Le génie pour l'industrie

RAPPORT TECHNIQUE
PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
DANS LE CADRE DU COURS LOG792

CONCEPTION D'UNE BASE DE DONNÉES DE GÉNOMIQUE EN HBASE

MICHAEL RUDEEN
RUDM13039006

DÉPARTEMENT DE GÉNIE LOGICIEL ET DES TI

Professeur-superviseur

Alain April

MONTRÉAL, 23 AVRIL 2014
HIVER 2014

REMERCIEMENTS

J’aimerais d’abord remercier le professeur Alain April pour son encadrement, son aide et ses conseils au long de ce projet. Son implication dans le domaine des technologies de la santé a permis de démarrer ce projet et sa confiance en ses étudiants en a mené à sa réalisation.

Je voudrais également souligner le travail inestimable de David Lauzon, étudiant à la maîtrise, pour les nombreuses heures qu’il a consacrées à la réalisation de ce projet, entre autres au niveau de la coordination des équipes, de l’expertise qu’il y a amené et de ses contributions directes. Un grand merci.

Ensuite, je remercie Patrice Dion pour son support au niveau de l’équipement et son suivi lors de l’installation de celui-ci.

J’aimerais remercier aussi Cédric Julien du CHUSJ pour son temps et son expertise qui ont permis de bien comprendre les processus d’affaires et donc de mieux cibler les exigences du projet.

Finalement, ce projet a grandement bénéficié du travail de Sébastien Servoles, ancien étudiant à la maîtrise qui a réalisé un projet similaire et dont la thèse a permis de répliquer les optimisations de séparation des familles de colonnes d’échantillons, ainsi que de création d’index secondaires (chapitre 4).

CONCEPTION D'UNE BASE DE DONNÉES DE GÉNOMIQUE EN HBASE

MICHAEL RUDEEN
RUDM13039006

RÉSUMÉ

Le rapport présent cherche à présenter la conception d'une base de données de génomique en HBase. En fait, le Centre hospitalier universitaire Sainte-Justine désire faire des recherches sur les variations génétiques potentiellement responsables pour la scoliose, mais leurs technologies sont désuètes et limitent leurs capacités. Le mandat pour ce projet a donc été d'établir une preuve de concept pour une base de données de génomique répondant aux exigences du laboratoire de recherche. Ce rapport s'adresse à tout individu cherchant à établir une base de données de génomique, ainsi qu'à toute personne s'impliquant dans les itérations futures de ce projet.

Le CHUSJ utilise présentement des fichiers Excel afin de transmettre leurs données et en faire l'analyse, mais ceci s'avère un format peu propice en raison du nombre volumineux de données. L'objectif de la base de données conçue serait donc de faciliter le travail des chercheurs en améliorant la vitesse de lecture et de recherche de données, en rendant le système flexible et facilement extensible et en garantissant la traçabilité de l'information.

La solution préconisée pour ce problème est une base de données en HBase optimisée pour les cas d'utilisation de l'équipe de recherche, complétée par une API faisant abstraction des mécanismes de la base de données. Cette solution répond à tous les critères élicitées plus haut et garantit la disponibilité du système.

Il reste beaucoup de travail à faire pour les itérations futures de ce projet, tel que des tests de performance pour des plus grands volumes de données et la mise en commun des données de plusieurs centres de recherche, mais cette première preuve de concept démontre la faisabilité du projet.

TABLE DES MATIÈRES

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

ADN : Acide désoxyribonucléique
CHUSJ : Centre hospitalier universitaire Sainte-Justine
ÉTS : École de technologie supérieure
GFS : Google File System
HDFS : Hadoop Distributed File System
JSON : JavaScript Object Notation
TSV: Tab-Separated Values
VCF : Variant Call Format
WAL : Write-Ahead Log
YARN : Yet Another Resource Negotiator

INTRODUCTION

Depuis plusieurs années, l'informatique de la santé est un domaine qui gagne de l'ampleur au sein des sociétés développées. De plus en plus, les systèmes logiciels permettent aux professionnels de la santé de pousser leur recherche et d'offrir des services mieux adaptés à leurs patients. Par contre, avec le temps, ces systèmes doivent être maintenus et améliorés afin de continuer de répondre aux besoins des utilisateurs.

Au Centre hospitalier universitaire Sainte-Justine (CHUSJ), des chercheurs tentent de trouver les causes génétiques de la scoliose. Pour ce faire, ils doivent analyser de grands volumes de données et leur solution actuelle, la transmission de fichiers Excel, devient de plus en plus inadéquate et limite la portée de leur recherche. Ils désirent donc la conception d'une base de données pour les assister.

Pour ce faire, les exigences du laboratoire de recherche ont été analysées et il a été décidé d'utiliser la plateforme Hadoop avec HBase afin de réaliser ce projet. Ce projet a donc comme objectif la conception d'une base de données de génomique en HBase qui contribuerait à la recherche des variations génétiques pouvant expliquer la maladie de la scoliose. En premier lieu, une mise en contexte est faite par l'entremise d'une introduction à la génomique et aux besoins spécifiques du laboratoire du CHUSJ. Ensuite, l'analyse des exigences mène à un choix et une justification des technologies utilisées. Les données et la planification du projet sont présentés, puis le modèle de données est détaillé. Enfin, les étapes nécessaires au déploiement de la solution sont expliquées.

CHAPITRE 1

CONTEXTE DU PROJET

1.1 Introduction à la génomique

La génomique est une discipline de la biologie qui a pour but l'étude du génome, c'est-à-dire l'ensemble des gènes qui composent tout être vivant. Les génomes varient d'une espèce à une autre et les variations génétiques à l'intérieur d'une même espèce nous permettent d'en distinguer les individus. En fait, le génome, qui chez l'humain est encodé sous la forme de la fameuse molécule en forme de double-hélice, l'ADN, est composé d'une multitude de chromosomes (46 chez l'être humain, formant 23 pairs), qui à leur tour sont composés de plusieurs gènes. On estime actuellement que chaque être humain compte environ 20 000 gènes et la composition et l'arrangement de ceux-ci expliquent la majorité des caractéristiques biologiques d'un individu, comme le type sanguin, la couleur des cheveux, le risque d'attraper certaines maladies, etc.

L'étude du génome se fait en trois grandes étapes. D'abord, le génome doit être séquencé. À cette étape, des échantillons d'ADN d'un individu sont lus par un analyseur génétique en segments, ces segments sont réassemblés afin de former le génome complet, puis les différents gènes sont identifiés et annotés. Une fois le génome reconstruit, il est sauvegardé, puis subit différentes transformations afin de réduire la taille des données et en faciliter l'analyse. À titre de référence, le génome complet d'un individu pré-traitement est de l'ordre de 10 à 100 Go (gigaoctets) dépendamment de la méthode de séquençage utilisée et de la précision de lecture voulue, alors que le fichier final, sous le format VCF (*variant call format*), est d'environ 100 Mo (mégaoctets). Enfin, plusieurs lectures (*samples*) de génomes sont assemblées afin d'en faire l'analyse, habituellement en faisant des comparaisons statistiques sur les génomes recueillis de plusieurs individus ou espèces.

Cette analyse de la génomique a plusieurs utilités. La liste suivante en énumère les principales :

- Étudier l'évolution génétique d'une espèce : on cherche à comprendre quelles mutations dans les gènes d'une espèce ont, avec le temps, contribué à son évolution.
- Analyser l'interaction génétique à l'intérieur d'un écosystème : on veut découvrir comment les différentes espèces d'un écosystème (par exemple, l'Amazonie) transmettent leur information génétique entre-elles.
- Identifier les fonctions des gènes et des variations possibles : on tente de faire un mappage entre les gènes composant un être et les caractéristiques biologiques qu'ils affectent, ainsi que de déterminer les effets des variations de chaque gène.
- Comprendre le comportement des maladies génétiques : on cherche à établir une relation entre la susceptibilité d'un individu à une maladie et les variations génétiques qu'il possède.

Pour ce projet, c'est une instance spécifique de ce dernier champ d'étude qui a fait l'objet d'analyse.

1.2 Problématique spécifique du projet

Le projet actuel a eu pour but la conception et l'implémentation d'une base de données de génomique qui servirait à l'identification des variations génétiques pouvant expliquer le développement de la scoliose, une maladie d'origine génétique qui déforme la courbure de la colonne vertébrale. Ce projet a été réalisé pour le Laboratoire de génétique moléculaire des maladies/malformations musculo-squelettiques du Centre hospitalier universitaire Sainte-Justine (CHUSJ) et toutes les données utilisées proviennent de fichiers anonymisés du Laboratoire. La solution actuelle du Laboratoire consiste en l'analyse et le partage de fichiers Excel par courriel.

CHAPITRE 2

PRÉSENTATION DE LA TECHNOLOGIE UTILISÉE

2.1 Exigences liées au choix de technologies

Plusieurs exigences ont orienté le choix des technologies à employer pour ce projet. D'abord, la recherche de variations génétiques est une recherche d'information disparate, c'est-à-dire qu'une très faible proportion des données correspond font l'objet de la recherche et sont dispersées dans l'entièreté des données. En effet, pour la plupart des variations génétiques, la proportion de la population possédant cette variation (ce qu'on appelle la fréquence allélique) est très faible, donc si l'on recherche seulement les individus possédant une certaine variation génétique, il sera relativement long de retrouver cette information parmi tous les gènes de tous les individus de la population étudiée. L'architecture utilisée doit donc permettre de rapidement effectuer ces recherches disparates.

Ensuite, les technologies utilisées pour effectuer le séquençage évoluent rapidement et peuvent changer le format des données en sortie. D'ailleurs, le Laboratoire compte passer à une nouvelle technologie nommée Illumina l'année prochaine et s'attend à devoir modifier la façon dont les données sont traitées en conséquence. À cette fin, le modèle de données développé doit être flexible et adaptable aux changements dans les données, sans pour autant perdre les données existantes.

Cette dernière exigence est d'ailleurs liée au besoin de traçabilité des données. En fait, l'on doit pouvoir à la fois pouvoir répliquer les transformations des données sources vers les données étudiées, ce qui requiert le stockage de données à grand volume (voir section 1.1 pour une approximation de la taille des données brutes), ainsi que les changements de valeur pour un individu lors d'un reséquençage. Bien que cette exigence n'a pas fait l'objet de l'implémentation lors de cette première itération du projet, elle a été retenue pour les besoins de la sélection des technologies en envisageant des itérations futures.

Enfin, étant donné la grande quantité de données stockées, il est important de pouvoir facilement augmenter la capacité de la base de données, à la fois au niveau du stockage, de la mémoire et de la puissance de calcul (parcourir plus de données à la même vitesse requiert une vitesse de lecture plus rapide). Préférablement, le système devrait être propice à une expansion horizontale (plus de composantes à coût abordable), plutôt que d'être limité à une expansion verticale (meilleures composantes plus dispendieuses).

Tenant compte de ces exigences, une famille de technologies a semblé propice à la tâche, soit la plateforme Hadoop d'Apache et plus spécifiquement, un de ses sous-projets, HBase.

2.2 Hadoop : une architecture distribuée

Hadoop est un cadriciel développé par Apache utilisé pour le traitement distribué de grandes quantités de données (communément appelé du « big data ») basé sur les publications MapReduce et Google File System (GFS) de Google et écrit en Java. Il comporte deux composantes principales : d'abord, son système de fichiers distribué intitulé HDFS (Hadoop Distributed File System), puis son mécanisme de parallélisation et distribution de tâches, MapReduce/YARN (Yet Another Resource Negotiator). Pour adéquatement décrire le fonctionnement et les avantages de l'utilisation de Hadoop, nous nous attarderons à la description de ces deux composantes.

HDFS est un système de fichiers distribué, c'est-à-dire qu'il répartit ses fichiers sur plusieurs noeuds d'une même grappe et en fait ensuite abstraction à l'utilisateur en présentant une interface d'accès servant de façade. Il est composé de deux types de noeuds : le *NameNode* (ainsi que les *NameNodes* secondaires) stocke les métadonnées nécessaires à la gestion et à la réplication des fichiers sur plusieurs noeuds (en cas de panne d'un noeud) et les *DataNodes* stockent les fichiers sous forme de blocs. La taille d'un bloc est fixe, donc si un fichier individuel dépasse cette taille, il est découpé en plusieurs blocs, qui sont eux répartis sur les *DataNodes*. De par ce fait, on facilite l'expansion horizontale du système, puisqu'on peut simplement ajouter d'autres *DataNodes* à la grappe lorsque nécessaire, et on assure la

disponibilité du système, puisque la réplication fait en sorte qu'aucun noeud individuel peut arrêter le système complet en tombant en panne. En fait, un logiciel nommé ZooKeeper (lui aussi distribué sur plusieurs noeuds) assure la redondance du *NameNode* en élisant un nouveau *NameNode* parmi les *NameNodes* secondaires si celui-ci tombe en panne.

MapReduce, qui s'appelle maintenant YARN depuis la version 2, permet de paralléliser une tâche et de distribuer les parties afin qu'elles s'exécutent sur différents noeuds simultanément et ainsi accélérer la complétion de la tâche. Il accomplit ceci via une architecture très similaire à celle de HDFS. Effectivement, MapReduce est composé d'un *JobTracker* qui est responsable de diviser une tâche en entrée et de la répartir sur les noeuds esclaves, les *TaskTrackers*, qui exécutent chacun la partie de la tâche qui leur a été assignée et retournent le résultat au *JobTracker*. Depuis la version 2 de MapReduce, *JobTracker* et *TaskTracker* se nomment respectivement *ResourceManager* et *NodeManager*. MapReduce offre ainsi les mêmes bénéfices d'expansion horizontale et de disponibilité que HDFS, mais permet aussi de faire des gains de performance lors d'exécution de longs traitements hautement parallélisables comme l'agrégation de champs dans une base de données.

2.3 HBase : une base de données non-relationnelle

HBase est un sous-projet de Hadoop et agit comme base de données non-relationnelle distribuée à l'aide de HDFS. Elle est basée sur la publication BigTable de Google.

Étant une base de données non-relationnelles (ou NoSQL), HBase a un modèle de données très différent des bases de données traditionnelles, comme Oracle ou MySQL. En fait, HBase est hautement dénormalisé et stocke habituellement toutes ses données dans une seule table. Cette table est composée d'une ou plusieurs familles de colonnes (*column family*) statiques qui sont déclarées à la création de la table et qui dictent comment les données seront stockées. Ces familles de colonnes sont à leur tour divisées en colonnes, qui sont elles dynamiques et peuvent varier d'une rangée à une autre. En réalité, une colonne n'est qu'un qualificatif pour une famille de colonnes et sert uniquement à identifier une instance de celle-

ci pour une rangée particulière. Par exemple, les notes des étudiants pour un cours de physique pourraient être stockées dans une famille de colonnes intitulée « cours », dont le qualificatif est « physique », qui serait identifié ainsi : « cours:physique ». Un étudiant n'étant pas inscrit à ce cours de physique, mais suivant plutôt des cours de mathématiques et de chimie n'aurait pas la colonne physique de déclarée, mais aurait des entrées identifiées « cours:mathematiques » et « cours:chimie ».

Chaque rangée de la table est, tant qu'à elle, identifiée par une clé unique (*rowkey*), de façon semblable à la clé primaire d'une table dans une base de données relationnelle, et l'intersection d'une rangée et d'une colonne forme une cellule qui contient une valeur. Chaque cellule d'une table est versionnée, habituellement en utilisant la date-heure de mise à jour, donc l'historique complet de chaque cellule est conservé et peut être accédé en spécifiant la version désirée (par défaut, une requête retourne la version la plus récente d'une cellule). La figure 2.1 ci-dessous illustre un exemple de table composée de deux familles de colonnes, « User » et « Social ».

Column Family: User

rowid	Col_name	ts	Col_value
u1	name	v1	Ricky
u1	email	v1	ricky@gmail.com
u1	email	v2	ricky@ya
u2	name	v1	Sam
u2	phone	v1	650-3456

Column Family: Social

rowid	Col_name	ts	Col_value
u1	friend	v1	u10
u1	friend	v1	u13
u2	friend	v1	u10
u2	classmate	v1	u15

Figure 2.1 - Exemple de table comportant deux familles de colonnes

L'attribut le plus distinctif du modèle de données de HBase est que toutes les cellules sont triées d'abord par famille de colonnes, puis par clé de rangée, par nom de colonne et par

version. C'est dans cet ordre que les données sont physiquement stockées dans la base de données. Nous verrons sous peu comment HBase tire profit de cet ordonnancement pour accélérer ses recherches.

2.4 L'architecture de HBase

Obéissant aux mêmes paradigmes architecturaux que HDFS et MapReduce, HBase est composé d'un noeud primaire, le *Master*, et de plusieurs noeuds esclaves, les *RegionServers*. Les *RegionServers* stockent les données des tables, alors que le *Master* s'occupe de la répartition et de la réplication des données sur ces noeuds. Une table, en fait, est divisée en plusieurs entités physiques nommées des régions et ce sont ces régions qui sont réparties sur les *RegionServers*. Chaque région est identifiée par les clés d'accès (la famille de colonne, la clé de rangée, le nom de colonne et la version) de sa première et dernière cellules et une table spéciale nommée .META sur le *Master* catalogue l'emplacement de chaque région en ordre de la clé d'accès de leur dernière cellule. Une recherche de données s'effectue donc en consultant la table .META pour trouver l'emplacement de la première région dont la dernière clé est inférieure à la clé recherchée, puis on consulte la région elle-même sur le *RegionServer* appropriée. Si la clé d'accès distribue bien les cellules par région (une distribution relativement uniforme, mais pas trop, est idéale), ceci permet de rapidement réduire le nombre de données à parcourir afin de retrouver l'objet de la recherche.

HBase assure l'intégrité des données en écrivant toute mise à jour de façon atomique à la fin du *Write-Ahead Log* (WAL), ou journal de pré-enregistrement, du *RegionServer* approprié et puisque HBase stocke tous ses fichiers dans HDFS, ce journal est répliqué sur d'autres noeuds en cas de panne. Une fois enregistrée dans le WAL, toute nouvelle donnée est d'abord stockée dans un espace de la mémoire vive du noeud réservé à la région, le MemStore. Ceci est un autre mécanisme qui permet d'accélérer les recherches puisque les données les plus récentes (et donc les plus susceptibles d'être recherchées) sont déjà en mémoire et n'ont pas besoin d'être lues à partir du disque. Enfin, lorsque le MemStore se remplit, les données qu'il contient sont écrit sur disque dans un HFile, aussi répliqué à l'aide

de HDFS, et le MemStore et le WAL du *RegionServer* sont vidés. La figure 2 ci-dessous représente une vue statique de cette architecture.

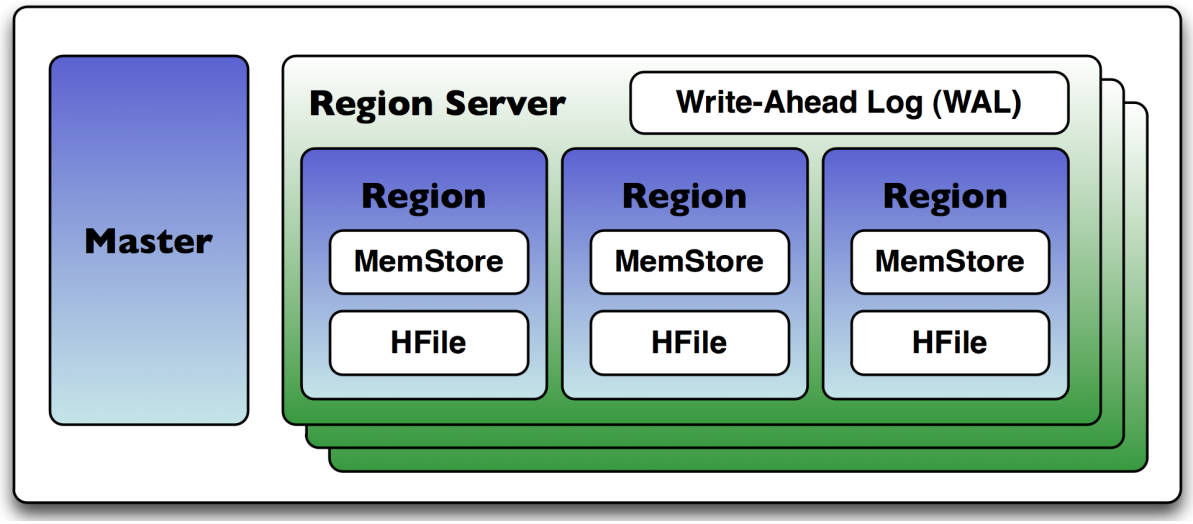

Figure 2.2 - Vue architecturale de HBase

2.5 Justification de la technologie choisie

On voit donc par le fonctionnement de Hadoop et HBase que plusieurs avantages sont accordés par leur utilisation, notamment :

- La disponibilité du système et l'intégrité des données sont assurées via ses mécanismes de redondance et de réplication.
- Le schéma de la base de données est flexible grâce à la déclaration dynamique des colonnes, ce qui permet également la traçabilité des données en conjonction avec son système de versionnement.
- Le filtrage rapide des données lors de recherches est propice à la recherche de données disparates.
- Les tables sont dénormalisées, éliminant la nécessité de jointures massives, un des facteurs de ralentissement les plus importants de bases de données relationnelles.
- Le système est facilement extensible de façon horizontale simplement en ajoutant de nouveaux noeuds.

On remarque que ces avantages correspondent pour la plupart aux exigences technologiques élicités à la section 2.1 et c'est donc pour ces raisons que ces technologies ont été retenues pour l'implémentation du projet actuel.

CHAPITRE 3

ANALYSE DES DONNÉES ET PLANIFICATION DU PROJET

3.1 Profil des données

Le Laboratoire a fourni trois fichiers de données sous le format Excel au long du projet correspondant aux documents d'analyse des chercheurs contenant les variations génétiques étudiées, ainsi que les données relatives aux génomes prélevées sur les patients du Laboratoire et sur des individus de contrôle. Une analyse des données a été faites sur le premier fichier fourni, mais il a par la suite été signalé que ce format n'était plus valide et un deuxième fichier a été remis. La grille d'analyse pour le premier fichier a été inclus dans l'Annexe I de ce document à titre consultatif, mais elle n'a pas servi par après au projet.

Le deuxième fichier est celui qui a été retenu pour la conception de la base de données et est celui auquel fera référence le reste de ce document. Le troisième et dernier fichier est une légère modification du deuxième qui inclue quelques nouveaux champs statistiques qui ont été subséquemment ajoutées à la base de données.

Le fichier source contient 40 champs contenant des données génériques et statistiques pour chaque variation génétique retenue. Il présente également deux champs pour chaque patient et individu de contrôle par variation, correspondant au génotype de l'individu et à la qualité de lecture du gène en question pour cet individu. Il contient des données sur 20 855 variations pour 129 individus, dont 67 patients ayant la maladie, classés en trois groupes biochimiques. La grille d'analyse complète du fichier de données a été inclus dans l'Annexe II de ce document.

3.2 Étapes du projet

Le projet s'est déroulé en quatre grandes phases :

1. La conception de la base de données et de ses mécanismes d'optimisation
2. L'installation de la grappe de serveurs, de Hadoop et de HBase, ainsi que le déploiement du modèle de données
3. L'importation des données de Excel vers HBase
4. La création d'une API (Application Programming Interface) afin d'abstraire les mécanismes complexes de HBase et simplifier son utilisation

Le prochain chapitre couvre la première de ces phases, alors que les trois autres phases seront abordées dans le chapitre 5.

CHAPITRE 4

CONCEPTION ET OPTIMISATION DE LA BASE DE DONNÉES

4.1 Cas d'utilisation

Deux cas d'utilisation principaux ont été retenus pour cette itération du projet. D'abord, le cas d'utilisation principal est la recherche de variations génétiques correspondant à différents critères. Les critères principaux utilisés pour cette recherche sont le chromosome concerné, ainsi que les différentes statistiques calculées à partir de la population de génomes comme la fréquence allélique et le chi-deux.

Le deuxième cas d'utilisation est la recherche des données spécifiques aux individus qui possèdent une ou plusieurs parmi une liste de variations génétiques. À cette étape, le chercheur a déjà cerné les variations qui l'intéressent et veut connaître plus d'informations sur les individus ayant une de ces variations.

4.2 Facteurs importants à la conception

La conception de la table primaire détermine principalement la clé de rangée qui sera utilisée, ainsi que les familles de colonnes statiques qui seront déclarées à la création de la table. Ces deux choix sont très importants et doivent être adaptés aux cas d'utilisation, à défaut de subir des pertes de performance importantes et d'implémenter une solution inutilisable. En effet, plusieurs facteurs doivent être considérés :

- Les familles de colonnes doivent distinguer des données qui sont rarement demandées en conjonction. Puisque les cellules sont d'abord triées par famille de colonne, les différentes familles de colonnes ne seront pas stockées dans la même région. Ceci réduit le nombre de données à lire si l'on n'a besoin que des données d'une famille de colonnes, mais ralentit la lecture lorsque plusieurs familles de colonnes doivent être lues. Une famille de colonnes correspond souvent à un cas d'utilisation.

- Les familles de colonnes devraient aussi être de tailles comparables, puisque la compression des données se fait sur une table complète lorsqu'une famille de colonnes devient trop grosse, donc une grande différence de taille entre les familles de colonnes causerait la surcompression inutile des familles de colonnes plus petites, ce qui en ralentit la lecture.
- La clé de rangée (*rowkey*) doit permettre d'identifier une entité unique afin d'éviter les collisions. Elle doit également permettre de rapidement filtrer les données afin de réduire le nombre de données qui doivent être lues. Pour ce faire, la clé de rangée devrait correspondre aux critères de recherche les plus communs du cas d'utilisation principal.
- Les familles de colonnes et la clé de rangée devraient tendre vers une distribution presque uniforme afin de pouvoir faire une dispersion équitable et facile à chercher des données parmi les multiples régions. Par contre, il faut éviter de trop uniformiser la distribution des données parce que la sous-division d'une région lorsqu'elle est pleine rend temporairement inaccessible les données de cette région, donc la sous-division synchronisée de plusieurs régions pourrait causer des périodes de ralentissement importantes pour le système.

4.3 Conception de la table primaire

Trois familles de colonnes ont été créées pour ce projet : « variations », « samples » et « nosamples ». Cette division s'inspire de la thèse de Sébastien Servoles, étudiant de maîtrise à l'ÉTS qui a également conçu une base de données de génomique en HBase, et permet de répondre aux besoins des deux cas d'utilisation énoncés.

D'abord, la famille « variations » contient toutes les données génériques et statistiques pour une variation génétique, comme le nucléotide de référence, le nucléotide de la variation, le domaine fonctionnel, le chi-deux, les fréquences alléliques, etc. Une liste complète des colonnes pour cette famille est disponible à l'Annexe III. Cette famille de colonnes sert à répondre aux besoins du cas d'utilisation principal.

Les familles de colonnes « samples » et « nosamples » contiennent essentiellement les mêmes informations. Toutes deux utilisent l'identifiant de l'individu comme nom de colonne pour stocker les données d'un individu pour la variation spécifiée dans la clé de rangée. La valeur contenue dans la cellule est en fait une agrégation de tous les champs d'un individu pour une variation (génotype, qualité de lecture du gène et groupe biochimique de l'individu) séparés par « : », mais ceci est acceptable puisqu'aucune de ces valeurs ne constitue un critère de recherche et que la création d'une famille de colonnes intermédiaire causerait une perte de performance inutile. La différence entre les deux familles de colonnes réside en le fait que la famille « samples » contient les données seulement des individus ayant la variation génétique, alors que la famille « nosamples » contient les données des individus qui n'ont pas la variation. Ceci accélère la recherche lors du deuxième cas d'utilisation, car les données relatives aux individus qui n'ont pas la variation ne nous intéressent pas. De plus, la famille « nosamples » ne stocke pas la valeur du génotype, car celui-ci sera toujours de 0 puisqu'il correspond au génotype de référence.

Enfin, la clé de rangée est composée d'abord du chromosome, puis de la position du gène et d'une valeur incrémentale qui sert à éviter les collisions. Le chromosome sert de préfixe à la clé puisque c'est un critère de recherche commun pour le cas d'utilisation principal et il permet de rapidement scinder les données étant donné que le génome humain comporte seulement 23 pairs de chromosomes. Ensuite, la position du gène est utilisé pour identifier le gène subissant la variation. En fait, pour les besoins de recherche actuels, ces deux paramètres à eux seuls permettent d'identifier une variation unique, mais une valeur incrémentale est affixée à la clé en vertu de changements futurs possibles où plusieurs variations plus complexes peuvent exister pour un gène particulier. Les statistiques, autres critères de recherche communs dans le cas d'utilisation principal, n'ont pas été utilisées dans la clé puisque ce sont des attributs calculés et sont donc très volatiles en fonction des individus étudiés, les rendant peu propices dans une clé identifiante.

La figure 4.1 ci-dessous illustre de façon abrégée le modèle de données :

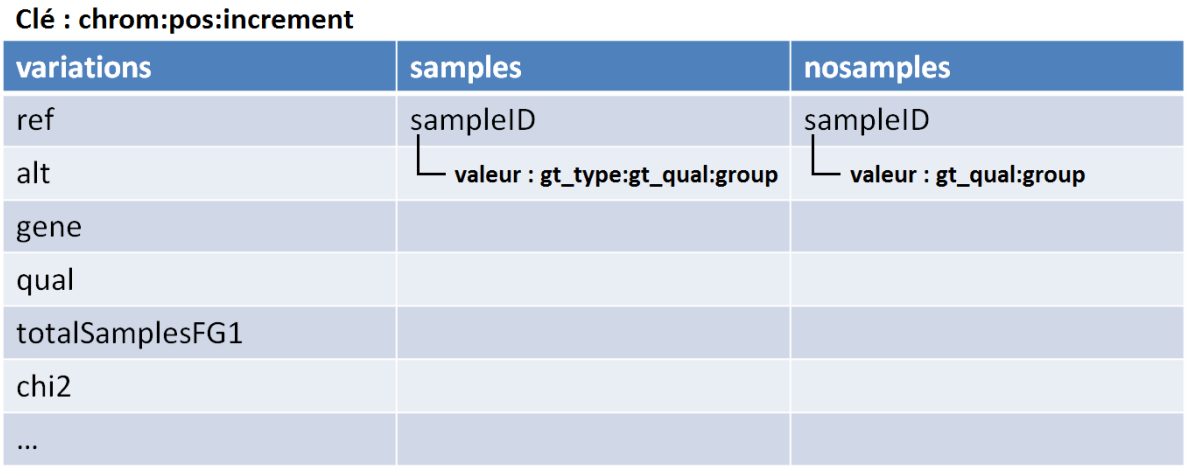

Figure 4.1 - Représentation du modèle de données

On voit par cette représentation que les facteurs ayant un impact important sur la performance ont été respectés. Les colonnes de familles sont lues lors de recherches distinctes pour différents cas d'utilisation, la clé de rangée permet d'identifier une entrée sans causer de collisions et est composée d'un champ souvent utilisé pour la recherche et qui filtre bien les données. La taille des familles de colonnes peut paraître très différentes, mais elles sont comparables. En fait, bien que les familles « samples » et « nosamples » ne possèdent qu'une colonne, elles stockent ensemble l'information de plus de 100 patients par variation, alors que la famille « variations » stocke un peu plus de 40 colonnes par variation. C'est aussi ce qui contribue à uniformiser la distribution des données puisque la taille est semblable pour chaque famille et que la clé basée sur le chromosome permet de bien distribuer les données.

4.4 Index secondaires

Un index secondaire avait été contemplé pour permettre de facilement retrouver toutes les variations pour un individu particulier, mais l'idée a été abandonnée suite à une rencontre avec les utilisateurs du CHUSJ. Le concept est tout de même détaillé ici à titre de référence pour les itérations futures du projet au cas où cette fonctionnalité ou une autre similaire deviendrait désirable.

En fait, HBase n'a aucun mécanisme permettant de créer des index secondaires. Ses seules méthodes de recherche sont Get, qui prend en paramètre l'identifiant de la cellule recherchée et l'accède directement, et Scan, qui peut soit rechercher sur un intervalle de valeurs pour la clé de rangée ou faire une recherche sur la table complète. Pour contourner ceci, la façon commune de simuler un index secondaire dans HBase est de créer une deuxième table dont la clé de rangée est triée différemment et la valeur correspondante est la clé de la table primaire. Ceci permet de faire une recherche rapide dans la table index et d'utiliser la ou les clés de la table primaire pour faire des requêtes Get directement.

Pour ce projet, une table d'index avait été conçue dont la clé serait composée ainsi : « sampleID:chrom:increment » ayant une seule famille de colonnes « mainKey » et dont la valeur contenue dans la cellule serait la clé de la table principale. Ceci permettrait de rapidement filtrer toutes les valeurs par individu et de par la suite utiliser le chromosome désiré pour réduire le nombre d'entrées davantage si nécessaire. Ensuite, il s'agit uniquement de faire des requêtes Get individuelles avec les clés retournées.

4.5 Table « schema »

La table « schema » est une table de métadonnées qui contient toutes les colonnes de la table et indique si elles existent dans le modèle de données actuel. La table ne possède qu'une famille de colonnes, intitulée « exists », et la clé de rangée est composée du nom de la famille de colonnes, suivi du nom de la colonne. Enfin, la valeur contenue dans la cellule est soit 1 ou 0 pour indiquer si oui ou non la colonne existe dans le modèle de données actuel. De plus, grâce au versionnement des cellules, on peut garder la traçabilité du schéma de la table.

CHAPITRE 5

IMPLÉMENTATION DE LA SOLUTION

5.1 Installation

Pour le développement de la solution, cinq serveurs ont été empruntés de l'ÉTS et configurés en grappe. Pour l'installation de Hadoop et de HBase, Cloudera 5 a été sélectionné. Cloudera est une solution pré-configurée qui facilite l'installation de Hadoop et de plusieurs de ses sous-projets, comme HBase, Hive, Pig et Impala. Il offre également des outils de gestion afin de visionner l'état des serveurs et identifier tout problème qui peut survenir, ainsi que de dynamiquement assigner des rôles à chaque noeud. Cloudera a été choisi par souci de temps et de facilité d'utilisation, car l'installation manuelle de Hadoop en conjonction avec ses sous-projets et leur déploiement sur une grappe de serveur est une opération relativement complexe et ce n'était pas l'objectif du projet. La configuration des noeuds est illustrée dans la figure 5.1 ci-dessous.

Ceci fait, le schéma de la base de données a été créé en HBase à l'aide de l'API Java selon le modèle établi au chapitre précédent afin de préparer le système pour l'importation des données.

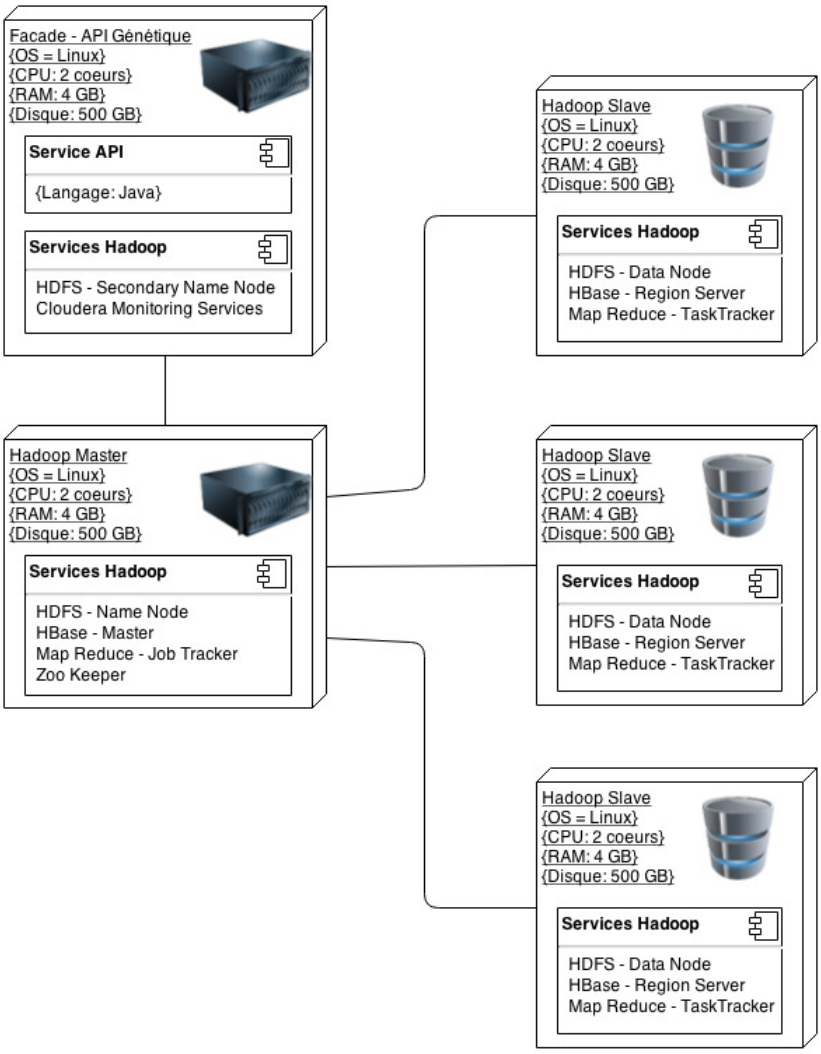

Figure 5.1 - Configuration des noeuds

5.2 Importation des données

L'importation des données, comme l'implémentation du modèle de données, s'est faite par l'API Java de HBase. En fait, il a d'abord fallu exporter les données de Excel vers le format séparé par des tabulations (TSV), puis elles ont été chargées en mémoire et été insérées dans la base de données ligne par ligne. Il serait préférable de paralléliser cette tâche à l'aide de MapReduce afin d'accélérer le processus, mais pour la quantité limitée de données contenues dans le fichier Excel, cette solution s'est avérée passable et s'exécute en moins d'une minute.

La simplicité a donc été préférée pour cette itération, mais il faudra développer une solution plus performante lorsque de plus grands volumes de données doivent être insérés.

5.3 L'API

L'API est installée sur le *NameNode* secondaire de HDFS et elle sert à masquer les opérations de HBase à l'utilisateur afin que celui-ci puisse manipuler les données et faire ses recherches sans devoir comprendre comment ses actions sont interprétées. L'API présente quatre fonctionnalités majeures et toutes les données retournées sont sauvegardées sous le format JSON (un exemple de réponse est disponible à l'Annexe IV).

D'abord, l'application client de l'utilisateur devrait automatiquement faire une requête à l'API afin d'obtenir le schéma de la table, avec optionnellement en paramètre une date. Cette requête se transforme en Scan sur la table « schema » et retourne le modèle de données actuel ou celui qui était en vigueur à la date spécifiée. Ceci permet à l'application client d'afficher la structure de la base de données sans devoir la connaître via une configuration spécifique même si le schéma est modifié.

Le deuxième appel possible est de retourner tous les groupes biochimiques étudiés ainsi que le groupe de contrôle. Encore une fois, il s'agit d'une façon de dynamiquement apprendre à l'application client les métadonnées du système.

La troisième requête qui peut être exécutée via l'API est une recherche simple pour une variation génétique spécifique. Le client doit fournir en paramètre la clé exacte de la variation recherchée et les données de la famille de colonnes « variations » relatives à cette variation seront retournées.

Enfin, la requête la plus complexe est celle d'une recherche paramétrée pour des variations génétiques. Pour ce type de requêtes, l'utilisateur spécifie les critères recherchés (par exemple : chrom in (chr1, chr2, chr3) & qual>80) et optionnellement les champs qu'il veut

afficher, l'ordre de tri des valeurs, le décalage depuis la première valeur, ainsi que le maximum de valeurs à retourner. Ces deux derniers paramètres servent surtout à la pagination. Par exemple, si l'on ne veut afficher que 10 résultats par page et que l'utilisateur veut voir la troisième page de résultats, le client demanderait à l'API de retourner les données avec un décalage de 20 et une limite de 10. Ce type de requête permet aussi de retourner des données de n'importe quelle famille de colonnes et celles-ci seront juste lues si une de leurs colonnes est utilisée dans la requête.

La documentation complète de l'API est disponible en ligne en anglais sur un site privé. Le professeur Alain April devrait être consulté si le lecteur désire y accéder.

CONCLUSION

L'étude de la génomique requiert l'analyse de très grands volumes de données, ce qui est communément appelé du « big data ». Au CHUSJ, le Laboratoire étudie les causes génétiques de la scoliose, mais se voit limité dans ses capacités de recherche par les technologies à sa disposition.

Le projet présent avait pour but la conception d'une base de données en HBase qui répondrait aux exigences des chercheurs du CHUSJ. Dans cette optique, une preuve de concept a été mise en place et sera sous peu démontrée aux membres du Laboratoire.

La solution en HBase est consituée d'une table primaire composée de trois familles de colonnes, présentant une variété d'avantages, tels que l'expansion horizontale simple du système, la traçabilité et l'intégrité garantie des données et une vitesse de recherche performante pour les besoins des utilisateurs.

Une API servant de façade pour HBase simplifie l'utilisation du système et permet aux utilisateurs et aux applications client d'utiliser les données en faisant abstraction des mécanismes qui les gèrent.

Lors de cette première itération du projet, seule une preuve de concept utilisant un échantillon de données a été implémentée. Il faudra par la suite développer une solution plus robuste avec les données complètes et faire des tests de performance afin de valider le modèle de données et y apporter des modifications s'il y a lieu. Il serait possiblement aussi envisageable d'étendre la portée du projet à d'autres centres de recherche en génomique afin de partager les données et ainsi augmenter les capacités de recherche de chaque centre.

LISTE DE RÉFÉRENCES

La figure 2.1 à la page 7 démontrant un exemple de table avec deux familles de colonnes a été prélevée du site suivant :

Han, Henry. 2010. « BigTable Model with Cassandra and HBase ». In *Huiwhenhan*. En ligne. <http://huiwenhan.wordpress.com/2010/11/18/repostbigtable-model-with-cassandra-and-hbase/>. Consulté le 16 avril 2014.

La figure 2.2 à la page 9 illustrant l'architecture de HBase a été prélevée du site suivant :

Leberknight, Scott. 2013. « Handling Big Data with HBase ». In *DZone*. En ligne. <http://java.dzone.com/articles/handling-big-data-hbase-part-3/>. Consulté le 16 avril 2014.

La figure 5.1 à la page 19 illustrant les responsabilités de chaque noeud de la solution a été créée par David Lauzon, étudiant de maîtrise à l'ÉTS, et est utilisée avec sa permission.

BIBLIOGRAPHIE

White, Tom. 2012. *Hadoop: The Definitive Guide,* 3rd Edition. Sebastopol: O'Reilly, 657 p.

The Apache Software Foundation. 2014. « The Apache HBase Reference Guide ». In *Apache HBase*. En ligne. <http://hbase.apache.org/book.html>. Consulte le 16 avril 2014.

George, Lars. 2011. *HBase: The Definitive Guide,* 1st Edition. Sebastopol : O'Reilly, 556 p.

Wikipedia. 2014. « Genomics ». In *Wikipedia.* En ligne. <http://en.wikipedia.org/wiki/Genomics>. Consulté le 22 avril 2014.

George, Lars. 2010. « HBase Architecture 101 – Write-ahead-Log ». In *Lineland.* En ligne. <http://www.larsgeorge.com/2010/01/hbase-architecture-101-write-ahead-log.html>. Consulté le 23 avril 2014.

ANNEXE I

ANALYSE DU FICHIER ORIGINAL

Voir analyse_orig.xlsx.

ANNEXE II

ANALYSE DU FICHIER UTILISÉ

Voir analyse_vrai.xlsx.

ANNEXE III

LISTE DES COLONNES DE LA FAMILLE VARIATIONS

- ref
- alt
- gene
- qual
- callRate
- rsIds
- pfamDomain
- rmsk
- inCpgIsland
- inSegdup
- isConserved
- gmsIllumina
- gmsSolid
- inCse
- Depth
- rmsMapQual
- impactSeverity
- aafEspAll
- aafEspEa
- aaf1kgAll
- aaf1kgEa
- numHomRef
- numHet
- numHomAlt
- totalFg1
- totalFg2
- totalFg3
- totalCtrl
- totalAis
- oddRatio
- chi2

ANNEXE IV

EXEMPLE DE RÉPONSE DE L'API


```
{
        “variations”: [ {
                “id”:“chr1:13539825:0”,
                “ref”:“T”,
                “alt”:“G”,
                “qual”:84,
                ...},
                {
                “id”:“chr1:24802530:0”,
                “ref”:“C”,
                “alt”:“A”,
                “qual”:42,
                ...}]
}
```