités proposées par la distribution HDP en commençant par HDFS, qui permet de stocker des fichiers de manière distribuée vers un cluster d'ordinateurs même si pour le moment il s'agit d'une seule machine.

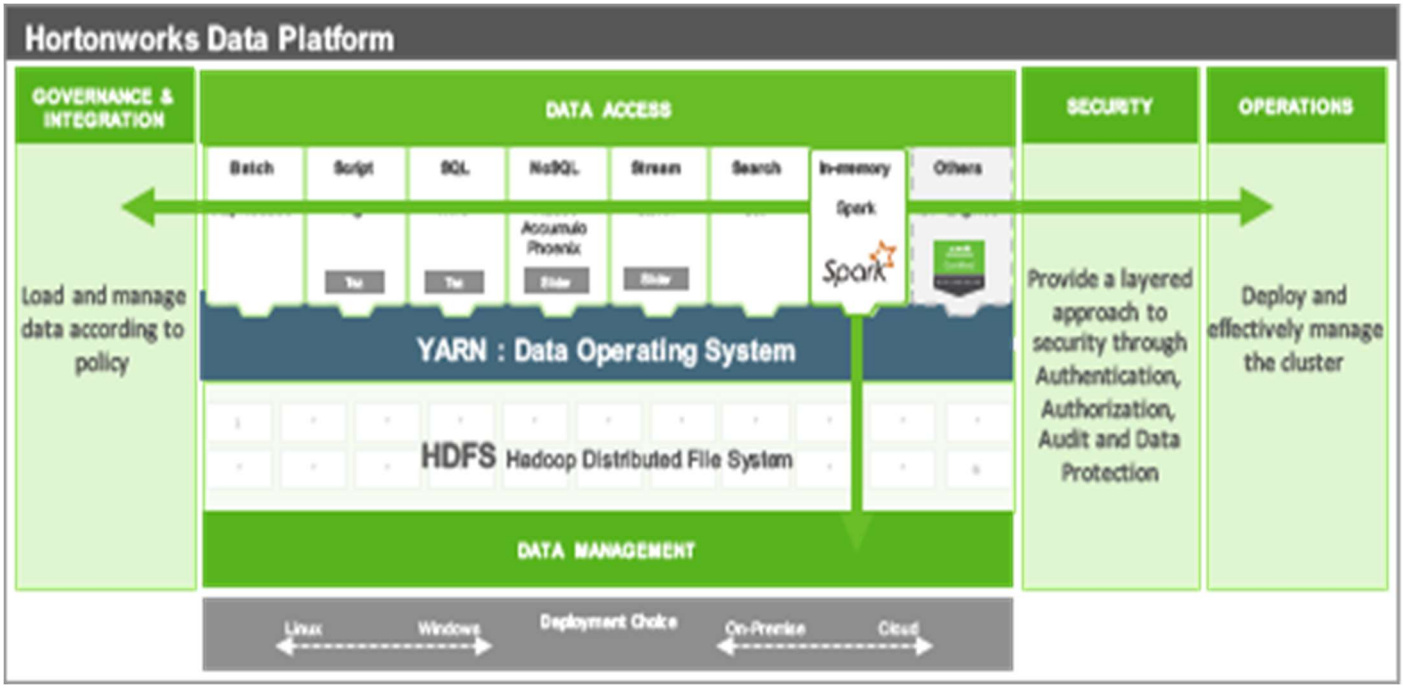
J'ai également testé Hive, un outil de traitement de données à travers des requêtes et des tables. Le langage utilisé est Hive Query Language (HiveQL), basé sur le même principe que le SQL avec quelques différences.
J'ai ensuite exécuté les mêmes types de traitement mais avec la plateforme Pig dont le langage est le Pig Latin. Les scripts Pig Latin permettent d'écrire des programmes MapReduce utilisés dans Hadoop et également de faire le lien avec des tables enregistrées dans Hive.

3. Apache Spark

Un autre composant essentiel de la distribution HDP est l'interface Spark. Apache Spark est un Framework de calcul distribué, comparable à Apache Hadoop avec un modèle de programmation plus simple et avec quelques différences dont le traitement en mémoire des applications, ce qui réduit de beaucoup les temps d'exécution comparés à ceux des applications MapReduce exécutées sur le disque. Développé après Hadoop, Spark est parfois vu comme un remplaçant de Hadoop, mais les deux Frameworks peuvent également être utilisés ensembles.

Spark peut en effet être lancé avec YARN, le gestionnaire de cluster de Hadoop et avec le système de fichiers distribué HDFS.

Intégration de Spark dans HDP



Crédits : HortonWorks

Pour prendre en main l'outil, j'ai d'abord utilisé Spark avec la distribution d'HortonWorks. Les traitements de Spark sont lancés depuis un terminal avec l'une des interfaces de programmation proposées qui sont Scala, Python et R ou bien depuis un code Java. Dans la distribution HDP il est possible d'utiliser des commandes HiveQL dans Spark pour créer et charger des tables par exemple. J'ai donc lancé quelques commandes avec les APIs en Python et en Scala pour des traitements simples sur des fichiers de données astronomiques.

B. Conception

Après une première phase de recherches liées au domaine du Big Data et après avoir pris connaissance et testé avec différents tutoriels les outils précédents liés aux traitements sur de larges volumes de données, le but était de déployer les outils à un cluster de plusieurs machines. Pour le développement des applications, il fallait choisir dans un premier temps entre Hadoop et Spark, le Framework Spark étant plus récent avec une nouvelle approche de calcul grâce aux traitements en mémoire, nous avons opté pour cette option.

Pour installer Spark, il suffit de télécharger la version souhaitée depuis le site d'Apache,

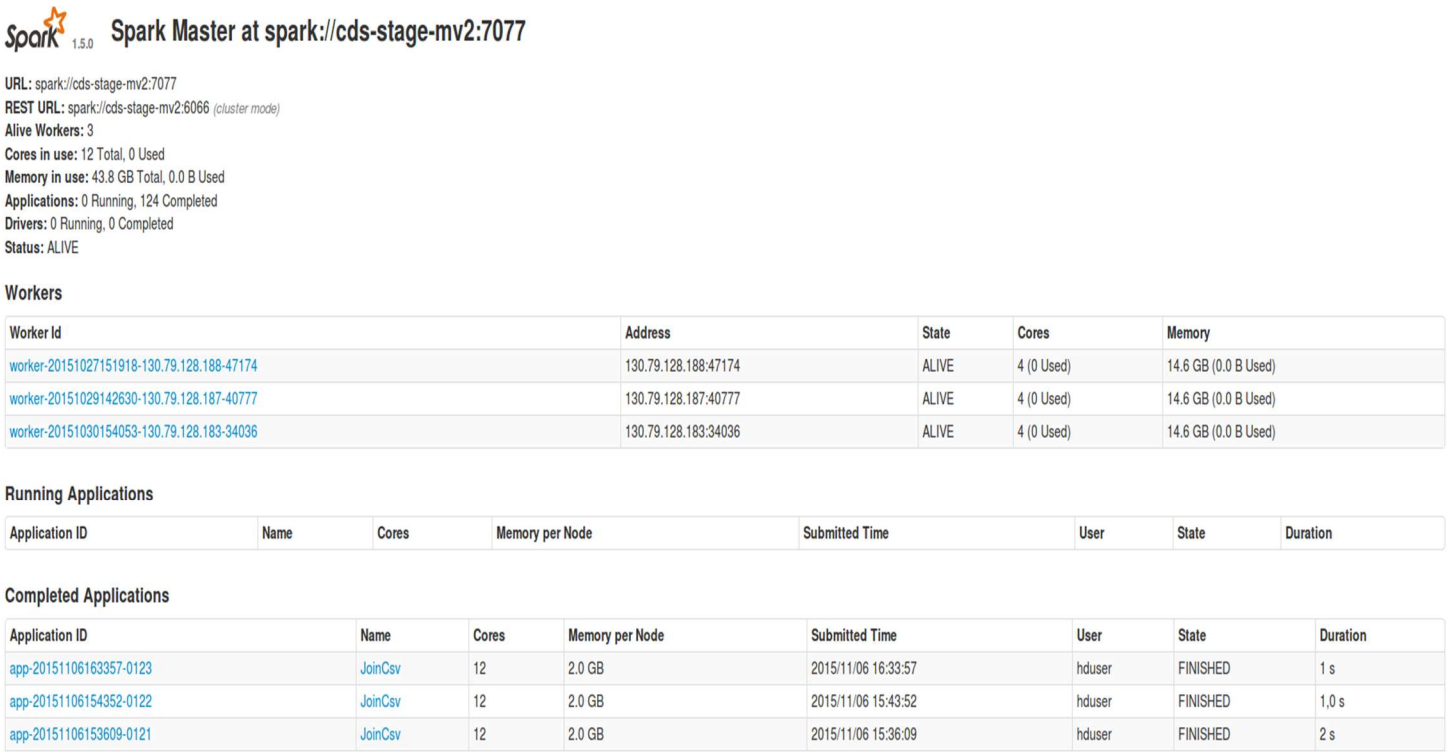
dans notre cas il s'agit de la version 1.5.0 pour Hadoop 2.6, en vue d'une utilisation simultanée de Spark et Hadoop.

Pour le déploiement sur plusieurs machines, nous avons choisi le mode standalone, dans lequel Spark a son propre gestionnaire de clusters. D'autres modes sont possibles, dans lesquels Spark tourne sur les gestionnaires de clusters Apache Mesos ou Apache Yarn.

Pour en revenir au mode standalone, on répète l'installation de Spark sur chacune des machines du cluster. Un cluster Spark se définit par une machine maître, appelée master et des nœuds, appelés workers. La machine maître peut également faire office de nœud.

Dans notre cas de figure, Spark est installé sur six machines avec un master et trois nœuds. Pour relier les nœuds au master, on modifie un fichier existant slaves.template, disponible dans la version téléchargée de Spark, en slaves et on y ajoute le nom ou l'adresse ip des nœuds du cluster. Ce fichier n'est modifié que sur la machine maître. Pour démarrer ou arrêter le master et les nœuds, on peut choisir d'utiliser les scripts de lancement ou d'arrêt fournis dans Spark ou le faire manuellement en ligne de commande depuis un terminal. Pour pouvoir utiliser les scripts et pour contrôler librement les nœuds depuis le master, il faut penser à configurer un accès SSH sans mot de passe sur les nœuds.

Interface web de la machine maître



Spark 1.5.0 **Spark Master at spark://cds-stage-mv2:7077**

URL: spark://cds-stage-mv2:7077
REST URL: spark://cds-stage-mv2:6066 *(cluster mode)*
Alive Workers: 3
Cores in use: 12 Total, 0 Used
Memory in use: 43.8 GB Total, 0.0 B Used
Applications: 0 Running, 124 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers

Worker Id	Address	State	Cores	Memory
worker-20151027151918-130.79.128.188-47174	130.79.128.188:47174	ALIVE	4 (0 Used)	14.6 GB (0.0 B Used)
worker-20151029142630-130.79.128.187-40777	130.79.128.187:40777	ALIVE	4 (0 Used)	14.6 GB (0.0 B Used)
worker-20151030154053-130.79.128.183-34036	130.79.128.183:34036	ALIVE	4 (0 Used)	14.6 GB (0.0 B Used)

Running Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration

Completed Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20151106163357-0123	JoinCsv	12	2.0 GB	2015/11/06 16:33:57	hduser	FINISHED	1 s
app-20151106154352-0122	JoinCsv	12	2.0 GB	2015/11/06 15:43:52	hduser	FINISHED	1,0 s
app-20151106153609-0121	JoinCsv	12	2.0 GB	2015/11/06 15:36:09	hduser	FINISHED	2 s

Les interfaces web du master et des workers affichent, en plus de certaines informations sur le cluster, des renseignements sur les applications en train d'être exécutées et celles qui sont terminées.

A ce stade, le cluster Spark est configuré et les applications peuvent être lancées. Pour la suite, les applications Spark seront développées en Java. Dans le programme, on lit un fichier de données, qui doit être présent sur chacun des nœuds du cluster Spark, soit en local sur les machines, soit par l'intermédiaire d'un système de fichiers distribué. On choisit d'installer le système de fichiers distribué de Hadoop – HDFS – pour y stocker nos fichiers d'entrée et nos résultats.

HDFS permet de stocker de larges volumes de données sur un très grand nombre de machines et permet également d'éviter les pertes de données grâce au facteur de réplication des fichiers. L'architecture du système HDFS se distingue par :

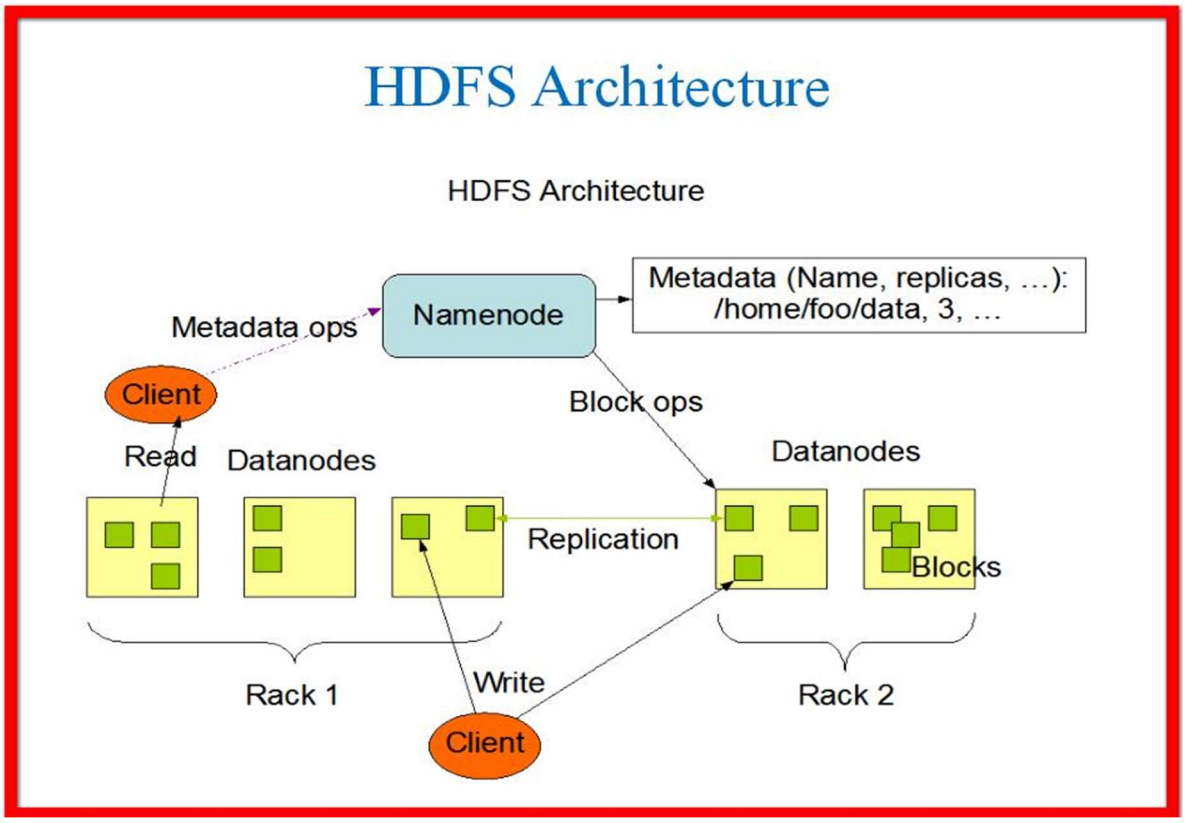
- un unique namenode : cette machine gère et contient toutes les métadonnées liées aux fichiers, telles que les noms, le facteur de réplication, l'emplacement, elle contrôle également la répartition des blocs de fichiers au sein du cluster.
- un ou plusieurs datanodes : une machine faisant office de datanode stocke les données et communique régulièrement avec le namenode pour l'informer de tous

les blocs de données qu'elle contient.

Lors de l'enregistrement d'un fichier vers HDFS, celui-ci est découpé en plusieurs blocs et chaque bloc est copié sur plusieurs datanodes en fonction du facteur de réplication. L'utilisateur peut contrôler et modifier la taille d'un bloc HDFS (par défaut à 64 ou 128MB) ainsi que le facteur de réplication des blocs (par défaut à trois) en fonction du nombre de datanodes dans le cluster.

Pour utiliser le système HDFS, on installe le Framework Hadoop sur toutes les machines du cluster. La version choisie 2.6 est la version compatible avec celle de Spark. Le but est d'utiliser Hadoop pour stocker les fichiers dans le cluster par l'intermédiaire du système HDFS et d'utiliser Spark pour l'exécution des applications Java.

Architecture du système HDFS



Crédits : Apache Hadoop

C. Implémentation

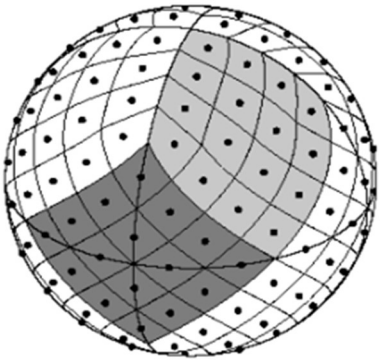
Suite à la mise en place de Spark et du système de fichiers HDFS de Hadoop, on peut maintenant lancer l'application de cross-match. Bien qu'implémenté de manière différente, le programme écrit pour Spark réalise le même traitement que celui du CDS c'est-à-dire une jointure floue entre deux tables allant jusqu'à plusieurs centaines de millions de données.

Le programme est divisé en deux parties. Les fichiers en entrée sont deux tables contenant des positions dans le ciel. Les données en sortie sont tous les éléments de la deuxième table étant à une certaine distance des éléments de la première.

1. Première partie : Préparation des données

Avant l'exécution de cette première partie, les fichiers d'entrée sont stockés dans HDFS.
Ces fichiers sont dans un premier temps chargés dans deux RDDs simples où chaque ligne du RDD est un élément contenant des informations sur un objet dans le ciel.
Le RDD ou Resilient Distributed Dataset est une collection distribuée de données.
Chaque RDD est ensuite transformé en PairRDD : à chaque élément du RDD est attribuée une clé représentant le numéro de pixel de la source grâce à un découpage HEALPix du ciel.

Découpage HEALPix du ciel



Crédits : HEALPix – arXiv:astro-ph/0409513

Le ciel est en effet découpé en losanges de tailles identiques, appelés pixels, de sorte à ce que chaque source ou objet du ciel soit positionné dans un pixel numéroté.
Les éléments des PairRDDs sont alors des couples (clé, valeur) où la valeur contient

toutes les informations dont les coordonnées (ra, dec) de la source dans le système de coordonnées équatoriales : ra étant pour *Right Ascension* ou Ascension Droite et dec pour *Declination* ou Déclinaison.
Les PairRDDs sont ensuite distribués sur les différents nœuds du cluster. Cette distribution est faite sur la base d'un partitionnement par hachage où les PairRDDs sont découpés en partitions qui vont être stockés sur les nœuds. Un partitionnement par hachage consiste à regrouper tous les éléments ayant la même clé (même numéro de pixel) dans une même partition. Les partitions étant ensuite stockés sur des nœuds différents, les éléments de même clé se retrouvent sur les mêmes nœuds. Cette distribution des données est essentielle pour la deuxième partie du programme.

Enfin les PairRDDs sont enregistrés dans HDFS sous forme de fichiers binaires grâce à une méthode permettant de garder la structure (clé, valeur).

2. Seconde partie : jointure des données

Les fichiers binaires enregistrés précédemment sont directement chargés dans des PairRDDs.
Sur le deuxième PairRDD est appliquée une méthode qui duplique certaines sources dans les pixels voisins. J'explique cette duplication par la suite.

Les deux PairRDDs sont ensuite joints au niveau de la clé. La jointure donne lieu à un nouveau PairRDD où les éléments sont de type (clé, valeur1, valeur2).
Pour décrire le cross-match, j'ai employé le terme de « jointure floue ». En effet, la jointure étant faite sur la clé (numéro de cellule), deux sources proches peuvent être dans des cellules différentes et ne sont donc pas jointes. D'où la duplication des sources dans les cellules voisines pour limiter les effets de bord.
La duplication est effectuée de manière suivante : un cercle de rayon fixé est tracé autour de la source. Si des pixels voisins se trouvent en partie à l'intérieur de ce cercle, la source est alors dupliquée dans ces cellules voisines.

Les éléments joints sont ensuite filtrés : seuls les éléments joints dont la distance entre les deux sources est inférieure à un certain seuil sont gardés.

Le résultat final est enregistré dans HDFS sous format texte pour une visualisation et une utilisation ultérieures.

D. Tests et validation

Avant d'écrire le programme de cross-match présenté ci-dessus, j'ai d'abord écrit et testé plusieurs autres programmes de jointures pour d'une part prendre en main Spark et HDFS et d'autre part pour me familiariser avec la programmation en JavaSpark. J'ai donc testé différentes implémentations possibles pour une même jointure, et également différents types de jointures telles qu'une jointure simple ou une jointure à +/- 1 où une clé est jointe à une même autre clé, à la clé + 1 et à la clé – 1.

Pour le programme de cross-match, j'ai réalisé les tests en commençant avec des fichiers de petite taille (~500 MB), puis de taille plus importante (~60 GB). Etant donné que les performances et la réussite de l'exécution du code sont liées en partie à la capacité de mémoire disponible, un même code peut fonctionner avec des fichiers de petite taille et non avec des fichiers de taille plus importante. Les premiers tests permettent donc de voir plus rapidement les résultats et voir ainsi si les données en sortie correspondent à ceux attendus et de voir si les problèmes rencontrés viennent du code ou des capacités physiques des machines.

Spark et Hadoop donnent un accès à un grand nombre de paramètres à configurer qui ont pour la plupart une valeur par défaut. En fonction de ces paramètres, un même code peut durer quelques minutes ou plusieurs heures. J'ai donc testé plusieurs configurations possibles, en modifiant les paramètres internes de Spark et HDFS, en modifiant également l'écriture du code de cross-match jusqu'à trouver la configuration la plus optimale possible, chaque configuration ayant des avantages et des inconvénients.

E. Déploiement sur matériel spécifique

J'ai d'abord réalisé les tests sur les machines de l'Observatoire, en augmentant la taille du cluster petit à petit jusqu'à six machines. Les machines mises à disposition à l'Observatoire avaient une capacité en mémoire vive de 16Go et une capacité sur le disque d'environ 700Go.

Les traitements dans Spark sont réalisés en mémoire et les résultats intermédiaires sont également stockés en mémoire, c'est pourquoi la capacité en mémoire vive d'un nœud

influence beaucoup les performances.

Par conséquent et également pour augmenter le parallélisme dans le traitement des données, plus on a de nœuds dans le cluster, plus les performances sont meilleures.

Pour ces raisons, le CDS a financé un prêt de douze serveurs dédiés chez OVH, un hébergeur de sites web français, pour une durée d'une semaine.

Chaque serveur avait une capacité en mémoire vive de 32 Go et une capacité sur le disque de 4To, faisant un total de 384Go de RAM et 48To de disque.

F. Résultats

Les résultats des tests avec les serveurs dédiés OVH sont ceux qui ont permis d'atteindre les objectifs fixés au départ en terme de temps : on a réussi à obtenir un temps de cross-match inférieur à celui du cross-match du CDS. La comparaison des temps est faite sur la 2ème partie seulement, sur le temps de jointure et non sur la préparation des données, car les données sont déjà préparées dans le cadre du cross-match du CDS.

Le tableau ci-dessous présente les temps obtenus avec les serveurs dédiés OVH. Les temps sont exprimés en minutes.

Cross-Match (duplication des sources faite dans la 2e partie ; avec toutes les données en sortie)											
Taille des blocs HDFS = 128MB pour les fichiers en entrée ; sdss7.csv et 2mass.csv répliqués 2x											
HashPartitioner	**60 partitions**										
Taille des blocs HDFS en sortie	**32MB**										
Nombre de nœuds Spark/HDFS	1	2	3	4	5	6	7	8	9	10	11
1ère partie : préparation des données		**40,0**	**28,0**		**23,0**		**16,0**		**14,0**	**14,0**	**13,0**
mapToPair (sdss7.csv)		7,8			5,1		4,9		4,9	4,8	4,7
saveAsHadoopFile (sdss7.bin)		10,0			5,7		2,7		2,0	2,3	1,5
mapToPair (2mass.csv)		8,5			5,7		5,2		5,2	5,1	5,0
saveAsHadoopFile (2mass.bin)		13,0			6,5		3,6		1,9	1,6	1,4
2ème partie : jointure		**53,0**	**45,0**		**31,0**		**21,0**		**13,0**	**11,0**	**9,9**
mapToPair (sdss7.bin)					7,2		4,7		3,5	3,0	2,9
flatMapToPair (2mass.bin)					11,8		8,3		5,5	4,9	4,3
saveAsTextFile (crossMatch_D.txt)					12,0		7,6		3,4	2,4	2,3
TOTAL		**93,0**	**73,0**		**54,0**		**37,0**		**27,0**	**25,0**	**22,9**

La configuration retenue a donc été la suivante :

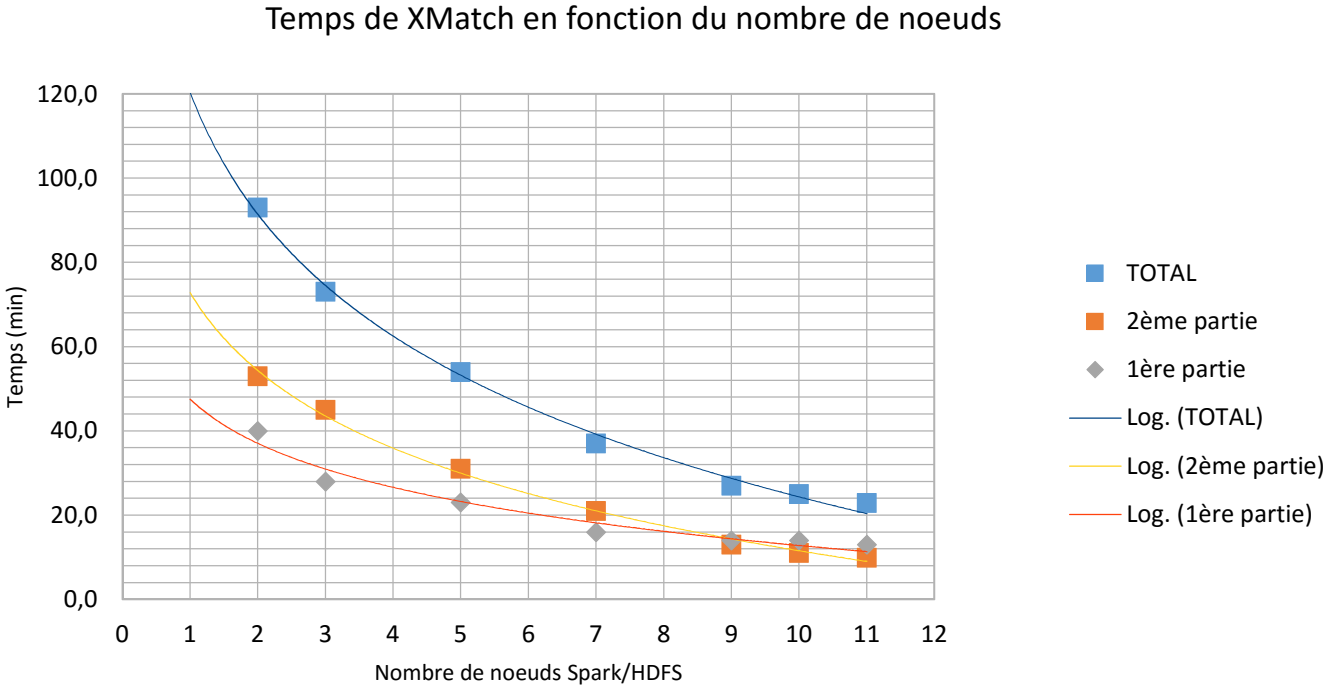
- La duplication des sources est faite dans la 2ème partie, car lorsqu'elle est faite en 1ère partie, les temps sont plus longs dus aux entrées/sorties.

- On enregistre toutes les données en sortie
- Les fichiers en entrée sont des fichiers contenant des informations sur des sources observées dans le ciel
    - sdss7.csv : 54 Go ; 355 000 000 de sources
    - 2mass.csv : 58 Go ; 470 000 000 de sources
    - Ces fichiers sont placés sur 2 nœuds avant le lancement du programme
- Dans HDFS, les fichiers sont découpés en blocs. La taille des blocs HDFS est fixée à 128 MB pour les fichiers en entrée. En fonction de cette taille, le nombre de blocs par fichier est également défini.
- Lors du partitionnement par hachage des PairRDDs, on choisit le nombre de partitions dans lesquels les données vont être placées. Ce nombre est fixé à 60.
- Pour les fichiers sauvegardés dans HDFS depuis le programme, la taille des blocs est configurée à 32MB.

Le résultat de la jointure donne un fichier de 49 208 820 d'éléments en sortie.

Je n'avais les serveurs à disposition que pour une semaine et ils n'ont pas été mis à disposition en même temps, c'est pour cela que l'accès aux serveurs a été perdu progressivement et le temps de mettre à nouveau en place Spark et HDFS et de régler des problèmes techniques, je n'avais plus accès au premier serveur. Les tests ont donc été faits à partir de 11 serveurs.

On obtient les courbes suivantes :



Pour les mêmes fichiers d'entrée, le cross-match du CDS donne un temps de 15 min environ, il s'agit du temps de jointure ce qui correspond à la deuxième partie de ce cross-match. On peut donc noter qu'avec 11 serveurs dédiés OVH, on obtient un temps de 9.9 min, ce qui est bien en-dessous du premier temps.

D'après les courbes, au-delà de huit serveurs, les temps d'exécution diminuent de moins en moins.

En supposant que l'on prenne huit serveurs dédiés, on aurait un coût d'environ 600€ / mois, un prix qui inclut la maintenance et l'entretien des machines. De plus, on dispose d'une installation flexible. Ce prix serait donc négligeable par rapport au salaire d'un ingénieur en charge du développement d'un cross-match en dehors des structures Spark et Hadoop comme celui du CDS.

G. Difficultés rencontrées

J'ai eu à faire face à de nombreux problèmes tout au long de mon stage, que ce soit des difficultés techniques ou des difficultés de compréhension des outils utilisés.

Les principales difficultés ont été au niveau de la connaissance des technologies Spark et HDFS. Ces outils demandent une étude très approfondie du sujet car il y a beaucoup de paramètres internes à configurer pour obtenir des résultats optimisés. De plus il n'y avait pas à l'Observatoire de spécialiste de ces technologies, il m'a donc fallu être très autonome, trouver les réponses par moi-même et faire des choix quant aux configurations à adopter.

J'ai en effet fait beaucoup de recherches sur différents sites pour trouver les réponses aux questions que je me posais. Les recherches sur Internet n'ont pas toujours abouti car ce sont des domaines encore récents sur lesquels peu d'entreprises ou de développeurs travaillent pour le moment. La documentation en ligne n'est donc pas encore complète. Par ailleurs, Spark étant écrit en scala, les nombreux exemples sont donc dans ce langage.

Un des éléments pour lesquels il m'a été très difficile de trouver une réponse a été la co-location des données. Lors du partitionnement par hachage des RDDs, les éléments

de même clé sont placés sur les mêmes nœuds pour un RDD donné. Cependant ceci n'implique pas que des clés communes à deux RDDs soient également sur les mêmes nœuds. Dans ce cas, cela implique un temps de transfert des données entre les nœuds lors de la jointure, ce qui affecte les performances.
La co-location des données semble actuellement non-paramétrable et pour avoir cette information j'ai posé la question sur le forum d'Apache Spark et également sur le site Stack Overflow sans n'avoir aucune réponse. La co-location des données concerne un domaine avancé de Spark ce qui peut expliquer l'absence d'information.

Une autre difficulté que j'aimerais mentionner concerne les serveurs dédiés OVH. Je n'y avais en effet accès que le temps d'une semaine, ce qui a été court pour la configuration, la résolution des problèmes et enfin la réalisation des tests qui prenaient de plus en plus de temps au fur et à mesure que l'on diminuait le nombre de nœuds.

Enfin lorsque je rencontrais des erreurs lors de l'exécution du code, il n'était pas évident de trouver l'origine de l'erreur car il n'y avait pas d'outil spécifique à Spark pour débugger. Je procédais souvent par élimination.

Conclusion

En conclusion de ce rapport et au terme de six mois de stage passés à l'Observatoire astronomique de Strasbourg, le travail réalisé a permis de trouver une alternative au service de cross-match du CDS grâce aux nouvelles technologies du Big Data, Spark, un Framework de calcul distribué et HDFS, le système de fichiers distribué de Hadoop.

L'application de cross-match a été testée et fonctionne. Les résultats et temps obtenus correspondent aux objectifs du stage. Avec 11 nœuds OVH on obtient même un temps inférieur à celui du service de cross-match du CDS.

Le travail réalisé a également permis d'observer que la location de huit serveurs dédiés chez OVH serait plus rentable pour une entreprise que de développer elle-même son propre service de cross-match sans l'aide des structures Spark ou HDFS.

La fin du stage laisse quelques perspectives. Parmi elles, une possible co-location des données dans les versions futures de Spark et/ou de HDFS. Cette possibilité permettrait de gagner du temps sur le temps de transfert des données entre les nœuds.

D'autres tests pourraient être faits avec un nombre supérieur de nœuds pour voir le temps minimum que l'on pourrait avoir avec cette même configuration.

Des tests avec d'autres changements dans les paramètres internes de Spark et HDFS peuvent également être envisageables pour améliorer les performances.

Enfin, ce stage a été très formateur tant pour la connaissance et la maîtrise des technologies que pour les aspects recherches et autonomie.

Bibliographie

Observatoire astronomique de Strasbourg, https://astro.unistra.fr/

Sites officiels des services du Centre de Données astronomiques de Strasbourg
CDS, http://cdsweb.u-strasbg.fr/index-fr.gml
VizieR, http://vizier.u-strasbg.fr
Simbad, http://simbad.u-strasbg.fr/simbad/
Aladin, http://aladin.u-strasbg.fr/aladin.gml
CDS X-Match Service, http://cdsxmatch.u-strasbg.fr/xmatch

Technologies utilisées
Apache Spark, http://spark.apache.org/
Apache Hadoop, http://hadoop.apache.org/

Spark : Cluster Computing with Working Sets, Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, Ion Stoica, University of California, Berkeley, http://static.usenix.org/legacy/events/hotcloud10/tech/full_papers/Zaharia.pdf

Optimizing Shuffle Performance in Spark, Aaron Davidson, Andrew Or, UC Berkeley, http://www.cs.berkeley.edu/~kubitron/courses/cs262a-F13/projects/reports/project16_report.pdf

Resilient Distributed Datasets : A Fault-Tolerant Abstraction for In-Memory Cluster Computing, Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, Ion Stoica, University of California, Berkeley, https://www.cs.berkeley.edu/~matei/papers/2012/nsdi_spark.pdf

JavaSpark Api, http://spark.apache.org/docs/latest/api/java/

HEALPix, http://healpix.jpl.nasa.gov/

Wikipédia, l'encyclopédie libre, https://fr.wikipedia.org
Stack Overflow, http://stackoverflow.com/

Mots clefs

11 – Centre ou laboratoire de recherche ; 16 – Fonction publique
13 – Recherche ; 07 - Informatique
06 – Base de données ; Autres : Big Data, Spark, Hadoop
03 – Logiciel d'analyse de données ; Autre : Cross-Match

WALI Noémie

Rapport de stage ST40 - A2015

Résumé

Ce rapport résume six mois de stage passés au sein du Centre de Données astronomiques de Strasbourg (CDS), une équipe de l'Observatoire astronomique de Strasbourg, dont les objectifs sont de collecter, d'enrichir et de distribuer les données astronomiques à la communauté internationale.

En collaboration avec mes deux tuteurs de stage, j'ai installé et configuré des outils de traitement et de stockage de données de type Big Data, dans un réseau allant d'une à douze machines. A l'aide de mon tuteur, j'ai également développé une application de type cross-match (identification croisée) permettant de joindre des objets astronomiques sur des critères définis.

Ce rapport décrit donc les objectifs de mon stage, le travail réalisé, les contraintes techniques et les difficultés rencontrées. J'y décris également les différents tests réalisés et les résultats obtenus en accord avec les objectifs fixés initialement.

Observatoire astronomique de Strasbourg
11 rue de l'Université
67 000 Strasbourg

Observatoire astronomique de Strasbourg

utbm université de technologie Belfort-Montbéliard