ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

RAPPORT DE PROJET PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAÎTRISE EN GÉNIE

PAR
Sébastien SERVOLES

UTILISATION DU BIG DATA EN GÉNOMIQUE POUR L'ANALYSE DES VARIATIONS GÉNÉTIQUES

MONTRÉAL, LE 19 AVRIL 2013

PRÉSENTATION DU JURY

CE RAPPORT DE PROJET A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

M. April, directeur de projet
Département de génie logiciel et des TI à l'École de technologie supérieure

M. Gherbi, codirecteur de projet
Département de génie logiciel et des TI à l'École de technologie supérieure

Mme El Boussaidi, présidente du jury
Département de génie logiciel et des TI à l'École de technologie supérieure

REMERCIEMENTS

Ce projet a été rendu possible à l'aide des initiatives d'Alain April dans les domaines des technologies et de la santé, et je tiens donc à le remercier pour ses efforts et son aide qui m'ont permis de mener à bien mes travaux. Je remercie Abdelouahed Gherbi pour ses conseils et son soutien tout au long du projet.

Je remercie également Patrice Dion pour son aide et son support au niveau des ressources et des systèmes informatiques de l'ÉTS.

Enfin, je remercie les étudiants Anna Klos, Jean-Philippe Bond et Jean-François Hamelin pour leurs commentaires et conseils en début de projet.

UTILISATION DU BIG DATA EN GÉNOMIQUE POUR L'ANALYSE DES VARIATIONS GÉNÉTIQUES

SÉBASTIEN SERVOLES

RÉSUMÉ

Le Big Data est un ensemble de technologies qui permettent de traiter de très grands volumes de données informatiques. Celles-ci sont apparues avec les besoins toujours plus importants des entreprises au niveau du traitement de données, et les limites des bases de données relationnelles pour certains cas d'utilisation. Certains domaines scientifiques qui ont recours à l'analyse de données peuvent également recourir à ces technologies. La génomique, qui est l'étude du génome, soit l'information génétique des espèces vivantes, est l'un de ces domaines. En effet, le génome est une source d'information gigantesque, qui nécessite des systèmes informatiques adaptés pour son analyse.

Ce projet vise à étudier les technologies de Big Data pour une application particulière en génomique : le stockage des variations génétiques chez l'être humain, et la mise à disposition d'une interface web permettant l'accès à des informations utiles.

USING BIG DATA FOR GENOMIC TO ANALYZE GENETIC VARIANTS

SÉBASTIEN SERVOLES

ABSTRACT

Big Data is a set of technologies used to process very big amount of digital data. These technologies have been introduced with the constant growing of the requirements companies are facing, and the limits of relational databases for specific use cases. Some scientific fields that need to analyze some data also have an interest about using these technologies. Genomic, which is the study of the genome, the genetic information of living species, is one of these fields, because the genome is a huge source of information that needs adapted information systems in order to be analyzed.

This project aims to study some Big Data technologies for a specific application: the storage of human genetic variants and the development of a web interface to query and access useful information about these variants.

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

ADN	Acide Désoxyribonucléique
API	Application Programming Interface
AWS	Amazon Web Service
CHUM	Centre Hospitalier de l'Université de Montréal
EC2	Elastic Compute Cloud
ÉTS	École de Technologie Supérieure
HDFS	Hadoop Distributed File System
IaaS	Infrastructure as a Service
INDEL	Insertion-Deletion
JAR	Java ARchive
REST	REpresentational State Transfer
S3	Simple Storage Service
SGBD	Système de gestion de base de données
SNP	Single Nucleotide Polymorphism
SQL	Structured Query Language
SV	Structural Variation
VCF	Variant Call Format

INTRODUCTION

Depuis plusieurs décennies, le modèle relationnel s'est imposé pour le stockage de données informatiques de manière structurée. Ce modèle est aujourd'hui utilisé par la plupart des systèmes de gestion de base de données (SGBD) car il possède plusieurs avantages. Il est suffisamment flexible pour gérer un très grand nombre de situations différentes, et le langage SQL s'adapte parfaitement à ce modèle et permet d'utiliser la plupart de ces bases de données de la même manière, ce qui facilite grandement leur utilisation.

Mais si ces bases de données ont permis de contenir l'information de manière efficace jusqu'à maintenant, elles ne sont plus adaptées à toutes les situations. En effet, le volume de données à stocker augmente continuellement. Avec l'apparition d'Internet, ce phénomène s'est accentué, et certains sites web importants ont commencé à générer des quantités de données suffisamment grandes pour que les bases de données classiques ne soient plus suffisantes pour leurs applications (Chang et al., 2008). Il a donc fallu inventer de nouveaux moyens pour traiter une telle quantité d'information.

C'est ainsi que sont nées les technologies de Big Data. Il s'agit de l'ensemble des techniques et outils utilisés pour gérer un très grand volume de données. Ces volumes de données peuvent représenter des pétaoctets, voir des exaoctets (un milliard de gigaoctets). Parmi les domaines qui requièrent de tels volumes à analyser, on retrouve la météorologie, la biologie, la génomique, etc. Dans le domaine des technologies de l'information, la gestion des « logs » (journaux applicatifs) et l'indexation web sont deux exemples couramment utilisés pour représenter les besoins en matière de Big Data.

Les technologies développées depuis quelques années ont généralement la particularité de s'éloigner du modèle relationnel des bases de données principalement utilisées aujourd'hui, afin de viser l'optimisation des performances et la prise en charge de volumes de données dont la taille peut être multipliée à l'infini.

Google a présenté en 2006 sa technologie BigTable (Chang et al., 2008). Il s'agit d'une base de données répartie à hautes performances. L'indexation web, nécessaire pour le moteur de recherche de Google, est le premier système qui va utiliser cette base de données. HBase est une base de données soutenue par la fondation Apache qui a été développée sur le modèle de BigTable (Khetrapal et Ganesh, 2006). Elle est aujourd'hui couramment utilisée, et disponible sous forme de logiciel libre.

Ce rapport présente un projet qui vise à utiliser la base de données HBase afin de proposer une solution à une problématique liée à un grand volume de données dans un domaine en particulier : la génomique. Pour cela, de nombreux aspects ont été évalués, du schéma de la base de données à l'utilisation d'un cluster dans le « cloud » afin de valider la mise en place de la solution et d'obtenir des résultats de tests de performance.

Dans un premier temps, le contexte du projet et le domaine de la génomique seront présentés plus en détail. Dans le deuxième chapitre, les technologies existantes, la problématique étudiée et les objectifs seront abordés. Dans le troisième chapitre, le travail effectué au cours de ce projet sera présenté. Enfin, les résultats des expérimentations et leur synthèse seront présentés dans le chapitre 4.

CHAPITRE 1

Contexte du projet et terminologie

1.1 Définition de la génomique

Le génome est l'ensemble de l'information génétique des individus ou des espèces vivantes. Cette information génétique a un support : l'ADN, molécule présente dans toutes les cellules vivantes. C'est cette molécule qui permet à un organisme de se développer et de fonctionner.

Les gènes sont des séquences d'ADN, qui peuvent être vues comme des unités d'information génétique. Ils sont eux-mêmes composés d'une suite de nucléotides, dont il existe quatre types dans l'ADN : A pour Adénine, G pour Guanine, T pour Thymine et C pour Cytosine.

Le phénotype est l'ensemble des caractères observables chez un individu à différentes échelles. Il s'agit de l'expression des gènes. Enfin, la génomique est l'étude du génome. Elle a pour objectif de mieux connaître le lien entre le génome et le phénotype afin notamment de favoriser la compréhension du fonctionnement des organismes ou de certaines maladies génétiques.

Depuis la fin du XXème siècle, d'importants progrès ont été réalisés pour les techniques de séquençage du génome, c'est-à-dire l'extraction de l'information génétique d'un individu, et aujourd'hui, le génome humain complet de plusieurs individus a déjà été réalisé. Ces techniques continuent à évoluer rapidement, et le rythme du séquençage va continuer à augmenter dans les prochaines années (Altshuler et al., 2010).

Les données résultantes de ces séquençages sont volumineuses. En effet, pour le génome humain, on compte près de 26 000 gènes, ce qui correspond à environ 3 milliards de paires de bases (Initial sequencing and analysis of the human genome, 2001). Une problématique apparait donc pour le stockage et le traitement de ces données, et le Big Data est une solution possible pour répondre à cette problématique.

1.2 La problématique étudiée

Chaque individu d'une espèce est unique puisqu'il possède des caractéristiques qui lui sont propres. Il possède pour cela un ensemble d'informations génétiques différentes des autres. La plupart des gènes sont identiques entre les individus d'une même espèce, il s'agit de la base commune. Au contraire, certains gènes vont varier, et auront une combinaison de paires de bases différente. Ce sont ces variations génétiques qui permettent la diversification d'une espèce.

Le nombre de variations génétiques d'un individu se chiffre en millions, et l'une des applications de la génomique est l'étude de ces variations. Pour cela, il est nécessaire d'effectuer de séquençage de plusieurs individus, et d'étudier les différences et les similitudes entre leurs génomes.

Les objectifs du projet présenté ici ont été définis à partir d'une réelle problématique pour l'analyse des variations génétiques de plusieurs individus. Les systèmes informatiques qui doivent permettre de faire cette analyse doivent être adaptés pour le stockage d'un très grand volume de données. Le laboratoire de recherche Guy Rouleau effectue ce type de recherche, et utilise pour cela un système basé sur une base de données relationnelle, mais il n'est pas adapté au traitement de certaines requêtes. Les données sont trop volumineuses et les temps de traitement sont trop longs. C'est à partir de ce problème qu'une solution Big Data a été envisagée.

Ce projet a connu plusieurs itérations puisqu'il a démarré avec des étudiants au baccalauréat au cours de l'année 2012. Les avantages d'une solution Big Data pour analyser ce type de données ont déjà été soulevés. C'est donc à partir des résultats obtenus qu'une nouvelle itération de recherche a commencé, avec de nouveaux objectifs qui visent notamment à proposer une solution plus complète.

Le prochain chapitre va dans un premier temps présenter les technologies de Big Data en rapport avec le projet, puis la problématique étudiée sera présentée plus en détail, et les nouveaux objectifs du projet seront donnés.

CHAPITRE 2

Présentation du sujet et revue de littérature

2.1 Les technologies existantes

Aujourd'hui, il existe plusieurs bases de données de type Big Data qui répondent souvent à des besoins différents. Tel que mentionné précédemment, BigTable est l'une de ces bases de données. Elle repose sur une architecture distribuée à l'aide du système de fichier GFS. Loin du modèle relationnel utilisé par les bases de données traditionnelles, elle utilise un nombre de tables réduit, dont les informations sont principalement accessibles à l'aide d'une clé. Les données sont triées à l'aide de cette clé, ce qui permet, si la structure de la clé a été choisie avec soin, de les parcourir de manière séquentielle. Les caractéristiques de BigTable lui permettent de s'adapter à de très grands volumes de données tout en conservant un niveau de performance acceptable. Utilisée dans un premier temps par l'indexation web, cette base de données est maintenant utilisée par de nombreux logiciels et services de Google, dont Google Earth et Google Finance (Chang et al., 2008).

Parmi les projets Big Data soutenus par Apache, on retrouve notamment HBase, qui propose un fonctionnement similaire à BigTable. C'est une base de données « open source » développée en Java, et qui a maintenant atteint une maturité suffisante pour être utilisée pour de nombreux projets, dont par exemple le système de messagerie de Facebook (Borthakur et al., 2011), ou bien encore LastFM (Lu, Hai-Shan et Ting-Ting, 2012). Cassandra est un projet initialement développé par Facebook avant l'adoption de HBase. Depuis, ce projet a été repris par Apache. Il fonctionne d'une manière similaire à BigTable, et s'en inspire sur plusieurs points (Lakshman et Malik, 2010).

Il existe des bases de données de type Big Data « orientées documents », c'est-à-dire qu'elles permettent de gérer une grande quantité de documents par opposition aux données structurées composées de types simples tels que des entiers et les chaînes de caractères. On retrouve par exemple dans cette catégorie les bases de données Apache CouchDB ou MongoDB.

De manière générale, le domaine du Big Data est encore très jeune et en pleine évolution, et la plupart de ces projets sont aujourd'hui en développement actif. Ce manque de maturité pose plusieurs problèmes. Tout d'abord, l'évolution rapide des fonctionnalités rend difficile le support et la compatibilité à long terme. Ensuite, il n'y a pas de standard permettant d'utiliser l'une ou l'autre de ces bases de données, et encore moins de langages tels que le SQL permettant de profiter des performances tout en faisant abstraction du système utilisé. On notera toutefois certains projets récents qui visent à ajouter cette couche d'abstraction, comme Impala et son moteur SQL. Cependant, ces projets ne fonctionnent pas tous de la même manière, et leur utilisation n'est pas encore très répandue.

2.2 Présentation de Hadoop et HBase

Avant de présenter plus en détail la base de données HBase, il faut comprendre le fonctionnement de Hadoop sur lequel elle repose. Hadoop est un ensemble de logiciels et d'outils qui permettent de créer des applications distribuées. Il repose principalement sur 2 composants : HDFS et MapReduce, présentés ci-dessous.

2.2.1 HDFS

HDFS (Hadoop Distributed File System) est un système de fichiers distribué, inspiré du système GFS développé par Google. Il est composé d'un composant maître, appelé « NameNode », et d'un ensemble de nœuds appelés « DataNode ». Le composant « NameNode » a plusieurs rôles. Il contient tout d'abord l'arborescence des fichiers présents sur le système de fichier. Il possède également la liste des « DataNode » connectés, et gère la répartition des données. Les « DataNode » contiennent les données sous forme de blocs occupant le même espace, typiquement 64 Mo. Cela facilite la répartition et la synchronisation des données dans le cluster.

Afin d'améliorer la fiabilité du système de fichiers, un « NameNode » secondaire est généralement utilisé afin d'avoir une alternative en cas de dysfonctionnement du premier « NameNode ». De plus, une réplication des blocs de données sur plusieurs « DataNode » permet d'éviter une perte de données en cas d'arrêt de l'un des nœuds. C'est au « NameNode » d'assurer un taux de réplication suffisant en contrôlant l'état de tous les nœuds et des données qu'ils contiennent. La figure ci-dessous présente le fonctionnement général de HDFS.

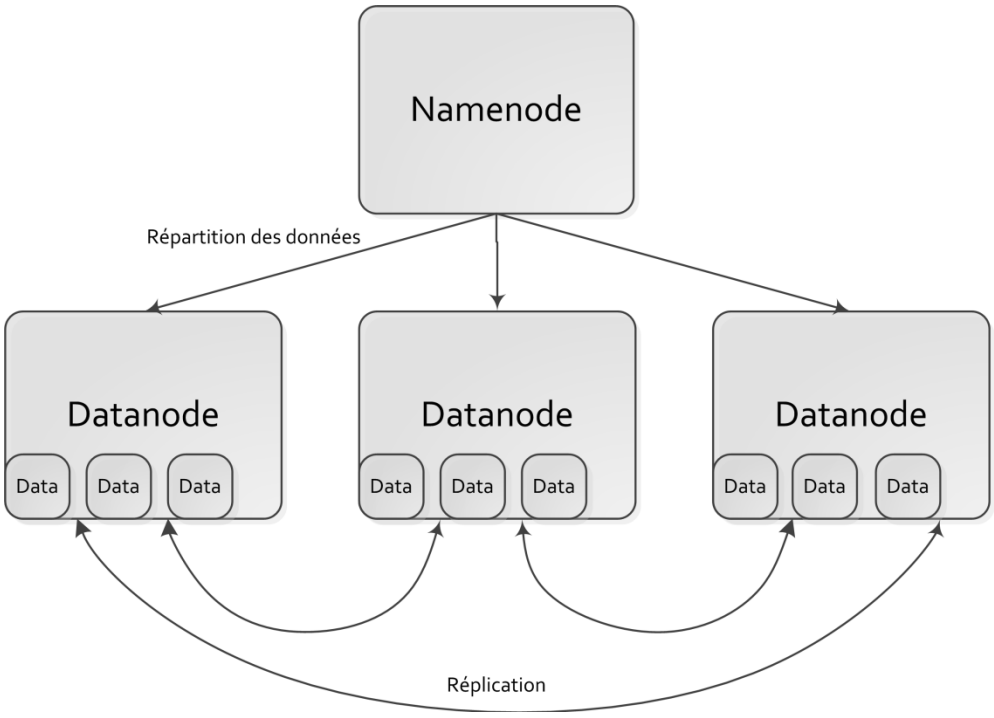


Figure 2.1 Architecture de HDFS

2.2.2 MapReduce

MapReduce est une technique qui permet d'effectuer des tâches sur de grandes quantités de données en utilisant un cluster de machines. Elle est implémentée au cœur du projet Hadoop. Tout comme HDFS, on retrouve deux types de composant. Le premier est appelé « JobTracker », et doit connaître l'état du cluster et les différents nœuds actifs. Il devra également recevoir les nouvelles tâches à effectuer afin de les distribuer aux autres nœuds.

Les nœuds appelés « TaskTracker » vont quant à eux effectuer les tâches données par le « JobTracker ».

Les tâches MapReduce sont exécutées à partir d'une source de données en entrée telle que des fichiers, et produisent une sortie. Elles sont réalisées en deux étapes. La première étape, appelée « map », consiste à découper les données en entrées afin de les envoyer sur différents nœuds enfants, puis d'associer à une clé une ou plusieurs valeurs à partir de ces données. Le résultat est ensuite envoyé au nœud parent. La deuxième étape, « reduce », est exécutée sur le nœud parent, et consiste à associer à regrouper les données reçues par clé, avant de retourner encore une fois le résultat.

Le schéma suivant présente un exemple d'une application utilisant MapReduce servant à calculer des moyennes pour des valeurs associées à des articles.

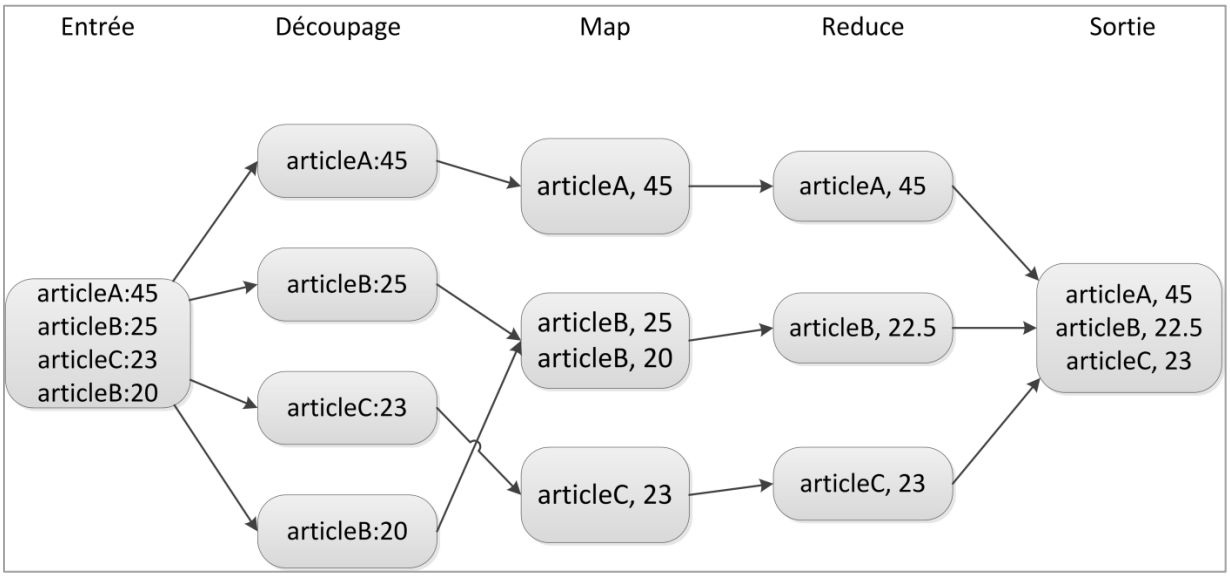


Figure 2.2 Exemple d'application MapReduce

En entrée, des valeurs sont associées à des articles. Dans un premier temps, le fichier est découpé ligne par ligne. L'étape « map » va alors associer chaque article, représentant une

clé, à une valeur, puis regrouper les mêmes articles ensemble. Enfin, l'étape « reduce » va calculer la moyenne des valeurs pour chaque article.

2.2.3 HBase

Tel que mentionné auparavant, HBase est une base de données Big Data utilisant le modèle de la base de données BigTable de Google. Elle a donc un fonctionnement distribué, et n'est pas relationnelle. C'est un projet libre soutenu par la fondation Apache et développé avec le langage Java.

Tout comme les composants HDFS et MapReduce, on retrouve deux types de composants pour HBase : le composant maître appelé « HMaster », qui contient le schéma des tables de données, et les nœuds esclaves appelés « Region Server », qui gèrent des sous-ensembles de tables, appelés « Region ».

HBase est habituellement utilisé sur un ensemble de machines, ou cluster, et utilise pour cela le système de fichier HDFS. Les nœuds de type « Region Server » fonctionnent alors sur des nœuds « DataNode » afin de stocker les données des tables. Cette architecture permet de profiter de l'espace et de la fiabilité de HDFS, tout en apportant une couche supplémentaire nécessaire pour proposer les fonctionnalités d'une base de données.

Les tables avec HBase sont différentes des tables généralement utilisées avec les bases de données relationnelles. Tout d'abord, il existe des familles de colonnes. Ces familles sont en général définies lors de la création de la table. Elles permettent de regrouper les colonnes par catégorie. Le nombre de colonnes par famille n'est lui pas limité, et on peut créer pour une même ligne des millions de colonnes. Le nom de chaque colonne est défini lors de l'enregistrement d'une cellule, et non dans le schéma de la table.

Pour accéder à une ligne, on doit utiliser une clé. Cette clé a une grande importance pour le fonctionnement de HBase, puisque les données sont triées dans les tables en fonction de

celle-ci. Cela permet de faciliter le parcours de lignes qui ont des clés similaires. Ainsi, il est important de choisir un bon modèle pour construire la clé afin d'optimiser les performances de l'application. La construction d'une clé sera détaillée dans le prochain chapitre.

Cette section n'aborde pas en détail de fonctionnement technique de la base de données HBase, mais la Figure 2.3 présente les grandes lignes de son architecture.

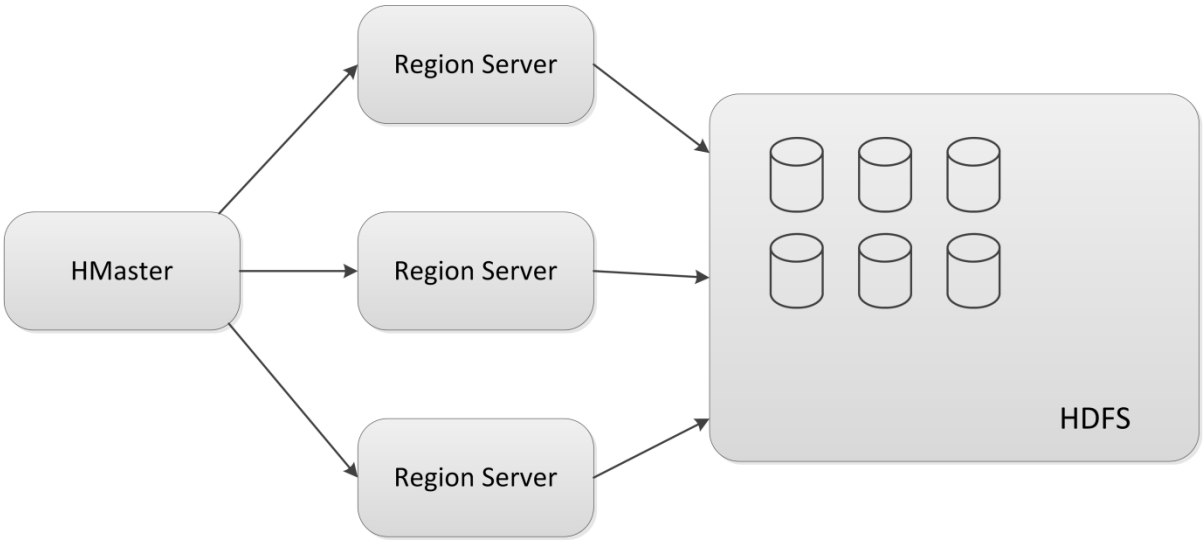


Figure 2.3 Fonctionnement général de HBase

2.3 Les projets Big Data en génomique

Le Big Data est un domaine particulièrement bien adapté pour traiter de très grands volumes de données en génomique. Depuis ces dernières années, plusieurs projets ont permis de démontrer les avantages de cette combinaison. On retrouve par exemple Cloudburst, un logiciel utilisant Hadoop, et notamment son implémentation de MapReduce, qui permet de faire le lien entre un génome de référence et des nouvelles séquences pour découvrir des variations (Taylor, 2010). SeqWare Engine est un autre projet qui vise à fournir une base de données pour l'analyse des variations génétiques à travers le Framework SeqWare. La base de données HBase a été proposée en tant que « backend » pour améliorer les temps de traitement (O'Connor, Merriman et Nelson, 2010).

Enfin, le projet présenté ici reprend les travaux de plusieurs étudiants qui ont travaillé avec le laboratoire du Dr. Guy Rouleau. Ces travaux ont consisté à la mise en place d'une solution performante pour effectuer des requêtes vers une base de données volumineuse. Contrairement aux programmes exploitant MapReduce, qui visent principalement à lancer de longues tâches, l'objectif ici est de pouvoir traiter ces requêtes très rapidement, afin de proposer ensuite une interface web simple d'utilisation. Cette problématique et les travaux entrepris sont expliqués plus en détail dans la section suivante.

2.4 La problématique du projet

2.4.1 Introduction

Les données constituent l'élément essentiel de ce projet d'application. Elles sont au cœur de la problématique. Avant de développer des méthodes et des procédures pour l'exploitation de ces données, il faut bien les comprendre. Il faut pour cela répondre à plusieurs points essentiels pour la poursuite du projet : leur définition tout d'abord, puis la problématique actuelle face à ces données.

La définition des données va dans un premier temps permettre de déterminer précisément leur nature, leur structure et leurs propriétés. Une définition précise permettra de comprendre ce que l'on attend de leur utilisation au sein d'une application informatique. La problématique des données est la raison d'être du projet, son point de départ, il faut donc la définir précisément avant de commencer à chercher une solution.

Dans la prochaine section, les données utilisées par le laboratoire seront présentées. Ensuite, nous verrons quelle est la problématique de ces données.

2.4.2 Présentation des données du laboratoire

Comme mentionné dans le chapitre précédent, le laboratoire Guy Rouleau utilise des données liées aux variations du génome. Celles-ci sont stockées dans une base de données relationnelle. Elles sont utilisées par une application permettant d'effectuer des recherches sur les informations des variations génomiques présentes en fonction de plusieurs critères.

Parmi les données stockées, on retrouve les variations génétiques avec une liste de propriétés associées. Il s'agit par exemple du génotype correspondant à la variation (génotype de référence et génotype de la variation), la fréquence de la variation, etc. On retrouve également une liste d'échantillons séquencés, et un lien entre ces échantillons et les variations détectées. Enfin, les outils et techniques utilisés pour faire le séquençage et la détection des variations sont également enregistrés dans la base de données.

Afin de donner une définition plus détaillée de ces données, la Figure 2.4 présente un schéma simplifié inspiré du schéma de la base de données du laboratoire. Le véritable schéma est sujet à des changements au cours de la réalisation de ce projet, et ne sera donc pas présenté ici.

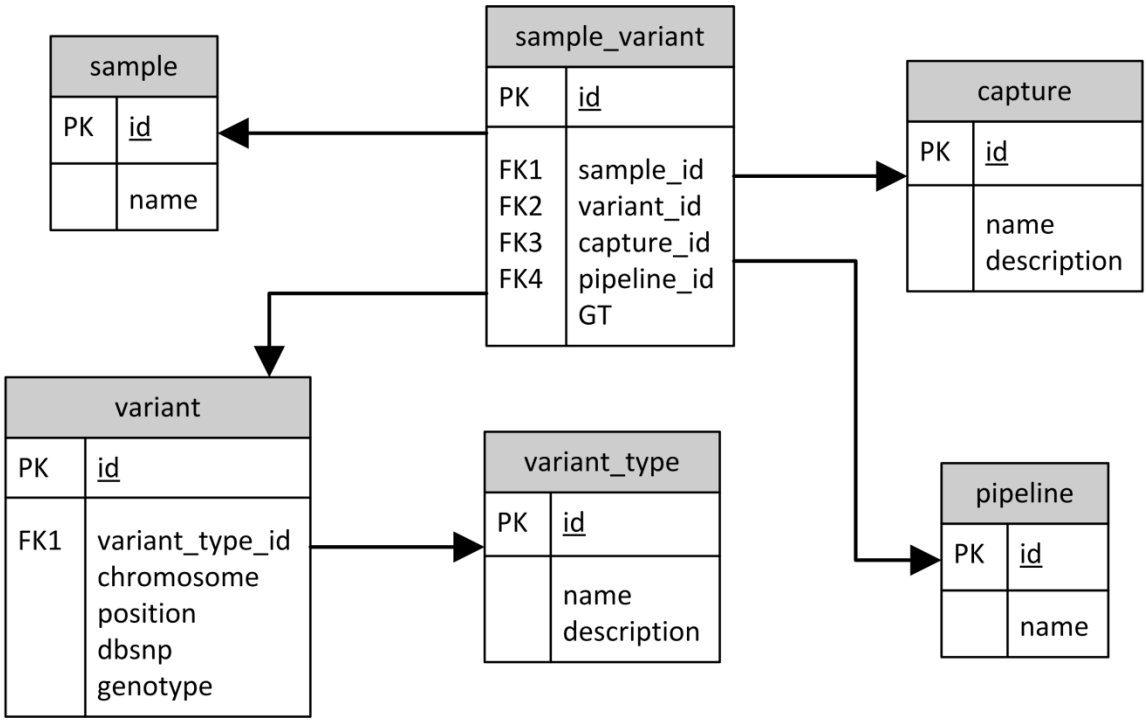


Figure 2.4 Schéma relationnel simplifié de la base de données

Voici le descriptif des différentes tables :

- variant : il s'agit de la table qui contient les informations sur les variations génétiques, notamment le génotype de référence (exemple : A, T, TAGCC…) et le génotype de la variation, la position de la variation au sein du génome de référence, la fréquence de la variation, etc.
- variant_type : cette table contient les différents types de variations. Pour le projet, trois types de variations sont enregistrés : SNP (Single-Nucléotide Polymorphism), variation la plus fréquente d'une seule paire de bases, INDEL (Insertion-Deleletion), pour l'ajout ou la suppression d'information génétique, et SV (Structural Variation), variation plus complexe que les précédentes et qui concerne un grand nombre de paires de bases.
- sample : cette table contient des informations sur la source de l'échantillon.
- sample_variant : cette table contient l'information sur la détection de la variation pour un échantillon. C'est cette table qui contient le plus d'enregistrements, typiquement plusieurs millions par individu.

- capture : informations sur les outils et techniques employés pour la détection.
- pipeline : informations sur les outils et techniques utilisés pour le séquençage de l'échantillon.

2.4.3 La problématique du laboratoire Rouleau

L'utilisation d'une base de données relationnelle pour stocker des données présente des avantages. Ce type de base de données est très utilisé aujourd'hui, ce qui facilite le travail de conception ainsi que l'accès aux ressources expertes du domaine. Les requêtes peuvent être facilement personnalisées afin de répondre à tout type de demande de la part des chercheurs.

Cependant, certaines requêtes peuvent être longues à traiter, notamment lorsqu'une grande quantité de données est concernée, ou bien lorsque de nombreuses jointures sont nécessaires. C'est le problème qui est survenu avec la base de données utilisée par le laboratoire pour au moins un cas d'utilisation.

Le cas d'utilisation problématique est la sélection d'un ensemble de variations en fonction de plusieurs paramètres. En effet, il est possible de sélectionner un ou plusieurs échantillons à partir desquels la recherche va avoir lieu. Seules les variations détectées avec les échantillons sélectionnés vont être retournées. Il est également possible de spécifier une condition sur ces échantillons afin d'obtenir les variations présentes avec tous les échantillons de la liste. Enfin il est possible d'exclure des échantillons. Cette option permet d'exclure toute variation présente avec l'un des échantillons exclus. Ces deux paramètres permettent donc de contrôler précisément les variations obtenues en fonction des individus. Les options de recherches incluent également la plage de position de la recherche. Il est possible de spécifier le chromosome et les positions de départ et de fin.

Lorsqu'on veut effectuer ce type de requête avec une grande quantité de données, les performances sont dégradées (Klos, 2012). C'est là le problème qui a conduit au principal objectif de ce projet de recherche : l'amélioration des performances dans le cas de

l'utilisation de ces requêtes problématiques, en expérimentant notamment une solution fondée sur les technologies de type Big Data.

2.5 Les travaux antérieurs effectués

Ce projet a été étudié par plusieurs étudiants de l'ÉTS dans le cadre de projets de fin d'études : Anna Klos tout d'abord, puis Jean-François Hamelin et Jean-Philippe Bond. Ainsi, on peut décrire plusieurs itérations d'études et des pistes de solution pour tenter de résoudre ce genre de problème. C'est à partir de ces éléments que l'on pourra ensuite établir des objectifs qui ont évolué depuis le début de la recherche.

2.5.1 Première itération : schéma et importation de données

Lors de la première itération de ce projet, début 2012, le cas problématique a été défini précisément et analysé. Ainsi, plusieurs requêtes nécessaires pour la réalisation de ce cas ont été identifiées. Tout d'abord, la liste des variations à exclure est obtenue à partir des échantillons exclus avec la requête suivante :

```
SELECT DISTINCT variant_id
FROM ngs_sample_variant
WHERE sample_id in ( :SAMPLE_IDS_À_EXCLURE )
ORDER BY variant_id
```

De manière similaire, les variations à inclure sont récupérées à partir de la liste des échantillons à inclure, avec éventuellement une condition supplémentaire pour inclure un type de « pipeline » en particulier. Les variations exclues sont retirées de ce deuxième résultat, puis les variations en elles-mêmes et les informations associées sont récupérées avec la requête suivante :

```
SELECT v.id, v.variant_class, v.strand, v.variant_genotype,
     v.dbSNP_rs, v.variant_type_id as s2d_type, v.gene,
v.freq_1000g,
     v.sift_score, qvp.chromosome_name as chromosome,
     qvp.hg18_position as hg18_chrom_position, qvp.hg19_position as
     hg19_chrom_position
FROM NGS_variant v, NGS_queryVariantPosition qvp
WHERE v.position_id = qvq.position_id
AND v.id = in ( @resultSet )
AND v.variant_type_id = 4
GROUP BY v.id, v.variant_class, v.strand, v.variant_genotype,
     v.dbSNP_rs, v.variant_type_id, v.gene, v.freq_1000g,
     v.sift_score, qvp.chromosome_name, qvp.hg18_,
qvp.hg19_position
```

Ici, la requête tient également compte d'un type de variation en particulier, correspondant à l'ID 4. Il est également possible de lancer une requête avec un autre type de variation.

Toutes ces requêtes sont nécessaires pour obtenir les résultats voulus. Elles imposent l'utilisation de plusieurs jointures, mais c'est surtout la quantité de données à traiter qui pose un problème de performance, puisque le nombre de variations à sélectionner avant de pouvoir les trier peut être de plusieurs millions. L'application du laboratoire est proposée sous la forme d'une interface web qui vise plutôt des traitements rapides, et ces problèmes de performance ne permettaient donc pas de l'utiliser de la manière voulue.

L'objectif de cette première itération de recherche était alors de concevoir un nouveau schéma de données avec la base de données HBase de manière à ce que ces requêtes puissent être traitées plus rapidement, puis de migrer les données vers cette nouvelle base de données. Une nouvelle problématique est alors apparue avec la migration des données actuelles provenant d'une base de données relationnelle vers cette nouvelle base de données HBase. Pour effectuer cette migration, l'utilitaire Sqoop a été sélectionné et utilisé. Cet outil a été

spécialement conçu pour copier des données d'un emplacement à un autre avec Hadoop. Il permet d'effectuer des migrations à partir d'une simple ligne de commande. Concrètement, une vue logique a été créé dans la base de données source afin d'accéder directement à toutes les informations utiles pendant l'importation (Klos, 2012). La table avec la base de données HBase est créée automatiquement par Sqoop.

Au terme de cette première étape, la procédure de migration des données vers la base de données HBase était fonctionnelle, et des premiers tests de performance ont été réalisés afin de comparer l'ancien système et le nouveau.

2.5.2 Deuxième itération : améliorations et application web

Pour la deuxième itération de recherche, tenue lors de l'été 2012, les requêtes problématiques ont été revues, et le schéma de la base de données HBase a été amélioré. Ainsi, plus de cas d'utilisation ont été pris en compte. Lors de cette itération, la base de données relationnelle était toujours utilisée afin de migrer les données concernant les variations.

Une interface web a été développée afin de démontrer la performance atteinte de cette nouvelle approche, en effectuant les requêtes à partir de la technologie HBase. Aussi, un petit cluster, composé de 3 machines virtuelles, mis en place dans le laboratoire de recherche GÉLOG de l'ÉTS a été configuré avec Hadoop et HBase. Cependant, l'application web n'a pas été installée sur ce cluster faute de temps, et les tests ont été effectués avec une installation de Hadoop et HBase en local (Hamelin et Bond, 2012).

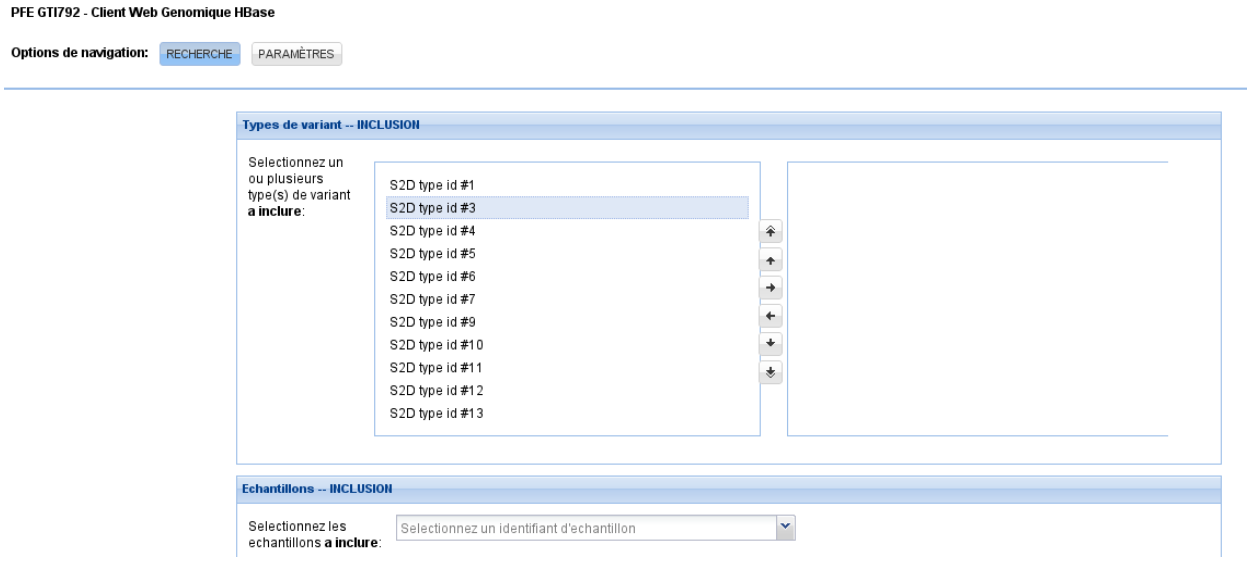
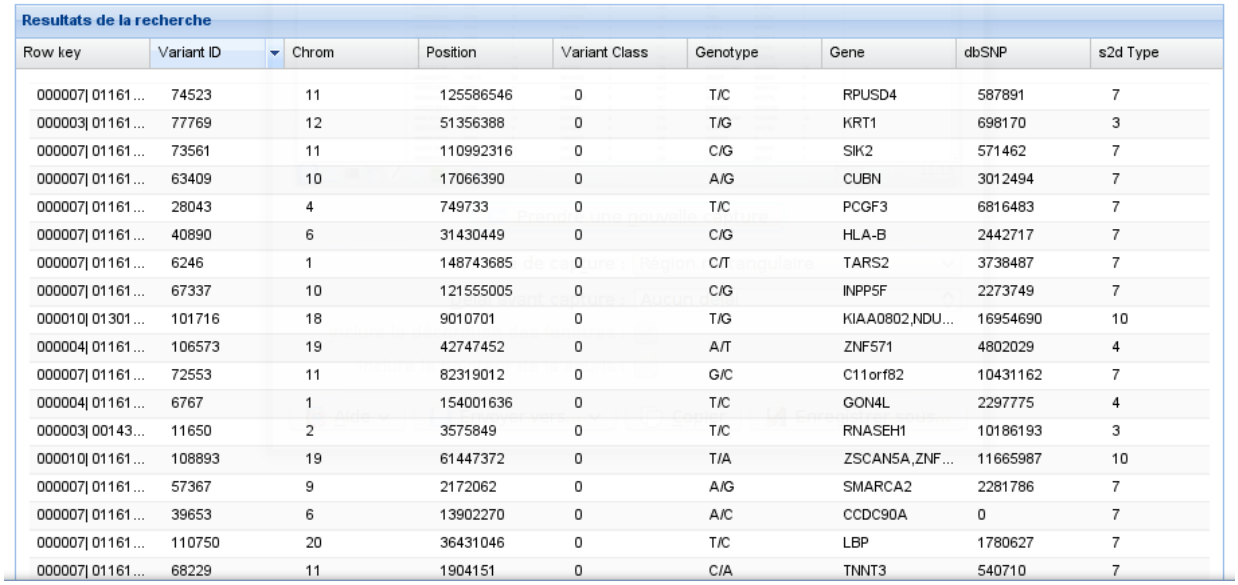


Figure 2.5 Formulaire de recherche de la première application web



Row key	Variant ID	Chrom	Position	Variant Class	Genotype	Gene	dbSNP	s2d Type
000007| 01161...	74523	11	125586546	0	T/C	RPUSD4	587891	7
000003| 01161...	77769	12	51356388	0	T/G	KRT1	698170	3
000007| 01161...	73561	11	110992316	0	C/G	SIK2	571462	7
000007| 01161...	63409	10	17066390	0	A/G	CUBN	3012494	7
000007| 01161...	28043	4	749733	0	T/C	PCGF3	6816483	7
000007| 01161...	40890	6	31430449	0	C/G	HLA-B	2442717	7
000007| 01161...	6246	1	148743685	0	C/T	TARS2	3738487	7
000007| 01161...	67337	10	121555005	0	C/G	INPP5F	2273749	7
000010| 01301...	101716	18	9010701	0	T/G	KIAA0802,NDU...	16954690	10
000004| 01161...	106573	19	42747452	0	A/T	ZNF571	4802029	4
000007| 01161...	72553	11	82319012	0	G/C	C11orf82	10431162	7
000004| 01161...	6767	1	154001636	0	T/C	GON4L	2297775	4
000003| 00143...	11650	2	3575849	0	T/C	RNASEH1	10186193	3
000010| 01161...	108893	19	61447372	0	T/A	ZSCAN5A,ZNF...	11665987	10
000007| 01161...	57367	9	2172062	0	A/G	SMARCA2	2281786	7
000007| 01161...	39653	6	13902270	0	A/C	CCDC90A	0	7
000007| 01161...	110750	20	36431046	0	T/C	LBP	1780627	7
000007| 01161...	68229	11	1904151	0	C/A	TNNT3	540710	7

Figure 2.6 Tableau de résultats de la première l'application web

2.6 Objectifs du projet

À partir des observations et publications des travaux déjà effectués, la troisième itération de recherche a démarré à l'automne 2012 avec des objectifs redéfinis.

2.6.1 Schéma de la base de données HBase

La première étape de cette itération de recherche vise à revoir le schéma de la base de données HBase proposé initialement par Jean-François Hamelin et Jean-Philippe Bond. La conception de celui-ci a beaucoup d'influence sur les performances, qui est le principal problème étudié. Afin de définir un schéma HBase plus performant, il faut effectuer plusieurs étapes de conception. Dans un premier temps, les données et leur type doivent être correctement identifiés. Ensuite, les cas d'utilisations doivent être précisément définis afin de savoir comment utiliser spécifiquement ces données. Enfin, il peut être nécessaire d'évaluer plusieurs alternatives en effectuant des expérimentations.

2.6.2 Nouvelle API

Une API est une couche logicielle permettant de faire le lien entre une application et un système. Dans le cadre de ce projet, l'API a pour objectif de fournir des fonctions permettant de récupérer des informations dans la base de données HBase. La solution proposée précédemment ne mettait pas en avant ce type de composant, mais plutôt une application complète embarquant toutes les fonctionnalités nécessaires. De plus, cette application logicielle permettait de tirer profit de certains cas d'utilisation et d'un schéma HBase qui ont été revus. Cette itération de recherche inclura donc la conception d'une API complète qui permettra de tirer profit des modifications apportées sur les données et le schéma de la base de données, et d'apporter une séparation nette entre celle-ci et le reste de l'application.

2.6.3 Nouvelle application web

Une application web présente une interface facile à utiliser pour exploiter le système mis en place. Lors de la précédente itération de ce projet, une application expérimentale a déjà été développée. Les changements qui doivent intervenir dans l'API et les cas d'utilisation revus

ont amené au développement d'une nouvelle application web tirant profit de ces changements.

Cette application sera développée avec la technologie Java EE afin de pouvoir profiter de nativement de l'API développée en Java. Elle sera fortement inspirée de la précédente interface. En marge d'une interface utilisateur fonctionnelle, il sera possible de récupérer les données sous un format XML afin que l'application puisse être utilisée par des applications tierces.

2.6.4 Mise en service de la solution

La mise en service de la solution proposée est son installation au sein d'un cluster. Il peut aussi bien s'agir d'un cluster dédié, installé et configuré pour cet usage, que d'un cluster virtuel exécuté à l'aide du cloud computing.

Le cloud computing est une pratique de plus en plus utilisée en informatique. Elle connait un développement très rapide depuis plusieurs années, et représente l'un des plus grands enjeux pour les entreprises pour la prochaine décennie.

Le principe est de dématérialiser l'informatique, afin de ne plus utiliser une infrastructure locale, des serveurs physiques ou des applications classiques. Pour cela, ces éléments sont amenés dans des centres de données spécialisés et accessibles à distance.

Le cloud computing est potentiellement adapté aux contraintes matérielles et logicielles de ce projet. En effet, il est question ici d'utiliser un grand nombre de machines pour réaliser des tests et valider l'implémentation d'une solution. Alors que la mise en place d'un cluster réel peut être longue et couteuse, l'utilisation de services de cloud computing ne nécessite qu'un minimum de préparation et leurs coûts correspondent à leur utilisation effective.

Les services de cloud computing existants seront donc étudiés afin de réaliser une installation de la solution sur l'un d'entre eux. Aussi, la procédure adaptée à cette technologie sera décrite en détail.

2.6.5 Tests de performance

Les tests de performances visent à valider les choix technologiques en démontrant que les performances ont atteint un niveau correct. Pour ce projet, il s'agit essentiellement de présenter les résultats d'un ensemble de requêtes exécutées sur le système de test. Les problématiques à résoudre pour cet objectif concernent principalement la mise en place d'un cadre pour ces tests et de définir les métriques. Enfin, il faudra interpréter les résultats afin de définir si la solution mise en place est satisfaisante.

CHAPITRE 3

Développement et travail effectué

3.1 Source des données

3.1.1 Introduction

Nous avons vu dans le chapitre précédent la nature des données génomiques qui seront utilisées pour ce projet. Nous allons maintenant nous intéresser à la source utilisée pour récupérer ces données. La source des données est le point d'ancrage du travail au niveau technique. Il s'agit du format que les méthodes et procédures vont utiliser pour le passage vers un nouveau système de type Big Data.

La source des données utilisée lors des travaux précédents sera présentée avant d'aborder les modifications apportées pour les nouveaux travaux.

3.1.2 Données utilisées lors des premiers travaux

Lors des précédents travaux sur ce projet, les données étaient fournies par le laboratoire Guy Rouleau sous la forme de scripts SQL. Ces scripts pouvaient être exécutés afin de peupler une base de données relationnelle.

Cette méthode a permis de répliquer l'environnement de départ avec une base de données similaire, et de faire une comparaison de performance entre le système original et le nouveau système.

Cependant, cette méthode présentait plusieurs inconvénients. Tout d'abord, le schéma utilisé par le laboratoire a été sujet à des changements au cours du projet, ce qui ne permet pas de réaliser des tests en fonction du système actuel, mais seulement d'un système antérieur.

Ensuite, les données fournies ne représentaient pas toutes les données utilisées réellement, mais seulement une partie. Enfin, les scripts SQL fournis étaient adaptés à une base de données en particulier, et pouvaient nécessiter quelques ajustements.

Ces problèmes pouvaient être en partie réglés en effectuant les corrections nécessaires. Deux solutions étaient donc envisageables : les corrections pouvaient être effectuées afin d'améliorer l'étape de construction de la base de données relationnelle comme source de données, ou bien une nouvelle source de données pouvait être utilisée.

Afin de se libérer des contraintes présentées, et de réduire les intermédiaires, la seconde solution a été retenue.

3.1.3 Données utilisées pour le projet

Les variations génomiques sont des données disponibles à tous. Plusieurs projets ont également rendu publics les résultats du séquençage de plusieurs individus. C'est le cas du projet 1000 Genomes, qui vise à récupérer le séquençage de plus d'un millier d'individus provenant de régions distinctes puis de distribuer les résultats obtenus à partir de ces séquençages. Parmi ces données, on retrouve le génome en lui-même, ainsi que les informations sur les variations présentes pour chaque individu. Le volume que représentent ces données est relativement important (environ 140 Go de données compressées pour les archives datant du 21 mai 2011), et ce sont donc elles qui ont été la source pour ce projet.

Ces données sont disponibles sous le format VCF (Variant Call Format), qui a été proposé par le projet 1000 Genomes pour l'enregistrement et la distribution de données de variations. Avec ce format, chaque variation identifiée occupe une ligne du fichier. On retrouve pour chacune d'entre elles des informations sur leur type, les paires de bases concernées, leurs fréquences… Enfin, pour chaque échantillon présent, on retrouve une information sur la présence ou non de la variation. Voici un exemple d'enregistrement qui peut être présent dans un fichier VCF :

```
#CHROM  POS     ID      REF     ALT     QUAL    FILTER  INFO                        FORMAT
20      14370           G       A       29      PASS    NS=3;DP=14;AF=0.5;DB;H2     GT:GQ:DP:HQ
```

La première ligne indique le type des différentes colonnes, séparées par des tabulations. La deuxième ligne contient un enregistrement pour une variation. Ici, il n'y a pas de données qui lient cette variation à des échantillons. L'ANNEXE I présente le format VCF plus en détail.

Les fichiers VCF créés par le projet 1000 Genome sont notamment disponibles sur les serveurs de l'EBI (European Bioinformatics Institute), du NCBI (National Center for Biotechnology Information), ainsi qu'à partir de divers services tels qu'Amazon S3.

Utiliser comme source des données textuelles brutes présente l'avantage de s'affranchir d'un système intermédiaire, ici une base de données relationnelle, pour la mise en place d'une solution. Pour la suite du projet, c'est cette source seulement qui sera utilisée.

Il existe cependant certains cas qui requièrent une autre source de données. Par exemple, la migration d'une base de données enrichie d'informations supplémentaires vers le nouveau système impose de travailler avec celle-ci. Dans ce cas-là, la procédure développée au cours des travaux précédents devra être reprise et si besoin améliorée afin de satisfaire cette contrainte.

3.2 Cas d'utilisation

Après avoir déterminé la source des données, il faut revoir les cas d'utilisation qui permettront de les exploiter, notamment avec des scénarios qui posaient problème avec le système précédent. La définition précise de ces cas d'utilisation permettra ensuite de modéliser le système final en répondant à des objectifs précis.

Le nouveau format pour les variations utilisé dans le projet fournit le même type de données que précédemment. Cependant, il existe quelques différences entre ce format et le format précédent :

- Les types de variations ne sont plus identifiés de la même manière. Avec les nouveaux fichiers de données, les types de variations sont SNP, INDEL et SV.
- L'information sur le type de « pipeline » utilisé n'est plus disponible. Le pipeline indiquait la combinaison des outils et techniques utilisés pour obtenir le séquençage d'un échantillon.
- Les variations non présentes dans les échantillons sont incluses.

Ces différences entrainent une revue des cas d'utilisation afin de s'adapter au nouveau format. Le Tableau 3.1 présente le premier cas d'utilisation identifié. Celui-ci est proche du cas identifié auparavant, et sera principalement étudié lors de la mise en place d'un schéma pour HBase. C'est également le cas le plus complet au niveau des attributs de recherche. De plus, le

Tableau 3.2 présente des cas d'utilisation secondaires qui sont dérivés du premier cas d'utilisation ou bien complémentaires. Ces cas sont mentionnés afin d'introduire les fonctionnalités qui seront disponibles avec l'application finale.

Tableau 3.1 Principal cas d'utilisation

Paramètres de recherche	**Informations retournées**
• Une variation ou une plage de positions • Un ou plusieurs échantillon(s) inclus(s) • Un ou plusieurs échantillon(s) exclu(s) • Une liste de types de variations	Une liste de variations avec, pour chacune d'entre elles, des informations sur sa nature et ses caractéristiques ainsi qu'une liste d'échantillons pour lesquels la variation est détectée.

Tableau 3.2 Cas d'utilisations secondaires

Paramètres de recherche	**Informations retournées**
• Une variation • Un ou plusieurs échantillon(s) inclus(s) • Un ou plusieurs échantillon(s) exclu(s)	Les caractéristiques de la variation, ainsi que la liste des résultats obtenus lors du séquençage avec les échantillons pour lesquels la variation a été détectée.
• Une plage de positions de variations • Une liste de types de variations • Un échantillon • Une condition d'inclusion des résultats négatifs dans les résultats	Une liste de résultats sur la détection des variations au sein de l'échantillon sélectionné. Il est possible de n'inclure que les résultats positifs, ou bien les résultats positifs et négatifs.
• Une variation	Les informations pour la variation indiquée.

3.3 Nouveau schéma de données

3.3.1 Introduction

Le schéma utilisé pour stocker les données avec HBase est l'un des points clés de ce projet. Avec un schéma classique décrivant un modèle relationnel pour le stockage de données, les relations entre les tables est l'un des points les plus importants à prendre en compte. Il faut également définir un type pour chaque colonne, des index, des clés primaires…

Avec HBase, la structure d'un schéma est complètement modifiée. Il n'est plus nécessaire de définir les relations entre les données ni même le type des colonnes, seules les chaînes d'octets étant prises en charge. Il faut définir une structure de table permettant de stocker un très grand volume de données et un accès rapide à celles-ci. Le nombre de tables est réduit, et l'usage d'une table unique permet de répondre à de nombreux cas d'utilisations.

L'élément le plus important d'un schéma HBase est la clé à utiliser. En effet, tous les enregistrements sont triés par ordre alphabétique en fonction de celle-ci, et ce sur l'ensemble des données. Le choix de la clé doit donc être fait avec soin. Celle-ci va, en général, être construite en fonction de plusieurs valeurs. Par exemple, dans le cas d'un système de stockage de courriels, la clé pourrait être composée du nom de l'utilisateur suivi d'un identifiant unique pour le courriel. De cette manière, les courriels seraient triés au sein de la base de données par utilisateur, et il serait rapide pour une application de parcourir tous les courriels selon un utilisateur en particulier, quel que soit le volume de données présent.

Les colonnes ne sont pas définies à l'avance avec HBase. Une ligne peut contenir un nombre indéterminé de colonnes, ce qui permet de regrouper un nombre variable d'éléments pour chaque enregistrement. Enfin, les colonnes sont regroupées par famille. Les familles sont définies au moment de la création de la table, et sont séparées au niveau du système de fichier.

Lors des précédentes itérations de ce projet, un schéma HBase a été proposé afin d'améliorer les performances des requêtes problématiques présentées précédemment. La clé était alors composée de la manière suivante :

variant_type_id | sample_id | pipeline_id | variant_id

La première particularité de cette clé est la présence d'identifiants dans sa composition. En réalité, lors de la recherche, les attributs étaient initialisés à partir d'une base de données relationnelle. L'identifiant unique de ces attributs était donc récupéré avant de faire des requêtes sur la base de données HBase. Les identifiants présents sont les suivants :

- variant_type_id : Identifiant d'un type de variation dans la base de données relationnelle.
- sample_id : Identifiant d'un échantillon dans la base de données relationnelle.
- pipeline_id : Identifiant d'un « pipeline » dans la base de données relationnelle, qui correspond à une combinaison de techniques et d'outils qui ont permis le séquençage.

- variant_id : Identifiant d'une variation dans la base de données relationnelle.

La structure de la clé est adaptée pour récupérer de nombreuses variations selon des paramètres de type, d'échantillon et de pipeline définis. Dans le cas où certains de ces paramètres peuvent varier, plusieurs « scans » seront nécessaires afin de récupérer toutes les données.

En ce qui concerne les colonnes pour ce schéma, elles représentent les données pour chaque variation, comme le génotype, la fréquence de la variation…. On peut noter que la dénormalisation des données a été poussée au maximum puisque ces informations se voient répliquées pour chaque échantillon qui possède la même variation. Cette stratégie a été appliquée afin de réduire de nombre de requêtes nécessaires pour accéder à ces informations, sans tenir compte de l'espace disque nécessaire.

3.3.2 Nouveau schéma à partir des cas d'utilisation

Le premier objectif de ce projet a été de revoir le schéma HBase proposé, afin d'essayer de trouver une solution plus optimisée. Celui-ci a été revu à partir des cas d'utilisation identifiés précédemment.

Comme nous l'avons vu, le cas d'utilisation principal est celui à partir duquel le schéma va être défini, parce qu'il s'agit à la fois du cas le plus utile et le plus complexe. Les attributs de recherche vont permettre de définir comment et dans quel ordre les données devront être stockées au sein de la table. La position des variations est ici le seul attribut obligatoire et l'attribut le plus important puisqu'il existe plusieurs millions de variations dans le génome humain. Les attributs secondaires sont donc les échantillons, les types de variations et l'identification des résultats positifs.

Une première solution testée pour cette nouvelle itération de recherche s'est basée sur le modèle précédent, c'est-à-dire qu'une clé spécifique permet d'accéder à l'information de

détection de la variation pour un échantillon ainsi qu'aux informations de la variation en elle-même. La partie droite de la clé est constituée de la position dans le génome, puisque c'est l'attribut qui varie le plus. À l'opposé, la partie gauche de la clé est constituée des attributs qui varient le moins, comme le type de variations, ou la présence de la variation pour l'échantillon. La stratégie mise en place concernant les variations possibles des attributs est donc cohérente. Un exemple de clés composées de l'échantillon, du type de variation et de la position de la variation est donné ci-dessous :

```
HG00096:SNP:1:00000001568
HG00096:SNP:1:00000002115
…
HG00097:SNP:1:00000001022
```

Ici, les clés correspondent aux échantillons HG0096 et HG00097. Les variations sont de type SNP, et sont présentes dans le premier chromosome, la dernière partie de la clé correspondant à la position dans le chromosome.

Ce modèle entraine un nombre élevé de lectures si les attributs qui constituent la partie gauche de la clé varient beaucoup pour une même requête, notamment pour les échantillons. Concrètement, lorsqu'un unique échantillon est choisi, une simple lecture séquentielle sera effectuée pour récupérer des variations présentes dans la même zone, mais si d'autres échantillons sont sélectionnés, le nombre de lectures nécessaires augmentera d'autant, et nuira aux performances.

La stratégie à mettre en place dans cette situation est le regroupement de valeurs au sein d'une même ligne, puisqu'il n'existe pas de limites quant au nombre de colonnes présentes. La clé est donc maintenant composée uniquement de la position de la variation et de son type, qui sont les seuls attributs nécessaires pour identifier une variation unique.

CHROMOSOME:POSITION:TYPE

Les colonnes sont regroupées en deux catégories. La première comprend les attributs de la variation, et la seconde comprend tous les échantillons, avec les informations sur la détection de la variation. Dans les deux cas, les colonnes ne sont pas déterminées à l'avance, et vont dépendre des informations récupérées dans les fichiers VCF.

3.3.3 Les familles de colonnes

Avec la base de données HBase, les colonnes peuvent être regroupées par famille. Ainsi, au niveau du système de fichier, chaque famille possède son propre bloc dans une région, ce qui permet de faire une vraie séparation entre elles. Cette séparation peut être utile pour plusieurs raisons. Tout d'abord, si l'on souhaite accéder à l'ensemble des colonnes qui répondent à une certaine propriété, on pourrait indiquer cette propriété dans le nom de la colonne, puis ajouter un filtre pour les noms de colonne. Cette solution fonctionnerait, mais dans certains cas un grand volume de données inutiles serait parcouru pour récupérer les bonnes colonnes. En utilisant des familles, on accède directement à toutes les données sans faire de tri supplémentaire. Enfin, si l'on doit récupérer beaucoup de lignes, le nombre de colonnes présentes pour chacune d'entre elles peut avoir un impact sur les performances, donc encore une fois, en séparant les données par famille et que l'on spécifie une famille spécifique lors du chargement des données, le traitement sera plus efficace.

Pour ce schéma, trois familles de colonnes ont été définies. Tout d'abord, la famille « variant » regroupe toutes les informations sur la variation, indépendamment des résultats des échantillons. On retrouve par exemple les colonnes « référence » et « alternative » qui font référence au génotype de la variation.

Les résultats pour les échantillons ont été séparés en deux familles : « sample » et « nosample ». La première famille contient les résultats positifs, c'est-à-dire les échantillons pour lesquels la variation a été détectée, alors que la seconde contient les résultats négatifs.

Cette séparation a été faite, car la plupart des requêtes de l'application porteront sur les résultats positifs, alors que ces résultats sont minoritaires par rapport aux résultats négatifs.

3.3.4 Exemples d'enregistrements

Le Tableau 3.3 présente des exemples d'enregistrement que l'on pourrait avoir pour la famille « variant ».

Tableau 3.3 Exemples d'enregistrements pour la famille "variant"

Clé HBase	REF	ALT	VCF	DBSNP	QUAL	FILT	VT	AA
1:000005367:SNP	A	T	18764	51448	100	PASS	SNP	T
1:000005372:INDEL	G	<DEL>	12364	98521	89	PASS	INDEL	C

Le premier enregistrement correspond à une variation dans le premier chromosome, à la position 5367, de type SNP. Le deuxième enregistrement correspond à une variation dans le premier chromosome, à la position 5372, de type INDEL. Le nombre de colonnes dépendra du nombre de propriétés disponibles pour la variation dans le fichier VCF. Cependant, certaines propriétés sont fixées par le format VCF lui-même. Il s'agit des six premières colonnes :

- REF : génotype de référence de la variation
- ALT : génotype alternatif de la variation
- VCF : identifiant de la variation provenant du fichier VCF
- DBSNP : identifiant de la variation pour la base de données dbSNP (projet non lié)
- QUAL : indice sur la qualité de l'information sur le génotype alternatif
- FILT : indique si la position a passé tous les filtres

La colonne « VT » indique le type de la variation, et la dernière colonne, « AA » correspond à un attribut optionnel sur l'allèle ancestral pour la variation.

Les tableaux suivants présentent des exemples d'enregistrements pour les familles « sample » et « nosample ».

Tableau 3.4 Exemple d'enregistrements pour la famille "sample"

Clé HBase	**LJ00032**	**MP00325**
1:000005367:SNP	0/1:0.400:-0.18	1|1:0.144:-0.400

Tableau 3.5 Exemples d'enregistrements pour la famille "nosample"

Clé HBase	**HF00167**	**BD01978**
1:000005367:SNP	0/0:0.500:-0.9	0|0:-0.18:-0.400

Les colonnes correspondent aux noms des échantillons. La valeur correspondante est le résultat de la détection de la variation pour cet échantillon.

3.3.5 Index secondaires

Les index secondaires sont une autre optimisation possible pour le schéma de la base de données. Avec HBase, les données sont triées avec la clé de la table, et il n'est pas possible d'ajouter automatiquement d'autres index pour accéder plus rapidement à des données suivant d'autres attributs. Cependant, il existe d'autres solutions pour gérer des index supplémentaires, comme les projets IHBase ou ITHbase (George, 2011).

Pour ce projet, les index secondaires sont gérés directement à l'aide d'une table supplémentaire. Un cas simple a été retenu pour illustrer leur utilisation : l'accès à de nombreuses variations pour certains types peu présents.

Pour comprendre ce cas, il faut rappeler le premier cas d'utilisation : l'accès à un ensemble de variations suivant certains attributs. Parmi ces attributs, on peut spécifier un ou plusieurs types de variations. Lorsqu'aucun type n'est donné, il n'y a pas de problèmes puisque les

lignes sont parcourues de la position de départ à la position d'arrivée. Lorsqu'un type qui revient fréquemment est spécifié, il suffit de spécifier un filtre au niveau de la clé afin d'exclure les autres types sans aucun impact important sur les performances. Mais dans le cas où un type peu fréquent est spécifié, le nombre de lignes à exclure devient trop important, ce qui va nuire aux performances.

Une table d'index secondaires est donc créée afin de stocker les variations par type. Une seule colonne dans cette table contient la clé pour la table principale. Lorsque l'application souhaitera récupérer des valeurs avec un type problématique, une première méthode récupère les clés correspondantes dans la table secondaire, puis une deuxième méthode récupère les données dans la table principale avec des accès directs. Le Tableau 3.6 présente un exemple de plusieurs enregistrements qui peuvent être présents dans la table de l'index secondaire.

Tableau 3.6 Exemples d'enregistrements pour la table d'index secondaires

Clé HBase	**Colonne « key »**
INDEL:1:000005372	1:000005372:INDEL
INDEL:1:000005391	1:000005391:INDEL
SNP:1:000005367	1:000005367:SNP
SNP:1:000005382	1:000005382:SNP
SNP:1:000005385	1:000005385:SNP

Pour le premier enregistrement par exemple, il s'agit d'une variation de type INDEL dans le premier chromosome à la position 5372. L'accès à plusieurs variations de type INDEL dans la même zone est très rapide, et une liste de clé pour la table principale pourra être obtenue rapidement.

3.4 Procédure d'importation

3.4.1 Introduction

La phase d'importation des données consiste à remplir la base de données HBase à partir des données des variations. Lors des précédentes itérations du projet, l'importation des données vers HBase était réalisée à partir d'une base de données MySQL, réplique de la base de données utilisée par le laboratoire Guy Rouleau. Cette tâche était alors effectuée à l'aide de l'utilitaire Apache Sqoop.

Apache Sqoop permet de faire des importations ou des exportations en ligne de commande à partir d'une base de données relationnelle. La compatibilité des bases de données est assurée avec l'utilisation du connecteur JDBC. Pour ce projet, une nouvelle vue a été créée sur le serveur MySQL afin d'accéder directement à toutes les données utiles. Sqoop a ensuite été utilisé pour exporter les données retournées par cette vue vers une nouvelle table HBase.

3.4.2 Révision

La procédure mise en place auparavant est fonctionnelle, mais elle implique l'utilisation d'une base de données relationnelle. Avec l'utilisation de fichiers de données brutes, il a été nécessaire de revoir cette procédure.

Les fichiers de données représentent un volume non négligeable : environ 140 Go de données compressées pour les archives utilisées pour le projet. Il est possible de développer un programme permettant de lire ces données et de faire des insertions directement dans HBase à l'aide de son API, mais il est plus judicieux de se servir de MapReduce pour réaliser cette tâche. En effet, MapReduce est spécialement conçu pour traiter de grandes quantités de données sur un cluster. Dans le cadre du projet, un cluster avec une installation de MapReduce est déjà disponible puisque HBase se base sur Hadoop pour fonctionner.

La procédure va donc consister à lire les fichiers de données à partir d'une tâche MapReduce lancée sur le cluster puis d'insérer ces données dans HBase. Le principal avantage de cette approche sera le temps nécessaire pour faire l'importation des données. En effet, avec MapReduce, chaque nœud du cluster va travailler avec une partie des données. On profite alors de la présence de plusieurs nœuds pour réduire d'autant la charge de travail à partir des données sources.

Dans le cas présent, un ou plusieurs fichiers au format VCF seront utilisés pour le lancement de la tâche. Il est intéressant de noter que ces fichiers peuvent être compressés ou non, MapReduce étant programmé de sorte qu'il peut lire les fichiers compressés de manière transparente pour l'utilisateur. La sortie de la tâche va dépendre de la stratégie choisie pour le chargement des données. Le premier cas est le chargement des données directement dans une table HBase. Pour cela, il faut paramétrer la tâche pour utiliser une table comme format de sortie. Lorsque la tâche sera exécutée, la partie « map » de la tâche va lire une ligne source, qui correspond ici à une variation et les résultats obtenus pour les échantillons. Elle va associer à une nouvelle clé au format « CHROMOSOME:POSITION:TYPE » une opération « Put ». Il n'y a pas d'étape « reduce » pour la tâche, car les données sont déjà chargées dans la base de données après l'étape « map ».

Ce type d'insertion est utile lorsque l'on souhaite ajouter des données dans des tables déjà existantes. Cependant, la procédure est relativement longue, et on pourra utiliser une fonction d'importation en masse pour une première initialisation des données.

3.4.3 Importation massive de données

L'importation massive (« bulk import ») peut être utilisée si aucune donnée n'est présente dans la base de données. Cette procédure va consister à générer dans un premier temps des fichiers HFile, qui possèdent la même structure que les fichiers manipulés par HBase, dans un répertoire HDFS. Ensuite, ces fichiers seront tous simplement copiés dans les répertoires gérés par HBase.

Afin d'éviter des divisions automatiques des tables sur plusieurs nœuds lors du chargement des données, opérations coûteuses en temps de traitement, il est également possible de créer manuellement les régions à l'avance. Pour cela, les données sont analysées avant de commencer l'importation. Pour chaque chromosome, le nombre de variations et la position maximale au sein du chromosome sont enregistrés. À partir de ces informations, chaque chromosome est découpé en un certain nombre de régions qui contiennent chacune un nombre équivalent de variations. Ces régions sont enregistrées dans un fichier qui contient la liste des clés qui délimitent les régions, par exemple :

```
1:0000820000
1:0001640000
1:0002460000
…
19:002456000
```

Une clé qui délimite une région ne doit pas nécessairement correspondre à une clé existante, c'est pourquoi on retrouve ici des valeurs arrondies et les types de variation ne sont pas présents. Ce fichier sera lu avant de créer des tables dans HBase. Lors de la génération des fichiers HFile, la structure de la table sera récupérée afin de définir le nombre de fichiers à écrire. Aussi, cela permet d'utiliser un nombre de tâches « reduce » égal au nombre de régions présentes, puisque chacune de ces tâches va écrire dans un fichier séparé.

La Figure 3.1 présente la procédure complète pour l'importation massive de données. Les tâches MapReduce sont identifiées en jaune.

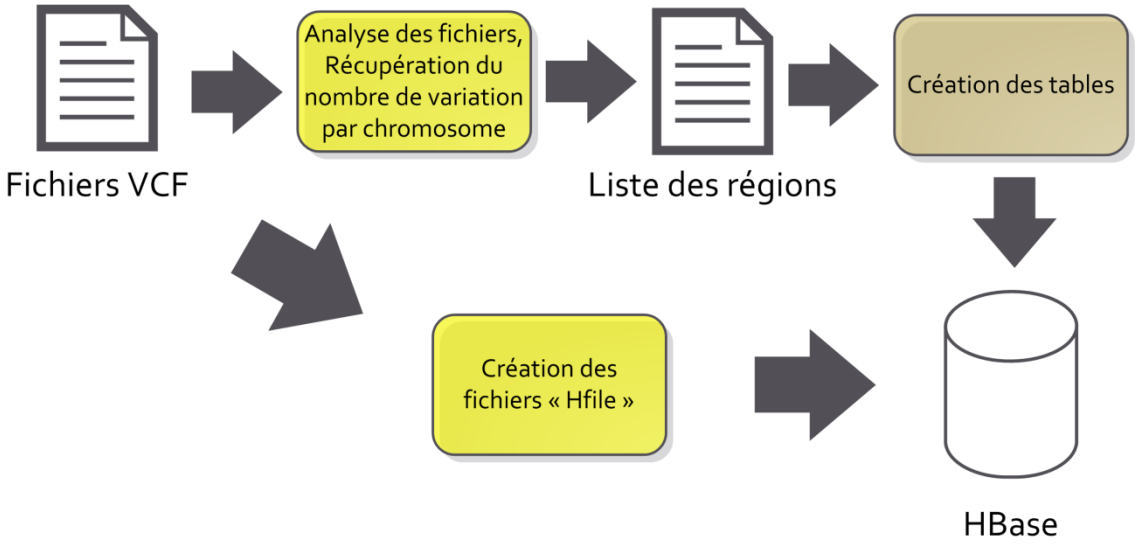


Figure 3.1 Procédure complète pour l'importation massive de données

3.4.4 Développement de la procédure

La mise en application de la procédure présentée ci-dessus passe par le développement de programmes permettant d'automatiser les tâches. Afin de profiter de l'API native de Hadoop et HBase, ceux-ci ont été développés avec le langage Java.

Toutes les classes et méthodes nécessaires pour faire l'importation ont été regroupées dans une même archive JAR avec plusieurs points d'entrées. Ainsi, chacune des étapes peut-être exécutées indépendamment si besoin. Le premier point d'entrée de l'archive JAR est « CreateRegions ». Il permet, à partir de fichiers VCF, de créer le fichier des régions. La syntaxe est la suivante :

```
hadoop jar import.jar gvdb.importation.CreateRegions <intervalle> <vcf> <sortie>
```

Le paramètre « intervalle » contrôle l'intervalle entre deux positions dans un chromosome afin de définir la taille des régions. Une valeur typique est 100 000 et permet d'avoir un bon ratio entre le nombre de régions et la taille des régions. Le paramètre « vcf » indique

l'emplacement des fichiers VCF. Le paramètre « sortie » indique le nom à utiliser pour le fichier des régions.

Le deuxième point d'entrée est « CreateTable ». Il permet de créer une table dans HBase en utilisant éventuellement un fichier de régions.

```
hadoop jar import.jar gvdb.importation.CreateTable [-r regions] <table> [famile1…]
```

Le paramètre « -r », optionnel, permet d'indiquer un fichier de régions à utiliser. Le paramètre « table » indique le nom de la table à créer. Ensuite, une liste de valeurs indique le nom des familles de colonnes à créer.

Les deux points d'entrée suivants permettent de créer les fichiers « HFile » pour la table principale et la table d'index.

```
hadoop jar import.jar gvdb.importation.CreateHFiles <table> <vcf> <destination>
hadoop jar import.jar gvdb.importation.CreateHFilesIndex <table> <vcf> <destination>
```

Le paramètre « table » indique la table à utiliser pour la génération des fichiers afin de récupérer les informations sur les régions. Le paramètre « vcf » indique l'emplacement des fichiers VCF, et le paramètre « sortie » indique la destination pour les fichiers « HFile ».

Le chargement des fichiers « HFile » dans la base de données est réalisé avec un programme directement inclus dans l'archive JAR de HBase :

```
hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles <path> <table>
```

La variable « path » correspond au chemin contenant les fichiers HFiles, et la variable « table » est le nom de la table HBase de destination.

3.5 API pour l'accès aux données

Une API est une composante logicielle qui fournit des fonctions permettant à des applications d'interagir avec un autre système ou une autre application. Dans le cadre de ce projet, l'API est une bibliothèque développée en Java qui permet d'accéder à des informations contenues dans la base de données HBase. Pour cela, elle utilise directement l'API de HBase, et lui apporte des classes qui décrivent les informations contenues dans les tables ainsi que des méthodes d'accès développées selon le schéma de ces tables. Pour cela, il est nécessaire de connaître la structure des données utilisées et les cas d'utilisation, ainsi que le schéma des tables. L'API de HBase étant développé principalement en Java, l'API du projet utilisera également ce langage. L'objectif est alors de construire un fichier JAR avec toutes les fonctions utiles pour qu'une application puisse accéder aux informations de la base de données. Dans un modèle MVC (Model View Controler), l'API correspondrait à la partie Model, c'est-à-dire la couche métier de l'application.

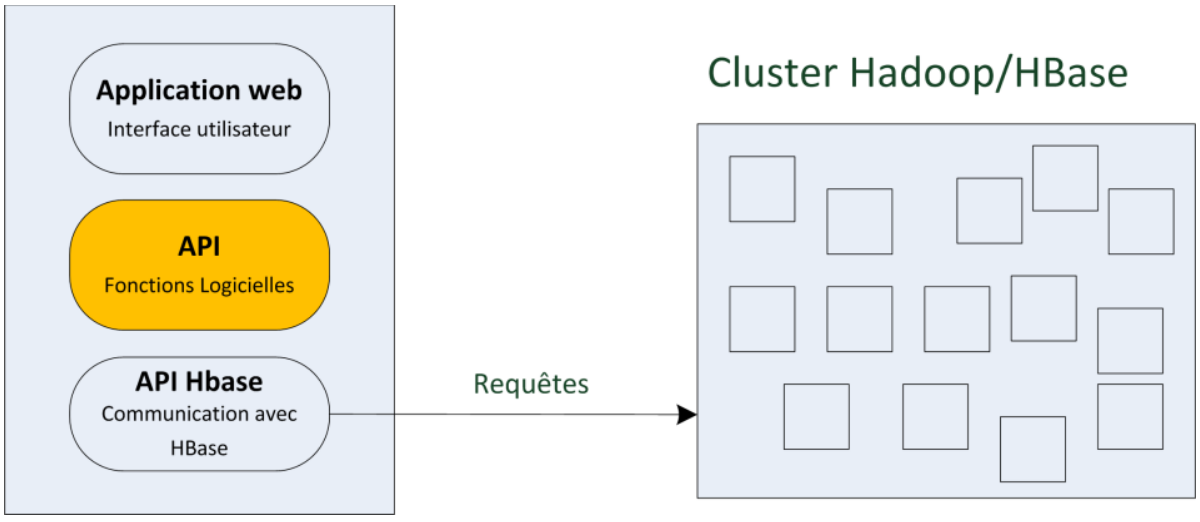


Figure 3.2 API dans l'architecture logicielle

La première des fonctions développées pour l'API est la mise à disposition de classes qui décrivent les données de la base de données. On retrouve notamment les classes suivantes :

- Sample : représente un échantillon.
- Variant : contient les informations pour une variation génomique.
- SampleData : contient les informations de détection d'une variation pour un échantillon.

Une fois les classes de données définies, il faut ajouter des méthodes permettant d'exécuter des requêtes vers la base de données pour récupérer les informations. Souvent, le modèle suivi pour développer ces fonctions consiste, pour chaque classe de données, à développer des méthodes d'accès, de mise à jour ou de suppression. Dans le cas de cette application, la performance de certains cas d'utilisation est l'un des principaux enjeux, et ce modèle n'a pas été suivi. Les cas d'utilisation ont été synthétisés en définissant des prototypes de fonctions pour chacun d'entre eux avec des interfaces. Ces interfaces sont ensuite implémentées avec des classes fonctionnelles. Ce modèle permet notamment de développer et de tester plusieurs implémentations en même temps afin de comparer plusieurs solutions au niveau des performances. La liste des interfaces utilisées pour le projet est donnée en ANNEXE II.

Plusieurs optimisations sont possibles pour l'implémentation des fonctions. Tout d'abord, il est possible avec l'API de HBase de spécifier un cache lors de l'utilisation de scanneurs. Les scanneurs parcourent une table en lisant plusieurs lignes contigües. Par défaut, chaque ligne sera récupérée avec une nouvelle requête vers la base de données. Augmenter la taille du cache permet de récupérer plusieurs lignes avec une même requête. Lors des précédentes itérations de recherche, des expérimentations ont été réalisées avec le cache des scanneurs, et une taille fixée à 500 lignes par requête a été jugée comme un bon compromis entre les performances et la mémoire utilisée. Avec le changement du schéma de la table HBase, chaque ligne contient beaucoup plus d'informations que précédemment, ce paramètre a donc été revu en tenant compte de la quantité de données à récupérer pour chaque ligne, afin de conserver des performances correctes : environ 100 lignes par requête pour un usage léger de la base de données, et jusqu'à 10 lignes par requête pour un usage intensif.

Un autre paramètre à prendre en compte pour la performance des requêtes est l'utilisation de la table d'index. Pour rappel, la table d'index mise en place pour cette expérimentation concerne les différents types de variations génétiques enregistrés. L'utilisation de la table d'index va être profitable dans certains cas, mais nuira aux performances dans d'autres cas. Il est donc nécessaire de définir les conditions d'utilisation de cette table avec des paramètres supplémentaires.

La gestion de la connexion avec la base de données est un autre rôle de l'API. Une application utilisant l'API n'aura ainsi qu'un minimum de configuration à faire avant de pouvoir commencer à exécuter des requêtes. Enfin, des fonctions supplémentaires ont été ajoutées pour contrôler l'exécution des requêtes, notamment au niveau du temps d'exécution. Pour cela, les objets permettant d'exécuter des requêtes doivent, dans un premier temps, s'enregistrer auprès d'un gestionnaire de requête qui pourra récupérer des informations utiles. Il est possible d'ajouter d'autres actions en créant de nouvelles classes héritées des classes existantes, comme l'enregistrement de toutes les requêtes exécutées afin de récupérer des statistiques.

L'ANNEXE III apporte plus d'informations sur les classes et les liens entre elles avec notamment un digramme de classe simplifié.

3.6 Application web

Avec la revue des cas d'utilisation et le développement d'une API séparée, une nouvelle application web a été développée afin de tirer profit des changements. Inspirée par la précédente application, elle reprend certaines technologies utilisées comme Java EE et les servlets. Elle est cependant plus simple et légère, notamment grâce au déchargement des fonctions maintenant assurées par l'API, et l'utilisation du format HTML pour l'interface utilisateur à la place d'une interface riche exploitant JavaScript.

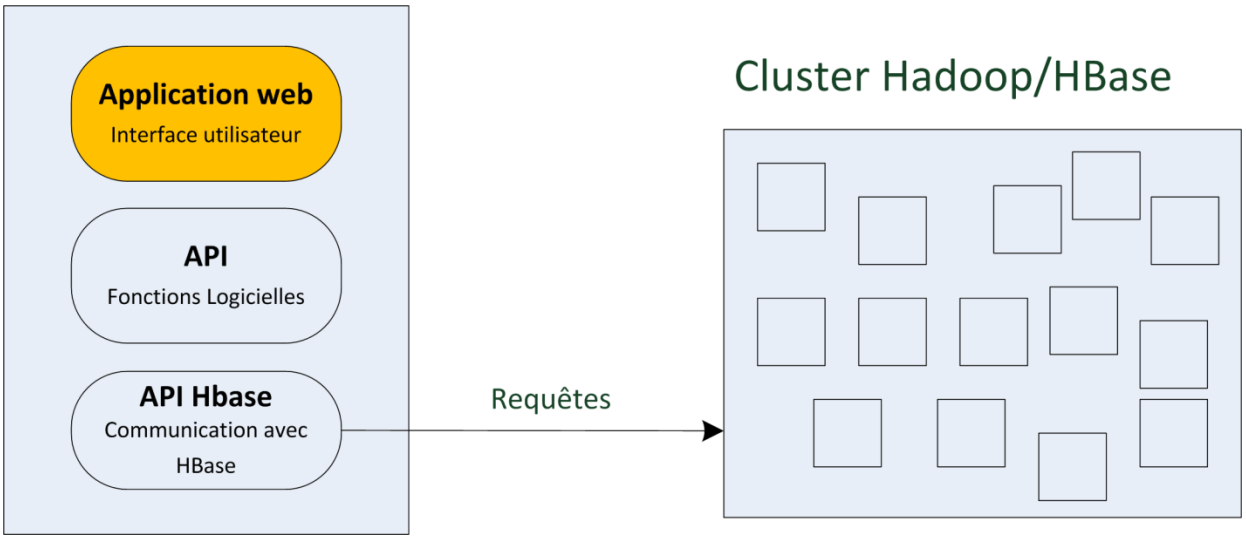


Figure 3.3 L'application web dans l'architecture logicielle

Les données peuvent être récupérées de deux manières différentes : avec une interface utilisateur ou bien dans un format XML. L'interface utilisateur permet de paramétrer et exécuter des requêtes très facilement. Les attributs de recherche permettent de profiter de la plupart des fonctions de l'API. Les fichiers XML sont une alternative, et peuvent être utiles dans le cas où une application cliente souhaite récupérer des informations à partir de l'application. Les fonctionnalités de l'application web sont découpées en 3 servlets.

Tableau 3.7 Description des servlets

Nom du servlet	**Description**
VariantResults	Récupérer des variations selon un intervalle de positions de variations génétiques au sein d'un chromosome. Il est possible de filtrer les échantillons et les types à inclure dans la requête.
VariantInfos	Récupération des informations sur une variation en particulier, ainsi que la liste des échantillons correspondants.
SampleInfos	Récupérer les variations présentes pour un échantillon en particulier.

Le premier servlet permet de lancer des requêtes selon le principal cas d'utilisation, c'est-à-dire la recherche de variations et d'échantillons correspondants au sein d'une plage de positions. Les attributs optionnels de recherche sont le type de variation et la liste des échantillons pour lesquelles la recherche doit être menée. Le deuxième servlet permet de récupérer toutes les informations sur une variation, avec la liste des échantillons pour lesquels la variation a été détectée. Enfin, le dernier servlet permet de récupérer la liste des variations pour un échantillon dans une certaine plage. Le fonctionnement de chaque servlet est donné en détail en ANNEXE IV.

Le point d'entrée de l'interface web est le formulaire de recherche pour le premier cas d'utilisation. Il est possible de spécifier un ou plusieurs échantillons à inclure en faisant une sélection multiple dans une liste, de même que pour la liste des échantillons à exclure. Il est aussi possible de spécifier un ou plusieurs type(s) de variation à inclure, le chromosome à inclure dans la recherche, et les positions de départ et de fin de la recherche.

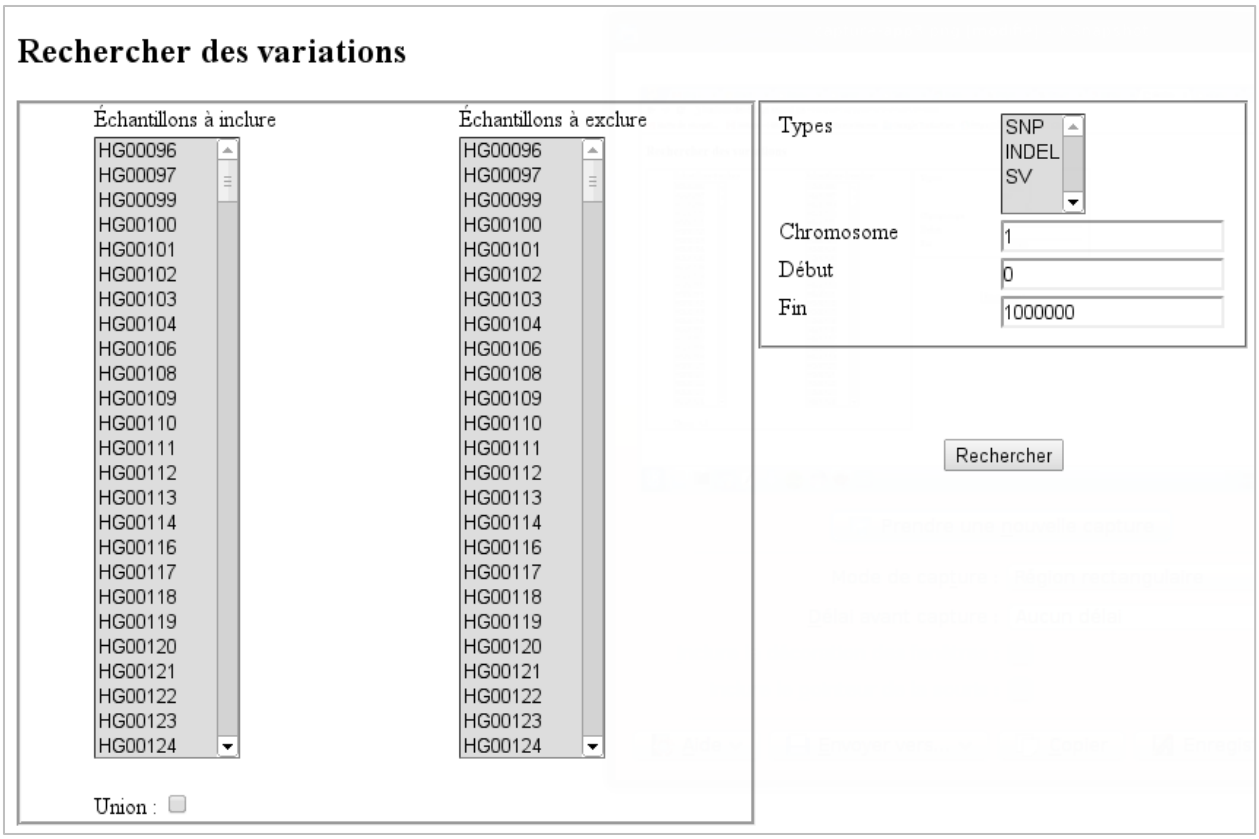


Figure 3.4 Interface utilisateur : formulaire de recherche

Les résultats sont présentés sous la forme d'un tableau. Plusieurs informations sont disponibles pour chaque variation, notamment la liste des échantillons pour lesquels la variation a été détectée. Il est possible de cliquer sur une position afin d'obtenir les informations complètes pour une variation avec une nouvelle requête exploitant le servlet « VariantInfos ». Il est également possible de cliquer sur un échantillon pour obtenir la liste de ses variations en passant par le servlet « SampleInfos ». Pour chaque requête exécutée à partir de l'interface utilisateur, le temps d'exécution est indiqué.

Chromosome 1

Positions 0000000000 - 0001000000

Chromosome	Position	Type	Référence	Alternative	Échantillons
1	0000013957	INDEL	TC	T	HG00110 HG00113
1	0000055249	INDEL	C	CTATGG	HG00101
1	0000063735	INDEL	CCTA	C	HG00101 HG00103 HG00104 HG00108 HG00109 HG00110 HG00112 HG00113 HG00114
1	0000084005	INDEL	AG	A	HG00103
1	0000094421	INDEL	TC	T	HG00106
1	0000109107	INDEL	G	GT	HG00114
1	0000165954	INDEL	GAATA	G	HG00104 HG00109 HG00110
1	0000247916	INDEL	CAGG	C	HG00112
1	0000249275	INDEL	G	GT	HG00111 HG00112 HG00114
1	0000251627	INDEL	AC	A	HG00102 HG00111 HG00112 HG00114
1	0000255923	INDEL	G	GTC	HG00102 HG00104 HG00106 HG00108 HG00109 HG00111 HG00112 HG00113 HG00114
1	0000532258	INDEL	AAAT	A	HG00109 HG00110
1	0000540540	INDEL	GC	G	HG00102
1	0000568745	INDEL	C	CCA	HG00108 HG00109
1	0000626686	INDEL	CCT	C	HG00110
1	0000668394	INDEL	AG	A	HG00103
1	0000691541	INDEL	AT	A	HG00101 HG00102 HG00103 HG00108 HG00110 HG00112

Figure 3.5 Interface utilisateur : résultats de recherche

Comme mentionné précédemment, un format XML est une alternative pour la récupération des données. Celui-ci présente les mêmes résultats formatés avec des balises. Un résultat d'une requête peut se présenter ainsi :

```
<variants>
  <variant>
    <chromosome>1</chromosome>
    <position>10583</position>
    <type>SNP</type>
    <reference>G</reference>
    <alternative>A</alternative>
    <sample>HG00100</sample>
    <sample>HG00102</sample>
    <sample>HG00106</sample>
    <sample>HG00111</sample>
    <sample>HG00112</sample>
    <sample>HG00119</sample>
    <sample>HG00120</sample>
  </variant>
  <variant>
    <chromosome>1</chromosome>
    <position>10611</position>
    <type>SNP</type>
    <reference>C</reference>
    <alternative>G</alternative>
    <sample>HG00110</sample>
    <sample>HG00113</sample>
  </variant>
</variants>
```

L'application web utilise un fichier de configuration Spring, un framework Java qui apporte de nombreuses fonctionnalités qui ne seront pas détaillées ici. Ce fichier permet notamment d'ajouter des méthodes d'inversion de contrôle dans l'application afin de spécifier les classes à utiliser et leurs paramètres. Dans le cadre de cette application, il s'agit principalement d'indiquer le chemin du fichier de configuration de HBase pour établir la connexion, et les classes à utiliser pour l'accès aux données. Ainsi, il est possible, sans modifier le code Java, de déployer l'application dans différents environnements cibles, tout en permettant l'utilisation de classes alternatives pour l'accès aux données, dans le cas par exemple où le schéma est modifié. Un extrait du fichier de configuration Spring utilisé à la fin de cette itération est donné en ANNEXE V.

3.7 Mise en place de la solution

3.7.1 Utilisation d'un cluster à l'ÉTS

Un petit cluster a été installé et configuré avec Hadoop et HBase pour réaliser des tests lors des précédentes phases du projet. Celui-ci est constitué de 3 machines virtuelles, chacune utilisant 2 cœurs logiques. Le système maître utilise 32 Go de RAM, et les deux autres systèmes utilisent 16 Go de RAM. Le système d'exploitation installé sur chacune des machines est Ubuntu 12.04. Hadoop est installé en version 1.0.3, et HBase en version 0.94.2. L'espace disque disponible au sein du système de fichier HDFS est de l'ordre de 240 Go.

Ce cluster a été utilisé tout au long du projet afin de tester tous les composants développés. Cependant, le nombre de nœuds configuré et l'espace disque disponible sont relativement limités pour ce type de projet. Un cluster plus performant est nécessaire pour réaliser des tests avec l'ensemble des données. Cependant, la mise à disposition d'un tel cluster est longue et couteuse, et une alternative a donc été préférée pour se libérer de cette contrainte. Il s'agit de l'utilisation de services de cloud computing, permettant notamment de valider le travail réalisé sans délai supplémentaire.

3.7.2 Utilisation du cloud computing

L'utilisation d'un service de cloud computing, ou informatique dans les nuages, va permettre de tester la solution proposée à bas coût. En effet, avec ce type de service, seules les ressources réellement utilisées sont facturées.

Dans le cadre des expérimentations de ce projet, c'est le service EC2 d'Amazon qui a été retenu (Elastic Compute Cloud). Il s'agit d'un service de type IaaS (Infrastructure as a Service) relativement populaire qui est mis en place dans plusieurs centres de données d'Amazon à travers le monde (Global Infrastructure). Il permet notamment d'exécuter des instances, qui sont en réalité des machines virtuelles disponibles à la demande. Un large choix de systèmes d'exploitation est disponible pour utiliser les instances, que ce soit pour les systèmes Windows ou GNU/Linux.

Concrètement, l'utilisation de ce service va permettre de mettre en place facilement un cluster sur un nombre choisi d'instances, d'installer la solution complète sur ce cluster et de réaliser des tests de performance.

Amazon propose également un service de stockage appelé S3 (Simple Storage Service). Avec ce service, les données sont enregistrées dans des compartiments. L'accès à ces compartiments peut être privé ou public. Les données fournies par le projet 1000 Genomes sont notamment disponibles pour tous dans un compartiment S3 public. La disponibilité de ces données et la proximité de ce service avec EC2 ont amené à l'utilisation de S3 pour le stockage des données lors des procédures de mise en place du cluster et des tests.

3.7.3 Automatisation de la configuration d'un cluster

Afin de faciliter l'installation et la configuration du cluster dans un service de cloud computing, il peut être utile d'utiliser un outil permettant d'automatiser complètement ou en

partie cette étape. Pour ce projet, deux outils ont été étudiés : Apache Whirr et Amazon Elastic Mapreduce. Le premier est une application logicielle développée en Java qui permet d'automatiser le lancement et la configuration de clusters en utilisant l'API mise à disposition par les fournisseurs de services IaaS tels qu'Amazon. Des profils sont notamment prévus pour le lancement de clusters Hadoop/HBase. Amazon Elastic MapReduce est quant à lui un service web permettant de lancer des clusters Hadoop. Depuis l'été 2012, il est également d'inclure HBase dans un cluster avec Elastic MapReduce, en version 0.92, ce qui a permet de considérer l'emploi du service pour ce projet. Les principales différences entre Apache Whirr et Elastic Mapreduce sont synthétisées dans le Tableau 3.8.

Tableau 3.8 Comparaison entre Apache Whirr et Amazon Elastic Mapreduce

Apache Whirr	**Amazon Elastic Mapreduce**
Application libre qui ne vise pas à fonctionner avec un unique fournisseur de services. Elle apporte une couche d'abstraction permettant de proposer les mêmes fonctionnalités avec un grand nombre de fournisseurs de services en utilisant l'API appropriée.	Service d'Amazon qui ne peut pas fonctionner avec un autre fournisseur de services.
Peut être modifié et amélioré librement. Il est très paramétrable et permet par exemple de choisir les logiciels à installer et leur version.	Service propriétaire qui ne peut pas être modifié. Les fonctionnalités proposées ne peuvent pas être étendues, le nombre de logiciels proposés est limité et le support ne concerne que des versions spécifiques de ces logiciels.
Une phase de configuration est nécessaire au préalable.	Peut être utilisé directement depuis une interface web. L'utilisation est simplifiée.
Aucun support officiel n'existe auprès des fournisseurs de services, le fonctionnement	Service officiel d'Amazon.

de l'application n'est pas garanti.	
Aucun coût autre que celui des instances utilisées chez le fournisseur de services.	Coût d'utilisation en sus du coût des instances.

C'est l'outil Apache Whirr qui a été envisagé dans un premier temps. Son principe de fonctionnement, son coût et son indépendance vis-à-vis d'un fournisseur de service ont été les principaux aspects retenus. Une configuration typique exploitant des instances ponctuelles larges à bas prix avec Amazon EC2 est donnée en ANNEXE VIII.

En pratique, l'utilisation d'Apache Whirr a posé quelques problèmes lors des tests, notamment :

- Le manque de support pour les distributions Linux récentes.
- Le lancement du cluster peut échouer pour diverses raisons. Les tests ont été réalisés dans un premier temps dans le centre de données « us-east-1 » d'Amazon, où les clusters ont pu être créés sans problèmes. La fluctuation importante des prix des instances ponctuelles dans ce centre de données a conduit à continuer les tests dans le centre de données « us-west-1 », mais la configuration automatique des clusters échouait souvent dans cette zone.

Ces problèmes peuvent peut-être être contournés en modifiant des paramètres ou bien en ajoutant des contrôles supplémentaires, mais le manque de temps pour ce projet a conduit à l'utilisation d'Amazon Elastic Mapreduce à la place pour réaliser les tests. Le lancement et la configuration de clusters avec ce service sont plus faciles et devraient théoriquement toujours fonctionner. Ainsi, les deux outils ont été réellement testés ce qui a permis de pouvoir tirer plus de conclusions. Le retour d'expérience avec la configuration automatique de clusters dans le « cloud » sera développé dans le CHAPITRE 4.

3.7.4 Procédure d'installation

3.7.4.1 Inscription et accès au service

La première étape avant de procéder à l'installation de la solution est l'inscription à Amazon AWS (Amazon Web Service). Cette inscription est libre et gratuite, et permet notamment de donner des informations de facturations. Après l'inscription, il est possible d'utiliser l'interface web des services pour les utiliser. Il existe également une API permettant à des outils tiers de contrôler les services. Avant de pouvoir utiliser cette API, il est nécessaire de récupérer des informations de connexions, sous la forme d'une clé d'accès publique et d'une clé d'accès secrète.

3.7.4.2 Préparation des données

Les tâches MapReduce avec Hadoop peuvent lire en toute transparence des données compressées ou des données non compressées. Lorsqu'il s'agit de données non compressées, les fichiers volumineux peuvent être découpés afin de répartir le traitement sur plusieurs nœuds. Puisque les données sont traitées ligne par ligne, ce découpage est opéré au niveau d'une nouvelle ligne.

Dans le cas des données compressées, le découpage ne va pouvoir être fait que pour des formats de compressions qui proposent une lecture à une position spécifique du fichier, et non seulement une lecture séquentielle de l'ensemble du fichier. Cette possibilité est apportée par un découpage par blocs des données ou par l'utilisation d'un index, mais reste limitée à certains formats de compression.

Les données offertes par le projet 1000 Genomes sont compressées avec GZip, qui ne propose pas cette fonctionnalité. Ainsi, les fichiers sources tels qu'ils sont disponibles ne pourront pas être découpés afin de profiter d'une répartition optimale de la charge de travail lors de la lecture.

Afin de répondre à ce problème, les fichiers ont été découpés à l'avance afin de fournir à la tâche MapReduce des fichiers avec un volume plus faible. Cette étape est optionnelle, et son unique rôle est d'accélérer l'étape suivante. Elle ne sera pas expliquée en détail.

La procédure suivie pour cette étape est la suivante : tout d'abord, une nouvelle instance de taille moyenne a été créée avec le service EC2. Tous les fichiers de données ont été téléchargés sur cette instance. Chaque fichier a été décompressé puis découpé par blocs de 10 000 lignes. Chaque fichier résultant a ensuite été compressé à nouveau avant d'être copié sur un nouveau compartiment S3. C'est ce compartiment qui sera par la suite utilisé comme source de données.

3.7.4.3 Création des tables HBase

Le chargement des données vers HBase suit la procédure détaillée dans le chapitre 3.4.3. Elle se fait en deux étapes : la création des tables de données, et le chargement sur un cluster. La première de ces étapes se déroule comme suit.

Tout d'abord, un nouveau cluster est exécuté en utilisant Apache Whirr ou Elastic Mapreduce. Les composants de Hadoop utilisés sont MapReduce, HDFS et HBase. Le cluster est composé d'une machine maître, et de 16 machines secondaires pour l'exécution des tâches. Les instances utilisées sont des instances larges, avec 7.5 Go de mémoire, 2 cœurs virtuels et 850 Go d'espace disque.

Lorsque le cluster est configuré et disponible, les données sont copiées à partir du compartiment S3 préparé précédemment sur le système de fichier HDFS. Pour cela, l'outil « distcp », fourni avec Hadoop, est utilisé. Cet outil permet de faire des copies avec de grands volumes de données en utilisant MapReduce. Chaque nœud accède en même temps à la source de données ce qui permet d'accélérer le transfert.

Une fois les données disponibles sur HDFS, la procédure de création des tables est exécutée. Celle-ci est relativement longue puisqu'il est nécessaire de décompresser toutes les données avant de les lire ligne par ligne pour créer les enregistrements correspondants dans les tables de données. Avec les données utilisées pour ce projet, le volume brut de données traitées est d'environ 1 To.

Lorsque la création des tables est terminée, celles-ci sont copiées vers un compartiment S3. L'outil « distcp » est à nouveau utilisé afin d'accélérer le transfert. Les tables de données sont alors persistantes, et le cluster peut être arrêté. La Figure 3.6 illustre la procédure complète.

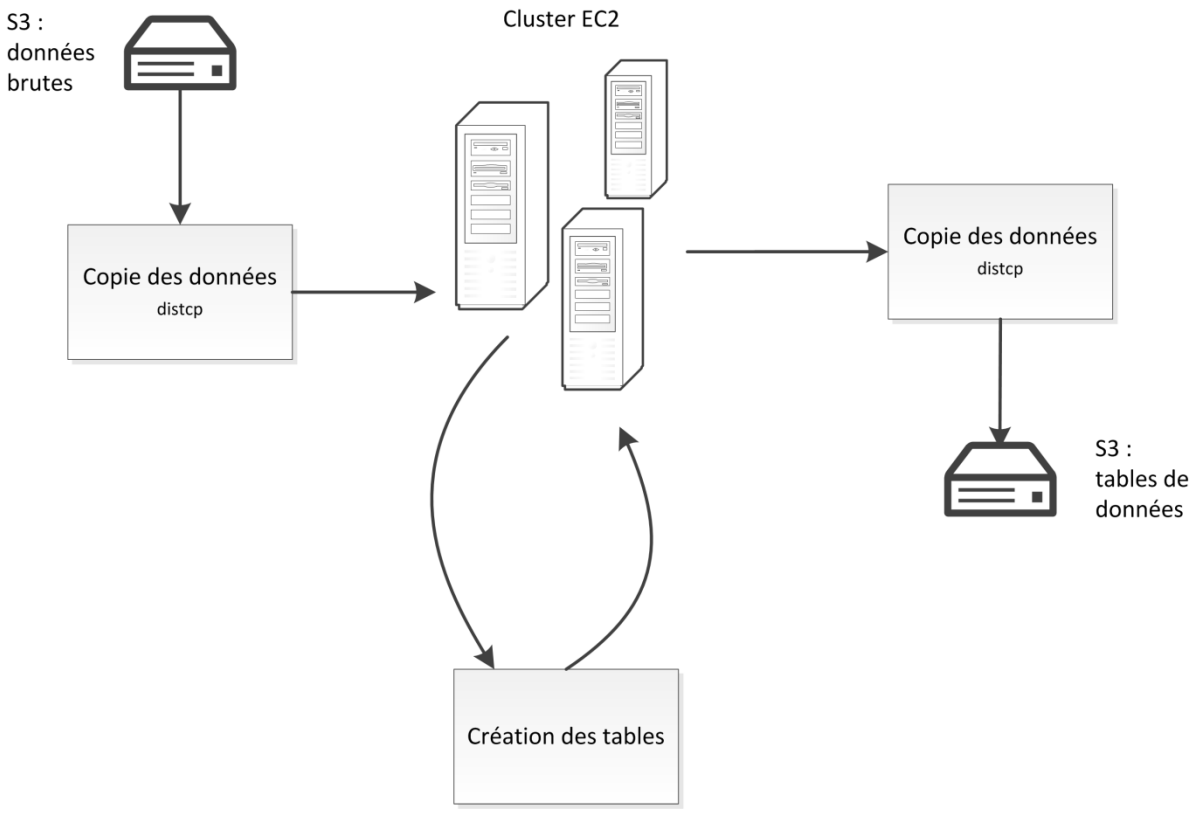


Figure 3.6 Étapes pour la création des fichiers de données

Avec un cluster composé d'une machine principale et de 16 nœuds, la procédure complète a pris environ 10 heures. La liste des commandes utilisées pour la procédure est donnée en ANNEXE VI.

3.7.4.4 Lancement du cluster

La mise en place de la solution suit une procédure similaire à la préparation des tables avec dans un premier temps le lancement d'un cluster avec EC2. Le nombre de nœuds dans le cluster va dépendre des performances souhaitées pour la solution. Dans le cadre de ce projet, la procédure a été testée avec des clusters de 4 à 16 nœuds afin de mesurer et de comparer les performances entre ces différentes configurations. Le même type d'instance que lors de l'étape précédente est utilisé ici.

Une fois le cluster démarré, les fichiers correspondants aux tables de données et stockés avec le service Amazon S3 sont copiés sur celui-ci. Cette étape est encore une fois réalisée avec l'outil « distcp » afin d'accélérer le transfert. Le chargement des tables vers la base de données est ensuite exécuté tel que décrit dans le chapitre 3.4.3. La base de données HBase est alors prête à être utilisée.

L'installation de l'application web est la dernière étape nécessaire pour la mise en place de la solution complète. Pour cette expérimentation, le serveur Tomcat a été choisi pour le déploiement de l'application. Il a été installé directement sur la machine principale du cluster, mais l'installation peut également être faite sur une machine hors cluster. La Figure 3.7 résume les étapes nécessaires pour la mise en place de la solution.

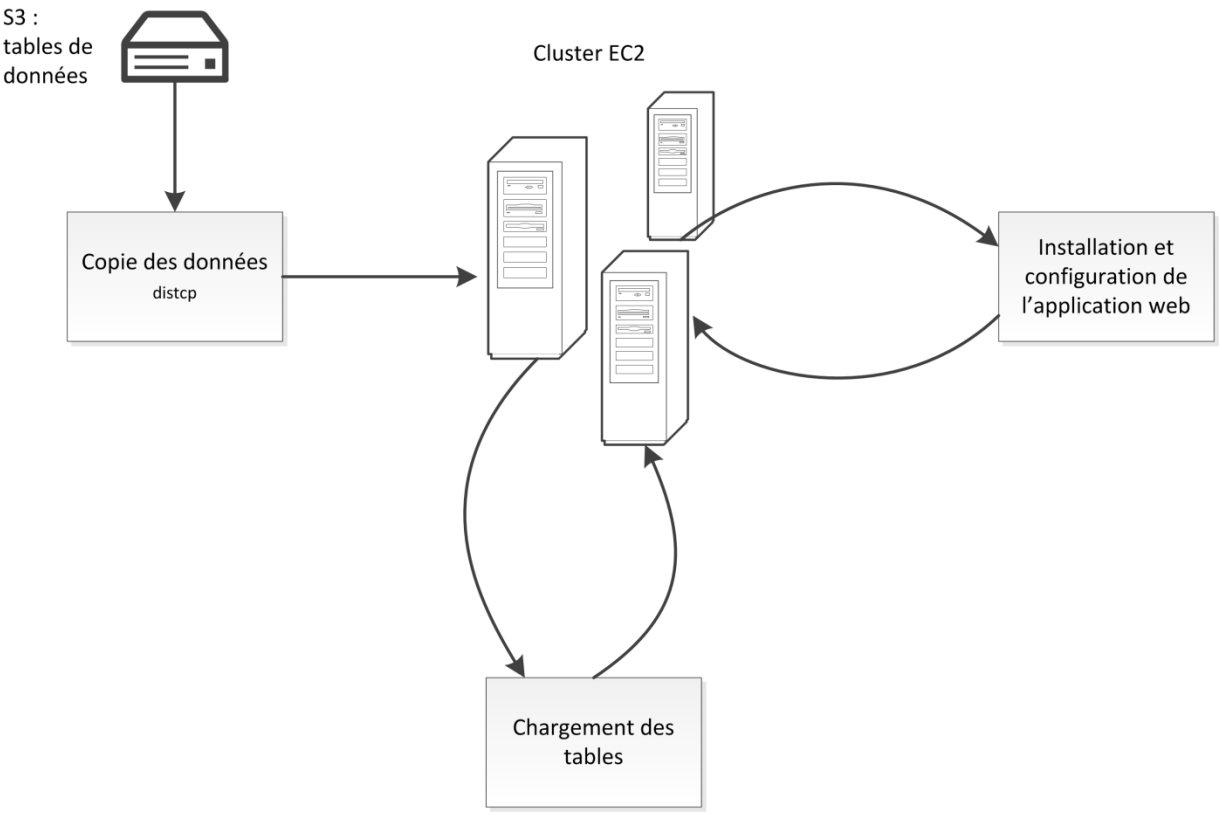


Figure 3.7 Étapes pour le lancement de l'application

La liste des commandes utilisée pour lancer un cluster est donnée en ANNEXE VII.

3.7.5 Remarques sur le cloud computing

Le cloud computing propose de nouvelles techniques pour l'utilisation de ressources informatiques. Pour ce projet et la réalisation de tests, les services de cloud computing sont particulièrement adaptés. Tout d'abord, la simplicité de ces services permet une mise en place très rapide de la solution. Aucun investissement de départ n'est requis, et la disponibilité est constante. Ensuite, l'utilisation de ces services est facturée de manière très précise, à l'heure par exemple pour EC2. Les expérimentations sur de courtes périodes sont donc facilitées. Enfin, la flexibilité du cloud computing permet de lancer un système avec plusieurs échelles possibles.

Après la mise en place de la solution, des tests de performance ont été réalisés selon différentes configurations. La procédure utilisée pour réaliser ces tests est donnée dans la section qui suit, et les résultats seront donnés dans le prochain chapitre.

3.8 Procédure de test de performance

Lors des précédentes itérations de recherche, le principal but des tests de performance était de démontrer l'avantage apporté par l'utilisation du Big Data et HBase pour parcourir des données volumineuses de génomique. Cette comparaison est utile, mais peut difficilement être réalisée avec une base commune au niveau du matériel. En effet, les bases de données relationnelles sont généralement utilisées avec une machine unique, alors que HBase est conçu pour fonctionner sur des clusters comportant plusieurs machines. Il existe des implémentations de bases de données relationnelles qui fonctionnent sur des clusters, mais leur mode de fonctionnement reste assez éloigné de celui de HBase.

Ce projet se base sur les acquis précédents, et notamment sur l'observation que le Big Data est adapté pour traiter un volume de données très grand, contrairement aux bases de données classiques. Les tests de performance visent donc maintenant à observer le comportement du système avec des cas proches de la réalité afin de mieux évaluer les besoins en termes de ressources nécessaires pour utiliser ce type de solution. La flexibilité apportée par le cloud computing permettra en outre de tester différentes configurations de clusters.

La principale métrique utilisée pour les tests de performance est la durée nécessaire pour réaliser un certain nombre d'actions. Les actions sont un ensemble de requêtes préparées avec l'API qui pourront être exécutées en utilisant plusieurs threads, c'est-à-dire qu'elles pourront être exécutées simultanément afin de tirer profit de la présence de plusieurs machines au niveau du cluster pour distribuer la charge parmi elles.

En pratique, un nouveau programme dédié a été développé avec le langage Java. Ce programme prend, en entrée, le nom de la table contenant les données, le nombre de requêtes à effectuer et le nombre de threads à utiliser. En sortie, le programme donnera la durée d'exécution de l'ensemble des requêtes.

```
hadoop jar perfs.jar <table> <requetes> <threads>
```

Les requêtes effectuées correspondent au principal cas d'utilisation, c'est-à-dire la récupération de variations avec une sélection d'échantillons. Ici, 10 échantillons aléatoires sont sélectionnés pour chaque requête, ainsi qu'un intervalle de 10 000 positions au sein d'un chromosome pris au hasard.

Les combinaisons requêtes/threads testées pour chaque cluster sont les suivantes :

Tableau 3.9 Combinaisons requêtes/threads testées

100 requêtes	**1000 requêtes**
1 thread	1 thread
2 threads	2 threads
5 threads	5 threads
10 threads	10 threads
50 threads	50 threads
100 threads	100 threads

Chaque combinaison a été exécutée 3 fois afin d'obtenir une valeur moyenne. Les valeurs relevées sont présentées dans l'ANNEXE IX. Ils sont commentés et analysés dans le prochain chapitre.

CHAPITRE 4

Résultats et analyses

4.1 Résultats de performance

Les tests de performance pour la solution proposée ont été réalisés sur plusieurs clusters avec le service Amazon EC2. La procédure suivie est celle décrite dans le chapitre 3.7.4. Les commandes utilisées pour préparer les clusters sont données en ANNEXE VII.

La seule différence entre chaque cluster est ici le nombre de nœuds présents. Chaque nœud correspond à une machine configurée avec les composants « DataNode », « TaskTracker » et « Region Server ». Un serveur maître est également utilisé en supplément. Les différentes configurations de cluster utilisées sont : 4 nœuds, 8 nœuds et 16 nœuds. Avec des instances EC2 ponctuelles, le coût de ces tests est de l'ordre de plusieurs dollars par heure, pour une durée de plusieurs heures en prenant en compte le temps de rapatriement des données depuis Amazon S3.

Lors des tests sur le cluster de 4 nœuds avec un grand nombre de threads (supérieur à 5 ou 10 suivant le nombre de requêtes par thread), certains « Region Server » se sont déconnectés, entrainant des temps d'exécution beaucoup plus longs ou bien un échec du test. Ce cluster a été relancé plusieurs fois afin de s'assurer qu'il ne s'agissait pas d'un problème matériel (chaque lancement étant normalement effectué sur des instances EC2 exécutées sur un serveur quelconque). Il est possible que ce problème soit dû à une mauvaise configuration du cluster ou de l'application pour des cas très gourmands en ressource. Ces résultats n'ont donc pas été inclus. L'ANNEXE IX présente tous les résultats obtenus.

La Figure 4.1 donne le temps d'exécution relevé pour chaque cluster avec les paramètres suivants : 1000 requêtes exécutées avec 5 threads.

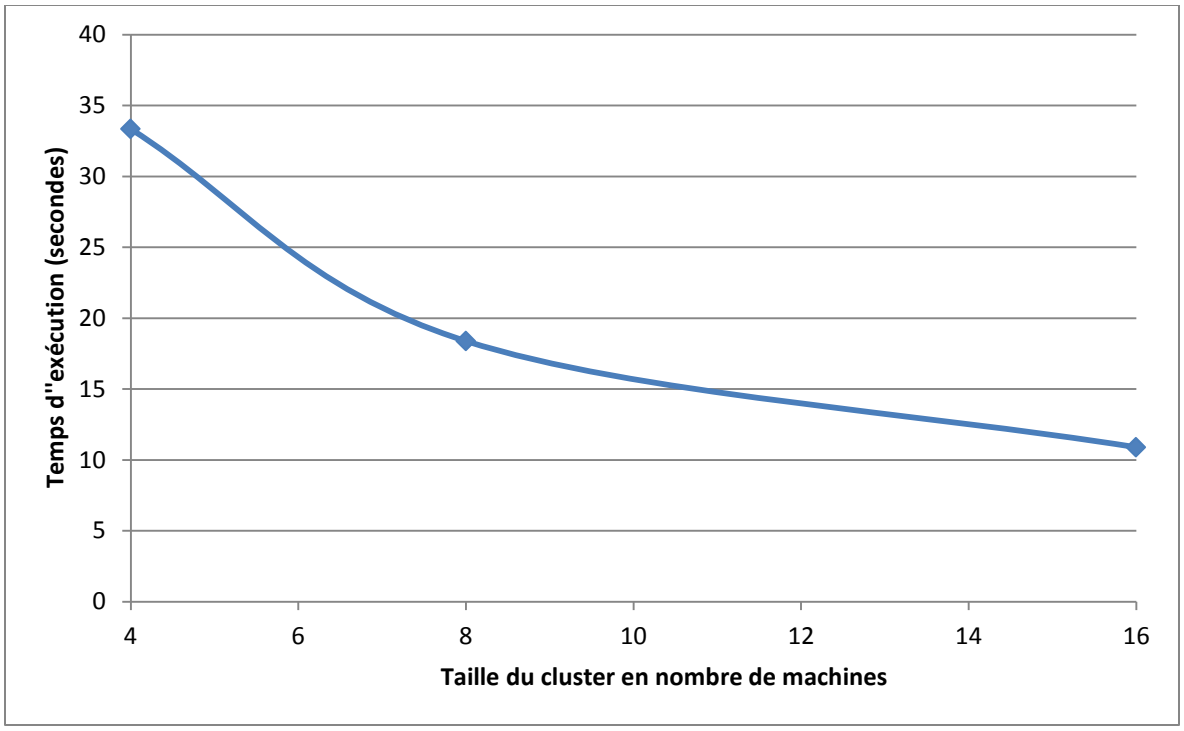


Figure 4.1 Temps d'exécution en fonction de la taille du cluster

On note un gain de performance avec l'utilisation de plus de nœuds, ce qui est le comportement attendu. La prochaine figure indique le temps d'exécution en fonction du nombre de threads pour les clusters à 8 nœuds et à 16 nœuds avec 1000 requêtes.

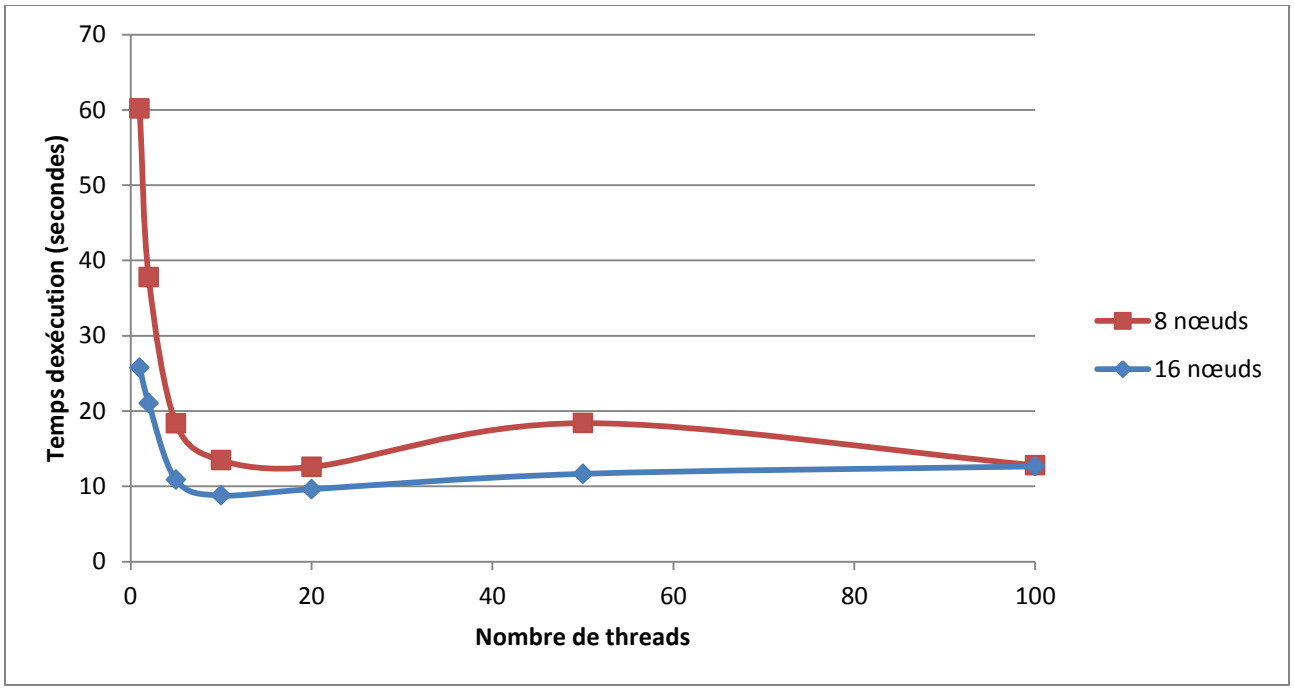


Figure 4.2 Temps d'exécution en fonction du nombre de threads

L'impact sur le nombre de threads est très marqué jusqu'à l'utilisation d'une dizaine de threads. Au-delà, il y a une légère perte de performance.

Les résultats obtenus sont cohérents par rapport à la taille du cluster utilisé, et on peut envisager que les ressources du cluster sont utilisées correctement.

Ces résultats donnent aussi un aperçu des performances attendues pour l'utilisation d'un cluster dans une situation réelle, et peuvent être utiles lors du choix de la quantité de ressource à accorder à un système de ce type. Cependant, dans le cadre d'une utilisation en production de cette solution, des questions supplémentaires doivent être adressées. Tout d'abord, l'espace disque disponible avec le cluster doit permettre de stocker toutes les données avec un niveau de réplication suffisant. Pour les tests effectués, l'espace disque disponible à partir de 2 nœuds était suffisant. Il faut notamment prendre en compte les situations où la quantité de données à stocker est variable afin de bien chiffrer les besoins.

Enfin, le choix d'un cluster devra prendre en compte le coût fonctionnel. Lors de ce projet, des instances ponctuelles ont été utilisées avec le service Amazon EC2. Ces instances présentent un intérêt au niveau des coûts, bien inférieurs aux autres types d'instance, mais l'inconvénient réside dans la faible disponibilité de ces instances, et le risque de voir l'arrêt du fonctionnement du cluster en cas de coûts excédants une certaine limite.

4.2 La conception d'une solution Big Data

Le CHAPITRE 2 a présenté la procédure suivie pour la conception d'une solution orientée Big Data pour le stockage de données génomique. Cette partie tente de reprendre les problèmes et les apprentissages abordés afin d'apporter des pistes et des conseils lors de la conception d'un système similaire. L'évaluation du besoin d'une solution Big Data ne sera pas abordée, et on considèrera que ce besoin a été identifié au préalable.

4.2.1 Type de projet : migration ou nouveau système

Un projet Big Data est généralement entrepris après avoir analysé les besoins pour une situation précise. Les ressources disponibles au départ peuvent varier, puisqu'il peut s'agir d'un nouveau projet ou bien de l'amélioration d'un système précédent. La nature de ces ressources a un impact sur la procédure de migration vers le nouveau système. Quelques questions utiles qui reviennent souvent à cette étape sont :

- Quel type de données veut-on analyser ?
- Comment se procurer ces données ?
- Existe-t-il un système déjà en place pour traiter ces données ?
- Comment interagir avec ce système ?
- Est-ce que le projet vise à remplacer ce système ?
- Est-ce que ce système va continuer à fonctionner ?
- Est-ce que la migration va être réalisée en une fois ou bien au cours d'une certaine période de temps ?

Lors de ce projet, l'ancien système a été vu comme une référence au niveau fonctionnel. Au niveau technique, la première approche a été d'utiliser ce système pour proposer une transition vers une solution Big Data. La dépendance avec l'ancien système n'avait d'ailleurs pas complètement disparu puisque la base de données originale était utilisée pour récupérer certaines informations. La seconde approche s'est focalisée sur la mise en place d'un système indépendant au niveau technique. Cela a permis de se passer des étapes intermédiaires, et d'apporter une plus grande liberté au niveau des choix technologiques, notamment pour la récupération des données à traiter.

Pour chaque projet, il pourra exister des liens différents avec un éventuel système précédent. Ce lien pourra amener à des configurations hybrides où le nouveau système est utilisé en complément de l'ancien, ou bien à une nouvelle configuration qui ne comprend que la nouvelle solution.

Souvent, l'importation de données sera effectuée à partir d'une base de données relationnelle, puisque ce type de base de données est très présent aujourd'hui. L'avantage apporté par ces bases de données est la standardisation de leur fonctionnement. Le langage SQL apporte une couche d'abstraction utile pour construire des outils qui ne dépendent pas du système sous-jacent. Cela a permis par exemple à l'utilitaire Sqoop de proposer une migration facile et rapide depuis ou vers tout type de base de données relationnelle avec la bibliothèque JDBC.

À l'inverse, la solution proposée ici se base sur l'utilisation de données brutes, importées vers HBase à l'aide d'une procédure plus complexe. Cette méthode présente l'avantage de profiter de la puissance du cluster pour démontrer l'efficacité du Big Data pour traiter de grands volumes de données. On est donc parti d'une approche rapide et facile à mettre en œuvre pour aller vers une approche orientée vers l'optimisation. En pratique, l'approche utilisée pour effectuer une migration de données va principalement dépendre de la disponibilité des données sources et des ressources disponibles pour mettre en place la procédure.

4.2.2 Conception du schéma

La conception du schéma de la base de données HBase est une étape importante puisque c'est ce même schéma qui va avoir un impact important sur le niveau de performance de la solution. Dès lors que celui-ci est assez différent d'un schéma relationnel classique, les questions à se poser ne seront pas les mêmes. Il s'agit ici d'obtenir des performances satisfaisantes. Avec les très grands volumes de données concernés, un schéma mal optimisé peut entrainer beaucoup de pertes, que ce soit au niveau du temps de traitement ou bien au niveau des ressources allouées.

Lors de ce projet, plusieurs points essentiels ont été abordés avant la conception du schéma. Tout d'abord, les spécificités techniques de la base de données choisie doivent être maîtrisées, sachant que ces technologies sont encore très récentes et qu'aucune base de données Big Data ne fonctionne de la même manière. La connaissance de ces spécificités permettra non seulement de valider le choix technologique d'une base de données en particulier, mais aussi de profiter de certaines caractéristiques qui pourraient améliorer ou faciliter les étapes suivantes.

Le point suivant est la définition des données, et notamment les types, les attributs et les relations entre les différentes informations. Enfin, les cas d'utilisation permettront de faciliter encore la modélisation. Ces cas d'utilisation sont à définir avec le client. Avec une base de données classique, la gestion des données est plus flexible avec notamment l'utilisation des jointures. Avec HBase, il faut penser dès le départ à stocker les données de telle sorte que les jointures ne soient jamais nécessaires puisqu'elles ne sont pas implémentées.

Lors de la modélisation du schéma de la base de données, plusieurs stratégies peuvent être mises en place. L'une d'entre elles est la manière de stocker les données : est-il préférable de limiter le nombre de colonnes afin d'avoir un maximum de lignes qui peuvent être facilement parcourues (modèle couramment appelé « Tall-narrow »), ou bien faut-il au contraire ajouter le maximum d'information au sein d'une ligne afin de maximiser la quantité d'information à

récupérer (modèle appelé « Flat-wide ») ? La bonne approche dépendra encore une fois de la définition des données et des cas d'utilisation. Avec HBase, la séparation des données est effectuée entre les lignes, donc si de nouvelles données sont fréquemment ajoutées à la base de données, il est préférable d'ajouter de nouvelles lignes au lieu d'ajouter des informations à des lignes existantes.

Au final, le schéma présenté ne prétend pas être la meilleure solution possible, mais les problèmes et réflexions abordés, soutenus par les tests réalisés, permettent de valider ce schéma comme apportant de nettes améliorations par rapport aux autres solutions proposées précédemment, et apportent des pistes pour la modélisation d'une solution de Big Data similaire.

4.3 La mise en production d'une solution Big Data

4.3.1 Infrastructure physique ou virtuelle

Le cloud computing est aujourd'hui un sujet omniprésent. Certains le considèrent comme une nouvelle ère dans l'informatique, alors que d'autres ne le considère que comme une tendance actuelle. Dans tous les cas, l'utilisation du cloud computing pour une solution Big Data est un sujet qui est intéressant de développer.

Comme présentée dans ce rapport, la mise en place de la solution proposée a utilisé pleinement les technologies du cloud computing. Celles-ci se prêtent particulièrement bien à plusieurs contraintes, comme la mise à disposition immédiate et l'expérimentation sur de courtes périodes.

Plus concrètement, un certain type d'instance a été utilisé avec Amazon EC2 pour réaliser les tests : les instances ponctuelles. Ces instances permettent d'utiliser les ressources disponibles dans un centre de données à un prix qui peut varier à tout moment. Ce prix est en général bien inférieur au prix des instances classiques. C'est à l'utilisateur de définir le prix

maximum qu'il est prêt à payer pour utiliser ces instances, et tant que celui-ci n'est pas dépassé, les instances continueront à être exécutée, avec une nouvelle évaluation toutes les heures. Ce mode de fonctionnement ne permet pas de garantir un bon niveau de service, mais il est bien adapté pour les systèmes en test, ou bien pour ceux qui n'ont pas besoin de fonctionner continuellement.

Il existe d'autres types d'instances avec Amazon EC2, par exemple les instances réservées. Il s'agit là d'effectuer un paiement à l'avance pour utiliser des instances dont le coût de fonctionnement sera alors réduit. Ces instances seront plus utilisées pour réduire les coûts d'un système qui doit être mis en production.

La variété des niveaux de service amène à se poser la question de l'utilisation d'une infrastructure classique ou bien d'une infrastructure dans les nuages. Voici les remarques qui découlent de l'usage observé pendant ce projet.

Tout d'abord, l'installation et la configuration d'un cluster avec une infrastructure classique nécessitent un minimum de moyens financiers et un personnel dédié. Les grandes entreprises ont souvent les ressources nécessaires pour cela, alors que les PME se tourneront souvent vers des prestataires externes. Au contraire, l'utilisation des services de cloud computing est souvent simplifiée et ne nécessite que peu de préparation.

Avec une infrastructure classique, un investissement de départ est effectué afin d'acquérir le matériel adéquat. Avec le cloud computing, aucun investissement matériel n'est requis puisque les services sont déjà en place. Cependant, les coûts d'utilisation d'une infrastructure classique sont plus transparents, alors que les coûts pour le cloud computing dépendent de l'utilisation faite et sont susceptibles de varier.

Le cloud computing apporte une grande souplesse au niveau de l'infrastructure informatique. Il est possible de changer la configuration des serveurs et d'augmenter ou de réduire les ressources avec une simple interface ou bien à travers une API. Cela a permis lors de ce

projet de tester différentes configuration de cluster en modifiant simplement un fichier de configuration. Avec une infrastructure physique, ces opérations sont plus complexes puisqu'elles nécessitent une intervention humaine. Le cloud a donc l'avantage de la souplesse.

En ce qui concerne la sécurité, il est souvent considéré que le modèle du cloud computing est moins adapté dès lors que l'on n'a plus un accès physique aux données. En réalité, la question de la sécurité est beaucoup plus nuancée puisque les fournisseurs de services sont bien souvent capables de fournir un niveau de sécurité équivalent si ce n'est meilleur que les services internes, cela va finalement dépendre de la politique mise en place.

L'un des principaux inconvénients du cloud computing est le coût, bien souvent supérieur à une infrastructure classique. En effet, le niveau de service plus « haut niveau » nécessite également plus d'investissement.

En conclusion, aucune des deux approches n'est meilleure que l'autre, puisqu'elles ne sont pas adaptées aux mêmes cas. Pour une application qui doit être déployée en production et dont la taille est bien définie, il est plutôt recommandé d'utiliser une infrastructure classique dont les coûts seront rapidement amortis. Pour une application temporaire, qu'il s'agisse de tests ou bien d'un traitement de courte durée, ou encore pour les applications qui ont besoin de ressources très rapidement ou en constante évolution, le modèle du cloud computing est parfaitement adapté.

4.3.2 Configuration du cluster

La configuration d'un cluster Hadoop peut être longue et complexe dès lors que l'on recherche une optimisation poussée en modifiant un grand nombre de paramètres. L'entreprise Cloudera propose des distributions de l'écosystème Hadoop qui permettent de simplifier ces étapes. Avec Apache Whirr ou Elastic MapReduce, l'installation et la configuration sont presque entièrement automatisées. La question qui se pose alors est : est-il

nécessaire d'effectuer une configuration manuelle ou bien les gains par rapport à une configuration automatique ne sont pas significatifs ? Lors de ce projet, l'une des étapes préliminaires a consisté à installer et configurer un cluster pour réaliser des tests fonctionnels, avant de se tourner vers des outils d'automatisation. Ainsi, certains commentaires peuvent être donnés pour ces deux approches.

La configuration manuelle de Hadoop permet une meilleure compréhension de son fonctionnement. C'est encore une étape recommandée pour le développement d'une application avec Hadoop puisque le système est encore jeune et n'est pas à l'abri de problèmes lors du déploiement. La documentation de Hadoop apporte beaucoup d'informations sur les différents paramètres et leurs effets. En pratique, il existe de très nombreuses options, et les paramètres par défaut ont généralement été utilisés. Ils sont adaptés à un grand nombre de cas.

L'impact réel de la configuration de Hadoop sur les performances n'a pas été mesuré pour ce projet. Le cluster physique mis à disposition n'avait pas une configuration suffisante pour obtenir des résultats intéressants, puisque l'espace disque disponible ne permettait de charger qu'une fraction des données. En ce qui concerne les clusters utilisés dans le « cloud », la plupart des paramètres par défaut ont été repris avec les outils d'automatisation, et c'est seulement en variant le nombre d'instances qu'une comparaison de performance a été effectuée.

L'une des raisons de cette restriction est la difficulté d'établir une configuration optimale pour plusieurs types de clusters. Certains articles apportent plus de détails sur ce point (Kambatla, Pathak et Pucha, 2009), et la principale source de ce problème est une gestion des ressources différentes entre des clusters composés d'un nombre différent de nœuds. Le type d'application et ses principales tâches vont également avoir un impact sur l'efficacité de certains paramètres.

En résumé, le paramétrage de Hadoop est lié à la configuration matérielle et à l'application déployée. Il est donc effectué lors de la mise en production. Il pourra permettre d'améliorer les performances dans le cas où toutes les caractéristiques du système sont connues. Pour ce projet, l'accent a été mis sur la simplicité de mise en œuvre, et sur l'adaptation à différentes configurations, en profitant des paramètres par défaut proposés par les outils utilisés. Cela apporte un gain de temps appréciable et facilite grandement l'étape de mise en production, notamment pour le déploiement dans le cloud computing. Si une solution similaire est déployée sur une configuration fixe, il peut être intéressant d'évaluer l'impact de certains paramètres sur les performances.

Enfin, il est intéressant de souligner qu'avec la démocratisation des technologies Big Data, la configuration des outils va être toujours simplifiée, et des services actuellement très utilisés tels que Amazon Elastic Mapreduce ne proposent déjà que très peu de personnalisation au niveau du système Hadoop.

4.3.3 Autres points à considérer

D'autres points sont à considérer pour la mise en production d'une solution de ce type sur un cluster, et certains d'entre eux sont brièvement présentés ci-dessous.

Comme toute base de données, il peut être nécessaire de fixer un certain niveau de sécurité pour les données stockées avec HBase. Il dépend principalement de trois critères : la confidentialité, l'intégrité et la disponibilité (Perrin, 2008).

En ce qui concerne la confidentialité, il s'agit d'une part de restreindre l'accès au cluster, et d'autres part de développer des mécanismes sécurisés pour l'accès à la base de données à travers les applications qui y accèdent. Enfin, le chiffrement des données dans HBase peut être envisagé au niveau applicatif, HBase n'intégrant pas directement cette fonction.

L’intégrité des données est gérée au niveau de HDFS puisqu’une somme de contrôle est calculée et stockée pour chaque bloc de données. Cette somme de contrôle est fréquemment vérifiée. De plus, la réplication permet de s’assurer que les données persistent suite à une panne d’un ou plusieurs nœuds.

La disponibilité des données va tout d’abord dépendre de la disponibilité du cluster. Au niveau de Hadoop, la réplication des données améliore la résistance aux pannes et par conséquent améliore la disponibilité au sein du cluster. Enfin, il peut être nécessaire de mettre en place une politique de sauvegarde.

CONCLUSION

Les technologies de Big Data sont aujourd'hui en plein essor. Dans les prochaines années, ces technologies seront de plus en plus utilisées pour répondre à de nouvelles problématiques pour la gestion de données. Avec ce projet, une solution basée sur le Big Data a été proposée à partir d'une problématique réelle pour l'accès à des données de génomiques.

La mise en place de la solution en utilisant des services de cloud computing a également permis de démontrer les capacités de ce modèle qui prend de plus en plus de place dans l'informatique aujourd'hui. Une installation sur un cluster classique reste possible, et a été réalisée pour conduire des tests sur des échantillons de données.

Au final, 4 grands domaines ont été étudiés et mis en jeu pour la mise en place de la solution :

- La génomique
- Le Big Data
- La programmation avec le langage Java et les technologies associées, comme les servlets
- Le cloud computing

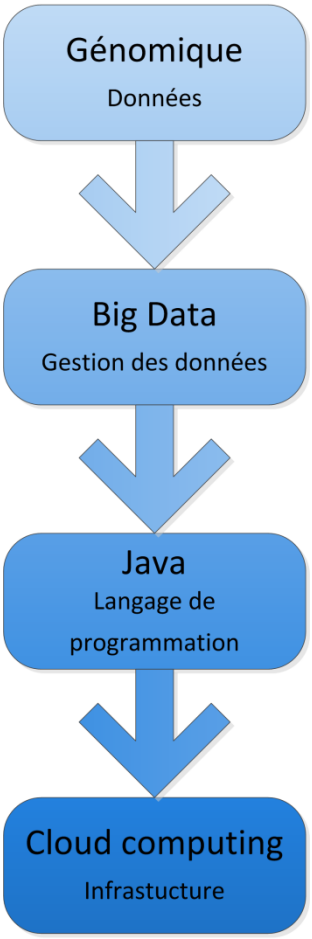


Figure 4.3 Domaines mis en relation pour le projet

Le prototype proposé n'a pas la vocation d'être déployé dans son état actuel par un laboratoire tel que le laboratoire Rouleau. Il propose seulement une base de connaissance pour répondre à des besoins similaires, cela à partir d'une problématique bien réelle. Il permet également de démontrer l'utilité du Big Data dans les domaines scientifiques.

RECOMMANDATIONS

Ce projet ne propose pas une solution finale, et peut être amélioré sur différents points. Tout d'abord, les cas d'utilisation peuvent être revus à nouveau avec un laboratoire de recherche. Ici, ils se sont basés sur les cas d'utilisation d'une application exploitant une base de données relationnelle classique. Avec les progrès réalisés chaque année dans le domaine de la génomique, de nouveau cas d'utilisation de ces données sont susceptibles d'apparaitre.

Ensuite, l'optimisation d'un cluster Hadoop/Hbase est un travail à part entière. De nombreux paramètres peuvent avoir une influence plus ou moins importante sur les performances, que ce soit pour une utilisation sur une infrastructure classique ou avec le cloud computing.

Enfin, bien que Hadoop et HBase soient aujourd'hui très utilisés, il existe d'autres solutions de Big Data. Les bases de données Big Data s'intégrant avec le langage SQL prennent notamment de plus en plus d'importance, et pourraient permettre de proposer un moyen plus standard de contrôler ces bases de données. On peut envisager ce type de technologie pour ce projet afin d'améliorer l'intégration avec des applications conçues pour effectuer des requêtes SQL.

ANNEXE I

Le format de fichier VCF (Variant Call Format)

VCF est un format de fichier proposé par le projet 1000 Genomes. Il est conçu pour stocker les informations sur les variations génétiques, tout en intégrant des résultats d'analyse pour des échantillons. Un tel format a l'avantage de faciliter l'échange et le traitement de données.

Un fichier VCF est composé de deux parties : l'en-tête et le corps du fichier. L'en-tête contient des lignes décrivant les méta-informations présentes. Celles-ci apportent des informations supplémentaires sur les variations et leur détection. La dernière ligne de l'en-tête décrit les différentes colonnes du corps, séparées par une tabulation. On retrouve notamment l'identifiant du chromosome, la position de la variation au sein de ce chromosome, un numéro d'identifiant pour la variation, le génotype de référence, le ou les génotype(s) alternatif(s) etc.

Le corps du fichier contient les informations. Chaque ligne correspond à une variation, avec en début de ligne les informations sur celle-ci, puis le résultat de la détection de la variation pour chaque échantillon présent.

L'exemple suivant est tiré de la documentation du projet 1000 Genomes pour le format VCF (VCF (Variant Call Format) version 4.0 | 1000 Genomes).

```
##fileformat=VCFv4.0
##fileDate=20090805
##source=myImputationProgramV3.1
##reference=1000GenomesPilot-NCBI36
##phasing=partial
##INFO=<ID=NS,Number=1,Type=Integer,Description="Number of Samples With
Data">
##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth">
##INFO=<ID=AF,Number=.,Type=Float,Description="Allele Frequency">
##INFO=<ID=AA,Number=1,Type=String,Description="Ancestral Allele">
##INFO=<ID=DB,Number=0,Type=Flag,Description="dbSNP membership, build 129">
##INFO=<ID=H2,Number=0,Type=Flag,Description="HapMap2 membership">
##FILTER=<ID=q10,Description="Quality below 10">
##FILTER=<ID=s50,Description="Less than 50% of samples have data">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
##FORMAT=<ID=HQ,Number=2,Type=Integer,Description="Haplotype Quality">
#CHROM POS     ID        REF ALT    QUAL FILTER INFO
FORMAT      NA00001        NA00002        NA00003
20     14370   rs6054257 G      A       29   PASS   NS=3;DP=14;AF=0.5;DB;H2
GT:GQ:DP:HQ 0|0:48:1:51,51 1|0:48:8:51,51 1/1:43:5:.,.
20     17330   .         T      A       3    q10    NS=3;DP=11;AF=0.017
GT:GQ:DP:HQ 0|0:49:3:58,50 0|1:3:5:65,3   0/0:41:3
20     1110696 rs6040355 A      G,T     67   PASS
NS=2;DP=10;AF=0.333,0.667;AA=T;DB GT:GQ:DP:HQ 1|2:21:6:23,27 2|1:2:0:18,2
2/2:35:4
20     1230237 .         T      .       47   PASS   NS=3;DP=13;AA=T
GT:GQ:DP:HQ 0|0:54:7:56,60 0|0:48:4:51,51 0/0:61:2
20     1234567 microsat1 GTCT   G,GTACT 50   PASS   NS=3;DP=9;AA=G
GT:GQ:DP    0/1:35:4       0/2:17:2       1/1:40:3
```

Figure-A I-I Exemple de fichier VCF

ANNEXE II

Interfaces de l'API

Les interfaces permettent de traduire les cas d'utilisation identifiés en fonctions Java, et de proposer un prototype à implémenter pour les classes usuelles. 3 interfaces ont été mises en place.

Tableau-A II-1 Interface IRequestVariant

IRequestVariant
`Variant getVariantInfos(Variant variant);`

Tableau-A II-2 Interface IRequestVariantResult

IRequestVariantResult
`List<VariantResult> getVariantResults(List<Variant> variants, List<Sample> include, List<Sample> exclude, int limit, int offset);`
`List<VariantResult> getVariantResults(VariantPosition position1, VariantPosition position2, List<String> types, List<Sample> include, List<Sample> exclude, int limit, int offset);`

Tableau-A II-3 Interface IRequestSampleData

IRequestSampleData
`List<SampleData> getSampleData(Variant variant, List<Sample> include, List<Sample> exclude);`
`List<SampleData> getSampleData(List<Variant> variants, Sample sample, Boolean variantOnly, int limit, int offset);`
`List<SampleData> getSampleData(VariantPosition position1, VariantPosition position2, List<String> types, Sample sample, boolean variantOnly, int limit, int offset);`

ANNEXE III

Diagrammes UML de l'API

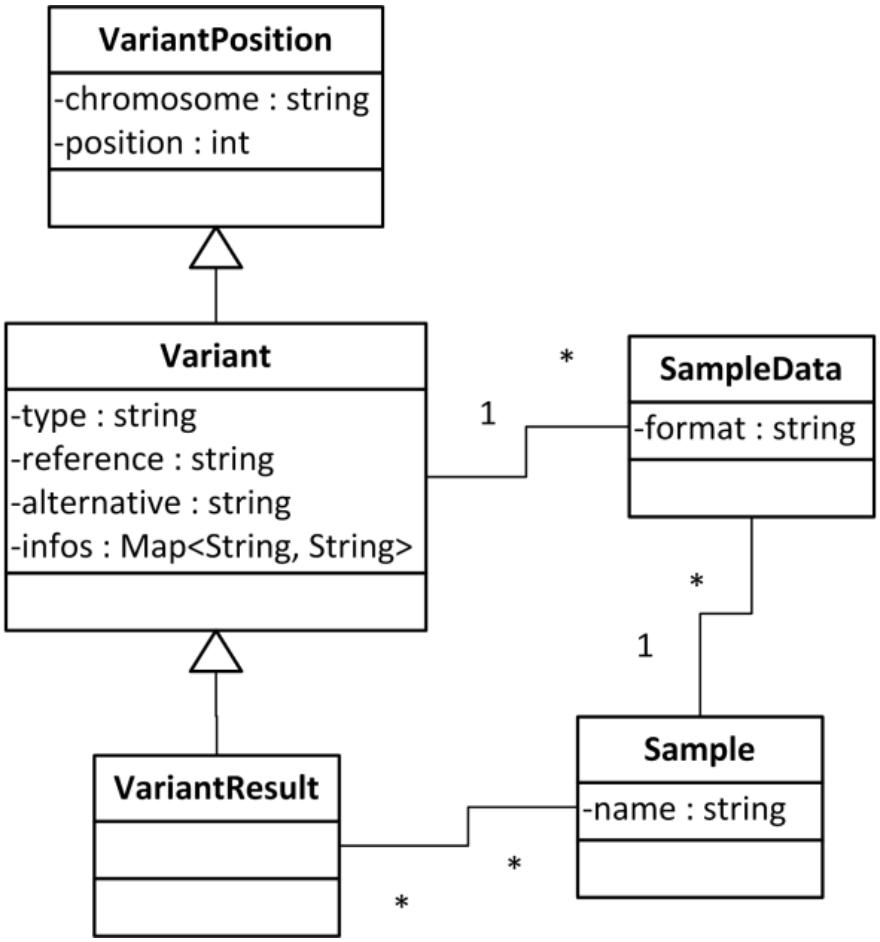


Figure-A III-I Diagramme UML pour les classes de données

La classe VariantPosition contient seulement l'information nécessaire pour localiser une variation au sein du génome. Elle est utilisée lorsque l'on souhaite définir des limites pour le parcours de données. La classe Variant hérite de VariantPosition, et comprend toutes les informations sur la variation. Cela comprend les annotations optionnelles qui sont listées dans l'objet infos, qui associe une valeur à un nom de propriété. La classe VariantResult permet de stocker ces mêmes informations ainsi qu'une liste d'échantillons associés.

La classe Sample représente un échantillon. La classe SampleData est liée à un échantillon et à une variation. Elle contient également l'information sur la présence ou non de la variation pour l'échantillon.

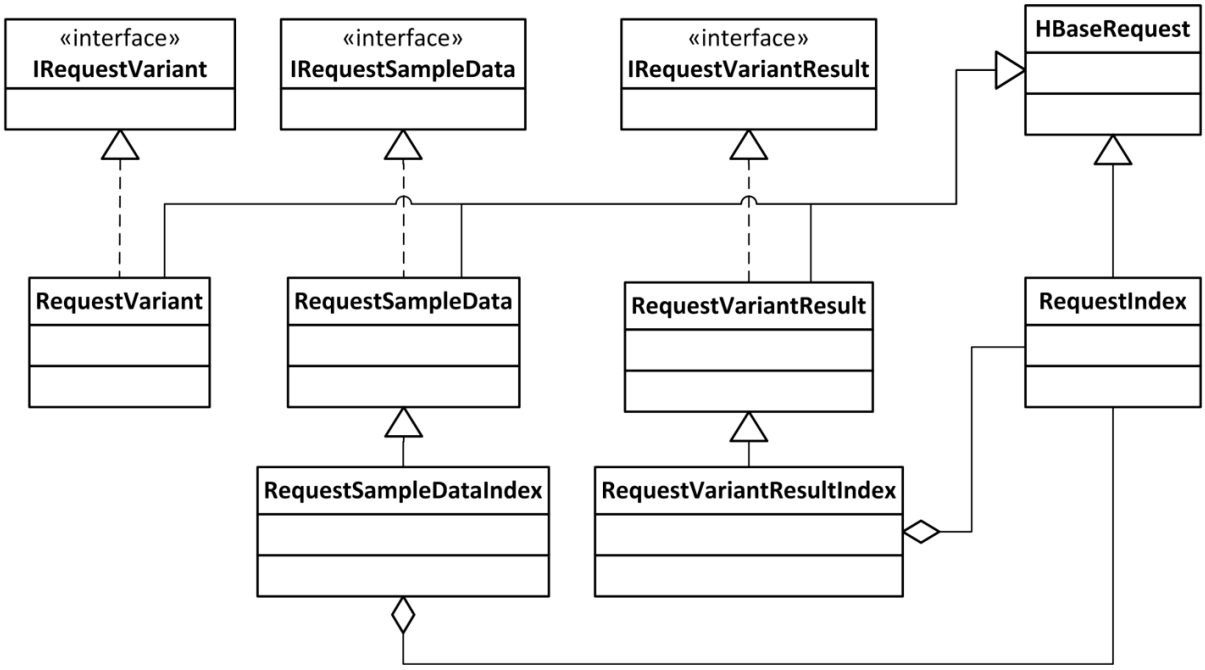


Figure-A III-II Diagramme UML simplifié pour les classes de gestion des requêtes

Chaque interface est implémentée par une classe qui dérive de la classe HBaseRequest. Celle-ci comprend des méthodes pour faciliter la connexion à la base de données. De nouvelles classes ont aussi été définies pour la prise en charge de la table d'index. Elles utilisent la classe RequestIndex qui elle-même utilise cette table d'index.

ANNEXE IV

Description des servlets

Les attributs en gras sont requis, les autres attributs sont optionnels.

Tableau-A IV-1 Description des servlets

Nom du servlet	Attribut	Type	Description
VariantResult	**chr**	string	Chromosome
	start	integer	Position de départ
	end	integer	Position de fin
	type	string	Filtrer les types de variation (défaut : aucun)
	sample	string	Filtrer les échantillons à inclure (défaut : aucun)
	exsample	string	Filtrer les échantillons à exclure (défaut : aucun)
	union	boolean	Variations présentes avec tous les échantillons sélectionnées (défaut : false)
	limit	integer	Nombre maximum de résultats (défaut : 100)
	page	integer	Accéder à une page de résultat (défaut : 1)
	gui	boolean	Utiliser l'interface web (défaut : false)
VariantInfos	**chr**	string	Chromosome
	pos	integer	Position
	type	string	Type
	gui	boolean	Utiliser l'interface web (défaut : false)
SampleInfos	**sample**	string	Échantillon
	chr	string	Chromosome
	start	integer	Position de départ
	stop	integer	Position de fin

	type	string	Filtrer les types de variation (défaut : aucun)
	allvariants	boolean	Affichage de toutes les variations (défaut : false)
	limit	integer	Nombre maximum de résultats (défaut : 100)
	page	integer	Accéder à une page (défaut : 1)
	gui	boolean	Utiliser l'interface web (défaut : false)

ANNEXE V

Extrait du fichier de configuration pour Spring

```
    <bean id="TableManager" class="gvdb.api.hbase.TableManager">
      <property name="conf">
          <value>/home/hadoop/hbase-site.xml</value>
      </property>
    </bean>

    <bean id="RequestIndex" class="gvdb.api.request.impl.RequestIndex"
scope="prototype">
       <property name="tableManager">
           <ref bean="TableManager" />
       </property>
       <property name="table">
           <value>genomic-variants-index</value>
       </property>
       <property name="indexTypes">
           <list>
              <value>SV</value>
           </list>
       </property>
    </bean>
    <bean id="RequestVariantResult"
class="gvdb.api.request.impl.RequestVariantResultIndex" scope="prototype">
       <property name="tableManager">
           <ref bean="TableManager" />
       </property>
       <property name="requestIndex">
           <ref bean="RequestIndex" />
       </property>
       <property name="table">
           <value>genomic-variants</value>
       </property>
       <property name="cache">
           <value>100</value>
       </property>
    </bean>
    <bean id="RequestSampleData"
class="gvdb.api.request.impl.RequestSampleDataIndex" scope="prototype">
       <property name="tableManager">
           <ref bean="TableManager" />
       </property>
       <property name="requestIndex">
           <ref bean="RequestIndex" />
       </property>
       <property name="table">
           <value>genomic-variants</value>
       </property>
       <property name="cache">
           <value>100</value>
       </property>
    </bean>
```

ANNEXE VI

Commandes utilisées pour la génération des fichiers « HFile »

```
S3_BUCKET="genomic-chrall"
LIBJARS=lib/guava*

hadoop distcp –i s3n://$S3_BUCKET/archives/ import/

hadoop jar import.jar gvdb.importation.CreateRegions 100000 import/ regions
hadoop jar import.jar gvdb.importation.CreateTable –r regions genomic-variants variant sample nosample
hadoop jar import.jar gvdb.importation.CreateTable genomic-variants-index key
hadoop jar import.jar gvdb.importation.CreateHFiles genomic-variants import/hfiles/ -libjars $LIBJARS
hadoop jar import.jar gvdb.importation.CreateHFilesIndex genomic-variants-index import/ hfiles-index/ -libjars $LIBJARS

hadoop fs -cp regions s3n://$S3_BUCKET/regions
hadoop distcp –i /user/hadoop/hfiles/ s3n://$S3_BUCKET/hfiles/
hadoop distcp –i /user/hadoop/hfiles-index/ s3n://$S3_BUCKET/hfiles-index/
```

ANNEXE VII

Commandes utilisées pour préparer un cluster

```
S3_BUCKET="genomic-chrall"

hadoop fs -cp s3n://$S3_BUCKET/regions ./regions
hadoop distcp s3n://$S3_BUCKET/hfiles/variant/ hfiles/variant/
hadoop distcp s3n://$S3_BUCKET/hfiles/sample/ hfiles/sample/
hadoop distcp s3n://S3_BUCKET/hfiles/nosample/ hfiles/nosample/
hadoop distcp s3n://S3_BUCKET/hfiles-index/key/ hfiles-index/key/

HADOOP_CLASSPATH=$(hbase classpath) hadoop -jar import.jar
gvdb.importation.CreateTable –r regions genomic-variants variant
sample nosample

HADOOP_CLASSPATH=$(hbase classpath) hadoop -jar import.jar
gvdb.importation.CreateTable –r regions genomic-variants-index key

hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles
hfiles/ genomic-variants

hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles
hfiles-index/ genomic-variants-index

sudo apt-get install tomcat7 -y
sudo cp webapp.war /var/lib/tomcat7/webapps/
sudo /etc/init.d/tomcat7 restart
```

ANNEXE VIII

Configuration typique d'Apache Whirr

```
whirr.cluster-name=hbasecluster # Nom du cluster

# 1 serveur principal et 16 noeuds
whirr.instance-templates=1 zookeeper+hadoop-namenode+hadoop-
jobtracker+hbase-master, 16 hadoop-datanode+hadoop-tasktracker+hbase-
regionserver

# Configuration des instances EC2
whirr.provider=aws-ec2
whirr.location-id=us-east-1b
whirr.image-id=us-east-1/ami-bffa6fd6
whirr.hardware-id=m1.large
whirr.aws-ec2-spot-price=0.100 # Utilisation d'instances ponctuelles
whirr.private-key-file=${sys:user.home}/.ssh/id_rsa_whirr
whirr.public-key-file=${sys:user.home}/.ssh/id_rsa_whirr.pub

# Spécifie la version de Hadoop et HBase
whirr.hadoop.version=1.0.4
whirr.hbase.version=0.94.5
whirr.hadoop.tarball.url=http://apache.osuosl.org/hadoop/common/hadoop-
      ${whirr.hadoop.version}/hadoop-${whirr.hadoop.version}.tar.gz

whirr.hbase.tarball.url=http://archive.apache.org/dist/hbase/hbase-
      ${whirr.hbase.version}/hbase-${whirr.hbase.version}.tar.gz
```

ANNEXE IX

Résultats des tests de performance

Taille du cluster	Nombre de requêtes	Nombre de threads	Temps d'exécution (en secondes)
4 nœuds	100	1	9,12
		2	5,72
		5	3,66
		10	3,19
		20	/
		50	/
		100	/
	1000	1	90,53
		2	57,35
		5	66,68
		10	/
		20	/
		50	/
		100	/
8 nœuds	100	1	7,38
		2	4,08
		5	3,13
		10	3,76
		20	2,89
		50	3,51
		100	3,85
	1000	1	60,18
		2	37,77
		5	18,38
		10	13,46
		20	12,57
		50	18,39
		100	12,8
16 nœuds	100	1	4,95
		2	1,81

		5	0,76
		10	2,45
		20	2,45
		50	2,85
		100	2,56
	1000	1	25,75
		2	21,06
		5	10,89
		10	8,78
		20	9,62
		50	11,66
		100	12,65

LISTE DE RÉFÉRENCES BIBLIOGRAPHIQUES

Altshuler, David Matthew, Eric Steven Lander, Lauren Ambrogio, Toby Bloom, Kristian Cibulskis, Tim J Fennell, Stacey B Gabriel, David B Jaffe, Erica Shefler et Carrie L Sougnez. 2010. « A map of human genome variation from population scale sequencing ». *Nature*, vol. 467, n° 7319, p. 1061-1073.

Borthakur, Dhruba, Jonathan Gray, Joydeep Sen Sarma, Kannan Muthukkaruppan, Nicolas Spiegelberg, Hairong Kuang, Karthik Ranganathan, Dmytro Molkov, Aravind Menon et Samuel Rash. 2011. « Apache Hadoop goes realtime at Facebook ». In *Proceedings of the 2011 international conference on Management of data*. p. 1071-1080. ACM.

Chang, Fay, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes et Robert E. Gruber. 2008. « Bigtable: A Distributed Storage System for Structured Data ». *ACM Trans. Comput. Syst.*, vol. 26, n° 2, p. 1-26.

George, Lars. 2011. *HBase : The Definitive Guide*, 1st. Sebastopol, CA: O'Reilly Media, xxviii, 522 p. p.

« Global Infrastructure ». < http://aws.amazon.com/fr/about-aws/globalinfrastructure/ >.

Hamelin, Jean-Francois, et Jean-Philippe Bond. 2012. *Base de données distribuée appliquée à la génétique dans le cadre de l'analyse du séquençage génomique - Rapport d'équipe*. Montréal, QC.

Initial sequencing and analysis of the human genome. 2001. « ». *Nature*, vol. 409, n° 6822, p. 860-921.

Kambatla, Karthik, Abhinav Pathak et Himabindu Pucha. 2009. « Towards optimizing hadoop provisioning in the cloud ». In *Proc. of the First Workshop on Hot Topics in Cloud Computing*.

Khetrapal, Ankur, et Vinay Ganesh. 2006. « HBase and Hypertable for large scale distributed storage systems ». *Dept. of Computer Science, Purdue University*.

Klos, Anna. 2012. *Optimisation de recherche grâce à HBase sous Hadoop*. Montréal, QC.

Lakshman, Avinash, et Prashant Malik. 2010. « Cassandra—A decentralized structured storage system ». *Operating systems review*, vol. 44, n° 2, p. 35.

Lu, Huang, Chen Hai-Shan et Hu Ting-Ting. 2012. « Research on Hadoop Cloud Computing Model and its Applications ». In *Networking and Distributed Computing (ICNDC), 2012 Third International Conference on*. p. 59-63. IEEE.

O'Connor, Brian, Barry Merriman et Stanley Nelson. 2010. « SeqWare Query Engine: storing and searching sequence data in the cloud ». *BMC Bioinformatics*, vol. 11, n^{o} Suppl 12, p. S2.

Perrin, Chad. 2008. « The CIA Triad ». *Dostopno na: http://www. techrepublic. com/blog/security/the-cia-triad/488*.

Taylor, Ronald. 2010. « An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics ». *BMC Bioinformatics*, vol. 11, n^{o} Suppl 12, p. S1.

« VCF (Variant Call Format) version 4.0 | 1000 Genomes ». < http://www.1000genomes.org/node/101 >.