PARTICLE ALGORITHMS FOR MASSIVELY PARALLEL MACHINES

Didier Issautier [1] , [2]

Abstract

We describe in this paper a strategy for parallelising a weighted particle method for solving the Boltzmann (B.G.K.) equation on MIMD and SIMD machines. The performance results obtained on two parallel machines, the Meiko Concerto and the Connection Machine are presented and compared with those obtained on a Cray-YMP.

ALGORITHMES PARTICULAIRES POUR DES CALCULATEURS A ARCHITECTURE MASSIVEMENT PARALLELES

Résumé

Nous présentons une stratégie pour implémenter sur des calculateurs massivement parallèles de type SIMD et MIMD, une méthode particulaire déterministe pour résoudre l'équation de Boltzmann (modéle B.G.K.). Des résultats de performance obtenus sur la machine Meiko Concerto [3] et sur la Connection Machine[3] sont présentés et comparés aux performances obtenues sur un Cray-YMP. [3].

[1]Laboratoire J.-A. Dieudonné, U.R.A. 168 du CNRS Université de Nice Sophia-Antipolis Parc Valrose, BP 71, 06108 NICE Cédex 02

[2]CERMICS, Sophia-Antipolis

[3]Les machines utilisées appartiennent au centre de calcul de la région P.A.C.A.

Introduction

La simulation numérique d'équations cinétiques de type Boltzmann, par des méthodes statistiques de type Monte-Carlo ou des méthodes déterministes, a connu un intérêt croissant au cours des dernières années. Cependant, pour simuler des phénomènes physiques complexes, de telles méthodes nécessitent un nombre très important de particules, ce qui se traduit par des temps calcul qui peuvent être prohibitifs. L'avènement récent de machines massivement parallèles suscite l'espoir de simuler ce type de phénomènes en des temps raisonnables. De nombreux travaux, parmi lesquels nous pouvons citer L.N.Long, M.Kamon et J.Myczkowski [6], F.Coron, P.Homsi [7], ont permis des avancées dans ce domaine.
Dans un précédent rapport nous rapportions les premiers résultats de performance obtenus sur une machine massivement parallèle de type SIMD, la Connection Machine [2]. Depuis lors, nous avons implémenté ce type de méthode sur une machine parallèle de type MIMD, la Meiko-CS1. Le but de ce rapport est de comparer les performances obtenues sur ces deux types de machine qui représentent les deux grands modèles existant à ce jour dans le parallélisme massif.
Dans la première partie nous rappellerons brièvement le modèle physique ainsi que la méthode numérique utilisée. Ensuite nous présenterons pour chacune des machines parallèles utilisées ses principales caractéristiques ainsi que l'algorithme employé. Enfin nous présenterons, dans la dernière partie, les résultats de performance obtenus pour les machines ci-dessus et les comparerons à ceux obtenus sur un Cray-YMP.

1 Modèle physique et approximation numérique

1.1 Le modèle B.G.K.

Le modèle **B.G.K.** pour l'équation de **Boltzmann** est un modèle cinétique simplifié pour décrire l'évolution d'un gaz [1]. En dépit de sa relative simplicité, ce modèle contient la plupart des propriétés fondamentales de la dynamique des fluides telles que la conservation de la masse, de l'impulsion et de l'énergie. L'état du gaz est modélisé à l'aide d'une fonction de distribution donnant la densité $f(t,x,v)$ de particules qui à l'instant $t > 0$ et à la position $x \in I\!R^N$ se déplacent à la vitesse $v \in I\!R^N$. L'évolution de la densité $f(t,x,v)$ est décrite par l'équation suivante :

$$\frac{\partial f}{\partial t} + v.\nabla_x f = \frac{1}{\tau}(M(f) - f) \quad , \tag{1}$$

où τ est un temps de relaxation, $M(f)$ est une Maxwellienne locale et peut s'écrire :

$$M(f) = \frac{\rho}{(2\pi T)^{\frac{N}{2}}} \exp(-\frac{(v-U)^2}{2T}) \quad . \tag{2}$$

La densité ρ, la vitesse U, la température T sont obtenues à l'aide des moments de la fonction de distribution f :

$$\begin{pmatrix} \rho(t,x) \\ \rho U(t,x) \\ E(t,x) \end{pmatrix} = \int_{I\!R^N} \begin{pmatrix} 1 \\ v \\ \frac{|\, v \,|^2}{2} \end{pmatrix} f(t,x,v)\, dv \quad . \tag{3}$$

La température est simplement déduite de l'énergie totale des molécules par la formule :

$$E = \rho \frac{|\, U \,|^2}{2} + \frac{N}{2} \rho T. \tag{4}$$

1.2 Approximation numérique

Pour résoudre numériquement cette équation, nous avons utilisé une méthode particulaire déterministe introduite par S.Mas-Gallic [4]. Nous rappelons ici brièvement la méthode, le lecteur pourra se référer à [4] et [5] pour une description plus précise.
Etant donné une distribution initiale f^0 sur un ensemble de points $(x_i^0, v_i^0)_{i \in I \subset Z}$ de l'espace des phases, on cherche $f_i(t), x_i(t)$ et $v_i(t)$ de telle façon que la mesure :

$$f_h = \sum_{i \in I} w_i f_i(t) \delta(x - x_i(t)) \otimes \delta(v - v_i(t)) \tag{5}$$

soit une approximation de la solution f au sens des mesures; w_i représente le poids positif de la particule d'indice i. A partir de (5), on introduit, pour une fonction g continue en (x, v), la formule de quadrature suivante :

$$\int\int g(t,x,v)\, d\,x\, d\,v \;\approx\; \sum_{i \in I} w_i g(t, x_i(t), v_i(t)) \tag{6}$$

où les particules servent de points de quadrature.
La méthode se décompose principalement en deux phases :

- Une phase de convection prenant en compte le déplacement des particules suivant les caractéristiques.
- Une phase de collision où l'on évalue l'opérateur de collision $M(f) - f$.

Phase de Convection

Pour chaque particule indicée i, $x_i(t)$ et $v_i(t)$ sont solutions du système différentiel suivant :

$$\begin{cases} \dfrac{d\,x_i}{d\,t}(t) = v_i \quad , \; x_i(0) = x_i^0 \quad , \\ \dfrac{d\,v_i}{d\,t}(t) = 0 \quad\;\; , \; v_i(0) = v_i^0 \quad . \end{cases} \tag{7}$$

Phase de Collision

Pour calculer l'opérateur de collision $M(f)-f$, il faut d'abord calculer les quantités macroscopiques $\rho, \rho U$ et E qui déterminent la Maxwellienne (2). Ces quantités sont obtenues en intégrant la fonction de distribution f sur l'espace des vitesses (3). Or (6) définit une formule de quadrature dans l'espace des phases. Il est donc nécessaire de transformer, à l'aide d'une fonction cut-off, les intégrales dans (3) en intégrales sur l'espace des phases tout entier, intégrales que nous évaluerons à l'aide de la formule de quadrature (6). Cette fonction cut-off que nous noterons ζ doit vérifier les propriétés suivantes :

$$\left\{ \begin{array}{l} \displaystyle\int_{I\!R^N} \zeta(x)\, d\,x = 1 \quad , \\ \displaystyle\int_{I\!R^N} \mid x^l \mid \zeta(x)\, d\,x = 0 \qquad 1 \leq l \leq k-1 \;\; ; \;\; k \geq 2 \quad , \\ \displaystyle\int_{I\!R^N} \mid x^k \mid \mid \zeta(x) \mid d\,x < +\infty \quad . \end{array} \right.$$

On pose : $\zeta_\epsilon(x) = \dfrac{1}{\epsilon^N}\zeta(\dfrac{x}{\epsilon})$ pour $\epsilon > 0$. Les valeurs approchées des quantités macroscopiques, à l'instant t, sont définies par :

$$\left(\begin{array}{c} \rho_h(t,x) \\ (\rho U)_h(t,x) \\ E_h(t,x) \end{array} \right) = \sum_{j\in I} w_j \left(\begin{array}{c} 1 \\ v_j \\ \mid v_j \mid^2 \end{array} \right) f_j\, \zeta_\epsilon\,(x - x_j(t)) \tag{8}$$

En utilisant (4) et (8), la valeur approchée de la Maxwellienne (2), à l'instant t, est donnée par la formule: $M_h(t,x,v) = M(\rho_h, \rho U_h, E_h)(t,x,v)$ et on pose: $M_i(t) = M_h(t, x_i(t), v_i(t))$. Par la suite nous supposerons que la fonction ζ_ϵ est à support compact dans $[-\epsilon, \epsilon]$. Par conséquent les particules intervenant dans le calcul des quantités macroscopiques sont celles situées dans la boule de rayon ϵ centrée en x.

Intégration en temps

Le schéma en temps que nous proposons répond à deux critères essentiels. Il reste stable pour des temps de relaxation petits (régime collisionnel) et grands (transport libre) et surtout il est explicite, ce qui conduit à un algorithme plus facilement parallélisable. Ce schéma utilise l'expression semi-implicite de la solution du modèle B.G.K. à l'instant $t + \Delta t$ où Δt est le pas d'intégration en temps :

$$\begin{array}{rl} f(t+\Delta t, x, v) & = f(t, x - \Delta t v, v)\exp(-\dfrac{\Delta t}{\tau}) \\ & \quad + \dfrac{1}{\tau}\displaystyle\int_0^{\Delta t} M(t+s, x-(\Delta t - s)v, v)\exp(-\dfrac{\Delta t - s}{\tau})ds \quad . \end{array} \tag{9}$$

Par la suite nous noterons : $M(t+s) = M(t+s, x-(\Delta t - s)v, v)$.
On écrit maintenant (9) pour $t = t_n = n\Delta t$:

$$f^{n+1} = \exp(-\frac{\Delta t}{\tau})f^n + \frac{1}{\tau}\int_0^{\Delta t} M(t+s)\exp(-\frac{\Delta t - s}{\tau})ds \quad .$$

Pour calculer l'intégrale précédente on va supposer que la Maxwellienne M reste constante entre t_n et t_{n+1}. Par conséquent le schéma s'écrit :

$$\begin{aligned} f_i^0 &= f_0(x_i^0, v_i^0) \\ f_i^{n+1} &= \exp(-\frac{\Delta t}{\tau}) f_i^n + (1 - \exp(-\frac{\Delta t}{\tau})) M_i^n \end{aligned} \tag{10}$$

Le lecteur intéressé pourra trouver une analyse plus détaillée de ce schéma dans [3].

2 Parallélisme SIMD et MIMD

Dans cette partie nous présentons deux machines utilisées au cours de cette étude, représentant les deux grands modèles qui existent aujourd'hui dans le parallélisme massif.

2.1 Parallélisme SIMD

Nous rappelons tout d'abord que le type **SIMD** (**S**ingle **I**nstruction **M**ultiple **D**ata) met en jeu des processeurs qui, à un instant donné, exécutent la même instruction sur des données différentes. La Connection Machine **CM200** est un calculateur à architecture massivement parallèle de type SIMD à mémoire distribuée comprenant de **4 K** à **64 K** processeurs (1 K=1024 processeurs) disposant chacun d'une mémoire locale. Ces processeurs sont physiquement regroupés par **noeuds** (1 noeud est constitué de 32 processeurs, d'une interface de calcul et d'une unité de calcul flottant 32 ou 64 bits). Ces noeuds sont reliés entre eux par un réseau de communication de type hypercube. Il existe essentiellement deux types de communication :

- NEWS (North-East-West-South) : mécanisme permettant de faire communiquer des processeurs voisins dans une grille définie par l'utilisateur.
- Le Router : mécanisme général permettant de faire communiquer des processeurs quelconques. Ce modéle de communication est néanmoins plus coûteux que le précédent.

Le système de la Connection Machine permet de simuler un nombre de processeurs supérieur au nombre de processeurs physiques, par une division de la mémoire locale de chaque processeur. Le rapport du nombre de processeurs à simuler sur le nombre de processeurs physiques est appelé **VPR** (**V**irtual **P**rocessor **R**atio). Parmi les langages de haut niveau existant sur la Connection Machine, nous avons utilisé durant ce travail le CM Fortran, qui est basé sur le Fortran 77 standard mais qui reprend certaines extensions vectorielles du Fortran 90. Le lecteur qui le souhaite pourra trouver des renseignements plus précis dans [8].

Description de l'algorithme

Il s'agit tout d'abord de choisir la manière d'organiser la répartition des données entre les processeurs. Le choix optimal est celui qui minimise les communications

entre processeurs, n'entraîne pas d'opérations redondantes et ne nécessite pas de stockage supplémentaire en némoire. Une telle répartition n'existe pas toujours, on doit alors trouver un compromis entre ces trois contraintes.
Pour le modéle B.G.K., il y a essentiellement deux manières de répartir les données:

1. Le choix naturel semble être d'associer une particule à un processeur. La partie convection de l'algorithme est ainsi parfaitement parallélisée, et aucune communication n'est nécessaire pour déterminer les nouvelles positions des particules. Cependant une telle répartition est moins bien adaptée à la phase de collision car cette phase fait intervenir des particules associées à des processeurs différents et donc nécessitera des communications. Comme de plus cette phase de collision est très coûteuse du point de vue des opérations un tel choix de répartition des données conduirait à un algorithme très peu efficace sur la Connection Machine. Par conséquent nous avons préféré faire un autre choix concernant la répartition des données.

2. Comme nous venons de le voir il est nécessaire que la structure de données choisie se prête bien à la phase de collision. Il faut donc faire en sorte que durant la phase de collision aucune communication inter-processeurs n'intervienne. Une manière possible de respecter cette contrainte consiste à associer à chaque processeur les informations nécessaires au calcul de l'opérateur de collision. Cependant un tel choix va entraîner un stockage redondant en mémoire et pourrait donc conduire à une limitation concernant le nombre de particules.

Nous rappelons que nous avons choisi une fonction cut-off à support compact dans $[-\epsilon, \epsilon]$, on introduit ainsi une discrétisation fictive de l'espace physique en cellules de même taille (ϵ). Chaque cellule ainsi définie sera associée à un processeur virtuel. Par conséquent, pour calculer l'opérateur de collision sans qu'aucune communication n'intervienne, il est nécessaire de stocker dans la mémoire locale de chaque processeur virtuel les données des cellules voisines (2 en dimension 1, 8 en dimension 2), voir Figure 1.

Ainsi il ne sera nécessaire de communiquer qu'après la phase de convection des

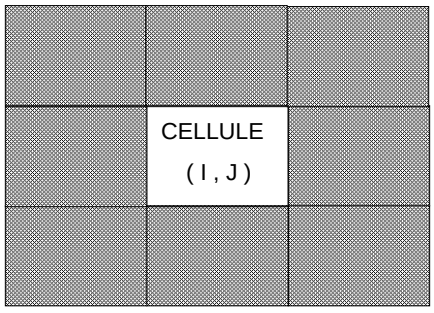

Figure 1: Cellule (I,J) et ses voisines

particules. Après chaque pas de temps, la nouvelle position de la particule i est donnée par :

$$x_i^{n+1} = x_i^n + \Delta t v_i^0$$

Sans contrainte sur Δt, une particule peut quitter sa cellule initiale et atteindre une autre cellule non nécessairement voisine. Dans ce cas, le mécanisme de communication le plus général (Router) sera utilisé et le coût des communications peut être alors très important.
Pour minimiser le coût des communications, on choisit Δt de telle façon qu'une particule ne puisse atteindre en un pas de temps qu'une des cellules avoisinantes (Figure 1). Ainsi les communications n'ont lieu qu'entre processeurs voisins et on peut donc utiliser NEWS pour communiquer et diminuer ainsi de façon significative le coût des communications.

L'algorithme

L'algorithme peut se résumer ainsi :
A chaque pas de temps Δt :

1. *Phase de convection*
2. *Prise en compte des conditions aux limites*
3. *Phase de communication*
4. *Calcul de l'opérateur de collision*
5. *Mise à jour des solutions*

2.2 Parallélisme MIMD

Contrairement au type SIMD, le type **MIMD** (**M**ultiple **I**nstruction **M**ultiple **D**ata) met en jeu plusieurs processeurs indépendants pouvant exécuter des tâches différentes sur des données différentes. Nous décrivons brièvement les caractéristiques de la machine utilisée.
La Meiko "Concerto" est une machine parallèle à mémoire distribuée de type MIMD. La configuration utilisée comprend 16 processeurs Intel 860 (pour les calculs flottants). Chacun de ces processeurs est relié à 2 transputers T800 (ayant chacun 4 liens physiques) pour les communications. Il est à noter que pour des raisons techniques, seuls 14 processeurs i860 étaient disponibles durant nos simulations. Nous avons utilisé, au cours de cette étude, le compilateur Fortran Greenhills (g860apx).

Décomposition du domaine de calcul

La parallélisation du code s'effectue en décomposant le domaine de calcul en plusieurs sous-domaines et en associant chaque sous-domaine à un processeur. L'algorithme utilisé ici ressemble beaucoup au précédent.
En effet après la phase de convection il y a une phase de communication où les processeurs s'échangent les informations concernant les particules ayant quitté le sous-domaine où elles se trouvaient. Notons au passage que comme précédemment on s'impose une contrainte sur le pas de temps Δt de telle façon qu'une particule

ne puisse atteindre en un pas de temps qu'un sous-domaine voisin.
Concernant la phase de collision, nous rappelons que le choix d'une fonction cut-off à support compact dans une boule de rayon ϵ, entraîne que seules les particules situées dans la boule de centre x et de rayon ϵ interviennent dans le calcul de la Maxwellienne au point x. Par conséquent pour qu'aucune communication ne soit nécessaire durant la phase de collision, il faut associer à chaque processeur un sous-domaine défini précédemment ainsi qu'un voisinage de longueur ϵ autour de ce sous domaine (Figure 2). On parlera alors de recouvrement des sous-domaines.

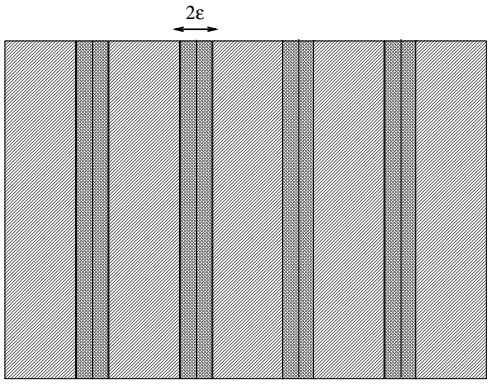

Figure 2: Décomposition du domaine de calcul

Utilisation de PARMACS

Pour assurer une portabilité maximale entre différentes machines parallèles de type MIMD et pour adapter facilement un code séquentiel sur une de ces machines nous avons utilisé le logiciel PARMACS qui contient des librairies de communication. L'utilisation de ces librairies permet de gérer plus facilement les échanges de message, d'une part et décharge l'utilisateur de l'assignation des tâches aux processeurs d'autre part. L'utilisation de ce logiciel rend ainsi le programme moins dépendant de la machine utilisée.

3 Résultats de performances

Nous donnons ici des résultats de performance obtenus en considérant le modèle B.G.K. (1) en dimension N=2. Plus précisément, nous considérons l'évolution de particules dans un domaine rectangulaire, décomposé en sous-domaines de même taille ce qui assure une répartition quasi équitable des tâches entre processeurs. Pour une description plus précise du cas test ainsi qu'une présentation des solutions physiques obtenues, nous renvoyons le lecteur à [2]. Nous comparons les résultats obtenus sur les trois machines suivantes: la CM200 (avec 8 K ou 16 K processeurs), la Meiko-CS1 avec 14 processeurs i860 et le Cray-YMP mono-processeur. Nous donnons des résultats en simple et en double précision (respectivement notée S.P et D.P) pour les deux premières machines. Notons que nous avons utilisé les librairies

de vectorisation VAST, pour optimiser les performances sur la Meiko-CS1. Nous présentons aussi les résultats obtenus avec et sans vectorisation sur les figures 3 et 4. Le gain obtenu en vectorisant est de l'ordre de 40% en moyenne.
Nous donnons dans les tables ci-dessous, le temps CPU par itération, les Mflops en fonction du nombre de particules. Nous avons fait varier ce nombre tout en laissant constant le nombre de particules par cellule.

Nous tenons à préciser que le seul critère objectif de comparaison entre les dif-

Nb parti.	Machine	Nb procs	Mflops	Temps CPU s/iter
101376	Meiko CS1	14	234	9.1
101376	CM200	8 K	748	7.8
202752	Meiko CS1	14	230.2	18.7
202752	CM200	8 K	879	13.2
405504	Meiko CS1	14	220.4	39.2
405504	CM200	8 K	966	24
811008	Meiko CS1	14	219	78.9
811008	CM200	8 K	1017	45.6
811008	CM200	16 K	1950	23.8
1216512	Meiko CS1	14	217	119.5
1216512	CM200	8 K	1036	67.2

Table 1: Résultats de performance en simple précision

férentes machines nous parait être le temps CPU, car les algorithmes sont légèrement différents (notamment entre la Meiko CS1 et la CM200), bien que le nombre d'itérations pour atteindre la solution physique soit identique pour toutes les machines. Il apparaît qu'en double précision, la CM200 8 K est environ $1,5$ fois plus rapide que la Meiko-CS1 et $1,8$ fois plus rapide que la Cray-YMP. En simple précision ces rapports deviennent respectivement $1,7$ et $2,5$. Nous remarquons également que lorsqu'on multiplie le nombre de particules et le nombre de processeurs par un même rapport, le temps CPU par itération reste à peu près constant, aussi bien pour la CM200 que pour la Meiko-CS1. D'autre part, le coût des communications est très faible: environ 5 % du temps CPU sur la Connection Machine et de l'ordre de 1 %

Nb parti.	Machine	Nb procs	Mflops	Temps CPU s/iter
101376	Meiko CS1	7	87.8	24.6
101376	Meiko CS1	14	174.2	12.4
101376	Cray YMP	1	150.8	14.2
101376	CM200	8 K	400	10.9
202752	Meiko CS1	7	86.9	49.7
202752	Meiko CS1	14	174.2	24.8
202752	Cray YMP	1	150.2	28.7
202752	CM200	8 K	540	18.9
405504	Meiko CS1	7	86.7	99.6
405504	Meiko CS1	14	173.8	49.7
405504	Cray YMP	1	150	57.6
405504	CM200	8 K	646	35.9
811008	Meiko CS1	14	172.9	99.9
811008	Cray YMP	1	149.8	117
811008	CM200	8 K	707	65.6

Table 2: Résultats de performance en double précision

sur la Meiko-CS1. Ces deux remarques montrent le très haut niveau de parallélisme de la méthode.

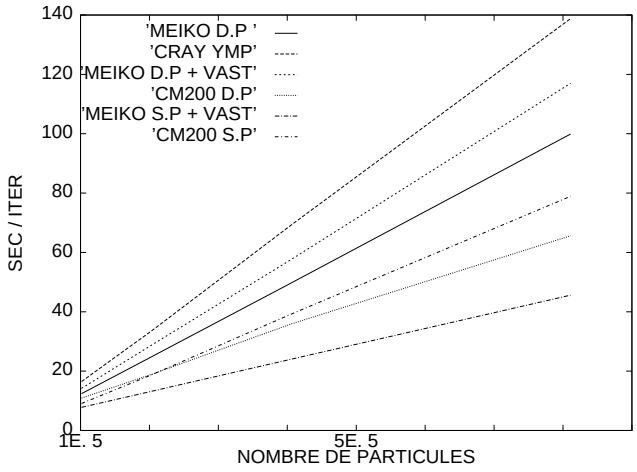

Figure 3: Temps CPU par itération

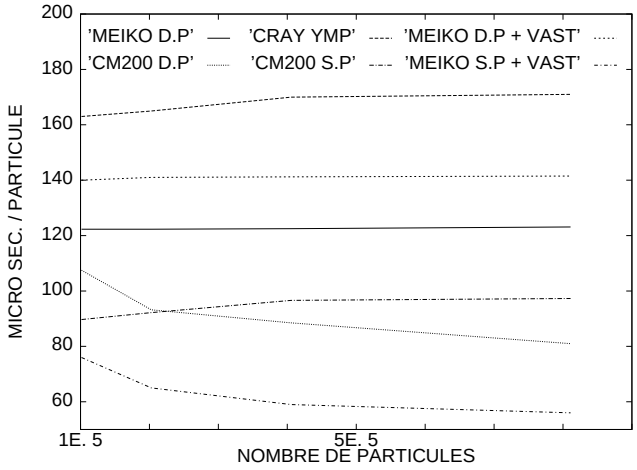

Figure 4: Temps CPU par particule

Conclusion

Les résultats de performance montrent, aussi bien sur la Connection Machine que sur la Meiko-CS1, le haut niveau de parallélisme de la méthode. On notera que les performances obtenues sur la Connection Machine sont supérieures à celles obtenues sur la Meiko-CS1 et ce en simple et double précision. D'autre part, les performances enregistrées sur ces deux machines sont supérieures à celles obtenues sur un ordinateur vectoriel : le Cray-YMP mono-processeur. Nous tenons à préciser que l'implémentation de la méthode a été plus rapide sur la Meiko-CS1 et que d'autre part avec l'utilisation du logiciel PARMACS, il est possible de porter le code sur d'autres machines parallèles de type MIMD, comme nCUBE ou l'iPSC860,

par exemple. De tels résultats sont encourageants et montrent tout l'intérêt du parallélisme massif. Ces premiers résultats incitent à l'extension de ces méthodes, d'une part, à des géométries plus complexes et, d'autre part, à des écoulements tri-dimensionnels.

Remerciements

L'auteur tient à remercier Loula Fezoui du CERMICS Sophia-Antipolis et Mark Loriot de SIMULOG pour leurs précieux conseils.

References

[1] P.L. BHATNAGAR, E.P. GROSS and M. KROOK, *A model for Collision Processes in Gases*, Phys.Rev.,**94**, (1954) p.511.

[2] D. ISSAUTIER, F. POUPAUD, L. FEZOUI, J.P. CIONI, *A parallel weighted particle algorithm for solving the Boltzmann (B.G.K.) equation on the Connection Machine*, Rapport CERMICS 93-14, (1993).

[3] D. ISSAUTIER, *Convergence of a weighted particle method for solving the Boltzmann (B.G.K.) equation*, à paraître.

[4] S. MAS–GALLIC, *A deterministic particle method for the linearized Boltzmann Equation*, Trans.Theory Stat.Phys., **16**, (1987) pp.885-887.

[5] S. MAS–GALLIC and F. POUPAUD, *Approximation of the Transport Equation by a weighted particle method*, Trans.Theory Stat.Phys., **17**, (1988) pp.311-345.

[6] L.N. LONG, M. KAMON and J.MYCZKOWSKI, *A Massively Parallel Algorithm to solve the Boltzmann (B.G.K.) Equation*, AIAA Paper, 92–0563.

[7] F. CORON,P. HOMSI, *Utilisation du Parallélisme Massif en calcul d'écoulements raréfiés*, Contrat DRET, 90/587.

[8] *Technical summary*, Thinking Machines Corporation, Version 6.0 (1990).