MASTER OF SCIENCE
IN ENGINEERING

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts
Western Switzerland

Master of Science HES-SO in Engineering
Av. de Provence 6
CH-1007 Lausanne

Master of Science HES-SO in Engineering

Orientation: Information and Communication Technologies (ICT)

Dockerisation d'environement pour Les projets de bioinformatique

Author:
Déruaz Vincent

Under the direction of:
Prof. Carlos Andrés Pena
CI4CB at HEIG-VD

External expert:
[Title] [FirstName] [LastName]
Company/Lab

Lausanne, HES-SO//Master, February 4, 2017

Sometimes a scream is better than a thesis.
— Manfred Eigen

To my parents...

Acknowledgements

Cette thèse as àtà réalisé dans le cadre du projet Inphinity à l'HEIG-VD.

Elle fait suite à la thèse de master [TODO: TITLE] réalisée par [TODO: prenom+nom DIOGO].

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Key words:

Résumé

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Mots clés :

Contents

List of Figures

List of Tables

1 Introduction

Ce travail a été réalisé dans le cadre du projet INPHINITY [TODO: pour qui ?]. Avec l'émergence de bactéries résistantes aux antibiotiques devenant une problématique mondiale qui menace les progrès de la médecine moderne, une alternative prometteuse pour lutter contre des bactéries multi-résistantes consiste à utiliser leurs prédateurs naturels, des bactériophages, virus mangeurs de bactéries. Ces bactériophages, innofensif pour l'homme, sont extrêmement spécifiques, ne reconnaissant qu'un type bien précis de bactéries. Ceci présente l'avantage de ne pas détériorer la flore bactérienne humaine mais pose, par contre, une limitation pour leur développement rapide. En effet, pour chaque type de bactérie il faut trouver le bactériophage correspondant. Face à la nécessité d'examiner systématiquement une multitude d'interactions possibles, le développement rapide des bactériophages comme alternative aux antibiotiques ne pourra se faire qu'avec l'aide d'un modèle permettant de prédire les interactions entre bactériophages et bactéries. Ceci permettra notamment de réduire le nombre de validations expérimentales nécessaires à l'identification du bactériophage approprié et contribuera à l'essor de cette voie thérapeutique.

Ce travail se place également dans la continuité d'une précédente thèse de master dont l'objectif était de prouver la pertinence d'une méthode d'analyse par *machine learning*. En effet, il s'agit d'une méthode permettant [TODO 1: compléter].

Dans la présente thèse, il est question de mettre en place plusieurs aspects permettant l'enrichissement du processus d'analyse de la thèse *[TODO 2: these name of diogo]*.

Afin de réaliser ces objectifs, une première phase du travail a consisté à réaliser des états de l'art pour les différents domaines utilisés (cf.chapitre 2-Etats de l'art).

Plusieurs phases distinctes de travail ont été nécessaires durant ce travail.

Premièrement, il a fallu reprendre la thèse [TODO: ref these diogo] et comprendre ce qu'il y a été fait. Les informations concernant la thèse de Mr.Leite Diogo nécessaires à la compréhension de ce travail ont été abordées dans l'introduction, pour davantage d'informations veuillez consulter la thèse en question.

Deuxièmement, une fois les objectifs de thèse fixés, il a été important de réaliser un état de l'art des différentes technologies et aspects techniques susceptibles d'êtres utilisés dans la présente thèse, voir chapitre chapitre 2-Etats de l'art .

Troisièmement, c'est uniquement après ces deux premières phases que le développement a pu commencer, voir chapitre chapitre 4-Parallelisation python3 et chapitre chapitre 5-Environnement et application. Durant cette phase un certain nombre d'aspect on été développé: Notamment, l'utilisation de python 3 afin de remplacer l'utilisation de python2, moins efficace.

De plus, on souhaite être capable d'automatiser le lancement de "l'application" et par la même occasion rendre le déploiement facile et unifier quelque soit la machine hôte, pour autant qu'elle utilise le système d'exploitation Linux. Ensuite, on souhaite pouvoir lancer l'analyse pour différentes configurations, créées à l'avance. Un autre objectif important était de remplacer l'utilisation d'une API en ligne par une utilisation de sa version locale cf.chapitre 6-Déploiement.

Finalement, le temps de travail étant limité, il faut penser aux utilisations futures de ce qui a été développé. Ceci passe notamment par l'utilisation de l'application réalisée dans ce travail de manière simple voir chapitre chapitre 7-Simplification d'usage, mais aussi par les améliorations possibles à cette thèse, voir chapitre chapitre 9-Améliorations. C'est pous cela qu'un environement de developpement et d'execution Docker à été produit dans ce travail, qui pourra êtres utile au autres membre du projet.

Il faut aussi préciser que certains résultats et métriques ont été réalisés et sont regroupés dans le chapitre chapitre 8-Résultats et Benchmarks.

TODO: Information concernant la volonté de réaliser un Docker for Bio-Informatique

2 | Etats de l'art

2.1 introduction

Dans ce chapitre nous aborderons les différentes pistes envisagées afin de remplir les objectifs fixés dans cette thèse, comme listé dans l'introduction (chapitre 1-Introduction).

Avant toutes choses il à fallu se mettre au niveau et comprendre la thèse *[Modélisation prédictive des interactions entre bactéries et virus bactériophages - Leite Diogo]* .

2.2 Automatisation

En terme d'automatisation, une pratique bien courante chez les developpeur s'agit à utiliser des script bash afin de pouvoir executer une certain nombre de commande et de code succecivement. Bien que cette méthode présente l'avantage d'etres simple, il suffit d'une console UNIX et d'un editeur de texte, elle présente un défault majeur. En effet, le developpeur du script contrôle quel commande et code sont executé et peut également definir des paramètres pour ceux-ci, mais il ne peu pas contrôlé l'environnement d'execution.

Une façcon de faire, en pleine essort depuis quelque temps, est l'utilisation de la platforme Docker. Il s'agit d'un logiciel de containerisation. C'est-à-dire la création de brique d'application qui mise en communes permette de réaliser un application global. De plus, le développement d'une telle solution, permet un partage facilité grâce à un déploiement facilité et autonome. Pour d'avantages d'explication sur le sujet je vous renvoi au chapitre chapitre 3-Bases de Docker.

Vous l'aurez bien compris, le choix qui à été fait est celui de l'utilisation de Docker.

2.3 Configuration

En ce qui concerne la recherche d'une méthode afin de réalisé facilement des fichiers de configurations pré-créées, beaucoup de solutions existent. Ces différentes méthodes sont plus ou moins flexible aux modifications.

Les fichiers de configurations dont il est question ici, sont spécifique à la partie python du code qui sera executé par notre application chapitre 5-Environnement et application. En effet, l'on souhaite entre autre être capable de donner des fichiers de configuration en entrée et d'obtenir pour chacuns un résultats en sortie.

Nous ne citerons ici uniquement la solution retenu, car les autres solutions trouvée sont soit trop incompatible soit presque identique à la solution retenu.

Nous utilisons le module python *Configparser*, qui permet de lire et parser des fichiers à l'extension .ini de manière simple. De plus, la structure d'un fichier .ini est très simple et ne laisse donc que très peu de place aux erreurs de format.

2.4 Hmmer

Dans la thèse *[Modélisation prédictive des interactions entre bactéries et virus bactériophages - Leite Diogo]* les séquences protéiniques sont recherché dans la base de données de profile-HMM à l'aide d'une interface de programmation applicative (API) en ligne. Cette API est disponible depuis le site *https://www.ebi.ac.uk/Tools/hmmer/*.

Comme dis précédement, un de objectifs de ce travail est de se passer de l'utilisation de cette API car sont accès n'est pas toujours disponible ou stable.

Une recherche rapide à permis de se rendre compte que l'application utilisé derrière cette API est disponible au téléchergement et peut donc etres utilisée de manière local. Pour d'avantage d'information chapitre 5-Environnement et application, sous-chapitre Hmmer.

2.5 Parallélisation

La version existante du code se trouvant dans la thèse *[Modélisation prédictive des interactions entre bactéries et virus bactériophages - Leite Diogo]* est une version sous forme de scrit, proof-of-concept, en python2 et non *multiprocessed*. Afin de garantir une utilisation optimal des ressources de la machine hote, sur laquelle le code est executé, nous souhaitons rendre le code parallèle là où il est possible de le faire.

Plusieures solution sont possible, encore une fois les solutions les plus compliquées ne sont pas toujours celle les plus efficaces. De plus une méthode trop complexe pourrait réduire la bonne transmition du code à d'autre développeurs.

La partie principal que l'on souhaite paralleliser est l'utilisation de la fonction de scanne de HMMMER, étant donné qu'un très grand nombre de séquences protéinique doivent être analysées.

2.5.1 Simple

Docker

Docker, mise à part de rendre le déploiement et l'execution d'application automatisée, parmet également de lancer plusieurs containers simultanéement, chapitre 3-Bases de Docker. Un container englobe un systeme de fichier complete possèdant tous se qu'y est necessaire a remplir sa fonction.

Python

En python on retrouve deux principal méthode permettant de réaliser de code parallel. En effet, on peut utiliser le *multiprocessing* ou le *multithreading*.

Notre bute est de réaliser et d'optimiser un code Central processing unit (CPU) dépendant, c'est-à-dire coeurs dépendant. Lors de l'utilisation du langage python il faut savoir qu'avec des code CPU dépendant, python limite les possibilité de parallelisme à cause de la Global Interpreter Lock (GLI). La GLI est necessaire en python, car python n'est pas *tread safe*. En effet, il y a, en python, un verrou global lorsque l'on essaye d'acceder a un objet depuis un thread.

A cause de se verrous les codes CPU dépendant ne gagnerons pas en pérformance lorsqu'ils sont parallélisé à l'aide de *multithreading*, mais uniquement avec le *multiprocessing*.

2.5.2 Avancée

Docker Swarm

Une autre méthode utilisant une librairie avancé de Docker, consiste à utiliser Docker Swarm. Docker Swarm apporte à DOcker une gestion native du *clustering*, afin de transformer un groupe de *Docker engines* en un unique et virtuel *Docker engine*. Grâce à cela il est possible d'executer une application sur un architecture partagée sur plusieurs système physiquement indépendant.

Spark

Spark est un framework *open source* de calcule distribué. Il permet d'effectuer des analyse complexes sur un grand nombre de données.

Il est également un ensemble d'outils pour le traitemnt de grandes source de données, notament grace à des fonctions *MapReduce*.

2.6 Optimisations

Le code repris de la thèse *[Modélisation prédictive des interactions entre bactéries et virus bactériophages - Leite Diogo]* est un code séquetielle, sous forme de script necessitant des input utilisateurs a chaque étapes. De plus ce code est ecrit en python dans sa version 2.

Grâce au travail du Dr. Brett Cannon, See here, on se rend compte que python 3.3 pourrait optimiser les performance de notre application. On peu lire ici que même 'appel des fonctions est en moyenne 1.20 fois plus rapide. De plus, les *threadded count* sont également plus rapide.

Une autre possibilité est d'utiliser *Cython*. Cython est un compilateur/language de python permettant d'utiliser de appele au langague C et de compiler un code python en executable C. Il faut savoir qu'un executable C est généralement plus rapide que l'execution de l'interpreteur python.

On trouve le tableau suivant dans la documentation de Cython, qui permet de nous rendre compte des différences.

Method	Time (ms)	Compared to Python	Compared to Numpy
Pure Python	183	x1	x0.03
Numpy	5.97	x31	x1
Naive Cython	7.76	x24	x0.8
Optimised Cython	2.18	x84	x2.7
Cython calling C	2.22	x82	x2.7

Figure 2.1 – Tableau de comparaison de Cython - `http://notes-on-cython.readthedocs.io/en/latest/std_dev.html/`).

2.7 conclusion

3 Bases de Docker

3.1 Introduction

3.1.1 Utilisations

3.1.2 Compatibilité inter-OS

3.1.3 Miracle or illusion

3.2 Pré-requis

3.2.1 Connaissance

3.2.2 Installations

3.2.3 Téléchargements

3.3 Fonctionnement

3.3.1 Docker

3.3.2 Docker-compose

3.3.3 Docker Swarm

3.4 Exemples

3.4.1 simple pull, build et run

3.4.2 Serveur Web

3.4.3 Biopython

3.4.4 Parallélisation

3.5 Conclusion

3.5.1 Docker

3.5.2 Alternatives

4 Parallelisation python3

4.1 Code de base

4.2 Utilisation

5 Environnement et application

5.1 Images Docker

5.1.1 Hmmer

5.1.2 Database

5.1.3 Core

5.2 Docker Compose

5.3 «Inphinity»

6 Déploiement

6.1 Obtention des sources

6.2 •

7 Simplification d’usage

7.1 Commandes et alias

7.2 Scripts

8 Résultats et Benchmarks

8.1 Parallèlisation

8.2 Dockers

8.3 Phases

9 Améliorations

9.1 Parallèlisation

9.2 Machines Amazone

9.3 Spark

10 | Conclusion

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

A | An appendix

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.