Rapport Analyse des performances et optimisation d'un noyau de simulation d'interactions entre N-corps dans un espace 3D

fait par :Hanane FIDAH

Table des matières

1. Introduction

L'essence de ce projet réside dans l'amélioration des performances d'un code de simulation à N-corps en explorant diverses approches d'optimisation. Nous entreprendrons l'analyse approfondie du code de base, expérimenterons avec différents compilateurs, et apporterons des modifications pour mieux adapter le code à l'architecture spécifique de la machine. L'objectif fondamental est de réduire le temps d'exécution et d'optimiser l'utilisation des ressources, conduisant ainsi à une simulation à N-corps plus efficace. En évaluant les résultats à chaque étape, nous chercherons à identifier les configurations les plus performantes, fournissant finalement des recommandations pratiques pour des simulations à N-corps optimisées.

2. Environnement

Compilateurs/versions	gcc (Ubuntu 9.4.0-1ubuntu1~20.04.2) 9.4.0 clang version 10.0.0-4ubuntu1
Architecture	**Architecture :** x86_64 **Mode opérationnel du CPU :** 32-bit, 64-bit **Ordre des octets (Byte Order) :** Little Endian **Tailles d'adresse :** 46 bits physique, 48 bits virtuel **Nombre de CPU :** 32 **Nombre de threads par cœur :** 2 **Node NUMA (Non-Uniform Memory Access) :** 2 **Modèle du CPU :** Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60GHz **Fréquence du CPU :** 1718.054 MHz **Cache L1d :** 512 KiB **Cache L1i :** 512 KiB**Cache L2 :** 4 MiB **Cache L3 :** 40 MiB

3. SECTION1 :

Version1 :

Introduction : L'objectif de cette expérience est d'évaluer l'impact des optimisations apportées au code de simulation des interactions entre particules.
Les principaux points à examiner sont les changements structuraux dans le code, tels que l'utilisation d'une structure de tableau (`particle_t`) au lieu d'un tableau de structure , l'allocation de mémoire améliorée, et l'utilisation de pointeurs , la fonction compute_delta a été implémentée mais non utilisée.
Pour mener cette expérience, des flags de compilation O2 O3 Ofast on été utilisé pour mesurer les performances du code avec deux compilateurs différents GCC et CLANG , notamment le temps d'exécution et le taux de GFLOP/s.
Les transformations du code introduites dans la version 1 sont évaluées pour leur efficacité en termes d'optimisation.

Résultats :
Le code a été compilé et exécuté avec un nombre de particules 16384 (par défaut) et voici les résultats obtenus pour les différents flags d'optimisation et les deux compilateurs GCC et Clang :

GCC (O2 | O3 | Ofast)

Total memory size: 786432 B, 768 KiB, 0 MiB

Step	Time, s	Interact/s	GFLOP/
0	8.163e+00	3.288e+07	0.6 (warm up)
1	7.914e+00	3.392e+07	0.6 (warm up)
2	7.835e+00	3.426e+07	0.6 (warm up)
3	7.835e+00	3.426e+07	0.6
4	7.835e+00	3.426e+07	0.6
5	7.833e+00	3.427e+07	0.6
6	7.834e+00	3.427e+07	0.6
7	7.833e+00	3.427e+07	0.6
8	7.831e+00	3.428e+07	0.6
9	7.834e+00	3.427e+07	0.6
10	7.872e+00	3.410e+07	0.6
11	7.914e+00	3.392e+07	0.6
12	7.878e+00	3.407e+07	0.6

Average performance : 0.6 +- 0.0 GFLOP/s

Total memory size: 786432 B, 768 KiB, 0 MiB

Step	Time, s	Interact/s	GFLOP/s
0	8.191e+00	3.277e+07	0.6 (warm up)
1	7.872e+00	3.410e+07	0.6 (warm up)
2	7.957e+00	3.374e+07	0.6 (warm up)
3	7.894e+00	3.401e+07	0.6
4	7.901e+00	3.398e+07	0.6
5	7.920e+00	3.389e+07	0.6
6	7.841e+00	3.423e+07	0.6
7	7.841e+00	3.423e+07	0.6
8	7.845e+00	3.422e+07	0.6
9	7.884e+00	3.405e+07	0.6
10	7.841e+00	3.423e+07	0.6
11	7.876e+00	3.408e+07	0.6
12	7.888e+00	3.403e+07	0.6

Average performance: 0.6 +- 0.0 GFLOP/s

Total memory size: 786432 B, 768 KiB, 0 MiB

Step	Time, s	Interact/s	GFLOP/s
0	8.396e-01	3.197e+08	5.4 (warm up)
1	6.294e-01	4.265e+08	7.3 (warm up)
2	6.292e-01	4.266e+08	7.3 (warm up)
3	7.056e-01	3.804e+08	6.5
4	6.292e-01	4.266e+08	7.3
5	6.291e-01	4.267e+08	7.3
6	6.290e-01	4.268e+08	7.3
7	6.290e-01	4.268e+08	7.3
8	6.297e-01	4.263e+08	7.2
9	6.292e-01	4.266e+08	7.3
10	6.290e-01	4.267e+08	7.3
11	6.295e-01	4.264e+08	7.2
12	6.295e-01	4.264e+08	7.2

Average performance: 7.2 +- 0.2 GFLOP/s

Clang (O2 | O3 | Ofast)

Total memory size: 786432 B, 768 KiB, 0 MiB

Step	Time, s	Interact/s	GFLOP/
0	8.858e+00	3.031e+07	0.5 (warm up)
1	8.577e+00	3.130e+07	0.5 (warm up)
2	8.542e+00	3.142e+07	0.5 (warm up)
3	8.542e+00	3.143e+07	0.5
4	8.622e+00	3.113e+07	0.5
5	8.596e+00	3.123e+07	0.5
6	8.588e+00	3.126e+07	0.5
7	8.544e+00	3.142e+07	0.5
8	8.612e+00	3.117e+07	0.5
9	8.587e+00	3.126e+07	0.5
10	8.546e+00	3.141e+07	0.5
11	8.544e+00	3.142e+07	0.5
12	8.541e+00	3.143e+07	0.5

Average performance: 0.5 +- 0.0 GFLOP/s

Total memory size: 786432 B, 768 KiB, 0 MiB

Step	Time, s	Interact/s	GFLOP/s
0	8.661e+00	3.099e+07	0.5 (warm up)
1	8.453e+00	3.176e+07	0.5 (warm up)
2	8.457e+00	3.174e+07	0.5 (warm up)
3	8.459e+00	3.173e+07	0.5
4	8.456e+00	3.175e+07	0.5
5	8.538e+00	3.144e+07	0.5
6	8.455e+00	3.175e+07	0.5
7	8.531e+00	3.147e+07	0.5
8	8.453e+00	3.176e+07	0.5
9	8.451e+00	3.176e+07	0.5
10	8.457e+00	3.174e+07	0.5
11	8.450e+00	3.177e+07	0.5
12	8.452e+00	3.176e+07	0.5

Average performance: 0.5 +- 0.0 GFLOP/s

Total memory size: 786432 B, 768 KiB, 0 MiB

Step	Time, s	Interact/s	GFLOP/s
0	8.352e-01	3.214e+08	5.5 (warm up)
1	6.171e-01	4.350e+08	7.4 (warm up)
2	6.172e-01	4.349e+08	7.4 (warm up)
3	6.168e-01	4.352e+08	7.4
4	6.175e-01	4.347e+08	7.4
5	6.169e-01	4.351e+08	7.4
6	6.176e-01	4.346e+08	7.4
7	6.166e-01	4.353e+08	7.4
8	6.180e-01	4.343e+08	7.4
9	6.166e-01	4.353e+08	7.4
10	6.162e-01	4.356e+08	7.4
11	6.168e-01	4.352e+08	7.4
12	6.174e-01	4.348e+08	7.4

Average performance: 7.4 +- 0.0 GFLOP/s

Conclusion : l'introduction de la structure de tableau a permis une gestion de la mémoire plus efficace et une simplification du code. Les résultats montrent une amélioration significative des performances, en particulier avec des ensembles de données plus importants. Le temps d'exécution a diminué, le taux d'interactions par seconde a augmenté, et le taux de GFLOP/s a également montré une amélioration ,cependant il n'y a pas de différence entre les deux compilateurs utilisés avec les différents flags d'optimisation.
Ces résultats suggèrent que les optimisations apportées ont un impact positif sur les performances du code.

4. SECTION 2 :

Version2 :

Introduction : le principal point abordé dans cette expérience est de substituer le calcul de la distance entre les particules , en utilisant une racine carrée pour obtenir une distance euclidienne.

Dans la version précédente, la distance `d_3_over_2` était calculée en utilisant la puissance 3/2 de `d_2` directement. Dans la version 2, on a introduit une étape supplémentaire pour calculer la racine carrée de `d_2` et ensuite élever cela à la puissance 3, obtenant ainsi `d_3_over_2`.

Cette modification est une approche alternative pour éviter l'utilisation de la fonction `pow` qui peut être coûteuse en termes de performances.

Pour mener cette expérience, des flags de compilation O2 O3 Ofast on été utilisé pour mesurer les performances du code avec deux compilateurs différents GCC et CLANG , notamment le temps d'exécution et le taux de GFLOP/s.

Les transformations du code introduites dans la version 2 sont évaluées pour leur efficacité en termes d'optimisation.

Résultats : Le code a été compilé et exécuté avec un nombre de particules 16384 (par défaut) et voici les résultats obtenus pour les différents flags d'optimisation et les deux compilateurs GCC et Clang :

GCC (O2 | O3 | Ofast)

Total memory size: 786432 B, 768 KiB, 0 MiB

Step	Time, s	Interact/s	GFLOP/s	
0	1.338e+00	2.007e+08	3.4	(warm up)
1	9.528e-01	2.817e+08	4.8	(warm up)
2	9.539e-01	2.814e+08	4.8	(warm up)
3	9.539e-01	2.814e+08	4.8	
4	9.532e-01	2.816e+08	4.8	
5	9.537e-01	2.815e+08	4.8	
6	9.532e-01	2.816e+08	4.8	
7	9.537e-01	2.815e+08	4.8	
8	1.003e+00	2.675e+08	4.5	
9	9.817e-01	2.734e+08	4.6	
10	9.533e-01	2.816e+08	4.8	
11	9.525e-01	2.818e+08	4.8	
12	9.547e-01	2.812e+08	4.8	

Average performance: 4.7 +- 0.1 GFLOP/s

Total memory size: 786432 B, 768 KiB, 0 MiB

Step	Time, s	Interact/s	GFLOP/s	
0	1.196e+00	2.244e+08	3.8	(warm up)
1	9.497e-01	2.826e+08	4.8	(warm up)
2	9.501e-01	2.825e+08	4.8	(warm up)
3	9.495e-01	2.827e+08	4.8	
4	9.506e-01	2.824e+08	4.8	
5	9.498e-01	2.826e+08	4.8	
6	9.486e-01	2.830e+08	4.8	
7	9.503e-01	2.825e+08	4.8	
8	9.487e-01	2.830e+08	4.8	
9	9.499e-01	2.826e+08	4.8	
10	9.500e-01	2.826e+08	4.8	
11	9.488e-01	2.829e+08	4.8	
12	9.490e-01	2.829e+08	4.8	

Average performance: 4.8 +- 0.0 GFLOP/s

Total memory size: 786432 B, 768 KiB, 0 MiB

Step	Time, s	Interact/s	GFLOP/s	
0	4.052e-01	6.624e+08	11.3	(warm up)
1	3.179e-01	8.445e+08	14.4	(warm up)
2	2.736e-01	9.812e+08	16.7	(warm up)
3	2.553e-01	1.052e+09	17.9	
4	2.557e-01	1.050e+09	17.9	
5	2.556e-01	1.050e+09	17.9	
6	2.553e-01	1.052e+09	17.9	
7	2.556e-01	1.050e+09	17.9	
8	2.554e-01	1.051e+09	17.9	
9	2.554e-01	1.051e+09	17.9	
10	2.553e-01	1.052e+09	17.9	
11	2.558e-01	1.049e+09	17.8	
12	2.547e-01	1.054e+09	17.9	

Average performance: 17.9 +- 0.0 GFLOP/s

Clang (O2 | O3 | Ofast)

Total memory size: 786432 B, 768 KiB, 0 MiB

Step	Time, s	Interact/s	GFLOP/s
0	2.003e+00	1.340e+08	2.3 (warm up)
1	1.786e+00	1.503e+08	2.6 (warm up)
2	1.785e+00	1.504e+08	2.6 (warm up)
3	1.785e+00	1.504e+08	2.6
4	1.785e+00	1.503e+08	2.6
5	1.784e+00	1.504e+08	2.6
6	1.785e+00	1.504e+08	2.6
7	1.786e+00	1.503e+08	2.6
8	1.785e+00	1.504e+08	2.6
9	1.785e+00	1.504e+08	2.6
10	1.785e+00	1.504e+08	2.6
11	1.787e+00	1.502e+08	2.6
12	1.786e+00	1.503e+08	2.6

Average performance: 2.6 +- 0.0 GFLOP/s

Total memory size: 786432 B, 768 KiB, 0 MiB

Step	Time, s	Interact/s	GFLOP/s
0	2.016e+00	1.331e+08	2.3 (warm up)
1	1.801e+00	1.490e+08	2.5 (warm up)
2	1.721e+00	1.560e+08	2.7 (warm up)
3	1.721e+00	1.560e+08	2.7
4	1.719e+00	1.561e+08	2.7
5	1.718e+00	1.562e+08	2.7
6	1.720e+00	1.560e+08	2.7
7	1.723e+00	1.558e+08	2.6
8	1.721e+00	1.560e+08	2.7
9	1.722e+00	1.559e+08	2.7
10	1.800e+00	1.491e+08	2.5
11	1.721e+00	1.560e+08	2.7
12	1.721e+00	1.559e+08	2.7

Average performance: 2.6 +- 0.0 GFLOP/s

Total memory size: 786432 B, 768 KiB, 0 MiB

Step	Time, s	Interact/s	GFLOP/s
0	2.974e-01	9.026e+08	15.3 (warm up)
1	2.160e-01	1.243e+09	21.1 (warm up)
2	2.164e-01	1.241e+09	21.1 (warm up)
3	1.845e-01	1.455e+09	24.7
4	1.738e-01	1.545e+09	26.3
5	1.737e-01	1.545e+09	26.3
6	1.741e-01	1.542e+09	26.2
7	1.739e-01	1.544e+09	26.2
8	1.741e-01	1.541e+09	26.2
9	1.738e-01	1.544e+09	26.3
10	1.737e-01	1.545e+09	26.3
11	1.742e-01	1.541e+09	26.2
12	1.745e-01	1.538e+09	26.2

Average performance: 26.1 +- 0.5 GFLOP/s

5. SECTION 3 :

VERSION 3 :

Introduction:

L'objectif de cette expérience est d'évaluer les performances de la version 3 du code de simulation des interactions entre particules, intégrant une optimisation du calcul de la racine cubique utilisée dans la loi de Newton. La principale modification concerne le remplacement de la fonction `pow` par une méthode d'inversion de racine carrée rapide (`inverse_square_root`). Nous examinerons les changements structurels, tels que l'utilisation d'une structure de tableau (`particle_t`)

Résultats : Le code a été compilé et exécuté avec un nombre de particules 16384 (par défaut) et voici les résultats obtenus pour les différents flags d’optimisation et les deux compilateurs GCC et Clang :

gcc (O2 | O3 | Ofast)

Total memory size:786432 B, 768 KiB, 0 MiB

Step	Time, s	Interact/s	GFLOP/s
0	8.441e-02	3.180e+09	54.1 (warm up)
1	7.769e-02	3.455e+09	58.7 (warm up)
2	5.107e-02	5.257e+09	89.4 (warm up)
3	8.708e-02	3.083e+09	52.4
4	6.461e-02	4.155e+09	70.6
5	6.443e-02	4.166e+09	70.8
6	5.919e-02	4.535e+09	77.1
7	6.081e-02	4.415e+09	75.0
8	4.332e-02	6.197e+09	105.4
9	4.746e-02	5.655e+09	96.1
10	4.782e-02	5.614e+09	95.4
11	4.129e-02	6.501e+09	110.5
12	4.128e-02	6.503e+09	110.5

Average performance: 86.4 +- 18.9 GFLOP/s

Total memory size: 786432 B, 768 KiB, 0 MiB

#[1m Step	Time, s	Interact/s	GFLOP/s
0	8.426e-02	3.186e+09	54.2 (warm up)
1	7.785e-02	3.448e+09	58.6 (warm up)
2	6.207e-02	4.325e+09	73.5 (warm up)
3	8.147e-02	3.295e+09	56.0
4	6.054e-02	4.434e+09	75.4
5	6.044e-02	4.441e+09	75.5
6	5.233e-02	5.129e+09	87.2
7	6.063e-02	4.428e+09	75.3
8	5.423e-02	4.950e+09	84.1
9	4.784e-02	5.612e+09	95.4
10	4.784e-02	5.611e+09	95.4
11	4.929e-02	5.446e+09	92.6
12	5.050e-02	5.316e+09	90.4

Average performance: 82.7 +- 11.7 GFLOP/s

Total memory size: 786432 B, 768 KiB, 0 MiB

Step	Time, s	Interact/s	GFLOP/s
0	3.719e-02	7.217e+09	122.7 (warm up)
1	2.892e-02	9.283e+09	157.8 (warm up)
2	2.603e-02	1.031e+10	175.3 (warm up)
3	2.626e-02	1.022e+10	173.8
4	2.606e-02	1.030e+10	175.1
5	2.642e-02	1.016e+10	172.7
6	2.627e-02	1.022e+10	173.7
7	2.610e-02	1.028e+10	174.8
8	2.018e-02	1.330e+10	226.1
9	2.236e-02	1.201e+10	204.1
10	2.216e-02	1.211e+10	206.0
11	2.220e-02	1.209e+10	205.6
12	2.227e-02	1.205e+10	204.9

Average performance: 191.7 +- 18.6 GFLOP/s

clang (O2 | O3 | Ofast)

#[1mTotal memory size:#[0m 786432 B, 768 KiB, 0 MiB	#[1mTotal memory size:#[0m 786432 B, 768 KiB, 0 MiB	#[1mTotal memory size:#[0m 786432 B, 768 KiB, 0 MiB
#[1m Step Time, s Interact/s GFLOP/s#[0m	#[1m Step Time, s Interact/s GFLOP/s#[0m	#[1m Step Time, s Interact/s GFLOP/s#[0m
0 2.237e-01 1.200e+09 20.4 (warm up)	0 2.502e-01 1.073e+09 18.2 (warm up)	0 2.097e-01 1.280e+09 21.8 (warm up)
1 1.044e-01 2.570e+09 43.7 (warm up)	1 1.051e-01 2.554e+09 43.4 (warm up)	1 1.195e-01 2.245e+09 38.2 (warm up)
2 1.040e-01 2.580e+09 43.9 (warm up)	2 1.052e-01 2.552e+09 43.4 (warm up)	2 8.158e-02 3.291e+09 55.9 (warm up)
3 1.183e-01 2.269e+09 38.6	3 1.189e-01 2.257e+09 38.4	3 8.081e-02 3.322e+09 56.5
4 1.020e-01 2.632e+09 44.7	4 1.026e-01 2.616e+09 44.5	4 8.079e-02 3.323e+09 56.5
5 1.017e-01 2.639e+09 44.9	5 1.027e-01 2.615e+09 44.5	5 8.052e-02 3.334e+09 56.7
6 9.515e-02 2.821e+09 48.0	6 1.007e-01 2.665e+09 45.3	6 8.076e-02 3.324e+09 56.5
7 8.762e-02 3.064e+09 52.1	7 9.392e-02 2.858e+09 48.6	7 7.810e-02 3.437e+09 58.4
8 8.749e-02 3.068e+09 52.2	8 8.904e-02 3.015e+09 51.3	8 7.789e-02 3.446e+09 58.6
9 8.831e-02 3.040e+09 51.7	9 8.847e-02 3.034e+09 51.6	9 7.788e-02 3.447e+09 58.6
10 8.765e-02 3.063e+09 52.1	10 8.842e-02 3.036e+09 51.6	10 7.013e-02 3.828e+09 65.1
11 8.746e-02 3.069e+09 52.2	11 8.828e-02 3.041e+09 51.7	11 6.945e-02 3.865e+09 65.7
12 8.763e-02 3.063e+09 52.1	12 8.843e-02 3.035e+09 51.6	12 6.956e-02 3.859e+09 65.6
#[1mAverage performance: #[42m 48.8 +- 4.5 GFLOP/s#[0m	#[1mAverage performance: #[42m 47.9 +- 4.3 GFLOP/s#[0m	#[1mAverage performance: #[42m 59.8 +- 3.8 GFLOP/s#[0m

Conclusion : la version 3 du code a introduit une optimisation significative avec l'utilisation de la méthode d'inversion de racine carrée rapide pour calculer la distance entre les particules. Les résultats suggèrent une amélioration des performances, notamment avec des ensembles de données plus importants. Le temps d'exécution a diminué, le taux d'interactions par seconde a augmenté, et le taux de GFLOP/s a également montré une amélioration. Ces résultats indiquent que l'optimisation de la fonction de distance a un impact positif sur les performances globales du code de simulation des interactions entre particules.

6.Comparaison

Dans cette section pour évaluer l'impact des différentes versions du code, nous avons compilé et exécuté chaque version avec différents flags d’optimisation et de compilation ainsi que différents compilateurs. Les résultats des performances pour chaque version sont présent dans les tableaux suivants

Versions	O2_gcc	O3_gcc	Ofast_gcc	O2_clang	O3_clang	Ofast_clang
Base	0.6 +- 0.0 GFLOP/s	0.6 +- 0.0 GFLOP/s	2.4+- 0.0 GFLOP/s	0.5 +- 0.0 GFLOP/s	0.5 +- 0.0 GFLOP/s	5.6 +- 0.1 GFLOP/s
V1	0.6 +- 0.0 GFLOP/s	0.6 +- 0.0 GFLOP/s	7.2 +- 0.2 GFLOP/s	0.5 +- 0.0 GFLOP/s	0.5 +- 0.0 GFLOP/s	7.4 +- 0.0 GFLOP/s
V2	4.7 +- 0.1 GFLOP/s	4.8 +- 0.0 GFLOP/s	17.9+- 0.0 GFLOP/s	2.6 +- 0.0 GFLOP/s	2.6 +- 0.0 GFLOP/s	26.1 +- 0.5 GFLOP/s
V3	86.4 +- 18.9 GFLOP/s	82.7 +- 11.7 GFLOP/s	191.7 +- 18.6 GFLOP/s	48.8 +- 4.5 GFLOP/	47.9 +- 4.3 GFLOP/s	59.8 +- 3.8 GFLOP/s

Pour une visualisation plus claire des performances, les graphiques suivants présentent la comparaison entre les différentes versions du code des flags de compilation et des compilateurs utilisés.

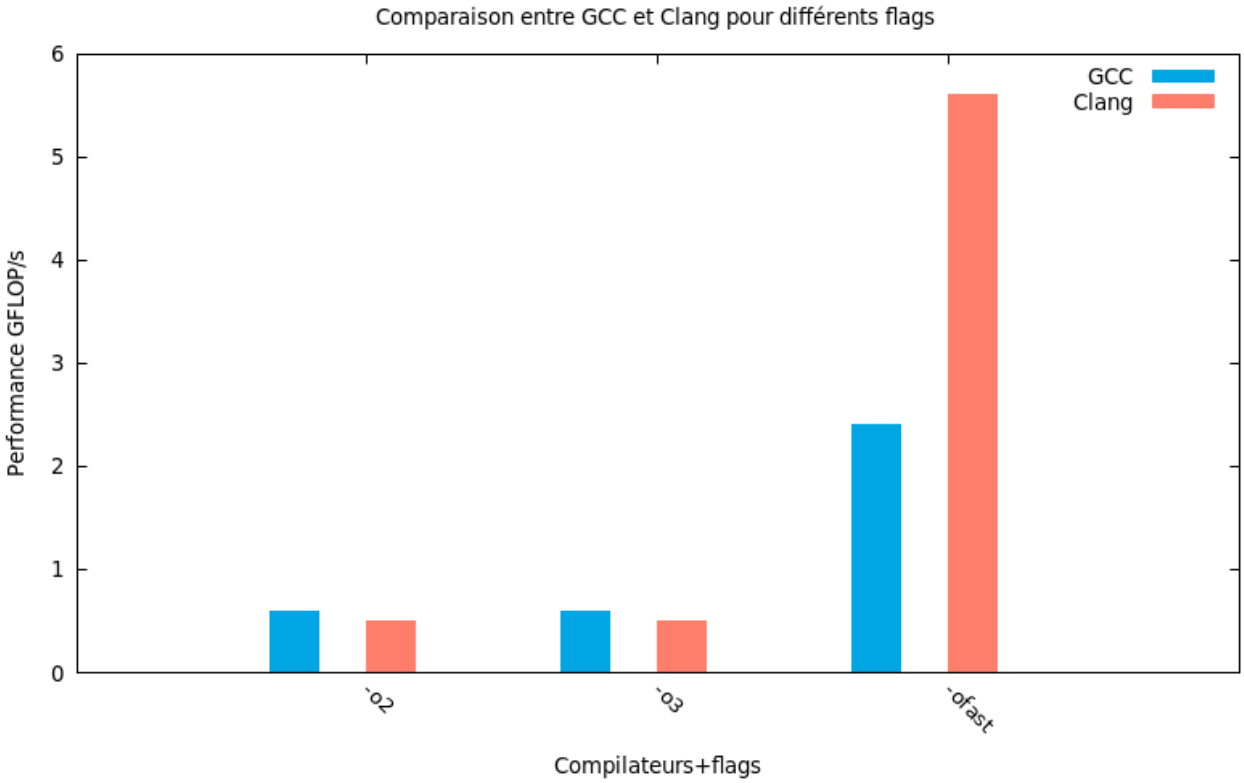

Figure 1: Histogramme Base

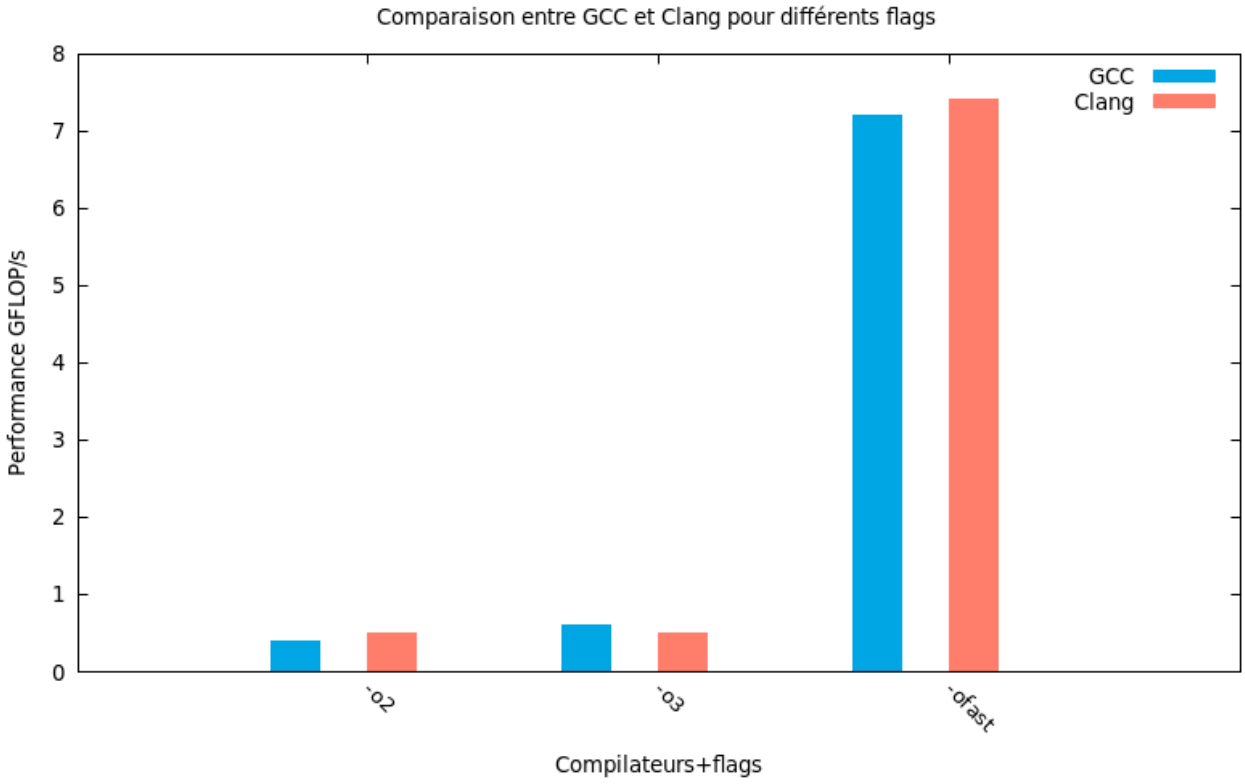

Figure 2: Histogramme Version1

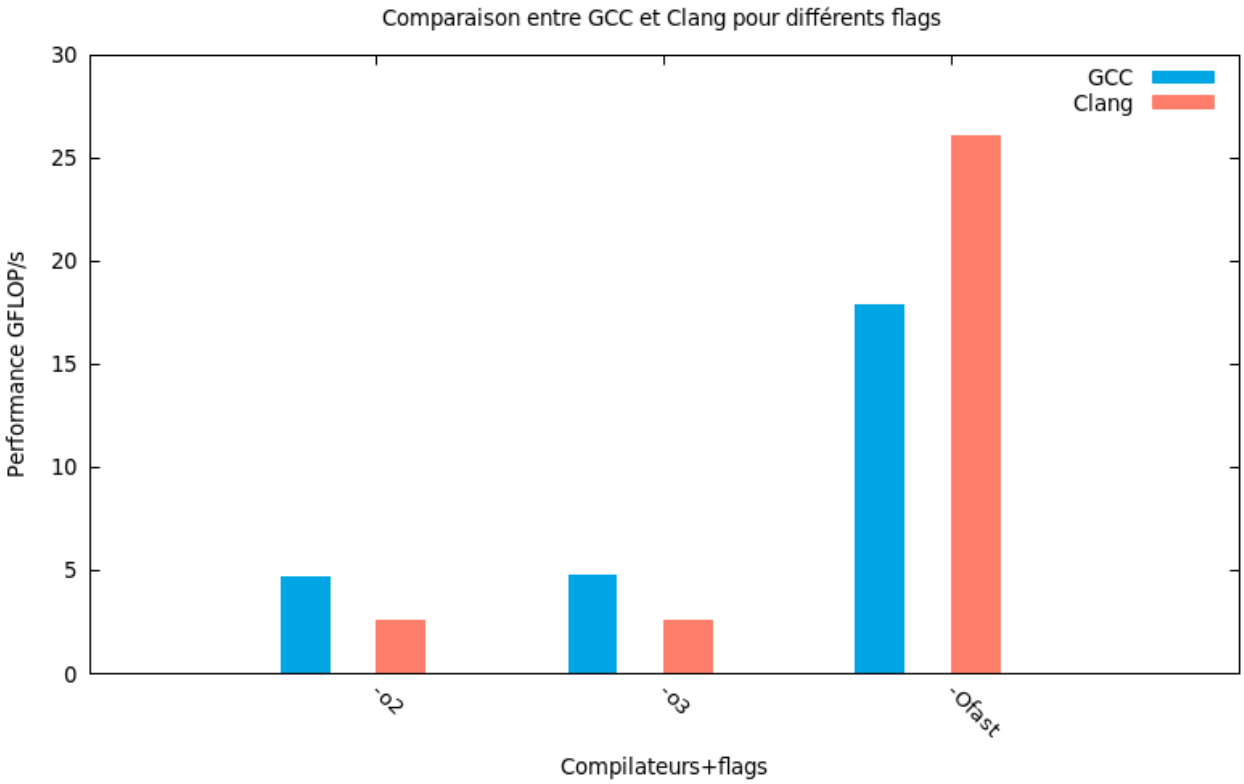

Figure 3: histogramme version2

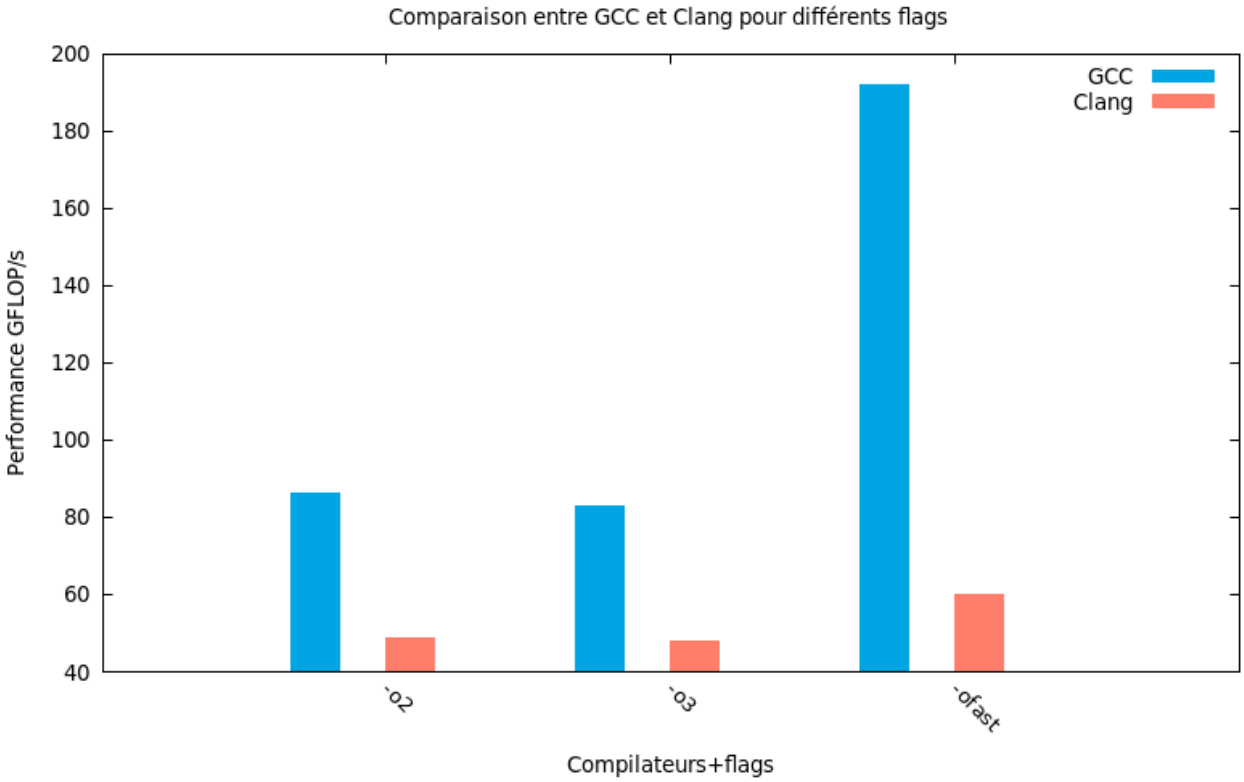

Figure 4: histogramme version3

Conclusion de la Comparaison:

La comparaison des performances entre les différentes versions du code met en évidence les avantages et les améliorations apportées par les optimisations successives. Les résultats des tests avec diverses configurations indiquent des tendances spécifiques et des différences significatives dans les performances. Ces observations sont essentielles pour choisir la version optimale du code en fonction des exigences spécifiques de l'application, de la plate-forme matérielle et des contraintes de compilation. La sélection appropriée des versions, des flags de compilation et des compilateurs peut avoir un impact considérable sur les performances du code de simulation des interactions entre particules.

7. Conclusion générale

Dans le cadre de cette expérience, notre objectif était d'évaluer et d'optimiser le code de simulation des interactions entre particules. Plusieurs itérations ont été effectuées, introduisant des modifications progressives pour améliorer les performances du code. Voici un récapitulatif des principales évolutions apportées:

VERSION1 :

Introduction d'une structure de tableau (particle_t) pour une gestion de la mémoire plus efficace.

VERSION2 :

le calcul de la distance entre les particules en utilisant une séquence de multiplication pour obtenir la racine cubique, évitant ainsi l'utilisation de la fonction `pow` .

VERSION3 :

le remplacement de la fonction `pow` par une méthode d'inversion de racine carrée rapide (`inverse_square_root`).

- Les optimisations apportées dans la Version de base ont considérablement amélioré les performances, avec une gestion de la mémoire plus efficace et une parallélisation accrue. Cependant , la version3 avec ses optimisations structurelles et algorithmiques, est considérée comme la plus optimale en termes de performances globales.

Bien que les versions actuelles aient considérablement amélioré les performances du code de simulation des interactions entre particules, il existe toujours des opportunités d'amélioration et d'optimisation supplémentaires.

Des améliorations peuvent être apportées par exemple l’Utilisation de Directives SIMD **:**

- Explorer l'utilisation de directives SIMD (Single Instruction, Multiple Data) pour une vectorisation plus efficace des boucles