CESi
ENSEIGNEMENT SUPÉRIEUR ET FORMATION PROFESSIONNELLE

projetdna

Projet DNA

Spécialité : Ingénierie logicielle

Niveau : 4ème année

$Id: Projet-RIL, 0.1.0-BETA 2017/01/01 04:47:09 rbello@cesi.fr Exp $

SYNTHESE DU PROJET

L'objectif de ce projet est de **concevoir une plateforme distribuée de traitement des séquences d'ADN**. En vous appuyant sur les technologies du Framework .NET, vous devrez réaliser un système complet composé de plusieurs applications distribuées en cluster sur plusieurs machines afin d'effectuer certaines opérations sur des séquences d'ADN humaines.

Voici une explication simple du fonctionnement attendu :

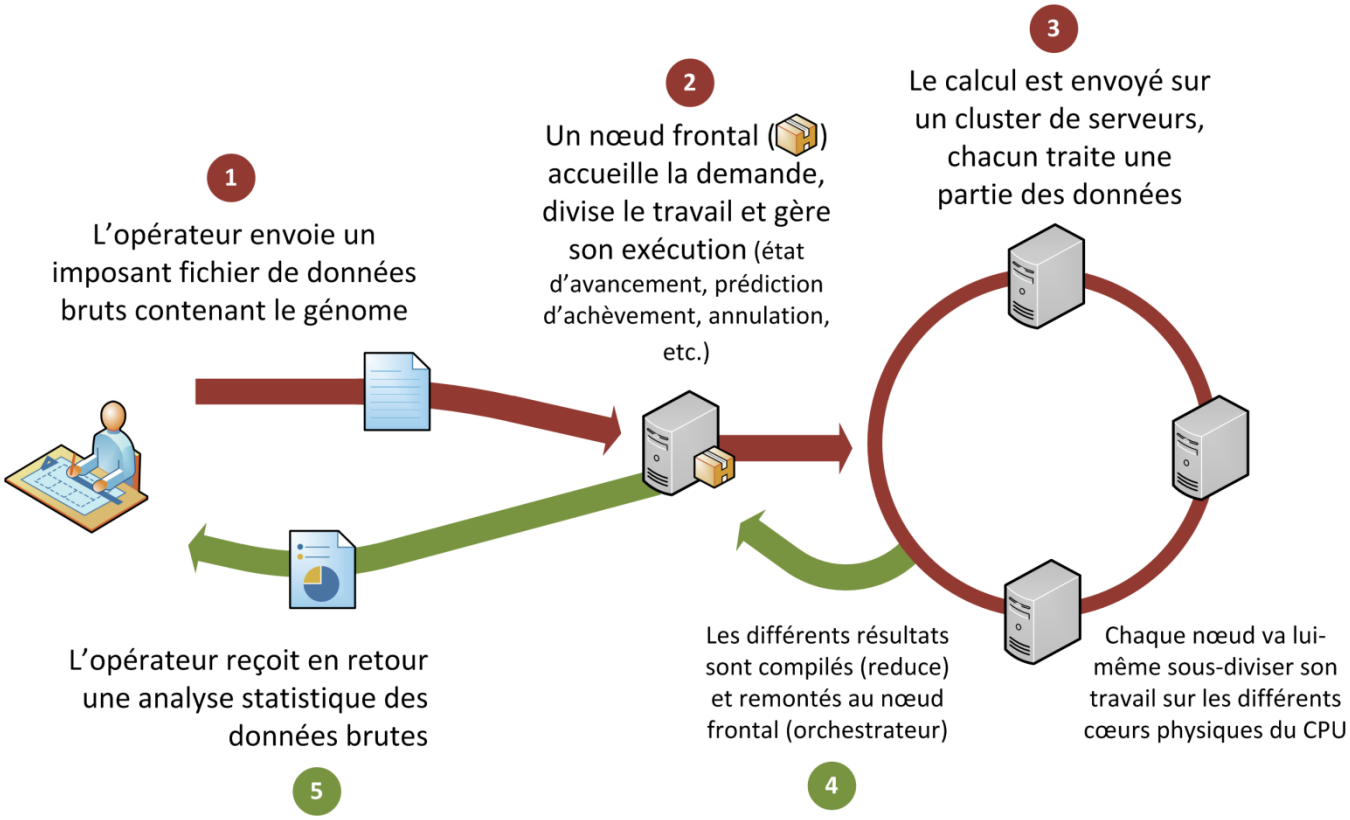

Figure 1 - Diagramme fonctionnel de la solution complète

Le système aura comme finalité de permettre certaines opérations sur les données :

- Rechercher une séquence d'ADN dans un génome, en renvoyant les séquences qui matchent le mieux avec l'échantillon donné et classées par pourcentage de correspondance avec l'échantillon.
- Extraire des données statistiques quantitatives concernant le génome.
- Traduire le génome en acides aminés, afin d'y rechercher des gènes.

Le projet devra être réalisé par un groupe de 2 à 3 personnes. Le livrable devra comporter le code source (de qualité et commenté sur les parties complexes) en archive ZIP, un exécutable EXE, ainsi qu'un document donnant une vue d'ensemble sur l'architecture logicielle de la solution. Une soutenance permettra de présenter l'application et les détails de sa conception logicielle.

***Mise en garde sur l'imprécision scientifique de cet exercice** : afin d'être principalement orienté sur la conception logicielle et non la génétique, cet exercice est volontairement vulgarisé. En réalité, le procédé utilisé dans cet exercice n'est pas adapté à l'analyse de l'ADN humain (eucaryote).*

ENONCE DES TRAVAUX A REALISER

Cette section débute par une introduction sur l'ADN et son analyse, puis énonce les différents travaux à réaliser.

Le génome

Le but du système est de traiter des données représentants un génome humain. Un génome est « enregistré » dans une macromolécule appelée Acide désoxyribonucléique, ou ADN. Cette molécule a la forme d'une double hélice, et elle est composée de plusieurs de paires de « bases azotées » que l'on peut assimiler à des connexions entre les deux hélices.

Chaque *base* (azotée) est reliée à une autre base sur l'hélice opposée, c'est pour cette raison qu'une fois séquencé l'ADN peut se représenter par une série de combinaison de 2 lettres correspondant aux bases :

- A pour l'Adenine
- T pour la Thymine
- G pour la Guianine
- C pour la Cytosine

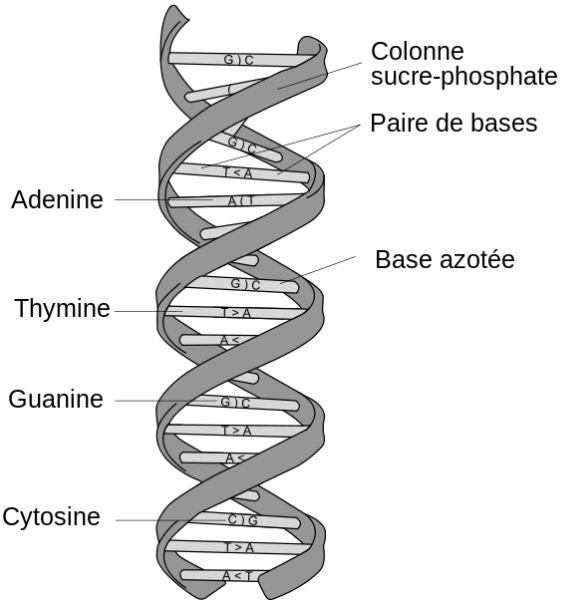

Figure 2 – Séquence d'ADN représentée en double hélice

Cette représentation est très pratique pour le traitement informatisé, même s'il faut être conscient qu'il ne s'agit que d'un modèle simplifié. Néanmoins, d'un point de vue informatique, les opérations que vous allez réaliser son assimilables à du traitement de caractères sur des chaînes, que l'on appellera séquences d'ADN. Ce sont ces séquences qui portent l'information génétique. Celle-ci est structurée en gènes, qui sont exprimés à travers la transcription en ARN. Votre travail va donc consister à :

- Offrir un logiciel capable de lire les séquences d'ADN et d'effectuer des recherches dans l'ADN
- Extraire des statistiques quantitatives sur les séquences d'ADN
- Transformer l'ADN en ARN afin d'y rechercher des gènes

Toute la difficulté de ces traitements est que l'ADN n'est pas encodée d'une manière idéale, avec par exemple le début et la fin d'un gène parfaitement clairs. Le vivant peut faire preuve de subtilités étonnantes : en effet, certaines séquences peuvent à la fois jouer un rôle précis comme le début d'un gène, mais également être simplement une information contenue dans un gène. L'approche utilisée est donc « statistique et probabiliste » : on va rechercher dans l'ADN des séquences qui ont la plus grande « probabilité » d'être ce que l'on cherche. Enfin, comme chacun le sait, l'ADN est structurée en chromosomes, il faudra donc prendre en compte cette organisation.

Il vous est conseillé de prendre connaissance des ressources qui vous sont fournies, et qui expliquent plus précisément l'approche informatique du traitement des séquences d'ADN.

Les chapitres suivants détaillent la solution architecturale logicielle qui a été retenue par le client et auquel vous devrez vous conformer.

Socle technique imposé

Les développements devront être réalisés avec la dernière version du Framework .NET de Microsoft dans le langage C#. Aucune librairie autre que celles du framework ne devra être utilisée. Ce sera donc à l'équipe qui réalisera ce projet de créer ses propres librairies, les plus réutilisables possibles.

Networking

Le but de ce projet est d'effectuer les traitements sur la donnée la plus vite possible. Pour ce faire, il faudra diviser le calcul sur plusieurs machines simultanément afin d'optimiser les performances. Dans cette configuration, chaque serveur physique peut être vu comme le *nœud* d'un réseau. Un nœud spécial, appelé « orchestrateur » aura pour but de centraliser les connexions et distribuer le calcul.

Vous devrez réaliser l'ensemble de l'interconnexion des nœuds de calcul sous la forme d'un*cluster* (ou *pool*) vous-même, sans utiliser de librairie tierce. Chaque nœud pourra soit rejoindre un cluster existant (en précisant l'IP du nœud frontal), soit héberger lui-même le cluster (dans ce cas il sera chargé lui-même de l'orchestration). Dans ce dernier cas d'hébergement, l'IHM permettra de consulter les nœuds connectés au réseau et de lancer des calculs. Dans le cas où le nœud servirait de nœud d'exécution, l'IHM permettra de saisir l'IP du nœud orchestrateur.

Le cluster devra en permanence connaitre l'état de connexion des nœuds qui le compose, et gérer la connexion de nouveaux nœuds et la déconnexion d'autres durant l'exécution. Egalement, l'IHM de l'orchestrateur affichera l'état de travail de tous les nœuds, et leurs charges de CPU. Une interface graphique soignée permettra de voir – sur le nœud orchestrateur – l'état de tous les autres nœuds.

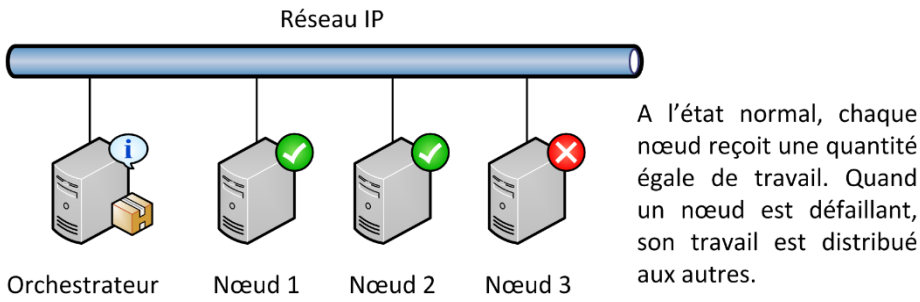

Figure 3 – Interconnexion des nœuds sur le réseau et répartition de la charge

En cas de déconnexion d'un nœud durant l'exécution d'un travail, l'orchestrateur devra être à même de redistribuer le travail non réalisé à un autre nœud. De plus, l'orchestrateur essayera toujours de répartir le travail au mieux entre les différents nœuds, pour équilibrer la charge et saturer chaque nœud.

Vous utiliserez un mécanisme de bas niveau appelé le « Socket » afin de faire communiquer vos nœuds sur un réseau IP. A vous donc de créer le *protocole* permettant d'échanger l'ensemble des transactions nécessaires au bon fonctionnement du système.

Votre code devra être propre et évolutif, le recours aux design patterns et l'application des principes SOLID sont attendus. Plusieurs traitements seront à effectuer sur les données, le projet sera donc découpé en plusieurs *Modules* détaillés plus loin dans ce document.

Division du traitement de la données

Afin de diviser le travail de traitement sur les différents nœuds du cluster, vous devrez implémenterez le patron d'architecture Map Reduce. Celui-ci sert à diviser et traiter la donnée, puis recomposer et renvoyer le résultat du traitement. Voici une représentation simplifiée du processus :

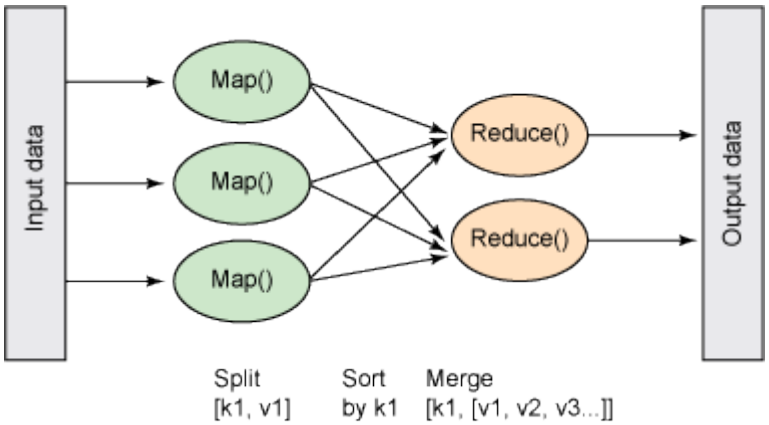

Figure 4 – Fonctionnement schématisé du Map Reduce

Cette technologie, très utilisée dans le traitement massif de données se prête parfaitement à une architecture de type cluster de machines Plus loin dans ce document viennent des informations sur les calculs que vous devrez réaliser sur les séquences d'ADN : ce sera à vous de déterminer comment ce travail pourra être divisé et réparti entre plusieurs unités de calcul.

Optimisation et complexité algorithmique

Temps de réponse

Le client souhaite que le système soit le plus réactif possible, en diminuant au maximum le délai de réponse. D'où l'architecture en cluster, la division du travail avec la méthode du MapReduce, et l'approche multi-threadée que vous devrez adopter.

En effet, chaque nœud de calcul qui se verra attribué une partie du traitement devra en interne re-segmenter le travail et déployer le calcul sur l'ensemble de ses cœurs physiques (CPU). Pour cela, le Framework .NET offre tous les mécanismes pour créer et utiliser des threads. De plus, intéressez-vous à la brique Parallel du Framework (voir les ressources). Attention également à proprement utiliser les bonnes pratiques en terme de threading avec l'IHM (voir ressources).

Complexité algorithmique

L'optimisation en temps du système passe aussi par une maitrise de la complexité algorithmique, et l'optimisation des fonctions de traitement de la donnée. Vous serez évalué par le client sur :

- L'implémentation d'algorithmes optimisés
- Votre capacité à expliquer le principe de complexité, et justifier vos choix d'implémentation

Le paragraphe suivant explique comment lire et interpréter les données fournies.

Les données

Le travail que vous devrez réaliser s'effectuera à partir de données provenant d'une analyse effectuée par séquençage automatique d'électrophorèse.

Des échantillons de données contenant des génomes humains vous seront remis, chaque fichier correspond à un génome et tous respectent le même format de données. Chaque ligne (non commentée) peut se décomposer en séparant les tabulations, ce qui donne quatre colonnes :

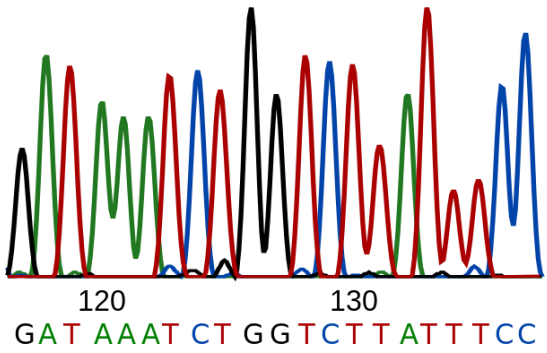

Figure 5 – Electrophérogramme

- Une chaîne qui identifie de manière unique la paire de base.
- Le chromosome portant la paire de base. Il y a 22 chromosomes numérotés de 1 à 22, puis 1 paire sexuelle (X et X ou X et Y). Les autres chromosomes ne seront pas utilisés.
- Une position (pour remettre les séquences dans le bon ordre)
- Et enfin l'information génétique, codée sur deux caractères d'un alphabet à 4 lettres (ATGC). Pour les chromosomes sexuels, un seul caractère est donné : il faudra recombiner les paires (en fonction de la position ordonnée). Enfin, il peut arriver que le génotype contienne le caractère tiret « - » qu'il faudra interpréter comme un inconnu : un caractère pouvant être n'importe quelle base.

MODULE #1 – ANALYSE QUANTITATIVE (COMPLEXITE : ★)

L'application réalisée devra afficher des statistiques quantitatives sur les génomes :

- Nombre total de paires de bases
- Nombre d'occurrence des bases A, T, G ou C dans le génome et pourcentage relatif au total
- Nombre de bases inconnues (le tiret)
- Nombre d'occurrence de la séquence de 4 bases la plus fréquente

MODULE #2 - IMPLEMENTER UN ALGORITHME DE RECHERCHE DANS LE GENOME (COMPLEXITE : ★★)

Il faudra implémenter la recherche d'une séquence donnée dans un génome entier d'ADN. Comme expliqué en préambule, la recherche dans le génome doit se faire de manière probabiliste. Ainsi, la recherche de la séquence « ATGC » dans le génome « ATGGCATGC » doit renvoyer plusieurs résultats :

- Rang 1, position 5, match 4/4, résultat : ATGC
- Rang 2, position 0, match 3/4, résultat : ATGG
- Rang 3, position 1, match 2/4, résultat : TGGC
- Etc…

Cette opération de recherche devra accepter un nombre illimité de caractères en entrée et renverra les N meilleurs résultats classés (avec possibilité de définir N dans l'IHM).

MODULE #3 – LA RECHERCHE DES GENES (COMPLEXITE : ★★★)

A partir de maintenant, si votre système est bien conçu et distribue bien la charge de travail, nous allons pouvoir partir à l'exploration des gènes dans le génome. Mais il faut pour cela avoir compris certains rudiments concernant la génétique : il vous faudra consulter les excellentes vidéos du professeur François Rechenmann de l'Institut national de recherche en informatique et en automatique (INRIA) données en ressources.

Première étape : transformer les séquences d'ADN vers des acides aminés du code génétique

Comme vu en préambule, l'ADN est une protéine. Une protéine, en tant que succession d'acides aminés, peut-être vue comme le résultat d'un processus de traduction d'une chaîne écrite dans un alphabet de 4 lettres en une autre chaîne écrite en un alphabet de 20 lettres. La table qui permet de passer de l'une à l'autre, est le code génétique. En détail, un gène va commencer sur un codon d'initiation de la traduction ATG, et se terminer sur un des codons de fin de traduction que l'on appelle codon-stop. Entre les 2, une succession de triplets (3 bases successives). Vous trouverez plus d'explications ici :

Vidéo 2.3 - Le code génétique : https://www.canal-u.tv/video/inria/2_3_le_code_genetique.24574

Vidéo 2.4 à 2.7 - Algorithme de traitement : canal-u.tv/video/inria/2_4_un_algorithme_de_traduction.24576

Deuxième étape : trouver des gènes

Une fois la séquence d'un génome complet obtenue, débute la phase d'annotation. L'annotation elle-même consiste tout d'abord à rechercher la localisation, c'est-à-dire la position des gènes sur cette séquence. Mais pour cela, il faut arriver à déterminer le mieux possible la position des codons qui indiquent le début et la fin d'un gène. Vous trouverez plus d'explications ici :

Vidéo 2.10 – Comment trouver les gènes : www.canal-u.tv/video/inria/2_10_comment_trouver_les_genes.24592

Vidéo 3.1 à 3.4 – Identifier les gènes : canal-u.tv/video/inria/3_1_tous_les_genes_se_terminent_sur_un_codon_stop.24584

Vidéo 3.5 à 3.8 – Amélioration la prédiction : canal-u.tv/video/inria/3_5_comment_ameliorer_la_qualite_des_predictions

MODULE #4 – RECHERCHE AVANCEE… (COMPLEXITE : ★★★★★)

Quelques épreuves pour les plus téméraires :

- Pourriez-vous trouver des gènes communs à chaque génome ?
- Seriez-vous capable de prédire la couleur des yeux des individus dont vous avez le génome à votre disposition ? (L'accomplissement de ce haut fait vous apporterait à lui seul une aura quasi divine !)

ORGANISATION

Le projet devra être réalisé par un groupe de 2 à 3 personnes. Les livrables demandés sont :

- Le code source de la solution (archivé au format ZIP) et un exécutable compilé
- L'accès au code source sur la plateforme de versionning utilisée (GitHub, BitBucket, etc…) devra être mis à disposition du jury le jour de la soutenance.
- Un document rassemblant les schémas de modélisation

Le projet donnera lieu à une soutenance permettant de présenter : 1) l'état d'achèvement en fin de projet ; 2) la démarche de conception ; 3) la démonstration de bon fonctionnement et le benchmarking de performance.

Il vous est conseillé de mettre en place le cluster dans un premier temps, de tester en parallèle le découpage d'un calcule, et enfin d'assembler le tout pour réaliser les modules.

> **Note sur les projets étudiants publics**
>
> Afin de vous confectionner un « book » de réalisations techniques, vous avez le droit de mettre votre code source sur un repository publique comme GitHub. Néanmoins, vous comprendrez qu'il est important que les futurs étudiants après vous réalisent leur projet sans pouvoir récupérer le travail des étudiants précédents. Vous ne devez donc faire aucune mention à des éléments permettant d'identifier directement l'exercice, notamment les termes : *RIL*, *Responsable en ingénierie des logiciels*, *CESI* ou *Projet DNA*.

RESSOURCES

Cours de Bio-informatique, l'Institut national de recherche en informatique et en automatique, François RECHENMANN

Partie 1 – Compréhension du vocabulaire
Partie 2 – Algorithme de transformation séquences ADN vers acides aminés 

Partie 3 – Recherche des gènes
https://www.canal-u.tv/producteurs/inria/cours_en_ligne/bioinformatique_algorithmes_et_genomes

L'utilisation des threads avec le Framework .NET

Cours d'introduction
http://emerica.developpez.com/csharp/threads/
API de la classe Thread
https://msdn.microsoft.com/fr-fr/library/system.threading.thread(v=vs.110).aspx
Index de la documentation Microsoft à ce sujet
https://msdn.microsoft.com/en-us/library/3e8s7xdd(v=vs.110).aspx

L'utilisation de la librairie Parallel sur le Framework .NET

Index de la documentation Microsoft à ce sujet
https://msdn.microsoft.com/fr-fr/library/dd460693(v=vs.110).aspx

Le threading avec les IHM

Article Microsoft
https://msdn.microsoft.com/en-us/library/ms951089.aspx

La complexité algorithmique

Cours d'introduction
https://openclassrooms.com/courses/algorithmique-pour-l-apprenti-programmeur/la-notion-de-complexite
Article Wikipédia sur l'analyser de la complexité algorithmique
https://fr.wikipedia.org/wiki/Analyse_de_la_complexit%C3%A9_des_algorithmes