Calcul de densité d'état de particule pour la modélisation de nanoparticule à transition de spin (SCO) selon un modèle d'Ising.

Moursal M.A , Ange .B.D, I. Ndiaye, Glen S.A

UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES
UFR DES SCIENCES
université PARIS-SACLAY
CALCUL HAUTE PERFORMANCE SIMULATION

Master 1 Calcul Haute Performance et Simulation

07 Mai 2019

Table de matière

Table de Matière

Introduction

- Matériaux à transition de spin (SCO).
- Densité d'état des particules

Table de Matière

Modèle Ising

Définition du modèle

Le modèle d'Ising est un modèle fréquemment utilisé pour tester de nouvelles idées et méthodes en physique statistique. Afin de représenter les atomes qui composent un matériau magnétique, on fixe deux hypothèses :

- chaque atome a un moment magnétique appelé spin. Pour simplifier, les spins n'auront que deux orientations possibles : haut (on dira que le spin est $+1$) ou bas (-1)
- initialement, ils sont orientés aléatoirement.
  Le but est de montrer l'existence d'une transition de phase dans ce modèle.

Illustration du modèle

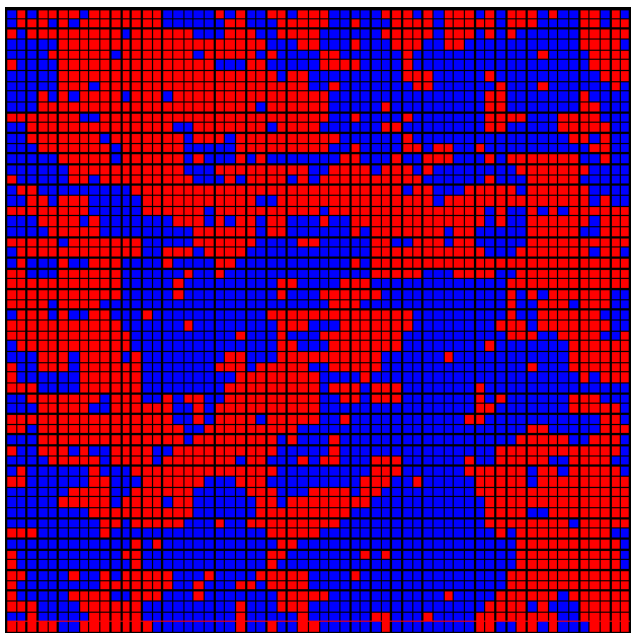


Figure – Configuration 2D

- $\uparrow$ $+1$ spin vers le haut en bleue
- $\downarrow$ -1 spin vers le bas en rouge

Configuration 2D Selon trois variables

σ represente un spin.

- $m = \sum_{i=1}^{n} \sigma_i$
- $s = \sum_{<i,j>} \sigma_i \sigma_j$
- $c = \sum_{k=1}^{M} \sigma_k^{'}$

Densité

La densité d'état d(m,s,c) est calculée, selon ces trois variables.

Monté Carlo

Définition

- Monté Carlo
- Monté Carlo Metropolis (Température, Constant de Bolztmann, l'Energie)

Table de Matière

Algorithme et implémentation

Implémentation

- Initialiser une matrice de taille N alétoirement avec une graine $rand()\%2$.
- Tirage aléatoire d'une case[i][j] de la matrice selon une probabilité de $\frac{1}{N}$ ($rand()\%N$) avec N le nombre de spin sur le réseau.
- L'énergie (E) est calculé, selon les voisins plus proches de la case tirer avec la fonction ***ith_energy***
- Multiplier la case(i,j) choisis par -1 pour faire subir un retournement et on recalcule la nouvelle énergie (E').

Algorithme et implémentation

Implémentation

- $\Delta E = (E') - (E)$, si $\Delta E \leq 0$ alors la configuration est stockée (accepter) dans un tableau (*), sinon on prend alétoire un nombre réel x avec la fonction *drand48()*, si $x \leq exp(-\Delta E * \beta)$
- $\Delta E = (E') - (E)$, si $\Delta E \leq 0$ alors la configuration est stockée (accepter) dans un tableau, sinon on prend alétoire un nombre réel x avec la fonction *drand48()*, si x $\leq exp(-\Delta E * \beta$) avec $\beta = \frac{1}{K_B * T}$: T étant la température et K_B la constante de Boltzmann $K_B = 1,3806485210^{23} JK^{-1}$, alors on fait la même que $(*)$. Sinon la configuration est rejeter.
- Ces différentes étapes sont faites à plusieurs tours de boucle.

Algorithme et implémentation

Stockage de configuration

L'ajout d'une configuration avec la fonction ***add_config***

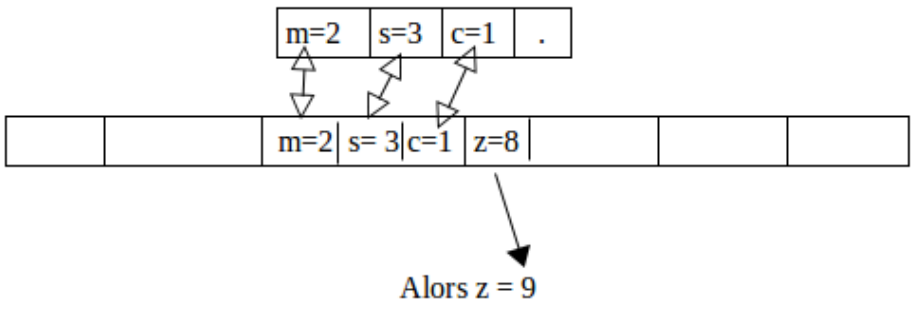


Table de Matière

Etude et Résultat

Etude

- Exemple d'une Matrice 5 par 5
- Le Cas idéal du critère d'arrêt de l'algorithme est si la densité vaut 2^5 (ou 2^N pour une taille de N*N)
- Entre 275 et 290K la densité diminue et de 290 à 320K elle varie.
- Complexité en temps dans le meilleur de cas est linéaire dans cette exemple.
- Complexité en mémoire dans le pire de cas est 2^N configuration à stocker.

Etude et Résultat

Résultat

- Exemple d'une Matrice 5 par 5
- Nombre d'itération $4 * 10e7$
- Temps d'exécution 9.5196 minutes

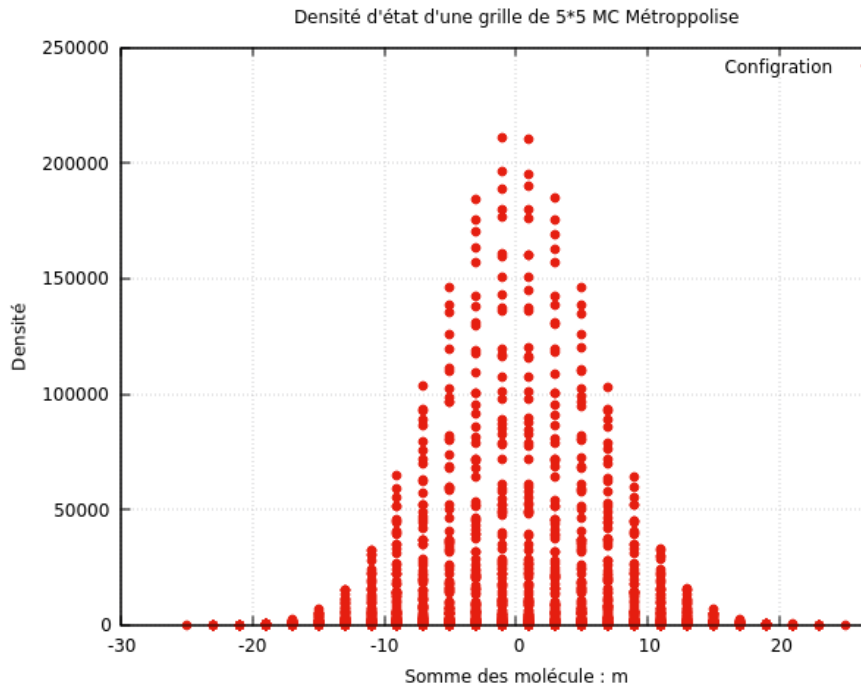
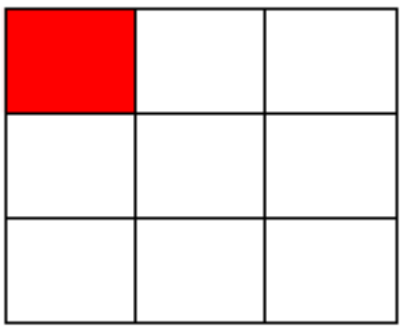


Figure – Densité Approché

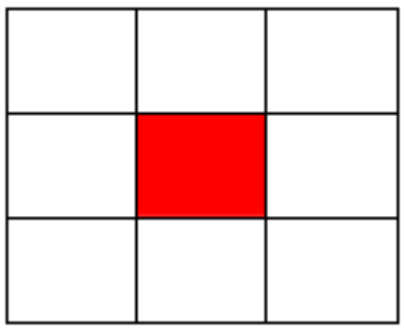
Table de Matière

Version MPI

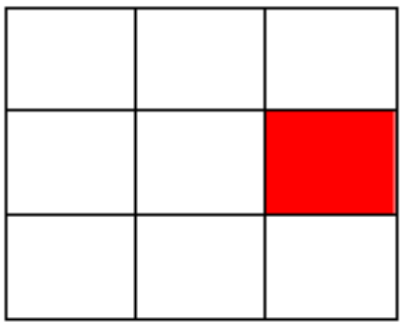
- Idée



P1



P2



P3

Version MPI

Explication de l'implémentation

- **srand(rang*t)**
- **Send(buffsnd_rang) - Recv(Buffrcv_rang)**
- **fusion_array(tab,Buff)**

Version MPI

- Influence du nombre de processus sur la densité
- Exemple d'une matrice 4*4 et 8 itération.

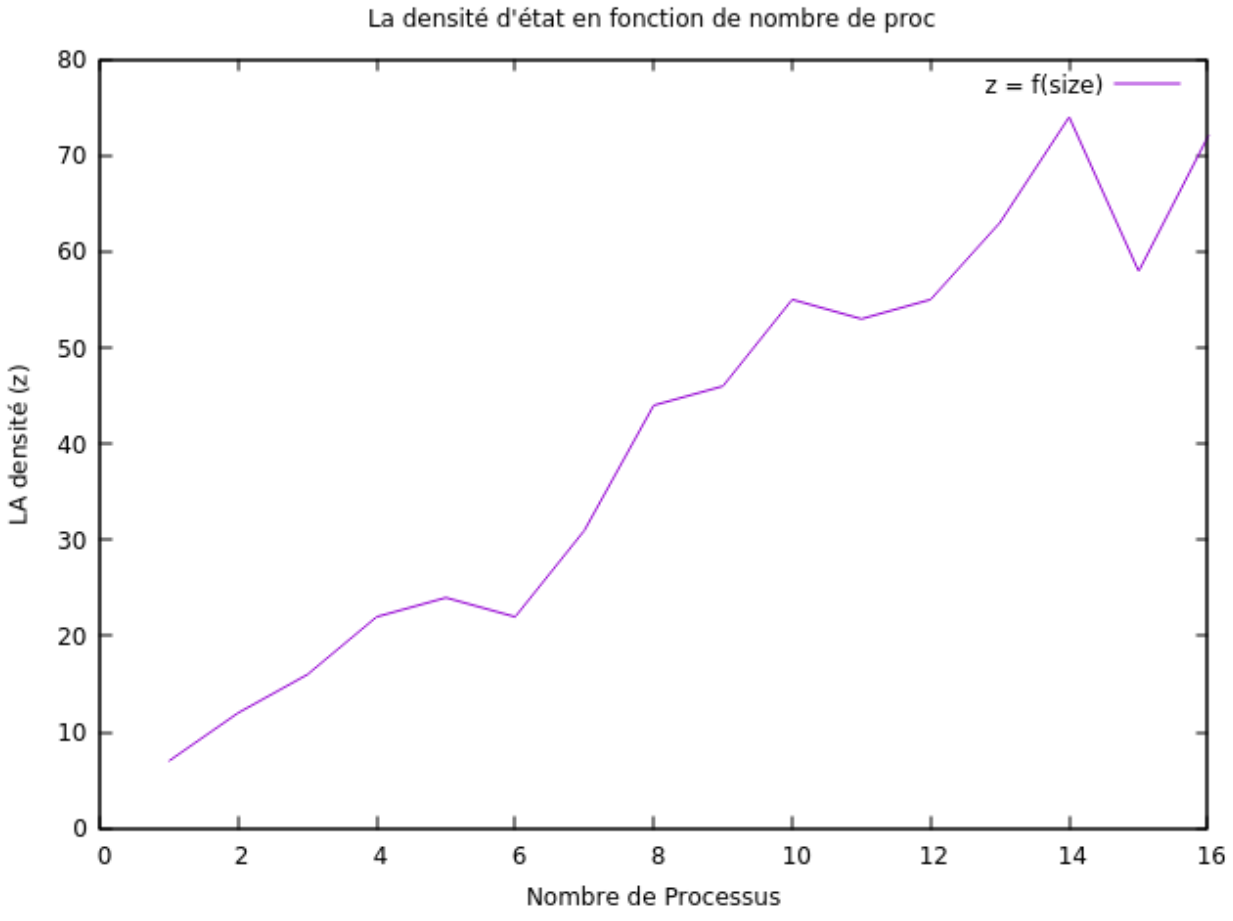
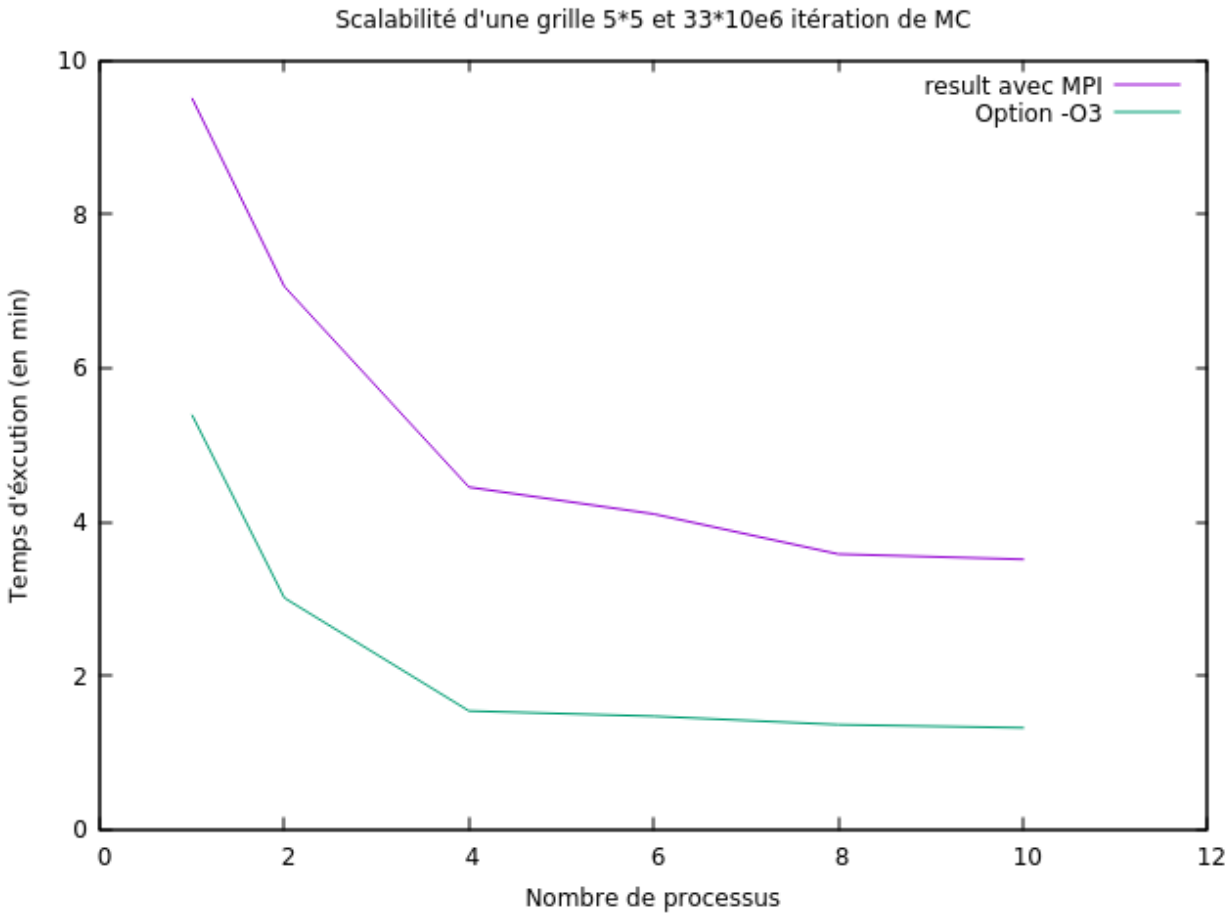


Tableau recapitulative

Nombre de processus	1	2	4	6	8	10
Durée globale (en min)	9m51	7m06	4m45	4m10	3m58	3m51
Speed UP (option -O3)	5m39	3m01	1m54	1m47	1m36	1m32
Temps comm (Send – Recv)	-	0.000185	2m30	0.283s	1m48	1m97
(Zmax/Nombre de processus)	33554432	16777216	8388608	5592405	4194304	3355443

Version MPI

- Etude de performance



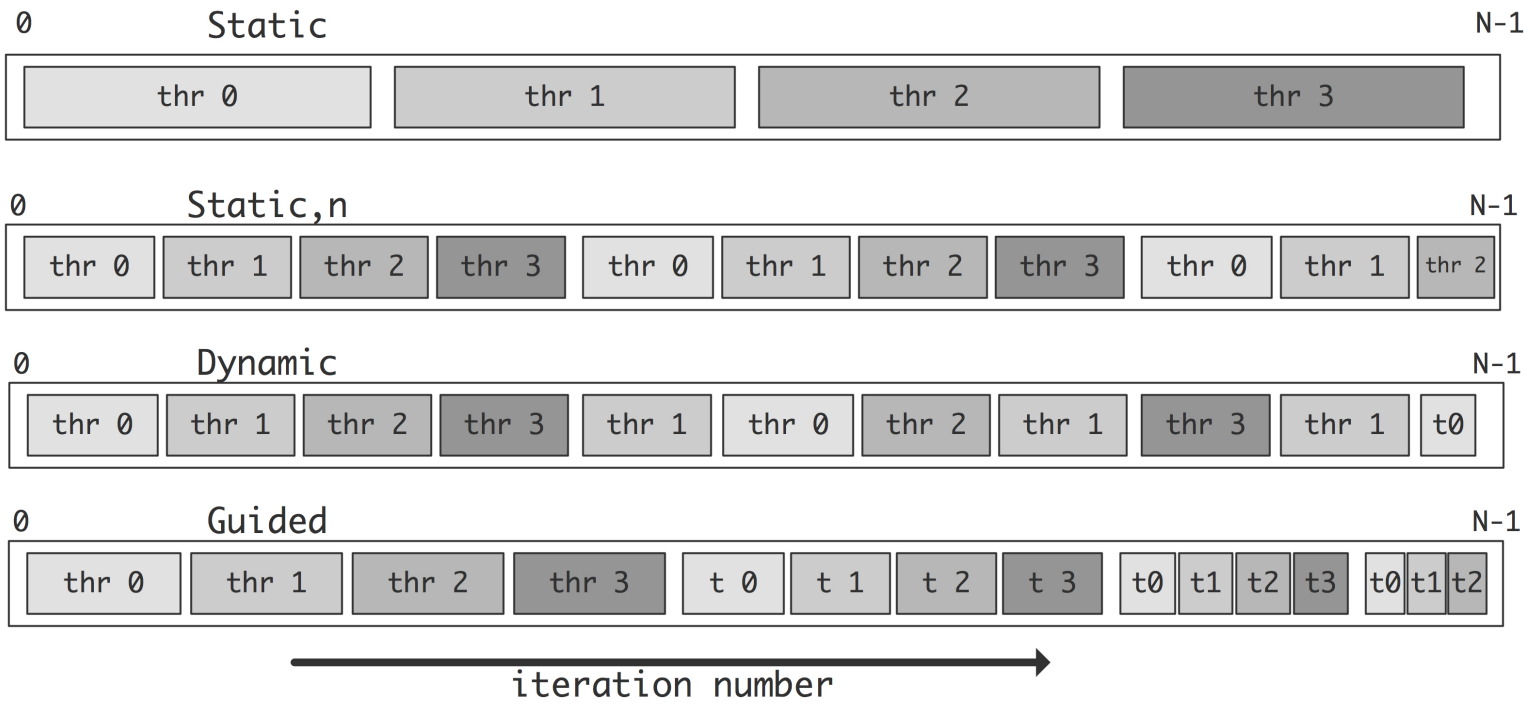
Version OpenMP

Hotspot

- **hotspot** de l'algorithme : la boucle de MC
- la répartition de tâche de celle-ci peut se faire avec différente clause : dynamic, static, guided, etc.
- Exemple : #*pragma* omp for schedule(static , 100) dispatching de la boucle en 100 itérations d'une manière static.

Version OpenMP

- Illustration



Version OpenMP

Démarche de l'implémentation

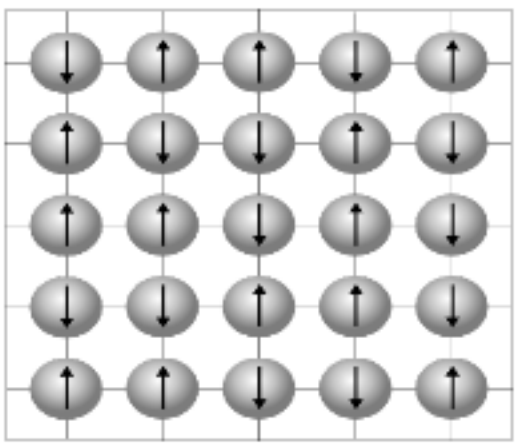
- Dispatcher équitablement et dynamiquement la boucle d'itération de MC en fonction de nombre de threads : *#pragma* **omp for schedule(dynamic,1000)**

- Choisir une graine différente pour chaque thread **srand(omp_get_thread_num())**.

Version OpenMP

Plusieurs threads

- Avoir plusieurs threads qui tirent en parallèle sur la même grille



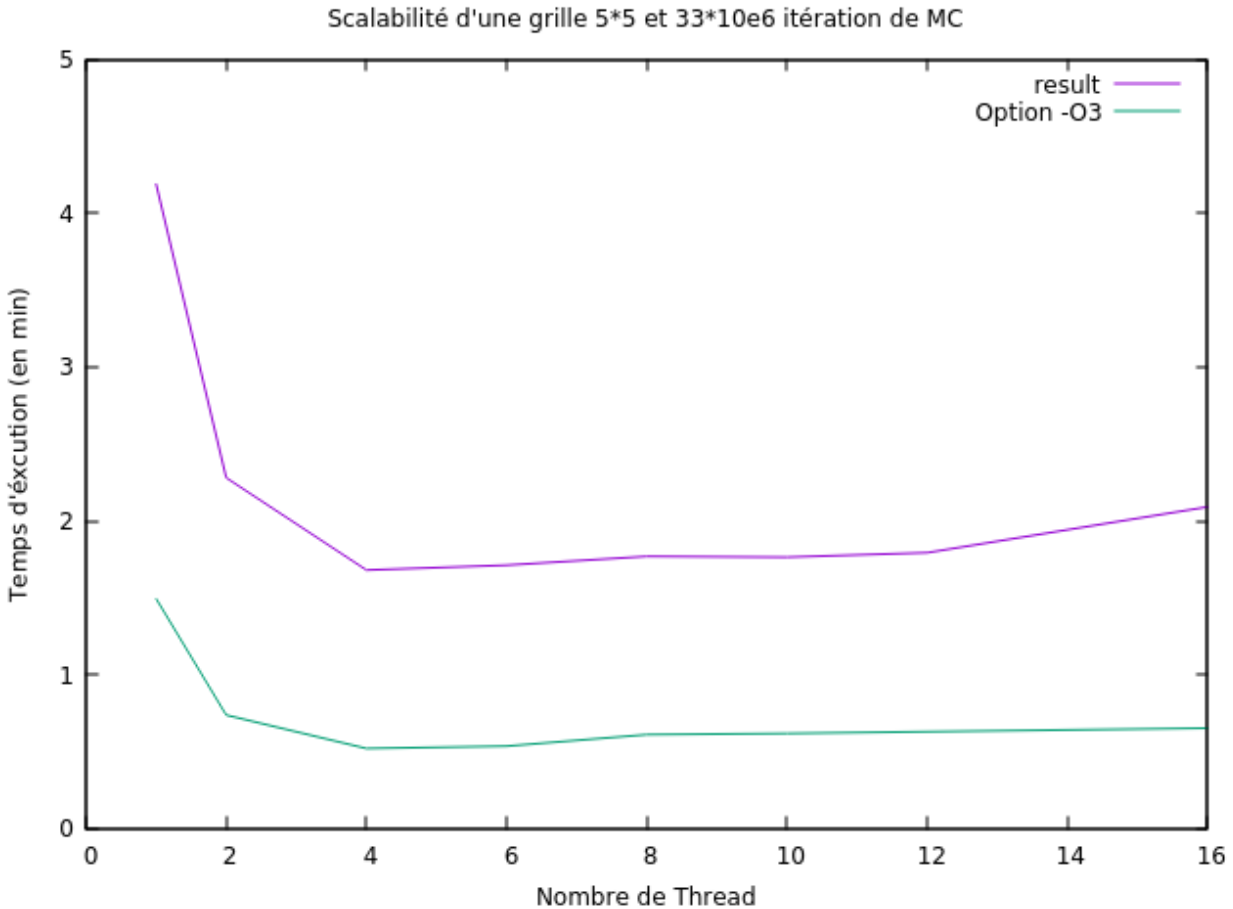
Version OpenMP

Résultats et Scalabilité

- nombre d'itération optimum 33 10e6 température 290 K.
- D'après nos résultats avec OpenMP on a constaté de 1 thread à 2 thread on a un gain de performance de environ 45%, de 2 thread à 4 thread environ 10% et au-déla de 4 thread on a une scalabilité faible.

Version OpenMp

- Etude de Performance



Version OpenMp

- Comparaison de la densité

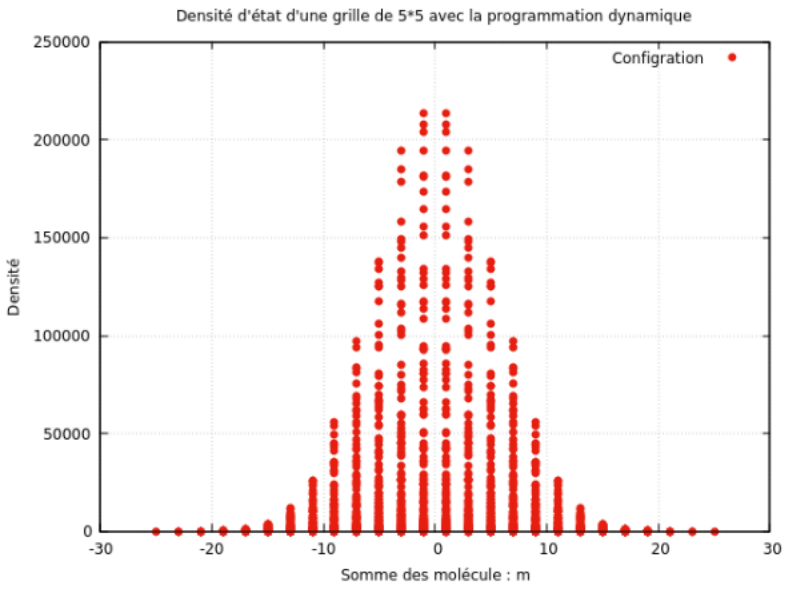


FIGURE 5: Densité Exacte

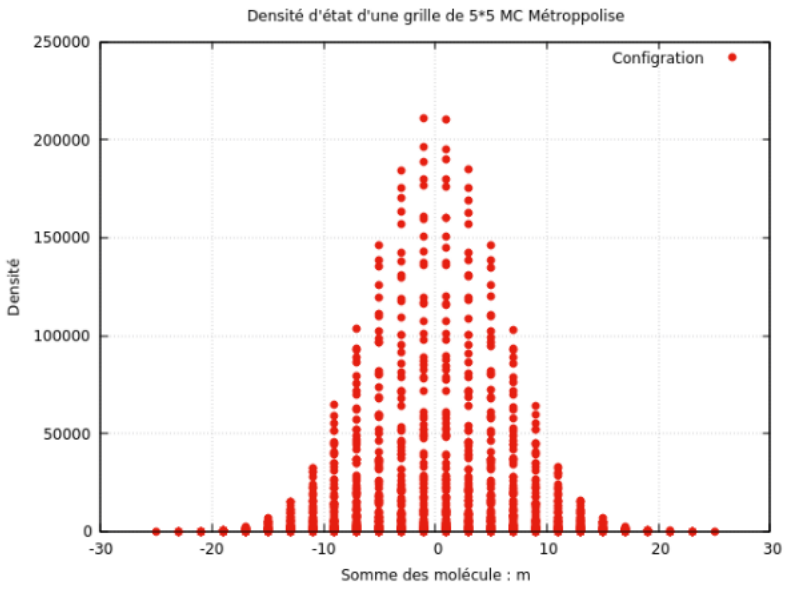


FIGURE 6: Densité Approché avec OpenMP

Table de Matière

Conclusion

- Séquentiel
- Parallèle

La version OpenMP étant plus rapide sur les premiers cores, en augmentant le nombre cores la version MPI donne des bons résultats bien que gourmande en mémoire. Une autre implémentaion MPI était envisageable avec "MPI Cart" .