UVSQ université PARIS-SACLAY | ISTY Institut des Sciences et Techniques des Yvelines
CAMPUS DE MANTES EN YVELINES
CAMPUS DE SAINT-QUENTIN-EN-YVELINES

CALCUL HAUTE PERFORMANCE SIMULATION

ISTY UVSQ
Université Paris-Saclay

M1 ***C****alcul* ***H****aute* ***P****erformance,* ***S****imulation*
Projet de l'Architecture Parallèle
Rapport de Projet

Code génétique, distance de Hamming et matrices stochastiques

---

Réalisé par :
Mme. Rim ACHRRAB

Encadré par :
M. Ibnammar M.Salah

Promotion : 2022/2023

Résumé

Mots clés :

Abstract

Keywords :

Table des matières

Résumé i

Abstract i

Introduction générale 1

1 Résumé de l'article 2

2 Réalisation 3

2.1 Explication du programme "genseq.c" 3
2.2 Explication du code "dist.c" 3
2.3 Fichier de type DNA 3
2.4 L'alignement mémoire 4
2.5 Remarque initiale du code 4
2.6 Optimisation 4

Table des figures

Introduction générale

1 Résumé de l'article

L'article se repose sur la construction d'une matrice stochastique basée sur le code génétique en utilisant la distance de Hamming et le code GRAY à 2 bits ($\{00, 01, 10, 11\}$) pour attribuer des valeurs aux bases génétiques $\{C, A, G, U\}$. Le domaine des mathématiques exprime cette relation par des fonctions utilisées pour coder les informations génétiques par des triplets de quatre caractères de l'ensemble $\{A, C, G, U\}$.

L'article nous montre trois types de tables de ce code génétique qui ont été révélées dans les différentes littératures et domaines grâce à leurs structures symétriques. La première table est basée sur le code Gray, la deuxième est appelée tableau bi-périodique et la troisième est générée à partir d'un arbre de 4-aires. Les chercheurs utilisent ces tables pour relier le code génétique avec les matrices en utilisant le code Gray, puis ils calculent la distance de Hamming pour chaque code Gray obtenu, puis ils génèrent des matrices numériques qu'ils doivent être stochastiques ou stochastiques doubles.

La distance de Levenshtein est plus sophistiquée.Elle est définie pour des chaînes de longueur arbitraire. Elle calcule les différences entre deux chaînes de caractères, alors que l'on compterait une différence non seulement quand les chaînes ont des caractères différents, mais aussi quand l'une a un caractère alors que l'autre n'en a pas.

2 Réalisation

2.1 Explication du programme "genseq.c"

Ce programme est un générateur de séquences d'ADN (A,C,G et T) d'une façon aléatoire.

En entrée le code prend le nom du fichier de sortie et la longueur souhaitée pour la chaîne de caractères d'ADN. La fonction srand est employée pour initialiser le générateur de nombres aléatoires en utilisant l'identifiant de processus (getpid) pour obtenir une séquence aléatoire différente chaque exécution (pour éviter les redondances). La fonction randxy a pour utilité la génération d'un nombre aléatoire entre 0 et 4 sera utilisé pour sélectionner un caractère des bases d'ADN (A, T, C ou G) pour l'ajouter ensuite à la séquence.

2.2 Explication du code "dist.c"

Ce programme calcule la distance de Hamming entre deux séquences d'ADN. La distance de Hamming est le nombre de positions où les deux séquences sont différentes (comme j'ai expliqué dans le résumé de l'article).

Le programme définit des structures de données pour les séquences d'ADN (seq_t) et des codes d'erreur pour les erreurs potentielles. Il utilise des fonctions pour charger les séquences d'un fichier donné, libérer de la mémoire utilisée pour les historiques et calculer la distance de Hamming entre les séquences une autre fois.

La fonction load_seq() est utilisée pour charger les séquences d'ADN à partir de fichiers en utilisant la fonction stat() pour obtenir la taille du fichier, puis en libérant de la mémoire pour stocker les séquences et en lisant les octets du fichier dans la mémoire allouée.

La fonction release_seq() est utilisée pour libérer la mémoire occupée allouée pour les séquences.

La fonction hamming() est utilisée pour calculer la distance de Hamming entre les séquences en utilisant la fonction __builtin_popcount() pour compter le nombre de bits différents entre les séquences en utilisant l'opérateur binaire XOR (renvoie 1 si les deux séquences sont différentes, et 0 s'elles sont égaux).

Dans la fonction main(), Le programme prend en entrée les noms des fichiers contenant les séquences d'ADN et utilise les fonctions précédentes pour charger les séquences afin de calculer la distance de Hamming et libérer la mémoire utilisée pour les séquences.

2.3 Fichier de type DNA

Un fichier de type ADN est utilisé pour stocker des informations génétiques sous forme de séquences d'ADN. Ces séquences d'ADN peuvent être utilisées pour une variété de domaines, notamment l'analyse génétique, médecine génétique, biotechnologie, conservation de la biodiversité,... .

2.4 L'alignement mémoire

En C, l'alignement mémoire est utilisé pour améliorer les performances en accélérant l'accès aux données en mémoire de l'ordinateur. Les processeurs utilisent des caches pour accélérer la vitesse de l'accès aux données en mémoire. Les caches sont organisés en cache lines, qui sont généralement de la taille de quelques octets. Lorsque le processeur lit ou écrit des données en mémoire, il lit ou écrit généralement un cache line complet, même si seul un octet de données est nécessaire.

Lorsque les données sont alignées sur les limites des cache lines, le processeur peut accéder aux données plus rapidement, car il n'a pas besoin de lire ou d'écrire des données inutiles. Si les données ne sont pas alignées sur les limites des cache lines, le processeur doit lire ou écrire des données supplémentaires, ce qui ralentit l'accès aux données.

En utilisant l'alignement mémoire, on peut améliorer les performances en s'assurant que les données sont alignées sur les limites des cache lines. Il est à noter que cela n'est pas nécessaire pour tous les types de données, et que cela peut causer des problèmes de mémoire si les données sont trop grandes pour tenir dans une cache line.

Dans le code, on doit utiliser l'alignement mémoire car en fait la distance de Hamming acceptent les caractères de qui ont un nombre d'octets identique.

2.5 Remarque initiale du code

Sans alignement du code, on remarque que la distance de Hamming est égale à 0 et le temps de l'éxecution du code se diffère suivant chaque flag d'optimisation.

2.6 Optimisation

L'utilisation de la fonction popcount() dans est optimale pour compter le nombre de bits qui sont différents entre deux octets (elle évite les duplication). Cette fonction est une intrinsic, ce qui signifie qu'elle est directement implémentée dans le compilateur et peut tirer parti de l'ensemble d'instructions de l'architecture cible pour effectuer l'opération efficacement. Cependant, dans certains cas, il peut être possible d'optimiser les performances de la fonction en utilisant un algorithme plus spécialisé, comme une table de recherche, qui est optimisé pour l'utilisation spécifique. Cela dépend des exigences et des contraintes spécifiques de l'application.

Vectorisation :
La directive pragma omp simd indique au compilateur de vectoriser la boucle, et la clause reduction (+ : h) spécifie que la variable h doit être traitée comme une variable de réduction, ce qui signifie qu'elle est cumulée sur toutes les itérations dans la boucle vectorisée.