



République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche
Scientifique



Université des Sciences et de la Technologie Houari Boumediène

Faculté d'Informatique

Département des Systèmes Informatiques (SIQ)

Mémoire de Master

Filière : Informatique

Spécialité : High Performance Computing (HPC)

Thème :
**Parallélisation de la métaheuristique
BSO-DOCK sur processeur graphique GPU avec
OpenACC**

Encadré par :

Mr. Hocine Saadi

Mme. Malika Mehdi

Présenté par :

Sonia Zemmour

Membre du Jury :

Mr. Ahmed Riadh Baba Ali

Mme. Karima Bouibede

Soutenu le :

XX/06/2024

Projet : HPC_005 / 2024

Dédicace

À la jeune Sonia de 18 ans, qui avait fait son premier pas à l'USTHB en octobre 2019,
YOU DID IT .

Remerciements

Tout d'abord, je loue Allah Tout-Puissant, pour m'avoir accordé la force et la persévérance nécessaires pour mener à bien ce travail.

Mes sincères remerciements vont à Monsieur Riadh Ahmed Baba Ali et Madame Karime Bouibede, membres éminents du jury, pour leur engagement et leur volonté d'évaluer mon travail.

Je suis particulièrement reconnaissante envers Monsieur Hocine Saadi et Madame Malika Mehdi, mes encadrants dévoués, pour leur soutien infaillible, leur patience et leur encouragement constant tout au long de ce travail. Leur expertise et leurs orientations ont été cruciales pour orienter mes efforts dans la bonne direction.

Je souhaite également exprimer ma profonde reconnaissance envers mes chers parents, ainsi qu'à mes sœurs aimantes, Nadia et Lynda. Leur amour indéfectible, leurs sacrifices et leur soutien sans faille ont été ma source de force et de motivation tout au long de ce parcours académique.

Enfin, je tiens à remercier toutes les personnes qui, de près ou de loin, ont contribué à la réalisation de ce travail.

Résumé

Dans les domaines de la bioinformatique et de la chimie computationnelle, le docking moléculaire est une technique essentielle pour prédire la position et l'orientation d'une molécule (ligand) lorsqu'elle interagit avec une protéine cible. Cette méthode est cruciale pour la découverte de médicaments, l'étude des mécanismes moléculaires et la compréhension des interactions biomoléculaires à l'échelle atomique. Cependant, la nature NP-difficile du docking moléculaire présente des défis computationnels importants. Ce projet explore l'utilisation de l'optimisation par essaim d'abeilles (Bee Swarm Optimization, BSO) et sa parallélisation avec OpenACC sur les accélérateurs (comme les GPU) pour relever ces défis. L'objectif est de déterminer dans quelle mesure la parallélisation sur GPU réduit le temps d'exécution par rapport aux implémentations séquentielles.

- **Mots-clés** : docking moléculaire, bioinformatique, , optimisation par essaim d'abeilles, BSO, OpenACC, parallélisation GPU.

Abstract

In the fields of bioinformatics and computational chemistry, molecular docking is a critical technique for predicting the position and orientation of a molecule (ligand) when it interacts with a target protein. This method is crucial for drug discovery, studying molecular mechanisms, and understanding biomolecular interactions at the atomic scale. However, the NP-hard nature of molecular docking poses significant computational challenges. This project explores the use of Bee Swarm Optimization (BSO) and its parallelization with OpenACC for accelerators like GPUs to address these challenges. The goal is to determine to what extent GPU parallelization reduces execution time compared to sequential implementations.

- **Keywords** : molecular docking, bioinformatics, Bee Swarm Optimization, BSO, OpenACC, GPU parallelization.

Table des matières

Résumé

Abstract

Liste des figures

Liste des tableaux

Liste des algorithmes

Liste des abréviations

Introduction Générale	1
1 Docking moléculaire et Méta-heuristiques	4
1.1 Introduction	4
1.2 Docking moléculaire	4
1.2.1 Définition de l'espace de recherche	5
1.2.2 Principe de fonctionnement	5
1.3 Cas d'utilisation du docking moléculaire	6
1.4 La classification du docking moléculaire	7
1.4.1 Docking rigide	7
1.4.2 Docking semi-flexible	7
1.4.3 Docking flexible	7
1.5 Les Défis du Docking moléculaire	8
1.5.1 L'exploration de l'espace de recherche	8
1.5.2 Les fonctions objectif	8
1.6 Les algorithmes de recherche : Les méta-heuristiques	9
1.6.1 Métaheuristiques à solution unique	9
Recuit simulé	9
Recherche Tabou	10
1.6.2 Les méta-heuristiques à population de solutions	12
les algorithmes génétiques	12
Algorithme de Colonie de fourmis	13
L'optimisation par essaim de particules	14
L'optimisation par essaims d'abeilles	15

1.7	Fonction objectif	16
1.7.1	Fonction de score basée sur les champs de force	16
1.7.2	Fonctions de score empiriques	16
1.7.3	Fonctions de score basées sur les connaissances	17
1.8	Conclusion	17
2	Programmation parallèle sur GPU et le standard OpenACC	18
2.1	Introduction	18
2.2	Limites de la programmation CPU	18
2.3	Notion de Parallélisme	20
2.3.1	Types de parallélisme	20
2.3.2	Classification des architectures parallèles	20
2.4	Programmation parallèle sur GPU (GPGPU)	22
2.4.1	Graphics Processing Unit (GPU)	22
2.4.2	Architecture d'un GPU VS. Architecture d'un CPU	22
2.5	Coopération CPU-GPU	23
2.6	Outils et API pour la coopération CPU-GPU	25
2.6.1	CUDA™	26
2.6.2	OpenCL™	27
2.6.3	OpenMP™	27
2.6.4	ROCm™	28
2.6.5	OpenACC™	28
2.7	OpenACC	28
2.7.1	Historique	29
2.7.2	Les directives	29
2.7.3	Les niveaux de parallélisme dans OpenACC	30
	Le parallélisme de vecteurs (vectorisation)	30
	Le parallélisme de travailleurs (workers)	31
	Le parallélisme de Gangs	31
2.7.4	Compilateur OpenACC	32
2.8	Conclusion	32
3	Conception	33
3.1	Introduction	33
3.2	Étape 01 : Analyse de la méta-heuristique BSO-DOCK séquentielle	33
3.2.1	L'encodage de la solution	34
3.2.2	La fonction d'évaluation (fonction objectif)	34
3.2.3	Déterminaisons des régions	35
3.2.4	Recherche du voisinage	35
3.2.5	Algorithme BSO-DOCK général	36
3.3	Étape 02 : Parallélisation de la métaheuristiques BSO-DOCK	37
3.3.1	Identification des parties parallélisables	37
3.3.2	Stratégies de parallélisation	37
	Stratégie 01 : Parallélisation automatique par le compilateur	37
	Stratégie 02 : Parallélisation guidée par le programmeur	41

3.3.3	Étape 03 : Optimisation avancée de la parallélisation de la fonction objectif	43
3.4	Conclusion	45
4	Implémentation et Résultats	46
4.1	Introduction	46
4.2	Environnement de développement	46
4.2.1	Environnement matériel	46
4.2.2	Environnement logiciel et outils de développement	47
4.3	Le data-sets utilisé	48
4.4	Les structures de données utilisés	48
4.5	Ajustement des paramètres pour les trois niveaux de parallélisme	50
4.6	Résultats de l'étude expérimentale	50
4.6.1	Test du Fonctionnement	50
	Sur GPU	51
	Sur CPU	53
4.6.2	Test de performance	53
4.7	Analyse des Résultats et Discussions	55
4.8	Conclusion	56
	Conclusion Générale	57
	Annexes	59
A	Annexe 01 : Architecture Ampere	59
A.1	Introduction	59
A.2	Caractéristiques Principales de l'architecture Ampère	59
A.3	Applications de l'Architecture Ampere	60
A.4	Conclusion	60
B	Annexe 02 : Fiche de Lecture	61
	Bibliographie	63

Listes des figures

1.1	Schéma explicatif du Docking moléculaire[5]	5
1.2	Les paramètres d'un ligand[6]	5
1.3	Schéma explicatif du fonctionnement du docking moléculaire[7]	6
1.4	Schéma explicatif sur les types du docking moléculaire [12]	8
1.5	Les catégories de méta-heuristiques	9
1.6	Schéma explicatif sur la recherche Tabou	11
1.7	Schéma explicatif sur les étapes d'un algorithme génétique	13
1.8	Schéma explicatif sur le processus de l'algorithme de colonie de fourmis[24]	14
2.1	Loi de Moore [31]	19
2.2	Fin de l'augmentation d'horloge pour un (01) processeur[32]	19
2.3	Les différents types d'architecture [37]	21
2.4	Architecture CPU VS. Architecture GPU [43]	23
2.5	Temps de tri coopératif par rapport aux temps de tri avec GPU seulement [45]	24
2.6	L'utilisation de la mémoire unifiée [48]	25
2.7	Structure d'un code CUDA	26
2.8	Un schéma représentant la subdivision d'une grille en blocs et chaque bloc en threads [51]	27
2.9	Comparaison de l'organisation du travailleur, du vecteur et du gang dans OpenACC avec la hiérarchie des threads CUDA [66]	31
3.1	Fonction objectif	34
4.1	La molécule Cutinase	48
4.2	Exécution du code séquentiel	51
4.3	Exécution du code parallèle (la stratégie 01) sur GPU	51
4.4	Exécution du code parallèle (la stratégie 02) sur GPU	52
4.5	Exécution du code parallèle optimisé sur GPU	52
4.6	Exécution du code parallèle optimisé sur CPU	53
4.7	Accélération pour différentes stratégies	55

Listes des tableaux

4.1	Temps d'exécution et accélérations par stratégie	54
B.1	Fiche de lecture de l'article - Partie 1	61
B.2	Fiche de lecture de l'article - Partie 2	62

Liste des algorithmes

1	L'algorithme général de la méthode recuit simulé	10
2	Algorithme de Recherche Tabou pour le Docking	12
3	Particle Swarm Optimization (PSO)	15
4	Algorithme général BSO-DOCK	36
5	Pseudo-code pour Calcul parallèle de l'énergie	38
6	Évaluation parallèle de la solution	39
7	Parallélisation de la fonction <i>searcharea</i>	40
8	Parallélisation de l'appel de la fonction <i>NeighborhoodComputation</i>	41
9	Pseudo-code pour le Calcul Parallèle de l'Énergie de Van der Waals (stratégie 02)	42
10	Parallélisation de la fonction <code>energy_sort</code> (stratégie 02)	43
11	Pseudo-code pour le Calcul Parallèle des énergies (optimisation)	44

Liste des abréviations

- **ACO** : Ant Colony Optimization
- **AMD** : Advanced Micro Devices
- **API** : Application Programming Interface
- **ARM** : Advanced RISC Machines (ou Architecture ARM)
- **BSO** : Bee Swarm Optimization
- **CUDA** : Compute Unified Device Architecture
- **DLSS** : Deep Learning Super Sampling
- **DM** : Docking Moléculaire
- **GPU** : Graphics Processing Unit
- **GPGPU** : General-Purpose Graphics Processing Unit
- **GCC** : GNU Compiler Collection
- **HCC** : Heterogeneous Compute Compiler
- **HPC** : High-Performance Computing
- **IA** : Intelligence Artificielle
- **MIMD** : Multiple Instruction, Multiple Data
- **MISD** : Multiple Instruction, Single Data
- **NP** : Non-deterministic Polynomial
- **OpenACC** : Open Accelerators
- **OpenCL** : Open Computing Language
- **OpenMP** : Open Multi-Processing
- **PGI** : The Portland Group

- **PSO** : Particle Swarm Optimization
- **RT** : Ray Tracing
- **SIMD** : Single Instruction, Multiple Data
- **SISD** : Single Instruction, Single Data
- **SM** : Streaming Multiprocessor
- **SPMD** : Single Program, Multiple Data
- **UAL** : Unité Arithmétique et Logique
- **VLSI** : Very-Large-Scale Integration