UNIVERSITE D'ORLEANS
Faculté des Sciences
LIFO

Laboratoire d'Informatique Fondamentale d'Orléans
4, rue Léonard de Vinci, BP 6759
F-45067 Orléans Cedex 2
FRANCE

Rapport de Recherche

www : http://www.univ-orleans.fr/SCIENCES/LIFO/

Une version parallèle extensible de l'algorithme Particle Mesh Ewald

Saïd Drouri et Gaétan Hains
Université d'Orléans, LIFO

Rapport N^o **2000-15**

1 Introduction

Ce rapport décrit les résultats d'une mini-action de validation applicative dans le cadre du programme iHPerf'98. Ce projet [1] a été mené en collaboration avec le CBM (CNRS-Orléans) et financé par le GDR ARP du CNRS. Il traite de l'analyse d'extensibilité et de l'implantation parallèle d'un algorithme fondamental en simulation moléculaire: l'algorithme Particle Mesh Ewald pour le calcul des interactions électrostatiques.

1.1 Le Centre de Biophysique Moléculaire d'Orléans

Le CBM d'Orléans est une unité propre de recherche du CNRS conventionnée avec l'université d'Orléans. Aux interfaces de la physique et de la chimie avec la biologie, sa mission porte sur la compréhension de processus biologiques fondamentaux en termes de structure, de dynamique et d'interactions de macromolécules naturelles ou modifiées. Un des axes de recherche du CBM est : **Systèmes biomoléculaires organisés.** Cet axe concerne l'étude de la structure, de la dynamique et des interactions de macromolécules biologiques. Huit équipes de recherche travaillent sur cet axe. Notre projet a été réalisé en collaboration avec l'équipe *Simulation moléculaire, méthodologie et calcul intensif* dont le responsable est le professeur Gerald KNELLER.

1.2 Particle Mesh Ewald

Particle Mesh Ewald est l'un des algorithmes les plus utilisés par les chercheurs du CBM pour la simulation des systèmes moléculaires. Le programme séquentiel implantant cet algorithme est très lent surtout pour des systèmes moléculaires de grande taille, d'où notre but de le paralléliser. La parallélisation de cet algorithme doit traiter deux problèmes : le placement des données et l'équilibrage des charges. Dans ce contexte une solution a été proposée dans [2], on y a aussi montré que dans les systèmes réels, les particules ne sont pas réparties de façon homogène et de ce fait, le choix de stratégie de placement des tâches de calcul sur les processeurs est important.

A partir d'un découpage géométrique de l'espace de simulation en des régions qui sont distribuées aux processeurs [1], nous avons estimé le coût de calcul, de synchronisation et de communication selon le modèle BSP [11]. Cette analyse permet d'optimiser l'équilibre entre communications, synchronisations et calculs dans le but de maximiser l'extensibilité de l'algorithme. Nous utiliserons le modèle BSP pour le calcul du coût de notre algorithme parallèle, nous l'utiliserons aussi comme modèle d'exécution avec la bibliothèque de communication MPI.

[1] `http://www.univ-orleans.fr/SCIENCES/LIFO/Members/ghains/ihperf98/ihperf.htm`

2 Description du projet

2.1 La somme d'Ewald

La somme d'Ewald [4] a été introduite en 1921 comme une technique pour calculer efficacement la somme des interactions entre les particules d'un système moléculaire. Nous allons définir la somme d'Ewald dans le cas du calcul de l'énergie de Coulomb dans les conditions périodique aux bornes. Dans des conditions périodiques aux bornes, on imagine que les particules sont situées dans une boîte et que cette boîte est dupliquée indéfiniment par des translations dans les trois diréctions de l'espace. Dans ce contexte, si une particule i se trouve à une position r_i dans la boite, on considère que cette particule représente un ensemble infini de particules, qu'on appelle *images périodiques* de la particule i, situées à des positions $r_i + (n_1 a, n_2 b, n_3 c)$ où n_1, n_2, n_3 sont des entiers et a, b, c sont les longueurs des arêtes de la boîte. Supposons maintenant que l'on ait N charges ponctuelles $q_1, q_2, \cdots, q_N$ situées dans un cube unitaire U d'arête L et qui satisfont la condition suivante $q_1 + q_2 + \cdots + q_N = 0$. Les charges ont des interactions entre elles suivant la loi de Coulomb. Une charge q_i située à la position i a des interactions avec toutes les charges q_j telles que $i \neq j$ qui se trouvent dans les positions r_j ainsi qu'avec toutes leurs images périodiques qui se trouvent dans les positions $r_j + (n_1 L, n_2 L, n_3 L)$ pour toutes les valeurs entières n_1, n_2, n_3. La charge i a des interactions aussi avec ses propres images périodiques qui se trouvent dans les positions $r_i + (n_1 L, n_2 L, n_3 L)$ pour tous les entiers n_1, n_2, n_3 tels que n_1, n_2 et n_3 ne sont pas tous égaux à 0. L'énergie électrostatique du cube unitaire U est donnée par l'équation suivante

$$E(r_1, \cdots, r_N) = \frac{1}{2} \sum_{\mathbf{n}}{}' \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{q_i q_j}{\mid r_i - r_j + \mathbf{n} \mid} \tag{1}$$

Dans la première somme le symbole $'$ indique que les termes correspondant à $i = j$ sont omis quand $\mathbf{n} = 0$. Le vecteur $\mathbf{n}$ est défini par $\mathbf{n} = (n_1 L, n_2 L, n_3 L)$.
Ewald a transformé le calcul de l'énergie décrit dans l'équation (1) en calcul de la somme de trois termes de la façon suivante $E(r_1, \cdots, r_N) = E_{dir} + E_{rec} + E_{self}$. Les termes E_{dir}, E_{rec} et E_{self} sont décrits par les équations (2), (3) et (4).

$$E_{dir} = \frac{1}{2} \sum_{\mathbf{n}}{}' \sum_{i,j=1}^{N} \frac{q_i q_j \mathrm{erfc}(\alpha \mid r_j - r_i + \mathbf{n} \mid)}{\mid r_j - r_i + \mathbf{n} \mid} \tag{2}$$

$$E_{rec} = \frac{1}{2\pi L^3} \sum_{\mathbf{m} \neq 0} \frac{exp\left(\frac{-\pi^2 \mathbf{m}^2}{\alpha^2}\right)}{\mathbf{m}^2} \mid S(\mathbf{m}) \mid^2 \tag{3}$$

$$E_{self} = -\frac{\alpha}{\sqrt{\pi}} \sum_{i=1}^{N} q_i^2 \tag{4}$$

Où $\mathbf{n} = (n_1L, n_2L, n_3L)$, $\mathbf{m} = \left(\frac{m_1}{L}, \frac{m_2}{L}, \frac{m_3}{L}\right)$ et le facteur $S(\mathbf{m})$ est défini par l'équation (5).

$$S(\mathbf{m}) = \sum_{j=1}^{N} q_j \mathrm{exp}\left(\frac{2\pi i(m_1r_{j1} + m_2r_{j2} + m_3r_{j3})}{L}\right) \tag{5}$$

erfc (*complementary error function*) est la fonction définie par l'équation (6).

$$erfc(x) = 1 - \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \tag{6}$$

α est un paramètre qui dépend du système étudié, de la rapidité et de la précision voulues pour le calcul de la somme d'Ewald.

2.2 l'algorithme Particle Mesh Ewald (PME)

2.2.1 Les approximations par coupures

Dans le contexte de la somme d'Ewald, des coupures de l'espace réel sont utilisées pour réduire le coût de la somme sur l'espace réel. Deux types de coupures sont utilisées: *l'image minimale* (minimum image) et *les coupures sphériques* (spherical cutoffs) (Fig 1 page 6). Dans le cas de la coupure sphérique la particule 5 qui se trouve dans la cellule E n'aura des interactions qu'avec les particules $F1, F4$ et $H2$. Dans le cas de la stratégie d'image minimale les particules qui auront des interactions avec la particule $E5$ sont $E3, F1, F4$ et $H2$.

2.2.2 Application des approximations à l'algorithme PME

L'algorithme PME calcule la somme de l'équation (2) en utilisant la stratégie de coupure sphérique. La somme de l'équation (3) est calculée avec une approximation basée sur des transformations rapides de Fourier (Fast Fourier Transforms) qu'on note FFT avec des convolutions sur une grille où les charges sont interpolées aux différents centres des cellules qui composent la grille. On appelle ces centres de cellules *les points de la grille*. Ceci permet de remplacer les termes complexes $exp\left(\frac{2\pi i m_\gamma r_{j\gamma}}{L}\right)$, $\gamma = 1, 2, 3$ de l'équation (3) par des combinaisons linéaires de leurs valeurs dans les points de la grille. Les forces sont calculées analytiquement par différentiation des énergies.
La figure (Fig 2 page 7) illustre un schéma de calcul de l'algorithme PME dans le cas de deux dimensions.

2.3 Parallélisation de l'algorithme PME

Les principales parallélisations de l'algorithme PME sont de deux types:

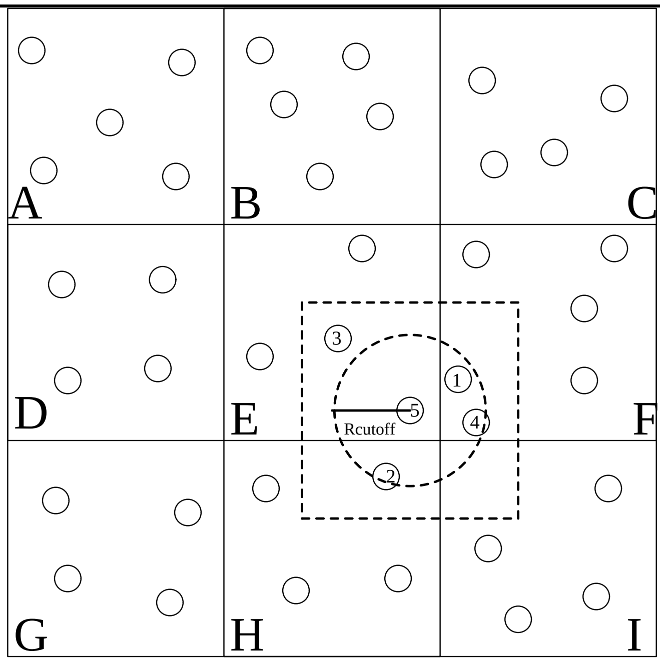

Figure 1: Exemple des deux stratégies: *image minimale* et *coupure sphérique*

1. La première parallélisation est celle présentée dans [9]. Cette parallélisation est basée sur une méthode Maître/Esclave, le maître calcule E_{rec} pendant que les esclaves calculent en parallèle E_{dir} et retournent le résultat au maître. Cette méthode donne de bons résultats pour des systèmes moléculaires pour lesquels les particules sont distribuées uniformément dans la grille. Dans le cas d'une distribution non uniforme des particules dans la grille, on rencontre un problème d'équilibrage des charges sur les esclaves : soit les parties vides de la grille correspondent à des périodes d'inactivités, soit elles engendrent de nombreuses requêtes des esclaves.

2. La deuxième parallélisation est celle présentée dans [5]. Elle est basée sur le calcul des préfixes et propose une solution au problème du déséquilibre. Dans cette méthode la distribution des données sur les processeurs est à la charge du compilateur, ses performances dépendent donc entièrement de l'environnement logiciel.

Notre travail dans ce projet consiste à trouver une nouvelle parallélisation plus efficace que les deux précédentes et dont les performances sont garanties extensibles.

Nous présentons dans la suite un type de problèmes auxquels nous serons confronté dans ce projet. Par exemple, pour calculer la somme de l'équation (2) en utilisant une

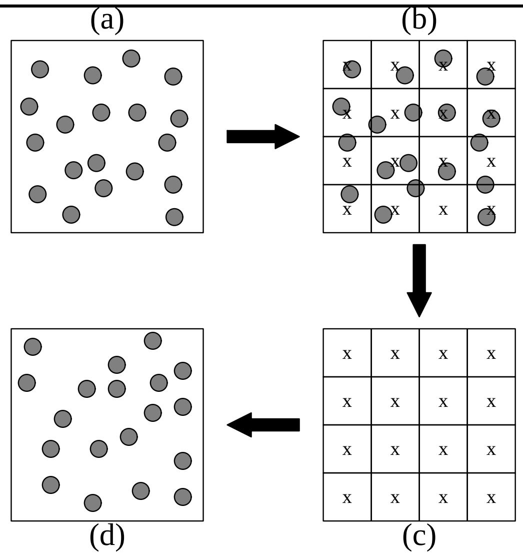

Figure 2: *(a)* Un système de particules chargées *(b)* Les charges sont interpolées sur une grille à deux dimensions *(c)* En utilisant des FFT, le potentiel et les forces sont calculés aux points de la grille *(d)* Interpolation des forces aux particules et mise à jour des coordonnées.

stratégie de coupure sphérique, nous devons effectuer les calculs suivants:

1. Localisation des particules dans la grille de simulation. A la fin de cette partie nous devons savoir pour chaque particule, la cellule de la grille dans laquelle elle se trouve.

2. Trouver pour chaque particule l'ensemble des autres particules avec lesquelles elle aura des interactions (i.e. celles qui se trouvent à l'intérieur de la sphère qui a pour centre la particule et pour rayon R_{cutoff}),

3. Calculer les éléments constituant la somme de l'équation (2),

4. Calculer la somme de ces éléments.

La troisième partie est la plus coûteuse en temps de calcul et celui-ci dépend de la stratégie de distribution des données. Elle nécessite aussi un équilibrage des charges. Supposons, par exemple, qu'on ait distribué les tâches de la façon suivante: On distribue les particules par blocs de tailles égales aux différents processeurs, ensuite chaque processeur calcule les interactions des particules qu'il contient dans sa mémoire. Nous remarquons que si une particule appartenant à un processeur x a plus de particules dans sa sphère d'interactions qu'une autre particule appartenant à un processeur y alors le processeur x aura plus de termes à calculer que le processeur y.

Pour résoudre ce problème, nous planifions d'utiliser un découpage géométrique initial du système à simuler, puis distribuer équitablement ce découpage de l'espace sur les processeurs. Les articles [8] et [1] ont montré l'efficacité de cet algorithme pour un ensemble de particules réparties de façon homogène. Pour des systèmes dans lesquels les particules ne sont pas réparties de façon homogène, on doit ajouter une phase d'équilibrage des charges sur les processeurs avec une bonne stratégie de placement des données et des calculs.

Une parallélisation de l'algorithme Particle Mesh Ewald

Définissons quelques notations que nous allons utiliser pour le reste de ce papier.

N : le nombre d'atomes du systèmes moléculaire

n : l'ordre d'interpolation

p : le nombre de processeurs de la machine parallèle

L_1, L_2, L_3 : les longueurs des arêtes de la grille de simulation

K_1, K_2, K_3 : les nombres de points dans les trois axes de la grille d'interpolation

3 Calcul de la partie directe de l'energie électrostatique

$$E_{dir} = \frac{1}{2} \sum_{\mathbf{n}}' \sum_{i,j=1}^{N} \frac{q_i q_j \operatorname{erfc}(\alpha \mid r_j - r_i + \mathbf{n} \mid)}{\mid r_j - r_i + \mathbf{n} \mid} \tag{7}$$

On divise la grille de simulation en $p = c_1 \cdot c_2 \cdot c_3$ régions, chaque processeur sera résponsable d'une région. On identifiera les différentes régions en les numérotant de la façon suivante

$$R_{i,j,k} \text{ avec } \begin{cases} 0 \leq i \leq c_1 - 1 \\ 0 \leq j \leq c_2 - 1 \\ 0 \leq k \leq c_3 - 1 \end{cases}$$

Les 26 régions voisines d'une région ($R_{i,j,k}$) dans les conditions périodiques sont

$$\{R_{\alpha,\beta,\gamma} \mid (\alpha,\beta,\gamma) \in \{i, (i-1) \bmod c_1, (i+1) \bmod c_1\} \times \{j, (j-1) \bmod c_2, (j+1) \bmod c_2\} \times \{\{k, (k+1) \bmod c_3, (k+1) \bmod c_3\}\} \setminus \{(i,j,k)\}$$

Chaque région contiendra $n_1 \cdot n_2 \cdot n_3$ boîtes. Les nombres n_i pour $i = 1, 2, 3$ sont définies de la façon suivante

$$n_i = \lfloor \frac{L_i}{R_{cutoff}} \rfloor$$

Les longueurs des arrêtes d'une boîte sont données par l'équation suivante

$$L_{b_i} = \frac{L_i}{n_i} \text{ pour } i = 1, 2, 3$$

On identifiera les différentes boîtes d'une région en leur donnant la numérotation suivante

$$b_{i,j,k} \text{ avec } \begin{cases} 0 \leq i \leq n_1 - 1 \\ 0 \leq j \leq n_2 - 1 \\ 0 \leq k \leq n_3 - 1 \end{cases}$$

exemple : les 26 régions voisines de la région $R_{0,0,0}$ dans les conditions périodiques sont présentées dans l'équation suivante

$$\begin{bmatrix} R_{c_1-1,1,0} & R_{0,1,0} & R_{1,1,0} \\ R_{c_1-1,0,0} & & R_{1,0,0} \\ R_{c_1-1,c_2-1,0} & R_{c_1-1,0,0} & R_{1,c_2-1,0} \end{bmatrix}$$ constitue la tranche de même niveau que $R_{0,0,0}$ selon l'axe z

$$\begin{bmatrix} R_{c_1-1,1,c_3-1} & R_{0,1,c_3-1} & R_{1,1,c_3-1} \\ R_{c_1-1,0,c_3-1} & R_{0,0,c_3-1} & R_{1,0,c_3-1} \\ R_{c_1-1,c_2-1,c_3-1} & R_{c_1-1,0,c_3-1} & R_{1,c_2-1,c_3-1} \end{bmatrix}$$ constitue la tranche de niveau inférieur

$$\begin{bmatrix} R_{c_1-1,1,1} & R_{0,1,1} & R_{1,1,1} \\ R_{c_1-1,0,1} & R_{0,0,1} & R_{1,0,1} \\ R_{c_1-1,c_2-1,1} & R_{c_1-1,0,1} & R_{1,c_2-1,1} \end{bmatrix}$$ constitue la tranche de niveau supérieur

Nous donnons dans la suite la liste des boîtes des régions voisines dont aurait besoin le processeur résponsable d'une région $R_{i,j,k}$ pour calculer la somme d'Ewald.

Pour les régions qui sont au même niveau selon l'axe z: Pour la région $R_{(i+1) \bmod c_1,j,k}$: les boîtes concernées sont

$$\{b_{0,\beta,\gamma} \mid 0 \leq \beta \leq n_2 - 1 \text{ et } 0 \leq \gamma \leq n_3 - 1\}$$

4 Calcul de la partie réciproque de l'energie

$$E_{rec} = \frac{1}{2\pi L^3} \sum_{\mathbf{m} \neq 0} \frac{exp\left(\frac{-\pi^2 \mathbf{m}^2}{\alpha^2}\right)}{\mathbf{m}^2} \mid S(\mathbf{m}) \mid^2 \tag{8}$$

4.1 L'algorithme séquentiel

Soient N charges ponctuelles $q_1, q_2, \cdots, q_N$ situées aux positions $r_1, r_2, \cdots, r_N$ dans une cellule unitaire U telle que $q_1 + q_2 + \cdots + q_N = 0$. Soient a_1, a_2, a_3 les vecteurs constituant les arêtes de la cellule unitaire U. On définit *les vecteurs réciproques conjugués* a_α^* par la relation $a_\alpha^* \cdot a_\beta = \delta_{\alpha\beta}$, avec $\delta_{\alpha\beta}$ désignant le delta de *Kronecker*

$$\delta_{\alpha\beta} = \begin{cases} 1 & \text{si } \alpha = \beta \\ 0 & \text{si } \alpha \neq \beta \end{cases} \tag{9}$$

Remarque : Les vecteurs a_1, a_2, a_3 ne sont pas nécessairement orthogonaux.

On définit *les coordonnées fractionnelles* de la charge q_i située à la position r_i par la relation $s_{\alpha i} = a_\alpha^* \cdot r_i$. Soit m le vecteur réciproque défini par la relation $\mathbf{m} = m_1 a_1^* + m_2 a_2^* + m_3 a_3^*$ avec m_1, m_2, m_3 des entiers qui ne sont pas tous égaux à zéro. On définit le *facteur de structure* $S(\mathbf{m})$ par l'équation (10).

$$S(\mathbf{m}) = \sum_{j=1}^{N} q_j \exp(2\pi i \mathbf{m} \cdot r_j) = \sum_{j=1}^{N} q_j \exp(2\pi i (m_1 s_{1j} + m_2 s_{2j} + m_3 s_{3j})) \tag{10}$$

Où $s_{\alpha j}$, $\alpha = 1, 2, 3$ sont les coordonnées fractionnelles de l'atome j. La partie réciproque de l'energie electrostatique est définie par l'équation (11), où $V = a_1 \cdot a_2 \cdot a_3$ désigne le volume de la cellule unitaire.

$$E_{rec} = \frac{1}{2\pi V} \sum_{\mathbf{m} \neq 0} \frac{\exp\left(-\pi^2 \mathbf{m}^2 / \beta^2\right)}{\mathbf{m}^2} S(\mathbf{m}) S(-\mathbf{m}) \tag{11}$$

Pour approcher les facteurs structurels éléctrostatiques, nous avons besoin d'interpoler les termes exponentiels apparaissant dans l'équation (10).

Soient K_1, K_2, K_3 les longueurs des arêtes de la grille d'interpolation et soit un point $\mathbf{r}$ dans la grille. On définit les coordonnées fractionnelles de $\mathbf{r}$: u_1, u_2, u_3 par la relation suivante : $u_\alpha = K_\alpha a_\alpha^* \cdot \mathbf{r}$ pour $\alpha = 1, 2, 3$. Si $\mathbf{r} = x_1 a_1 + x_2 a_2 + x_3 a_3$ alors on a $u_1 = K_1 x_1$, $u_2 = K_2 x_2$ et $u_3 = K_3 x_3$. D'où

$$\mathbf{m} \cdot \mathbf{r} = \frac{m_1 u_1}{K_1} + \frac{m_2 u_2}{K_2} + \frac{m_3 u_3}{K_3}$$

On obtient le résultat présenté dans l'équation (12).

$$\exp\left(2\pi i \mathbf{m} \cdot \mathbf{r}\right) = \exp\left(2\pi i \frac{m_1 u_1}{K_1}\right) \cdot \exp\left(2\pi i \frac{m_2 u_2}{K_2}\right) \cdot \exp\left(2\pi i \frac{m_3 u_3}{K_3}\right) \tag{12}$$

Soit $[u]$ la partie entière de l'entier u. On a $[u] \leq u \leq [u] + 1$. En utilisant une interpolation linéaire on peut approximer chaque terme de la partie droite de l'équation (12) par le terme de droite de l'équation (13).

$$\exp\left(2\pi i \frac{m_\alpha}{K_\alpha} u_\alpha\right) \approx (1 - (u_\alpha - [u_\alpha])) \cdot \exp\left(2\pi i \frac{m_\alpha}{K_\alpha} [u_\alpha]\right) + (u_\alpha - [u_\alpha]) \cdot \exp\left(2\pi i \frac{m_\alpha}{K_\alpha} ([u_\alpha] + 1)\right) \tag{13}$$

Cette approximation transforme une interpolation trilinéaire en une somme de huit termes. Deux méthodes d'interpolations sont utilisées pour approximer les facteurs structurels : *Interpolation de Lagrange* et *interpolation b-spline cardinale.* Dans la suite, on va s'interesser à la deuxième méthode d'interpolation.

4.1.1 Interpolation b-spline cardinale

Soit M_n la fonction définie récursivement par l'équation (14).

$$\begin{cases} M_2(u) = 1- \mid u-1 \mid & \text{pour } 0 \leq u \leq 2 \\ M_2(u) = 0 & \text{pour } u < 0 \text{ et } u > 2. \\ M_n(u) = \frac{u}{n-1} M_{n-1}(u) + \frac{n-u}{n-1} M_{n-1}(u-1) & \text{pour } n > 2. \end{cases} \tag{14}$$

Remarque : on déduit de l'équation (14) que $M_n(u) = 0$ en dehors de l'intervalle $[0, n]$.

Une propriété interessante des cardinales b-spline est le faite qu'elles sont faciles à dérriver analytiquement. Pour $n > 2$ on a

$$\frac{d}{du} M_n(u) = M_{n-1}(u) - M_{n-1}(u-1) \tag{15}$$

On définit le terme $b_i(m_i)$ par l'équation (16).

$$b_i(m_i) = \exp\left(2\pi i(n-1)m_i/K_i\right) \left(\sum_{k=0}^{n-2} M_n(k+1) \exp\left(2\pi i m_i k/K_i\right) \right)^{-1} \tag{16}$$

On peut montrer que quand n est pair on a le résultat de l'équation (17).

$$\exp\left(2\pi i \frac{m_i}{K_i} u_i\right) \approx b_i(m_i) \sum_{k=-\infty}^{\infty} M_n(u_i - k) \cdot \exp\left(2\pi i \frac{m_i}{K_i} k\right) \tag{17}$$

Dans le cas où n est impair et $2 \mid m_i \mid= K_i$ cette interpolation echoue. Dans ce cas on suppose que $b(m_i)$ est égale à zéro. En remplaçant le produit dans l'équation (12) par le produit des termes de droite de l'équation (17) on obtient le résultat décrit dans l'équation (19).

$$S(\mathbf{m}) \approx \widetilde{S}(\mathbf{m}) =$$

$$l \sum_{i=1}^{N} q_i b_1(m_1) b_2(m_2) b_3(m_3) \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} \sum_{k_3=-\infty}^{\infty} M_n(u_1 - k_1) M_n(u_2 - k_2) M_n(u_3 - k_3) \cdot$$
$$\exp\left(2\pi i \frac{m_1}{K_1} k_1\right) \cdot \exp\left(2\pi i \frac{m_2}{K_2} k_2\right) \cdot \exp\left(2\pi i \frac{m_3}{K_3} k_3\right) \tag{18}$$
$$= b_1(m_1) b_2(m_2) b_3(m_3) \sum_{k_1=0}^{K_1-1} \sum_{k_2=0}^{K_2-1} \sum_{k_3=0}^{K_3-1} Q(k_1, k_2, k_3) \exp\left(2\pi i \left(\frac{m_1 k_1}{K_1} + \frac{m_2 k_2}{K_2} + \frac{m_3 k_3}{K_3}\right)\right)$$
$$= b_1(m_1) b_2(m_2) b_3(m_3) F(Q)(m_1, m_2, m_3)$$

Où Q est la matrice définie par l'équation (19).

$$Q(k_1,k_2,k_3) = \sum_{i=1}^{N} \sum_{n_1,n_2,n_3} q_i M_n(u_{1i}-k_1-n_1K_1) M_n(u_{2i}-k_2-n_2K_2) M_n(u_{3i}-k_3-n_3K_3) \tag{19}$$

Rappelons que $u_{\alpha i}$ avec $\alpha = 1,2,3$ sont les coordonnées fractionnelles de la particule i.

$F(Q)$ est la transformée de Fourier discrète de la matrice Q.

L'approximation de la partie réciproque de l'energie est donnée par l'équation (20).

$$\begin{aligned}\widetilde{E}_{rec} = \\ \frac{1}{2\pi V} \sum_{\mathbf{m}\neq 0} \frac{\exp(-\pi^2 \mathbf{m}^2/\beta^2)}{\mathbf{m}^2} B(m_1,m_2,m_3) \cdot F(Q)(m_1,m_2,m_3)\ F(Q)(-m_1,-m_2,-m_3) \\ = \frac{1}{2} \sum_{m_1=0}^{K_1-1} \sum_{m_2=0}^{K_2-1} \sum_{m_3=0}^{K_3-1} Q(m_1,m_2,m_3) \cdot (\theta_{rec} \star Q)(m_1,m_2,m_3) \quad (20)\end{aligned}$$

Où B, C et θ_{rec} sont définis par les équations (22), (23) et (24). Le symbole $\star$ désigne la convolution.

$$\begin{aligned} B(m_1,m_2,m_3) &= \mid b_1(m_1) \mid^2 \cdot \mid b_2(m_2) \mid^2 \cdot \mid b_3(m_3) \mid^2 && (21) \\ C(m_1,m_2,m_3) &= \frac{1}{\pi V} \frac{\exp(-\pi^2\mathbf{m}^2/\beta^2)}{\mathbf{m}^2} \text{ pour } \mathbf{m} \neq 0, C(0,0,0) = 0 && (22) \\ \theta_{rec} &= F(B \cdot C) && (23) \\ \theta_{rec} \star Q &= F\left(F^{-1}(\theta_{rec} \star Q)\right) && (24) \\ &= K_1K_2K_3 \cdot F\left(F^{-1}(\theta_{rec}) \cdot F^{-1}(Q)\right) \\ &= K_1K_2K_3 \cdot F\left(B \cdot C \cdot F^{-1}(Q)\right) \end{aligned}$$

Avec $\mathbf{m}$ défini par $\mathbf{m} = m'_1\mathbf{a}^*_1 + m'_2\mathbf{a}^*_2 + m'_3\mathbf{a}^*_3$ où $m'_i = m_i$ pour $0 \leq m_i \leq K_i/2$ et $m'_i = m_i - K_i$ ailleurs .

4.2 Implantation de l'algorithme séquentiel

Dans [3], l'algorithme de calcul de $\widetilde{E}_{rec}$ est présenté de la façon suivante :

1. Calcul des coefficients $M_n(u_{\alpha i} - (k_\alpha + n_\alpha K_\alpha))$ avec $\alpha = 1,2,3$, $i = 1,\cdots,N$. Puisque, en général, on a $n < K_\alpha$ alors les valeurs de n_α pour lesquels on a $0 \leq u_{\alpha i} - (k_\alpha + n_\alpha K_\alpha) \leq n$ (i.e $M_n \neq 0$) sont -1 et 0.

$$\begin{cases} \text{si } u_{\alpha i} \geq n \Rightarrow n_\alpha = 0 \text{ et } \lceil u_{\alpha i} - n \rceil \leq k_\alpha \leq \lfloor u_{\alpha i} \rfloor \\ \text{si } u_{\alpha i} < n \Rightarrow n_\alpha = -1 \text{ et } \lceil K_\alpha + u_{\alpha i} - n \rceil \leq k_\alpha \leq \lfloor u_{\alpha i} + K_\alpha \rfloor \end{cases}$$

Donc, pour chaque particule i on calcule une matrice $(n \times n \times n)$ de termes qui contribueront au calcul de la matrice Q ces termes correspondent à des points (k_1, k_2, k_3) de la grille d'interpolation, les termes correspondant aux points restants de la grille d'interpolation sont égaux à 0.

2. A partir des coordonnées fractionnelles des particules et des coefficients M_p calculer la matrice Q en utilisant l'équation (19).

3. Calcul de la transformée de Fourier inverse de la matrice Q en utilisant $3DFFT$

4. En utilisant les équations (22) et (23) calculer la convolution de l'équation (25) en utilisant $3DFFT$

5. Calcul de $\widetilde{E}_{rec}$ en utilisant la formule de l'équation (20).

4.3 Calcul de la matrice Q en parallèle

Rappelons que les particules sont distribuées sur les différents processeurs selon un découpage géométrique correspondant à la grille de simulation.

1. On divise la grille d'interpolation qui est de taille $(K_1 \times K_2 \times K_3)$ en p régions de taille $\left(K_1 \times \frac{K_2}{\sqrt{p}} \times \frac{K_3}{\sqrt{p}}\right)$, où p désigne le nombre de processeurs de la machine cible. Chaque processeur sera responsable du calcul des éléments de la matrice Q appartenant à une région.

2. Chaque processeur calcule, pour chaque point de la grille d'interpolation, la somme des contributions des ses particules en ce point. Ensuite, on procède à une phase d'échange des données calculées dans laquelle chaque processeur envoit les éléments qu'il a calculés qui correspondent à des points de la grille d'interpolation appartenants à une région dont il n'est pas le responsable au processeur qui en est responsable. Ensuite, chaque processeur, pour chaque point de la région dont il est chargé, calcule la somme des éléments reçus des autres processeurs correspondant à ce point pour avoir la valeur de Q en ce point.

   Explication Cette méthode de calcul des éléments de la matrice Q revient à calculer localement par chaque processeur le terme suivant, $\text{Terme}_t(k_1, k_2, k_3) =$

$$\sum_{i \in P_t} \sum_{n_1, n_2, n_3} q_i M_n(u_{1i} - k_1 - n_1 K_1) M_n(u_{2i} - k_2 - n_2 K_2) M_n(u_{3i} - k_3 - n_3 K_3)$$

Où t désigne le numéro du processeur et P_t désigne l'ensemble des particules appartenant à la région de la grille dont le processeur reponsable est t. Donc

$$Q(k_1,k_2,k_3) = \sum_{t=1}^{p} \mathrm{Terme}_t(k_1,k_2,k_3)$$

Où p désigne le nombre de processeurs.

4.4 3D-FFT parallèle distribuée

Soit F la transformation de Fourier discrète de la fonction f dont on connaît la valeur aux points (j_1,j_2,j_3) avec $0 \le j_\alpha \le K_\alpha - 1$ pour $\alpha = 1,2,3$. On a

$$F_{k_1k_2k_3} = \sum_{j_3=0}^{K_3-1}\sum_{j_2=0}^{K_2-1}\sum_{j_1=0}^{K_1-1} \exp\left(2\pi i\left(\frac{j_1k_1}{K_1}+\frac{j_2k_2}{K_2}+\frac{j_3k_3}{K_3}\right)\right) f_{j_1j_2j_3} \tag{25}$$
$$\text{avec } 0 \le k_\alpha \le K_\alpha - 1 \text{ pour } \alpha = 1,2,3$$

On peut réecrire cette équation de la façon suivante

$$F_{k_1k_2k_3} = \sum_{j_3=0}^{K_3-1} \exp\left(2\pi i\frac{j_3k_3}{K_3}\right) \overbrace{\sum_{j_2=0}^{K_2-1} \exp\left(2\pi i\frac{j_2k_2}{K_2}\right) \underbrace{\sum_{j_1=0}^{K_1-1} \exp\left(2\pi i\frac{j_1k_1}{K_1}\right) f_{j_1j_2j_3}}_{F_{k_1}^{j_2j_3}}}^{F_{k_1k_2}^{j_3}} \tag{26}$$

Ce calcul est alors équivalent au calcul de trois 1D-FFT successives. On va étudier le coût de trois méthodes de parallélisation du calcul de l'équation (26).

Première méthode *le calcul d'une 3D-FFT est réalisé par un calcul local séquentiel de trois 1D-FFT successives avec des transpositions de données entre les trois 1D-FFT* [6].*Les transpositions des données entre les 1D-FFTs sont faites pour permettre à chaque processeur de calculer independement des autres processeurs les 1D-FFTs dont il est chargé.*

Le calcul de l'équation (26) se fait en trois étape :

Première étape calcul de la matrice $(F_{k_1}^{j_2j_3})$ définie par l'équation suivante

$$F_{k_1}^{j_2j_3} = \sum_{j_1=0}^{K_1-1} \exp\left(2\pi i\frac{j_1k_1}{K_1}\right) f_{j_1j_2j_3}$$

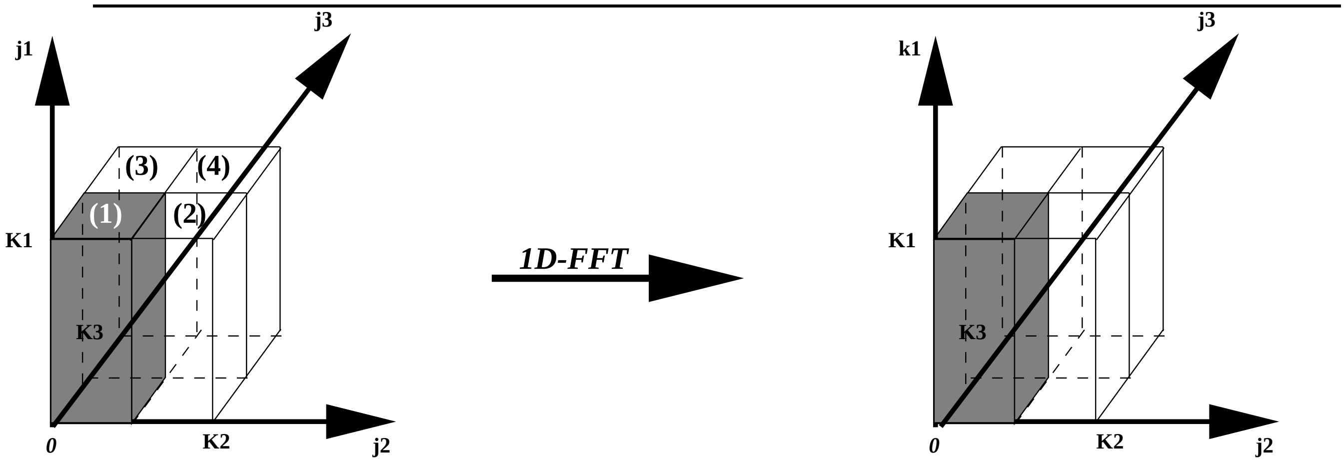

Figure 3: Exemple de calcul de la première étape sur 4 processeurs.

Le calcul de la matrice $\left(F_{k_1}^{j_2 j_3}\right)$ est égale au calcul de $K_2 K_3$ 1D-FFTs sur des vecteurs de taille K_1. Dans cette méthode, chaque processeur sera chargé de calculer en local et independement des autres processeurs $\frac{K_2 K_3}{p}$ 1D-FFTs sur des vecteurs de taille K_1.

Distribution de données : chaque processeur reçoit un bloc de la matrice f de taille égale à $\left(K_1 \times \frac{K_2}{\sqrt{p}} \times \frac{K_3}{\sqrt{p}}\right)$. Le processeur i tel que $i \in \{1..p\}$ aura le bloc de la matrice f défini par les équations suivantes

$$(f_{j_1 j_2 j_3}) \left\{ \begin{array}{rcccl} 0 & \leq & j_1 & \leq & K_1 - 1 \\ ((i-1) \textbf{ mod } \sqrt{p}) \frac{K_2}{\sqrt{p}} & \leq & j_2 & \leq & ((i-1) \textbf{ mod } \sqrt{p} + 1) \frac{K_2}{\sqrt{p}} - 1 \\ (i \textbf{ div } \sqrt{p}) \frac{K_3}{\sqrt{p}} & \leq & j_3 & \leq & (i \textbf{ div } \sqrt{p} + 1) \frac{K_3}{\sqrt{p}} - 1 \end{array} \right.$$

Dans la partie gauche de la figure (3) page 15, la colonne ombrée représente le bloc de la matrice $(f_{j_1 j_2 j_3})$ dont est chargé le processeur 1. La partie gauche de la figure représente la matrice $(f_{j_1 j_2 j_3})$ et la partie droite représente la matrice $(F_{k_1}^{j_2 j_3})$. Voici le pseudo-code pour le calcul de la première étape :

par $i \in \{1..p\}$ **do**
  begin
    for $j_2 = ((i-1) \textbf{ mod } \sqrt{p}) \frac{K_2}{\sqrt{p}}$ **to** $((i-1) \textbf{ mod } \sqrt{p} + 1) \frac{K_2}{\sqrt{p}} - 1$ **do**
      for $j_3 = (i \textbf{ div } \sqrt{p}) \frac{K_3}{\sqrt{p}}$ **to** $(i \textbf{ div } \sqrt{p} + 1) \frac{K_3}{\sqrt{p}} - 1$ **do**
        for $k_1 = 0$ **to** $K_1 - 1$ **do**

$$F_{k_1}^{j_2 j_3} = \sum_{j_1=0}^{K_1-1} \exp\left(2\pi i \frac{j_1 k_1}{K_1}\right) f_{j_1 j_2 j_3}$$

  end

Deuxième étape calcul de la matrice $(F_{k_1k_2}^{j_3})$ définie par l'équation suivante

$$F_{k_1k_2}^{j_3} = \sum_{j_2=0}^{K_2-1} \exp\left(2\pi i \frac{j_2k_2}{K_2}\right) F_{k_1}^{j_2j_3}$$

Le calcul de la matrice $\left(F_{k_1k_2}^{j_3}\right)$ est égale au calcul de K_1K_3 1D-FFTs sur des vecteurs de taille K_2. Dans cette méthode, chaque processeur sera chargé de calculer en local et independement des autres processeurs $\frac{K_1K_3}{p}$ 1D-FFTs sur des vecteurs de taille K_2.

Re-distribution des données : Pour pouvoir effectuer le calcul des 1D-FFTs en local et independement des autres processeurs, chaque processeur i doit recevoir des $\sqrt{p}-1$ processeurs j tel que $j \in \{(i \textbf{ div } \sqrt{p})\sqrt{p}+1, \cdot\cdot, i \textbf{ div } \sqrt{p}+\sqrt{p}\} \setminus \{i\}$ un nombre total de $(\sqrt{p}-1)$ blocs de taille $\left(\frac{K_1}{\sqrt{p}} \times \frac{K_2}{\sqrt{p}} \times \frac{K_3}{\sqrt{p}}\right)$. Le processeur i doit avoir le bloc de la matrice $(F_{k_1}^{j_2j_3})$ constitué des éléments suivants

$$\left(F_{k_1}^{j_2j_3}\right)\left\{\begin{array}{rcccl} ((i-1) \textbf{ mod } \sqrt{p})\,\frac{K_1}{\sqrt{p}} & \leq & k_1 & \leq & ((i-1) \textbf{ mod } \sqrt{p}+1)\,\frac{K_1}{\sqrt{p}}-1 \\ 0 & \leq & j_2 & \leq & K_2 \\ (i \textbf{ div } \sqrt{p})\,\frac{K_3}{\sqrt{p}} & \leq & j_3 & \leq & (i \textbf{ div } \sqrt{p}+1)\,\frac{K_3}{\sqrt{p}}-1 \end{array}\right. \tag{27}$$

Rappelons qu'à la fin de la première étape, le processeur i possedait les éléments définis par les équations suivantes

$$\left(F_{k_1}^{j_2j_3}\right)\left\{\begin{array}{rcccl} 0 & \leq & j_1 & \leq & K_1-1 \\ ((i-1) \textbf{ mod } \sqrt{p})\,\frac{K_2}{\sqrt{p}} & \leq & j_2 & \leq & ((i-1) \textbf{ mod } \sqrt{p}+1)\,\frac{K_2}{\sqrt{p}}-1 \\ (i \textbf{ div } \sqrt{p})\,\frac{K_3}{\sqrt{p}} & \leq & j_3 & \leq & (i \textbf{ div } \sqrt{p}+1)\,\frac{K_3}{\sqrt{p}}-1 \end{array}\right.$$

Alors le processeur p_i doit recevoir les $(\sqrt{p}-1)$ blocs suivants

$$\left(F_{k_1}^{j_2j_3}\right)\left\{\begin{array}{l} 0 \leq j_1 \leq K_1-1 \\ j\frac{K_2}{\sqrt{p}} \leq j_2 \leq (j+1)\frac{K_2}{\sqrt{p}}-1, \\ \qquad \text{avec } j \in \{0, \cdot\cdot, \sqrt{p}-1\} \setminus \{(i-1) \textbf{ mod } \sqrt{p}\} \\ (i \textbf{ div } \sqrt{p})\,\frac{K_3}{\sqrt{p}} \leq j_3 \leq (i \textbf{ div } \sqrt{p}+1)\,\frac{K_3}{\sqrt{p}}-1 \end{array}\right.$$

$$\left\{\begin{array}{llll} \left(F_{k_1}^{j_2j_3}\right) & 0 \leq k_1 \leq \frac{K_1}{\sqrt{p}}-1, & \frac{K_2}{\sqrt{p}} \leq j_2 \leq 2\frac{K_2}{\sqrt{p}}-1, & 0 \leq j_3 \leq \frac{K_3}{\sqrt{p}}-1 \\ \left(F_{k_1}^{j_2j_3}\right) & 0 \leq k_1 \leq \frac{K_1}{\sqrt{p}}-1, & 2\frac{K_2}{\sqrt{p}} \leq j_2 \leq 3\frac{K_2}{\sqrt{p}}-1, & 0 \leq j_3 \leq \frac{K_3}{\sqrt{p}}-1 \\ \vdots & & & \\ \left(F_{k_1}^{j_2j_3}\right) & 0 \leq k_1 \leq \frac{K_1}{\sqrt{p}}-1, & \frac{\sqrt{p}-1}{\sqrt{p}}K_2 \leq j_2 \leq K_2-1, & 0 \leq j_3 \leq \frac{K_3}{\sqrt{p}}-1 \end{array}\right.$$

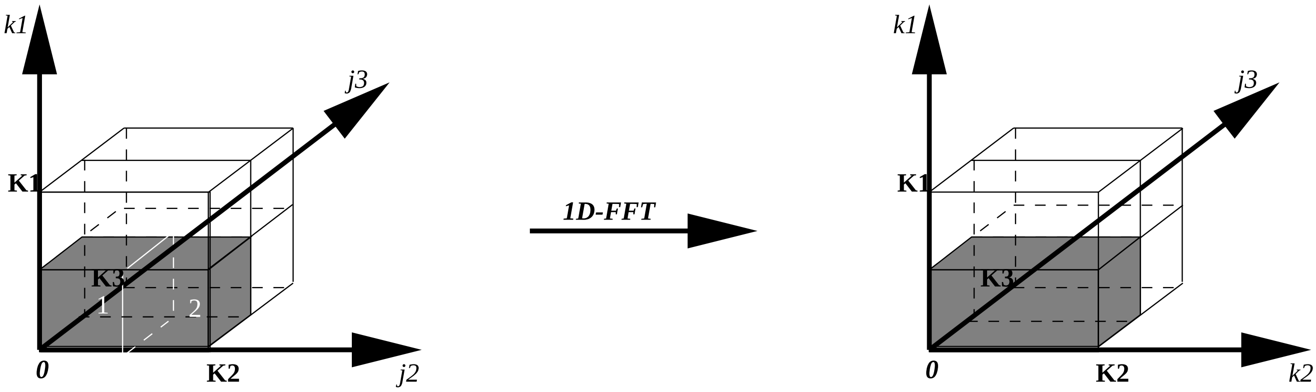

Figure 4: Exemple de calcul de la deuxième étape sur 4 processeurs.

Donc à la fin de la redistribution, chaque processeur aura un bloc de taille $\left(\frac{K_1}{\sqrt{p}} \times K_2 \times \frac{K_3}{\sqrt{p}}\right)$ de la matrice $\left(F_{k_1}^{j_2 j_3}\right)$. Ce bloc représente le volume des données qui sont utiles pour le calcul de la deuxième étape.

Dans la figure (4) page 17, la partie gauche de la figure représente la matrice $(F_{k_1}^{j_2 j_3})$ et la partie droite représente la matrice $(F_{k_1 k_2}^{j_3})$. Dans la partie gauche, la colonne ombrée représente le bloc de la matrice $(F_{k_1}^{j_2 j_3})$ dont est chargé le processeur 1, le bloc numéro 1 de cette colonne est une partie du résultat du calcul local au processeur 1 de la première étape. Le bloc numéro 2 est reçu du processeur 2 et représente une partie du resultat de son calcul de la première étape.

Pseudo-code pour le calcul de la deuxième étape :

par $i \in \{1..p\}$ **do**
  begin
    recevoir les $(\sqrt{p} - 1)$ **blocs manquants de la matrice** $\left(F_{k_1}^{j_2 j_3}\right)$
    for $k_1 = ((i-1) \textbf{ mod } \sqrt{p}) \frac{K_1}{\sqrt{p}}$ **to** $((i-1) \textbf{ mod } \sqrt{p} + 1) \frac{K_1}{\sqrt{p}} - 1$ **do**
      for $j_3 = (i \textbf{ div } \sqrt{p}) \frac{K_3}{\sqrt{p}}$ **to** $(i \textbf{ div } \sqrt{p} + 1) \frac{K_3}{\sqrt{p}} - 1$ **do**
        for $k_2 = 0$ **to** $K_2 - 1$ **do**

$$F_{k_1 k_2}^{j_3} = \sum_{j_2=0}^{K_2-1} \exp\left(2\pi i \frac{j_2 k_2}{K_2}\right) F_{k_1}^{j_2 j_3}$$

  end

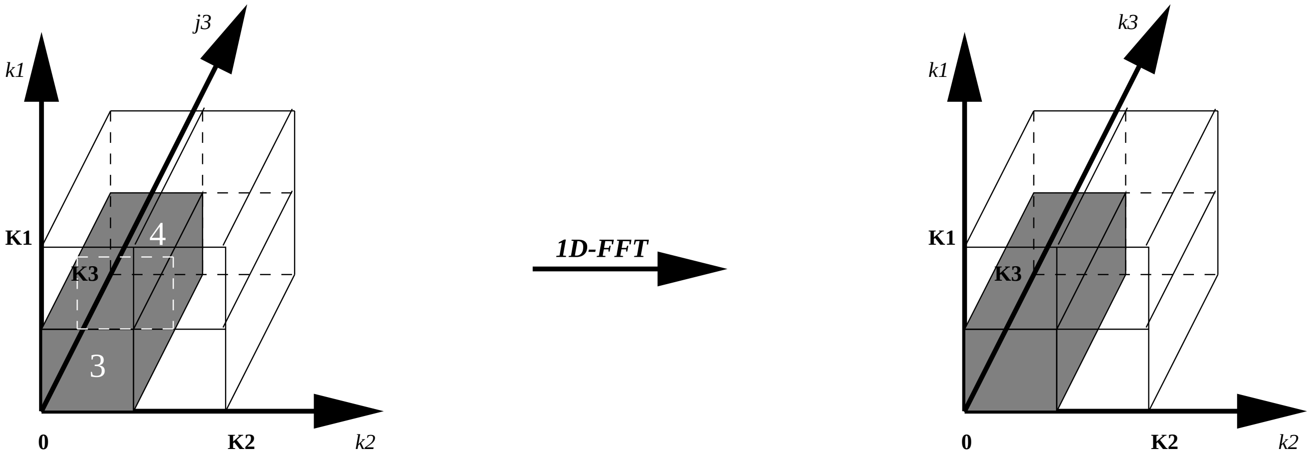

Figure 5: Exemple de calcul de la troisième étape sur 4 processeurs.

Troisième étape calcul de la matrice finale ($F_{k_1k_2k_3}$) définie par l'équation suivante

$$F_{k_1k_2k_3} = \sum_{j_3=0}^{K_3-1} \exp\left(2\pi i \frac{j_3 k_3}{K_3} F_{k_1k_2}^{j_3}\right)$$

Redistribution des données : Comme dans la deuxième étape, pour pouvoir effectuer le calcul des FFTs en local, chaque processeur doit recevoir des autres processeurs un nombre total de ($\sqrt{p}-1$) blocs de taille $\left(\frac{K_1}{\sqrt{p}} \times \frac{K_2}{\sqrt{p}} \times \frac{K_3}{\sqrt{p}}\right)$. Après cette redistribution chaque processeur aura un bloc de de données utiles pour le calcul de la troisième étape égale à ($\frac{K_1}{\sqrt{p}} \times \frac{K_2}{\sqrt{p}} \times K_3$) de la matrice $\left(F_{k_1k_2}^{j_3}\right)$.

Dans la partie gauche de la figure (5) page 18, la colonne ombrée représente le bloc de la matrice ($F_{k_1k_2}^{j_3}$) dont est chargé le processeur 1. La partie gauche de la figure représente la matrice ($F_{k_1k_2}^{j_3}$) et la partie droite représente la matrice ($F_{k_1k_2k_3}$). le bloc numéro 3 de cette colonne est une partie du résultat du calcul local au processeur 1 de la deuxième étape. Le bloc numéro 4 est reçu du processeur 3 et représente une partie du resultat de son calcul de la deuxième étape.

Coût BSP de la méthode le coût BSP de la première méthode est donné par l'équation suivante

$$
\begin{aligned}
T_1 &= O\left(\frac{K_2K_3}{p}\left(K_1\log\left(K_1\right)\right)\right) + \left(\left(\sqrt{p}-1\right)\frac{K_1K_2K_3}{p\sqrt{p}}\right)g + l + \\
&\quad O\left(\frac{K_1K_3}{p}\left(K_2\log\left(K_2\right)\right)\right) + \left(\left(\sqrt{p}-1\right)\frac{K_1K_2K_3}{p\sqrt{p}}\right)g + l + \\
&\quad O\left(\frac{K_1K_2}{p}\left(K_3\log\left(K_3\right)\right)\right). \\
&= O\left(\frac{K_1K_2K_3}{p}\left(\log(K_1)+\log(K_2)+\log(K_3)\right)\right) + \left(2\frac{\sqrt{p}-1}{p\sqrt{p}}K_1K_2K_3\right)g + 2l
\end{aligned}
$$

Deuxième méthode *les 1D-FFTs sont calculés en parallèle selon l'algorithme présenté dans* [7].

Dans cette méthode aussi, le calcul de l'équation (26) se fait en trois étape

Première étape : calcul de la matrice $(F_{k_1}^{j_2j_3})$.

Distribution des données : Chaque processeur reçoit un bloc de la matrice $(f_{j_1j_2j_3})$ de taille $(\frac{K_1}{p} \times K_2 \times K_3)$. On calculera alors K_2K_3 transformations de Fourier unidimentionnelles en parallèle de vecteurs de taille K_1 (i.e on appliquera K_2K_3 fois l'algorithme présenté dans [7] sur des vecteurs de taille K_1).

Deuxième étape : calcul de la matrice $(F_{k_1k_2}^{j_3})$.

Redistribution des données : Chaque processeur doit recevoir des autres processeurs $(p-1)$ blocs de la matrice $(F_{k_1}^{j_2j_3})$ de taille $(\frac{K_1}{p} \times \frac{K_2}{p} \times K_3)$. Nous allons illustrer ce calcul par un exemple. Supposons qu'un processeur p_0 avait le bloc de la matrice f défini par léquation suivante

$$
(f_{j_1j_2j_3})\begin{cases} 0 \le j_1 \le \frac{K_1}{p} - 1 \\ 0 \le j_2 \le K_2 - 1 \\ 0 \le j_3 \le K_3 - 1 \end{cases}
$$

Donc, à la fin de la première étape le processeur p_0 aura le bloc de la matrice $F_{k_1}^{j_2j_3}$ défini par l'équation suivante

$$
(F_{k_1}^{j_2j_3})\begin{cases} 0 \le k_1 \le \frac{K_1}{p} - 1 \\ 0 \le j_2 \le K_2 - 1 \\ 0 \le j_3 \le K_3 - 1 \end{cases}
$$

Or pour exécuter la deuxième étape le processeur p_0 doit avoir le bloc

$$
(F_{k_1}^{j_2j_3})\begin{cases} 0 \le k_1 \le K_1 - 1 \\ 0 \le j_2 \le \frac{K_2}{p} - 1 \\ 0 \le j_3 \le K_3 - 1 \end{cases}
$$

Donc le processeur p_0 doit recevoir des autres processeurs $(p-1)$ blocs de taile $(\frac{K_1}{p} \times \frac{K_2}{p} \times K_3)$ correspondants aux éléments

$$(F_{k_1}^{j_2 j_3}) \begin{cases} \frac{K_1}{p} \leq k_1 \leq K_1 - 1 \\ 0 \leq j_2 \leq \frac{K_2}{p} - 1 \\ 0 \leq j_3 \leq K_3 - 1 \end{cases}$$

Troisième étape : calcul de la matrice finale $(F_{k_1 k_2 k_3})$.

Redistribution de données pour les même raisons expliquées dans la deuxième étape, chaque processeur doit recevoir $(p-1)$ blocs de taille $(K_1 \times \frac{K_2}{p} \times \frac{K_3}{p})$ de la matrice $(F_{k_1 k_2}^{j_3})$.

Coût BSP de la deuxième méthode le coût BSP de la deuxième méthode est donné par l'équation suivante

$$\begin{aligned} T_2 &= K_2 K_3 \left(O\left(\frac{K_1 \log(K_1)}{p}\right) + \frac{K_1 \log(K_1)}{p \log\left(\frac{K_1}{p}\right)} g + \frac{\log(K_1)}{\log\left(\frac{K_1}{p}\right)} l \right) + \\ &\quad \left(\frac{p-1}{p^2} K_1 K_2 K_3\right) g + l + \\ &\quad K_1 K_3 \left(O\left(\frac{K_2 \log(K_2)}{p}\right) + \frac{K_2 \log(K_2)}{p \log\left(\frac{K_2}{p}\right)} g + \frac{\log(K_2)}{\log\left(\frac{K_2}{p}\right)} l \right) + \\ &\quad \left(\frac{p-1}{p^2} K_1 K_2 K_3\right) g + l + \\ &\quad K_1 K_2 \left(O\left(\frac{K_3 \log(K_3)}{p}\right) + \frac{K_3 \log(K_3)}{p \log\left(\frac{K_3}{p}\right)} g + \frac{\log(K_3)}{\log\left(\frac{K_3}{p}\right)} l \right) \\ &= O\left(\frac{K_1 K_2 K_3}{p} \left(\log(K_1) + \log(K_2) + \log(K_3)\right)\right) + \\ &\quad \frac{K_1 K_2 K_3}{p} \left(\frac{\log(K_1)}{\log\left(\frac{K_1}{p}\right)} + \frac{\log(K_2)}{\log\left(\frac{K_2}{p}\right)} + \frac{\log(K_3)}{\log\left(\frac{K_3}{p}\right)} + 2\frac{p-1}{p} \right) g + \\ &\quad \left(\frac{\log(K_1)}{\log\left(\frac{K_1}{p}\right)} + \frac{\log(K_2)}{\log\left(\frac{K_2}{p}\right)} + \frac{\log(K_3)}{\log\left(\frac{K_3}{p}\right)} + 2 \right) l \end{aligned}$$

Comparaison entre la première et la deuxième méthode : une comparaison des coûts BSP des deux méthodes de parallélisation du calcul de 3D-FFT montre que la première méthode est moins coûteuse que la seconde. Un autre avantage de la première méthode est que celle-ci évite de réecrire le code séquentiel du calcul de 1D-FFT. On peut alors utiliser les librairies de fonctions mathématiques du constructeur qui sont, en général, optimisées.

Troisième méthode *Cette méthode est semblable à la première méthode sauf pour la distribution des données. Cette nouvelle distribution des données permet de réduire les communications dues aux transpositions des données.*

Première étape *calcul de la matrice* $(F_{k_1}^{j_2 j_3})$.

Distribution des données : chaque processeur reçoit un bloc de la matrice f de taille égale à $(K_1 \times K_2 \times \frac{K_3}{p})$ (voir FIG. : 6).

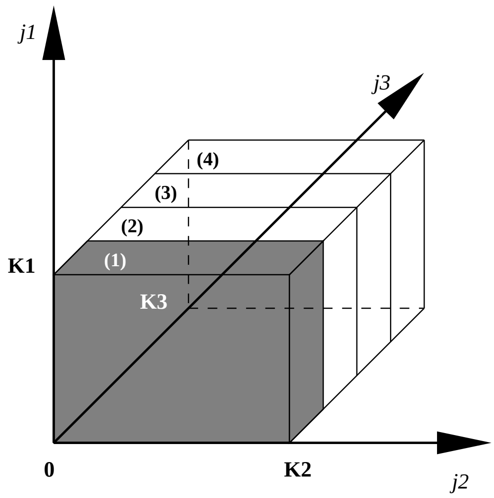

Figure 6: Exemple de distribution des données sur 4 processeurs.

Deuxième étape *calcul de la matrice* $(F^{j_3}_{k_1 k_2})$.

Ici il n'y aura pas de redistribution des données au contraire de la première méthode.

Troisième étape *calcul de la matrice finale* $(F_{k_1 k_2 k_3})$.

Redistribution des données : pour pouvoir effectuer le calucl des FFTs en local, chaque processeur doit recevoir des autres processeurs un nombre total de $(p-1)$ blocs de taille $(K_1 \times \frac{K_2}{p} \times \frac{K_3}{p})$. Après cette redistribution chaque processeur aura un bloc de taille égale à $(K_1 \times \frac{K_2}{p} \times K_3)$ (voir FIG : 7).

Coût BSP de la troisième méthode le coût BSP de la troisième méthode est donné par l'équation suivante

$$\begin{aligned}
T_3 &= O\left(\frac{K_2 K_3}{p}\left(K_1 \log\left(K_1\right)\right)\right) + \\
&\quad O\left(\frac{K_1 K_3}{p}\left(K_2 \log\left(K_2\right)\right)\right) + \left((p-1)\frac{K_1 K_2 K_3}{p^2}\right) g + l + \\
&\quad O\left(\frac{K_1 K_2}{p}\left(K_3 \log\left(K_3\right)\right)\right). \\
&= O\left(\frac{K_1 K_2 K_3}{p}\left(\log(K_1) + \log(K_2) + \log(K_3)\right)\right) + \left((p-1)\frac{K_1 K_2 K_3}{p^2}\right) g + l
\end{aligned}$$

Comparaison entre la première et la troisième méthode : une comparaison des coûts BSP de la première et de la troisième méthode montre que la troisième méthode est moins coûteuse que le première : il y a moins de communications que dans la première méthode. La taille des données distribuées est la même dans les deux méthodes. Dans la troisième méthode

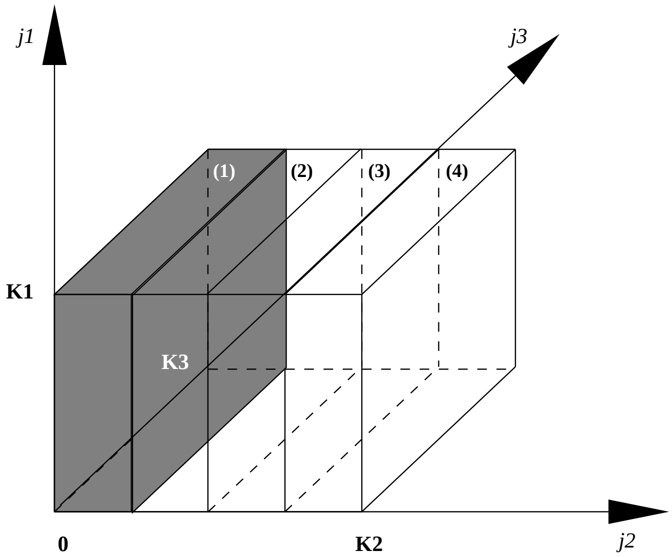

Figure 7: Exemple de redistribution des données pour le calcul de la troisième étape sur 4 processeurs.

exige la condition suivante $K_3 \geq p$ et $K_2 \geq p$. La première méthode exige la condition suivante $K_1 \geq \sqrt{p}, K_2 \geq \sqrt{p}$ et $K_3 \geq p$.

4.5 Calcul de la convolution $\theta_{rec} \star Q$ en parallèle

4.5.1 Calcul de la matrice B

On divise la matrice B en p blocs de taille égale à $\left(\frac{K_1}{\sqrt{p}} \times \frac{K_2}{\sqrt{p}} \times K_3\right)$. Chaque processeur calculera les éléments de la matrice B appartenant à un bloc.

4.5.2 Calcul de la matrice C

On divise la matrice C en p blocs de taille égale à $\left(\frac{K_1}{\sqrt{p}} \times \frac{K_2}{\sqrt{p}} \times K_3\right)$. Chaque processeur calculera les éléments de la matrice C appartenant à un bloc. Il faut que le bloc de C qu'un processeur reçoit corresponde avec le bloc de la matrice B déjà reçu, de sorte que chaque processeur puisse faire un produit point à point des deux blocs des matrice B et C.

4.5.3 Calcul du produit point à point $B \cdot C$

Chaque processeur calcul le produit point à point des blocs des matrices B et C dont il est chargé.

4.5.4 Calcul du produit point à point $(B \cdot C) \cdot F^{-1}(Q)$

Chaque processeur calcule le produit point à point des éléments des blocs des deux metrices $(B \cdot C)$ et $F^{-1}(Q)$ dont il est reponsable.

4.5.5 Calcul de la transformée de Fourier $F\left((B \cdot C) \cdot F^{-1}(Q)\right)$

Le choix de le méthode de redistribution des matrices B et C a été fait pour éviter une redistribution des données pour le calcul de $F\left((B \cdot C) \cdot F^{-1}(Q)\right)$. Rappelons qu'après calcul de la matrice $F^{-1}(Q)$ chaque processeur avait un bloc de taille $\left(\frac{K_1}{\sqrt{p}} \times \frac{K_2}{\sqrt{p}} \times K_3\right)$ de cette matrice. Pour éviter la redistribution des données de cette matrice, le calcul de $F\left((B \cdot C) \cdot F^{-1}(Q)\right)$ sera fait dans l'ordre inverse de celui qui a été fait pour le calcul de $F^{-1}(Q)$. Ceci est possible en inversant les trois somme de la 3D-FFT. Pour le calcul de $F^{-1}(Q)$ on avait suivi l'ordre

$$\sum_{j_3=0}^{K_3-1} \sum_{j_2=0}^{K_2-1} \sum_{j_1=0}^{K_1-1} \cdots$$

Pour le calcul de $F\left((B \cdot C) \cdot F^{-1}(Q)\right)$ on suivra l'ordre

$$\sum_{j_1=0}^{K_1-1} \sum_{j_2=0}^{K_2-1} \sum_{j_3=0}^{K_3-1} \cdots$$

4.5.6 Calcul de $\widetilde{E}_{rec}$

Chaque processeur calculera en local une partie de la somme contenu dans la formule de $\widetilde{E}_{rec}$. Par exemple un processeur calculera en local la somme

$$\sum_{m_1=0}^{K_1-1} \sum_{m_2=0}^{\frac{K_2}{\sqrt{p}}-1} \sum_{m_3=0}^{\frac{K_3}{\sqrt{p}}-1} Q(m_1, m_2, m_3) \cdot (\theta_{rec} \star Q)(m_1, m_2, m_3)$$

Ensuite une réduction en parallèle puis un produit par $\frac{1}{2}$ premettra de calculer $\widetilde{E}_{rec}$.

5 Calcul des forces

5.1 Calcul de la force directe

Soit $F_{dir}(i)$ la force directe appliquée à la particule i. On a

$$F_{dir}(i) = -\frac{\partial E_{dir}}{\partial \mathbf{r}_i}$$

On introduit ici quelques notations que nous allons utiliser dans la suite
Notations

$\mathbf{r}_{ij,\mathbf{n}} = \mid \mathbf{r}_i - \mathbf{r}_i + \mathbf{n} \mid$	avec $\mathbf{n} = (n_1L_1, n_2L_2, n_3L_3)\,; n_1, n_2, n_3 \in \mathbf{Z}$ non tous nuls
$F^t_{dir}(i)$	avec $t = x, y, z$ est la composante $tduvecteurF_{dir}(i)$
$(\mathbf{r}_{ij,\mathbf{n}})_t = \mathbf{r}_{it} - \mathbf{r}_{jp} + n_tL_t$	avec $t = 1, 2, 3$

La formule définissant la force directe appliquée à la particule i est présentée dans l'équation (28)

$$F^t_{dir}(i) = q_i \sum_{\mathbf{n}} \sum_{j=1, j\neq i}^{N} q_j \frac{(\mathbf{r}_{ij,\mathbf{n}})_t}{(\mathbf{r}_{ij,\mathbf{n}})^3} \left[\mathrm{erfc}(\alpha \mathbf{r}_{ij,\mathbf{n}}) + \frac{2\alpha}{\sqrt{\pi}} \mathbf{r}_{ij,\mathbf{n}} \exp\left(-(\alpha \mathbf{r}_{ij,\mathbf{n}})^2\right)\right] \tag{28}$$

Où α est un paramètre de l'algorithme PME qu'on choisit selon la vitesse et la précision voulues pour l'exécution de l'algorithme. Dans la deuxième somme, on ne considère que les particules j qui sont à l'intérieur de la sphère dont le centre est la particule i et dont le rayon est R_{cutoff}.

5.1.1 Calcul en parallèle de la force directe

Le calcul de la force directe en parallèle est similaire à celui du calcul de l'énergie directe i.e. chaque processeur calculera les forces qui s'appliqueront aux particules dont il est chargé.

5.2 Calcul de la force réciproque

La force réciproque pour chaque particule est obtenue en dérrivant la partie droite de l'équation ((20) page 12) par rapport à position $\mathbf{r}_i$ de la particule. On obtient l'équation suivante avec $\alpha = 1, 2, 3$

$$\frac{\partial \widetilde{E}_{rec}}{\partial \mathbf{r}_{\alpha i}} = \sum_{m_1=0}^{K_1-1} \sum_{m_2=0}^{K_2-1} \sum_{m_3=0}^{K_3-1} \frac{\partial Q}{\partial \mathbf{r}_{\alpha i}}(m_1, m_2, m_3) \cdot (\theta_{rec} \star Q)(m_1, m_2, m_3) \tag{29}$$

5.2.1 Calcul de $\frac{\partial Q}{\partial \mathbf{r}_{\alpha i}}$

On suppose que les boîtes de simulation et d'interpolation sont orthogonales. On a alors le résultat suivant

$$u_{\alpha i} = K_\alpha \frac{r_{\alpha i}}{L_\alpha}$$

Avec $r_{\alpha i}$ est la α ème composante du vecteur de position la particule i relative à la boîte de simulation (i.e $0 \leq r_{\alpha i} < L_\alpha$). Donc

$$\frac{\partial u_{\alpha i}}{\partial r_{\alpha i}} = \frac{K_\alpha}{L_\alpha}$$

On a aussi le résultat suivant

$$\frac{d}{du}M_n(u) = M_{n-1}(u) - M_{n-1}(u-1)$$

le calcul de $\frac{\partial Q}{\partial \mathbf{r}_{\alpha i}}$ est obtenu en utilisant l'équation

$$\frac{\partial Q}{\partial \mathbf{r}_{\alpha i}}(k_1, k_2, k_3) =$$

$$\sum_{i=1}^{N} \sum_{n_1,n_2,n_3} q_i \frac{\partial}{\partial \mathbf{r}_{\alpha i}} \left(M_n(u_{1i} - k_1 - n_1K_1)M_n(u_{2i} - k_2 - n_2K_2)M_n(u_{3i} - k_3 - n_3K_3)\right)$$

$$= \sum_{i=1}^{N} \sum_{n_1,n_2,n_3} q_i \{A + B + C\}$$

où A, B, C sont les termes

$$\frac{K_1}{L_1}\Big(M_{n-1}(u_{1i}-k_1-n_1K_1)-M_{n-1}(u_{1i}-1-k_1-n_1K_1)\Big)M_n(u_{2i}-k_2-n_2K_2)M_n(u_{3i}-k_3-n_3K_3)$$

$$M_n(u_{1i}-k_1-n_1K_1)\cdot\frac{K_2}{L_2}\Big(M_{n-1}(u_{2i}-k_2-n_2K_2)-M_{n-1}(u_{2i}-1-k_2-n_2K_2)\Big)M_n(u_{3i}-k_3-n_3K_3)$$

et

$$M_n(u_{1i}-k_1-n_1K_1)M_n(u_{2i}-k_2-n_2K_2)\cdot\frac{K_3}{L_3}\Big(M_{n-1}(u_{3i}-k_3-n_3K_3)-M_{n-1}(u_{3i}-1-k_3-n_3K_3)\Big).$$

Rappelons que $M_n(u_{\alpha i} - k_\alpha - n_\alpha K_\alpha) \neq 0$ dans les cas suivants

$$\begin{cases} \text{si } u_{\alpha i} \geq n \Rightarrow n_\alpha = 0 \text{ et } \lceil u_{\alpha i} - n \rceil \leq k_\alpha \leq \lfloor u_{\alpha i} \rfloor \\ \text{si } u_{\alpha i} < n \Rightarrow n_\alpha = -1 \text{ et } \lceil K_\alpha + u_{\alpha i} - n \rceil \leq k_\alpha \leq \lfloor u_{\alpha i} + K_\alpha \rfloor \end{cases}$$

Pour calculer en parallèle de la matrice $\frac{\partial Q}{\partial \mathbf{r}_{\alpha i}}$, on suivera la même méthode qu'on a utilisé pour calculer en parallèle la matrice Q.

5.3 Calcul de la force de correction

La force de correction appliquée à la particule i est donnée par l'équation suivante

$$F_{corr}(i) = -\frac{\partial E_{corr}}{\partial \mathbf{r}_i}$$

Dans [10], le calcul de cette force est négligé.

6 Conclusion

Les programmes C+MPI pour les algorithmes parallèles FFT et PME sont disponibles sur la page web du projet:
www.univ-orleans.fr/SCIENCES/LIFO/Members/ghains/ihperf98/ihperf.htm.

Les premiers tests effectués ont montré une très bonne accélération comme l'indiquent les temps de calcul (voir annexe) sur 4-64 processeurs d'un système Cray T3E. Nous croyons que l'optimisation de l'équilibre calcul-communication permettra de porter cette application sur des réseaux moins performants comme ceux des grappes de PC.

Nous remercions le centre ZAM de Jüelich en Allemagne pour l'accès fourni gratuitement à leurs systèmes Cray T3E.

References

[1] D. M. Beazley and P. S. Lomdahl. Message Passing Multi-Cell Molecular Dynamics on the Connection Machine 5. *Parallel Computing*, 1993.

[2] Pierre-Eric Bernard. *Parallélisation et multiprogrammation pour une application irrégulière de dynamique moléculaire opérationnelle.* Thèse de doctorat, Institut National Polythechnique de Grenoble, 1997.

[3] Ulrich Eßmann, Lalith Perera, Max L. Berkowitz, Tom Darden, Hsing Lee, and Lee G. Pedersen. A smooth particle mesh Ewald method. *Journal of Chemical Physics*, 1995.

[4] P. Ewald. Die Berechnung optischer und elektrostatischer Gitterpotentiale. *Ann. Phys.*, pages 253–287, 1921.

[5] Robert Ferrell and Edmund Bertschinger. Particle-Mesh methods on the connection machine. *International Journal of Modern Physics C*, 1994.

[6] H.Q.Ding, D.B.Gennery, and F.D.Ferraro. A Portable 3D FFT Package for Distributed-Memory Parallel Architectures. In *Proceedings of the 7th SIAM conference on Parallel Processing for Scientific Computing*. SIAM Press, 1995.

[7] W. F. McColl. Scalability, portability and predictability: The BSP approach to parallel programming. *Future Generation Computer Systems*, 12(4):265–272, 1996.

[8] S. Plimpton. Fast Parallel Algorithms for Short-Range Molecular Dynamics. *Journal of Computational Physics*, pages 1–19, 1995.

[9] Abdulnour Toukmaji, Daniel Paul, and John A. Board Jr. Distributed Particle-Mesh Ewald: A Parallel Ewald Summation Method. In *Parallel and Distributed Processing Techniques and Applications*, 1996.

[10] Abdulnour Y. Toukmaji and John A. Board Jr. Ewald Summation Techniques in Perspective: A Survey. *Elsevier Science*, 1996.

[11] L. G. Valiant. A bridging model for parallel computation. *Communications of the ACM*, 33(8):103, August 1990.

A Tests de performance sur système Cray-T3E

```
[sur une machine Cray T3E du centre ZAM-Juelich en Allemagne]
-----------
--> Execution en mode interactif.

# 5000 molecules d'eau (15000 atomes)
# Taille de la boite de simulation = 10
# Nombre de points dans la grille d'interpolation (dans chaque direction de l'espace) =
# Ordre d'interpolation = 4
# Rayon de coupure = 1.25

--------------------------------------------------------------------------
 | P = 4 |
 ---------
Energie directe = -11334.080671    temps = 75.403281
Energie reciproque = 3.268854      temps = 352.596192
Energie Self = -20602.336124       temps = 0.001756
--------------------------------------------------------------------------
| P = 8 |
---------
Energie directe = -11278.658300    temps = 22.246457
Energie reciproque = 1.934571      temps = 182.738225
Energie Self = -20602.336124       temps = 0.000967
--------------------------------------------------------------------------
| P = 16 |
----------
Energie directe = -11166.655554    temps = 10.277116
Energie reciproque = 0.894135      temps = 104.192052
Energie Self = -20602.336124       temps = 0.000628
--------------------------------------------------------------------------
| P = 32 |
----------
Energie directe = -11082.962222    temps = 4.216308
Energie reciproque = 0.000000      temps = 54.446030
Energie Self = -20602.336124       temps = 0.000444
--------------------------------------------------------------------------
```

```
| P = 64 |
----------
Energie directe = -10984.411878    temps = 1.900386
Energie reciproque = 0.000000      temps = 31.152724
Energie Self = -20602.336124       temps = 0.000407
```