PolyPublie
Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Titre: Title:	Parallélisation de la méthode des éléments discrets appliquée aux écoulements granulaires
Auteur: Author:	Louis-Alexandre Leclaire
Date:	2004
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Leclaire, L.-A. (2004). Parallélisation de la méthode des éléments discrets appliquée aux écoulements granulaires [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7192/

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/7192/
Directeurs de recherche: Advisors:	François Bertrand
Programme: Program:	Non spécifié

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal

https://publications.polymtl.ca

UNIVERSITÉ DE MONTRÉAL

PARALLÉLISATION DE LA MÉTHODE DES ÉLÉMENTS DISCRETS APPLIQUÉE AUX ÉCOULEMENTS GRANULAIRES

LOUIS-ALEXANDRE LECLAIRE

DÉPARTEMENT DE GÉNIE INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE INFORMATIQUE)
AVRIL 2004

© Louis-Alexandre Leclaire, 2004.

National Library of Canada
Bibliothèque nationale du Canada

Acquisitions and Bibliographic Services
Acquisisitons et services bibliographiques

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 0-612-91954-4
Our file Notre référence
ISBN: 0-612-91954-4

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou aturement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this dissertation.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de ce manuscrit.

While these forms may be included in the document page count, their removal does not represent any loss of content from the dissertation.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

PARALLÉLISATION DE LA MÉTHODE DES ÉLÉMENTS DISCRETS APPLIQUÉE AUX ÉCOULEMENTS GRANULAIRES

présenté par: LECLAIRE Louis-Alexandre

en vue de l'obtention du diplôme de: Maîtrise ès sciences appliquées

a été dûment acceptée par le jury d'examen constitué de:

M. ROY Robert, Ph.D., président

M. BERTRAND François, Ph.D., membre et directeur de recherche

M. GUIBAULT François, Ph.D., membre

Remerciements

La réalisation de cet ouvrage n'aurait pas été possible sans les conseils de mon directeur, François Bertrand qui m'a donné l'opportunité de travailler sur ce projet de recherche. Je remercie aussi M. Robert Roy et François Guibault pour leur participation au jury de ce mémoire.

Je tiens à remercier ma famille et mes amis pour leur encouragement et leur support, et bien sûr Elise pour m'avoir enduré.

Enfin, je tiens à saluer ceux qui ont travaillé à mes côtés sur le projet ainsi que tous mes collègues de l'URPEI.

RÉSUMÉ

La méthode des éléments discrets est une méthode numérique déterministe permettant l'analyse des écoulements granulaires. Les milieux granulaires sont complexes et composés de milliers de corps en interaction. Calculer les forces qui découlent de leurs collisions est excessivement long. Puisque ces procédés sont transitoires, plusieurs secondes de simulation sont requises. Les utilisateurs doivent alors s'armer de patience pendant que les stations de travail sont rudement mises à l'épreuve. L'accélération des temps de calcul passe premièrement par l'optimisation des algorithmes de détection de contacts adaptés aux besoins spécifiques de cette méthode. À ce sujet, nous proposons un nouveau mode de représentation des surfaces solides, combinant les avantages des outils de CAD et le moindre coût de détection des objets sphériques.

Heureusement, avec l'émergence des grappes de calcul de type Beowulf, la méthode des éléments discrets peut profiter de l'utilisation concourante de plusieurs unités de calcul. La parallélisation de la méthode des éléments discrets par décomposition de domaine est d'ailleurs le principal sujet de ce mémoire. En effet, bien implantée, elle permet en théorie de réduire linéairement les temps de résolution d'une simulation. L'étude de nombreux sujets intimement reliés à cette parallélisation a été réalisée. Nous avons constaté que l'équilibrage dynamique de tâches est crucial pour assurer l'efficacité et l'utilisation intelligente des ressources disponibles. Nous avons démontré qu'il était possible d'équilibrer les charges selon deux méthodes agissant à deux niveaux : en modifiant la quantité de travail par processus ou en tentant de rééquilibrer les processus occupés sur la grappe de calcul. Nous avons aussi abordé la création d'objets non-sphériques à partir de particules élémentaires. Nous concluons que cette représentation augmente la flexibilité sans les désavantages de la détection d'objets polyédriques, De plus, elle ne constitue aucun obstacle à l'implantation parallèle dont

l'efficacité par décomposition de domaine est durement touchée lorsque le rapport des tailles entre les objets augmente. Enfin, puisque les ordinateurs parallèles disponibles pour ce mémoire étaient composés de nœuds multiprocesseurs, une implantation en mémoire partagée a été réalisée. Suite à l'analyse de nos résultats, il est difficile de conseiller l'utilisation de ce paradigme ainsi que son dérivé, le mode hybride. Peu importe l'architecture, la parallélisation de la méthode des éléments discrets par décomposition spatiale du domaine est requise afin de diminuer les temps de calculs de cette méthode numérique très dispendieuse et permettre la simulation d'écoulements granulaires comportant un nombre réaliste de particules.

ABSTRACT

The Discrete Element Method (DEM) is a recent method to simulate the behaviour of granular materials. By tracking the interactions and the collisions between thousands of particles, this method can simulate complex phenomena that are unique to granular materials and that are still poorly understood. Such simulations are however very CPU-intensive and, consequently, high-performance computers and techniques are needed for DEM to be efficient. In particular, the use of an optimal algorithm for contact detection is required but, to get even faster results, the parallelization of DEM becomes essential.

With the use of a Beowulf cluster built from commodity processors, a parallelized DEM code has the potential to solve large problems. This thesis concerns the optimization and the parallelization of DEM using in particular domain decomposition techniques. Upon trying to reduce the communication cost between processes, we quickly realized that the work load imbalance was an important source of time waste. To achieve better speed-up, we investigated dynamic load-balancing by means of two different techniques. The first one is based on the displacement of the sub-domain boundaries to balance the work loads. The second method relies upon the dynamic distribution of a quantity of processes larger than the number of processors in the cluster.

Another topic that was investigated in this work concerns the simulation of the flow of non-spherical particles. To this end, we developed a method that models a non-spherical particle as a collection of spherical particles moving as a rigid body. One advantage of the proposed scheme is that it can use the standard contact detection schemes for spherical particles. Most importantly, the proposed method is compatible

with domain decomposition parallelization because a macro-particle can be split on many sub-domains.

Finally, we investigated the parallelization of DEM on SMP compute nodes using OpenMP-MPI hybrid parallelization and observed that domain decomposition based parallelization on a distributed memory system gives better results in the case of the flow of non-cohesive spherical particles.

Table des matières

Liste des tableaux

Liste des figures

Liste des sigles et abréviations

API	Application Programming Interface
CISC	Complex Instruction Set Computer
CPU	Central Processing Unit
DEM	Discrete Element Method
MPMD	Multiple Program Multiple Data
MD	Molecular Dynamic
MPP	Massively Parallel Processing
MPI	Message Passing Interface
NUMA	Non Uniform Memory Access
PVM	Parallel Virtual Machine
RAM	Random Access Memory
RISC	Reduce Instruction Set Computer
SCSI	Small Computer System Interface
SMP	Symmetric Multi Processing
SPMD	Single Program Multiple Data
URPEI	Unité de Recherche en Procédés d'Écoulements Industriels

1 Introduction

Automne 2003. Alors que les processeurs 64-bit sont proposés pour la première fois au grand public, les défis technologiques à relever pour faire progresser la fréquence sont de taille. Les constructeurs redoublent d'ingéniosité pour contourner le problème de la chaleur, qui atteindra bientôt le niveau de nos ampoules incandescentes. Il est à se demander si Moore finira par avoir tort. Pendant que la science-fiction rêve à l'ordinateur quantique, les besoins en calcul augmentent plus rapidement que les prouesses technologiques. Des modèles plus complexes nécessitent toujours plus de ressources informatiques et l'ingénieur d'aujourd'hui peut difficilement se contenter d'une seule unité de calcul.

Idée en vogue depuis une dizaine d'années, le parallélisme vient à la rescousse en combinant la puissance de plusieurs ordinateurs. Les applications pouvant tirer profit de cette nouvelle tendance sont nombreuses mais l'opération n'est pas toujours évidente. On retrouve déjà ces superordinateurs dans les centres météorologiques et astrophysiques, mais, de plus en plus, l'utilisation de grappes de calcul fait son apparition dans de nouveaux secteurs, dont les laboratoires pharmaceutiques, entre autres pour le design de nouvelles molécules. Et bientôt, les ordinateurs parallèles deviendront essentiels à l'étude des mélanges de poudres entrant dans la fabrication de médicaments sécuritaires et efficaces.

L'étude de ces procédés de mélange, bien qu'essentielle, est une entreprise très onéreuse. La simulation par ordinateur permet l'optimisation de procédés industriels à moindre coût. Cependant, les modèles numériques conventionnels basés sur la mécanique des milieux continus ne peuvent imiter le comportement unique des poudres :

avalanches, ségrégation, percolation. Ces modèles numériques permettent difficilement la simulation de ce type de phénomène.

La méthode des éléments discrets permet de suivre individuellement chaque particule et de simuler les phénomènes propres aux écoulements granulaires. Ces procédés granulaires impliquent généralement des milliers voire des millions de particules. Donc, des ressources informatiques très importantes sont requises afin de permettre d'effectuer des simulations dans des temps raisonnables. L'utilisation de plusieurs ordinateurs de façon concourante est de toute évidence souhaitable pour ce type d'application.

1.1 Objectif général

Ce mémoire relate ma contribution au sein d'une équipe de recherche tentant de concevoir un nouvel outil d'analyse des milieux granulaires. Étant le seul étudiant en génie informatique du groupe, mon rôle principal à l'Unité de Recherche en Procédés d'Écoulements Industriels (U.R.P.E.I) était de trouver et implanter des algorithmes efficaces dans le but d'accélérer le temps requis pour une simulation. En particulier, l'objectif consistait à paralléliser la méthode des éléments discrets au moyen de techniques de décomposition de domaine. La conception de l'application Powder3D ayant débuté au début de l'année 2002, une étape d'optimisation sur la version séquentielle de cette application fut requise. La validation des changements apportés au code a été effectuée en collaboration avec les autres membres de l'équipe. Ces modifications visaient en premier lieu la rapidité d'exécution et l'ajout de flexibilité du code de façon à faciliter son utilisation en rendant la parallélisation automatique et transparente.

1.2 Contribution

Ce mémoire porte sur la conception d'une version parallèle de la méthode des éléments discrets pour la simulation d'écoulements granulaires. On y retrouve notamment une description détaillée des algorithmes implantés ainsi qu'une discussion des techniques utilisées pour réaliser une parallélisation efficace. Le travail a aussi permis la simulation de cas de figure très complexes qui n'avaient pas encore été traités avec la méthode des éléments discrets.

2 Revue de littérature

L'idée d'utiliser plusieurs ressources en parallèle n'est pas nouvelle, elle a fait son apparition dès l'avènement de l'informatique moderne. Déjà en 1947, Amdahl dictait sa loi décrivant l'accélération potentielle A avec P processeurs :

$$A = \frac{T(1)}{T(P)} = \frac{T(1)}{\frac{(T(1)\times\alpha)}{P} + (1-\alpha)\times T(1)} = \frac{P}{P + (1-P)\times\alpha} \qquad \textbf{Équation 2.1}$$

où $T(1)$ est le temps séquentiel et α la fraction du code qui n'est pas parallélisable. L'accélération maximale limitée à $1/(1-\alpha)$ est atteinte lorsque le nombre de processeurs (P) est infiniment grand. Il est primordial de réduire la valeur de α qui est unique pour chaque application. Les techniques de parallélisation sont donc intimement reliées au problème à résoudre. Avant de discuter de la parallélisation de « Powder3D », nous allons présenter la méthode des éléments discrets.

2.1 Introduction aux écoulements granulaires

Au milieu des années 70, quelques scientifiques ont commencé à chercher une nouvelle approche pour l'analyse des milieux granulaires. Auparavant, les études étaient seulement expérimentales et les tentatives pour les reproduire avec des méthodes numériques de la mécanique des milieux continus n'arrivaient pas à simuler correctement tous les phénomènes rencontrés. En 1979, Cundall et Strack (1979) ont publié le premier article relatant la méthode des éléments discrets appliquée à l'étude des sols rocheux. Cette méthode, comme son nom l'indique, est basée sur la discrétisation individuelle des particules. Originale à l'époque parce qu'elle était appliquée pour la

première fois à autre chose que des atomes, cette façon de procéder n'était en fait qu'une extension de la dynamique moléculaire développée presque 30 ans plus tôt. Comme d'autres simulations physiques particulaires, telles l'astrophysique, les écoulements granulaires font intervenir des milliers de corps (Warren, 1992). Cependant, la méthode des éléments discrets (Discrete Element Method., DEM [i]) se démarque au niveau du champ d'application. Aujourd'hui, la DEM trouve preneur dans de nombreux domaines industriels impliquant de petites particules ou grains (< 1cm de diamètre). L'ordre de grandeur des corps en interaction est encore beaucoup supérieur aux molécules mais, on retrouve des applications de la DEM avec des poudres de plus en plus fines (Muzzio, 2002). De plus, la DEM se distingue dans la représentation des objets; alors que la source ponctuelle (point) et la sphère sont largement utilisées en astrophysique et en dynamique moléculaire respectivement, d'autres formes polygonales peuvent être générées pour représenter la multitude de formes de grains que l'on retrouve dans la nature. Cependant, le cercle et la sphère sont encore les plus étudiés. La méthode des éléments discrets demeure applicable lorsque les forces de collision sont dominantes, soit dans les environnements denses. *A priori*, les modèles simples ne supportent pas les forces à rayon d'action entre les particules. Toutefois, comme en dynamique moléculaire, des forces supplémentaires de cohésion (les forces de van der Waals, par exemple) peuvent être ajoutées lorsque les grains se rapprochent suffisamment. Il devient ainsi possible grâce à ce type de modèle d'étudier les mécanismes d'agglomération de particules (Yang et Yu, 2000).

D'autres méthodes numériques, de type probabiliste, peuvent être utilisées pour simuler les écoulements granulaires. Un algorithme basé sur la méthode Monte-Carlo a permis par exemple à Vidal et al. (2001) d'étudier la sédimentation de mélanges polydispersés dans le domaine du couchage du papier. Ces méthodes probabilistes ont l'avantage d'être extrêmement rapides, souvent au détriment de la qualité des résultats

[i] DEM sera utilisé pour désigner la méthode des éléments discrets.

obtenus. Leur application est donc limitée et l'utilisation de la DEM pour simuler des écoulements granulaires complexes reste inévitable.

2.2 Description de la méthode

La méthode des éléments discrets fait généralement intervenir des milliers de corps indépendants, tous compris à l'intérieur d'un domaine de simulation défini par des murs solides ou par des conditions périodiques. Grâce à un bilan de force individuel, l'application de la deuxième loi de Newton conduit à une méthode itérative qui permet de suivre la trace de chaque particule à tout moment :

$$m_i \frac{d^2 x_i}{dt^2} = F_{total,i} \qquad \textbf{Équation 2.2}$$

avec laquelle on trouve l'accélération de la particule i en fonction de sa masse et des forces qui s'exercent sur elle. Ces forces (F_{total}) comprennent entre autres les interactions de la particule *i* avec son entourage *N(i)* :

$$F_{total,i} = \sum_{\forall j \in N(i)} F_{ij} \qquad \textbf{Équation 2.3}$$

La boucle de résolution comporte trois étapes : La détection des contacts, le calcul des forces et le calcul du mouvement des corps.

2.2.1 Détection de contacts

Avant de pouvoir quantifier les forces résultant des collisions, il faut trouver les paires de particules en contact. Cette recherche est très dispendieuse si le nombre de particules est élevé. Des algorithmes de localisation ont été développés dans le but de réduire la recherche des $O(n^2)$ paires possibles aux paires probables seulement. Grilles,

tris et graphes, tous les moyens sont bons pour réduire l'ordre de complexité de la recherche à l'ordre linéaire *O(n)*. Le but de ces algorithmes est de déterminer s'il y a collision entre chaque paire de particules et, le cas échéant, trouver le point de contact. Du point de vue de la réduction des temps de calcul, cette étape est d'une importance comparable à celle de la parallélisation de la DEM. Nous discuterons des différents algorithmes au chapitre 3.

2.2.2 Force et collisions

La méthode des éléments discrets comprend deux écoles de pensée distinctes qui se démarquent au niveau du modèle de collisions. Le premier groupe affirme que les particules sont indéformables et ne peuvent se chevaucher mutuellement (hard sphere model) (Muller, 1996). Depuis son élaboration, cette technique n'a pas vraiment été beaucoup utilisée car le fait que les contacts et les forces sont instantanés est beaucoup trop contraignant. En pratique, ce modèle devient très difficile à utiliser pour des simulations comprenant plusieurs milliers de corps et des mouvements lents comportant des collisions très rapprochées dans le temps. De plus, pour simuler correctement les écoulements granulaires avec ce type de méthode, les calculs de détection doivent être classés en ordre chronologique de contacts, ce qui rend la parallélisation encore plus ardue.

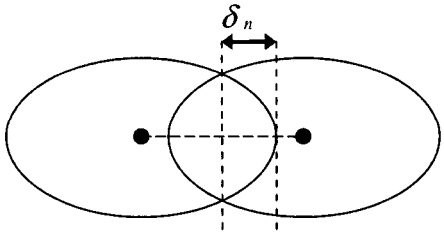


Figure 2-1 Représentation du contact entre deux particules pour le modèle des sphères déformables

Pour éviter cet inconvénient, il fallait éliminer la contrainte interdisant le chevauchement et remplacer le modèle de particules indéformables. C'est pourquoi la seconde école instaura un modèle de chevauchements munis d'une force de répulsion simulée par des ressorts. Avec ce type de modèle de sphères molles (*soft spheres*), deux particules peuvent se chevaucher pendant une certaine période, sans dépasser une limite raisonnable (maximum 1-10% du diamètre). Les forces proviennent des ressorts qui sont comprimés en fonction du chevauchement $\vec{\delta}_n$ (Figure 2.1). Plusieurs modèles de sphères déformables ont été proposés :

- Le modèle de Hertz : $\vec{F} = k\vec{\delta}^{3/2}$

où k est la constante de rappel du ressort.

Dans ce modèle les collisions sont élastiques et il n'y pas d'énergie perdue.

- Le modèle de Walton et Braun(1986): $\vec{F} = k_1 \vec{\delta}$ (pour le chargement)

$\vec{F} = k_2(\vec{\delta} - \vec{\delta}_0)$ (déchargement)

Ce modèle est linéaire et propose deux coefficients différents afin d'imiter le comportement plastique des particules.

- Le modèle de Cundall et Strack (1979) : $\vec{F} = k\vec{\delta} + c\vec{v}$

où k : la constante de rappel du ressort

c : le coefficient d'amortissement

$\vec{v}$: la vitesse relative des particules en collision

Ce modèle couramment implanté, aussi connu sous le nom de « spring and dashpot », modélise un système de ressort et de piston. On y retrouve en plus de la composante d'élasticité une composante d'amortissement qui est fonction du la vitesse relative pour tenir compte de l'énergie dissipée lors de la déformation.

La déformation n'est donc pas prise en compte exactement et les forces dissipatrices sont souvent ajoutées pour simuler des systèmes inélastiques. Remarquons que l'utilisation de ressorts ayant des constantes de rappel élevées et un chevauchement limité est équivalent à un modèle de corps indéformables.

Ces modèles se ressemblent en quelques points. Par exemple, la gestion des collisions multiples se fait habituellement en considérant individuellement chaque paire de particules en interaction. De plus, conséquence de la troisième loi de Newton : la force appliquée sur deux corps est de même intensité mais de direction opposée.

Enfin, la force appliquée sur une particule comprend une composante normale mais aussi une composante tangentielle qui permet de tenir compte du frottement. Le modèle complet tel que celui de Cundall et Strack peut être schématisé comme à la figure 2.2. On retrouve les forces de répulsion et d'amortissement normales (coefficients k_n et c_n) et tangentielles (coefficients k_t et c_t) ainsi que le frottement exprimé par la loi de Coulomb (μ). Précisons qu'un bilan angulaire permet aussi de calculer la vitesse angulaire des particules.

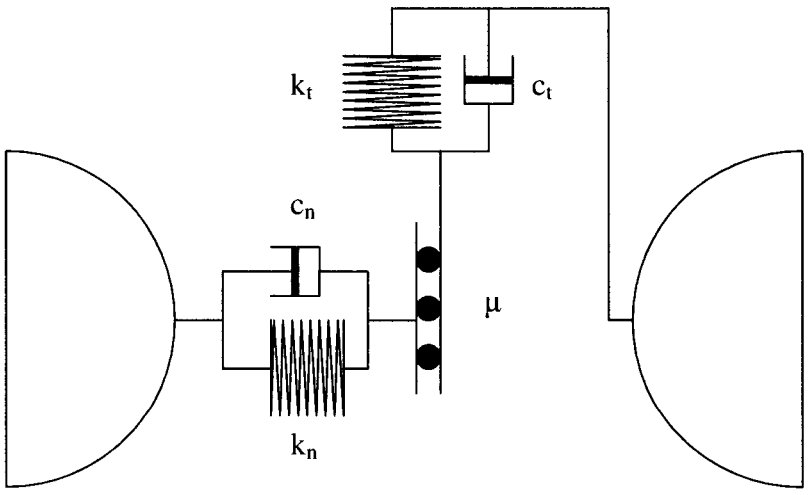


Figure 2-2 Représentation du modèle de Cundall et Strack (1979)

En résumé, lors de l'étape du calcul des forces, chaque particule accumule les énergies de collision dans un vecteur approprié (Équation 2.3). Il est possible d'ajouter à ce bilan de forces des forces externes telles la gravité qui s'applique sur tous les corps selon une direction prédéfinie. La présence d'un fluide impliquera de plus la poussée d'Archimède. Et, du déplacement relatif des particules par rapport à ce fluide, découlera une force de traînée en fonction de la viscosité du liquide.

2.2.3 Calcul du mouvement et de la position

A la fin du processus de sommation des forces de contact, chaque particule a accumulé les forces normales et tangentielles s'exerçant sur elle. La dernière étape consiste à déterminer les nouvelles positions. En appliquant la deuxième loi de Newton (équation 2.2), on déduit les accélérations. Il ne reste plus qu'à intégrer pour trouver les vitesses de déplacement ainsi que les nouvelles positions. Le modèle des corps déformables est valide à condition de n'avoir que de petits chevauchements. Par conséquent, le pas de temps doit être suffisamment petit afin d'éviter des déplacements relatifs exagérés. Pour conserver la stabilité du système, la valeur maximum oscille autour de 10^{-5} secondes, mais il n'est pas rare d'utiliser un pas de temps 100 fois plus petit pour des particules de quelques millimètres. Lorsque le pas de temps est grand, des intégrateurs numériques d'ordre élevé plus coûteux, tels l'algorithme de Runge-Kutta d'ordre 4, sont conseillés. Mais, dans les conditions de la DEM, un simple algorithme saute-mouton (*leap frog , en anglais*) du second ordre permet d'obtenir des résultats convenables. Très populaire, cet algorithme proposé par Verlet (1967) est économique tout en étant reconnu pour bien conserver l'énergie du système. Le calcul des vitesses et position se fait en trois étapes:

- Calcul de la vitesse $V_{i+1/2} = V_{i-1/2} + \frac{a \cdot \Delta t}{2}$ **Équation 2.4**

- Calcul de la vitesse moyenne $V_i = \frac{(V_{i-1/2} + V_{i+1/2})}{2}$ **Équation 2.5**

- Calcul de la position $X_{i+1} = X_i + V_{i+1/2} \cdot \Delta t$ **Équation 2.6**

La définition du pas de temps demeure problématique. Idéalement, il devrait être le plus grand possible et garantir la stabilité de la simulation en cours. Cundall et Strack (1979) ont proposé d'utiliser le critère suivant :

$$\Delta T_{CRITIQUE} = 2\sqrt{m/k} \qquad \textbf{Équation 2.7}$$

m = Masse de la plus petite sphère
k = Rigidité du ressort ~ module de Young

Dans Powder3D, sa valeur dépend de la vitesse maximale des particules du système, V_{max}, de la valeur du chevauchement maximal, δ_{max}, ainsi que du rayon de la plus petite particule R_{min} :

$$\Delta T_{CRITIQUE} = \frac{\delta_{max} \cdot R_{min}}{V_{max}} \qquad \textbf{Équation 2.8}$$

En pratique, Cundall et Strack proposent de trouver le pas de temps critique et de le diviser par un facteur typiquement compris entre 10 et 40. Donc, même si cette constante a beaucoup d'impact sur les temps de résolution et la stabilité de la simulation, on constate que le choix du pas de temps est très arbitraire. En fait, l'expérience de l'utilisateur et quelques tentatives infructueuses sont souvent nécessaires pour déterminer le pas de temps idéal. La prudence l'est aussi; il ne faut pas espérer un résultat trop rapidement car il en va de la qualité de la simulation. Quelques chevauchements excessifs peuvent créer une réaction en chaîne. Il n'est pas rare de voir une sédimentation exploser lorsque la vitesse maximale est plus élevée que prévue et

que le pas de temps n'a pas été ajusté en conséquence. À l'inverse, si le pas de temps est beaucoup trop petit, les temps de simulations seront inutilement trop grands.

En résumé, il suffit de boucler autant de fois que nécessaire sur les trois étapes présentées précédemment. Les particules se déplaceront en réaction aux forces externes et de contact jusqu'au moment où on atteint le temps réel de simulation voulu. L'analyse de l'écoulement granulaire peut avoir lieu en post-traitement en combinant les résultats récupérés à intervalles réguliers.

2.3 Extension du modèle et applications

Comme nous l'avons dit précédemment, les modèles sont nombreux et diffèrent par leur façon de gérer les contacts et le frottement. De plus, certains modèles plus complexes intègrent des forces capillaires ou de cohésion de type van der Waals. Dominantes en présence d'humidité ou en présence de particules très fines, ces forces de cohésion sont la cause de la création d'agglomérats. L'étude sur la formation et la destruction de ces amas particulaires devient cruciale, par exemple pour l'industrie pharmaceutique.

À la limite de la dynamique moléculaire, d'autres s'intéressent à l'étude des fissures d'amas granuleux (Owen, 2001). S'inspirant de cette technique, il est également apparu depuis une dizaine d'années des modèles permettant l'écoulement de structures de formes diverses composées de particules circulaires. Ces techniques permettent de simuler de tels écoulements sans avoir à supporter le coût de la détection de contacts pour des formes quelconques. Par exemple, une étude de la sédimentation de bâtonnets (Figure 2-3) a été réalisée par Gallas et Sokolowski (1993) à l'aide de la DEM.

Deux approches permettent de construire ces formes non-sphériques et gérer leurs liens. La première les représente à l'aide d'un ensemble de sphères qui se chevauchent (Pelessone, 2003). Dans la même optique, Favier (1999) a créé des objets elliptiques à surface lisse en combinant des dizaines de sphères. Son modèle de macro-particules permet entre autres de générer des tores (Figure 2-4). La deuxième approche est basée sur l'utilisation de forces d'attraction et de répulsion et considèrent que les sphères composant une macro-particule sont en contact et sans chevauchement. Plus simple à implanter, elle nécessite moins de particules élémentaires et la détection des contacts a lieu sans avoir à apporter de changements aux algorithmes (Ferrez, 2001).

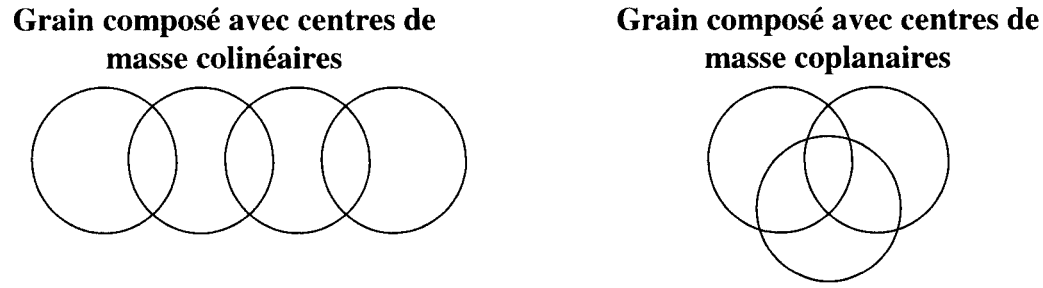


Figure 2-3 Exemple d'objets non sphériques proposés par Gallas et Sokolowski (1993)

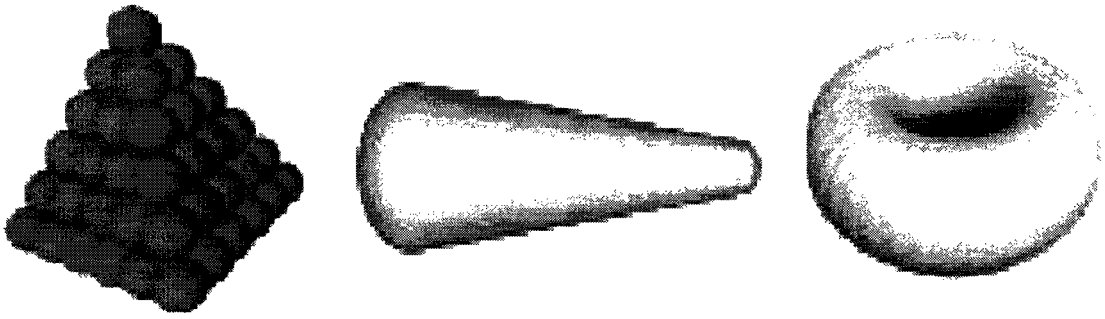


Figure 2-4 Macro particules en 3D sans chevauchement (a) Ferrez (2001) et avec chevauchement (b et c) Favier (1999)

Le choix d'une méthode DEM dépend du type de simulation. De nombreux paramètres doivent aussi être ajustés pour obtenir le comportement souhaité. Une étape de validation et de comparaison avec des résultats expérimentaux est nécessaire avant de pouvoir appliquer le même modèle à de nouveaux problèmes.

2.4 Le calcul parallèle de haute performance

Par sa nature itérative, la méthode des éléments discrets est très dispendieuse. Et comme la plupart des simulations numériques, elle demande les meilleurs systèmes informatiques pour être résolue dans des temps raisonnables. À l'époque où les ordinateurs personnels en étaient encore à leurs premiers balbutiements, seulement les grands centres de recherches pouvaient se procurer des ordinateurs massivement parallèles (MPP). Ces derniers combinaient des centaines de processeurs dignes des meilleures stations de travail de l'époque. Dans la catégorie des superordinateurs, on retrouve aussi l'architecture à mémoire partagée (SMP) comprenant plusieurs processeurs. Dans ce type de système, les temps d'accès pour modifier ou lire une zone mémoire sont les mêmes pour tous les processeurs. Ces architectures étaient toutefois coûteuses et vers 1994, on a assisté à la naissance d'une nouvelle classe de superordinateur: les grappes de type Beowulf. Ces grappes de calcul composées de processeurs indépendants reliées par un réseau tentent d'atteindre le niveau de performances des grands systèmes mais à une fraction du prix. Basés principalement sur le système d'exploitation Linux, ce type de système dispose aujourd'hui d'outils qui en permettent la gestion simplifiée. Grâce à la montée fulgurante en puissance des PC, cette architecture à mémoire distribuée est devenue très compétitive et gagne constamment en popularité et en efficacité.

Aujourd'hui, il existe donc deux grandes familles d'ordinateurs qui sont offertes aux groupes de recherche ayant des applications nécessitant le calcul haute performance.

Pour les moins démunis, il y a les systèmes à mémoire partagée dont le coût peut atteindre quelques millions de dollars. À l'opposé, on retrouve les grappes de calcul Beowulf composées de centaines de nœuds propulsés par des processeurs puissants et bon marché.

Si on exclut les processeurs vectoriels, les processeurs des superordinateurs offrent une puissance de calcul comparable, qu'ils soient de type RISC ou CISC. Cependant chaque architecture possède ses avantages et inconvénients. Ils se démarquent principalement au niveau des temps d'accès aux données. Les systèmes à mémoire distribuée sont très abordables mais la communication inter-processus doit passer par le réseau. Afin de limiter le ralentissement dû au transfert et à la synchronisation, des supports plus performants ont été développés, Myrinet par exemple. Les architectures SMP n'ont pas ce problème, la mémoire étant partagée et accessible par tous les processeurs. Le nombre de processeurs est toutefois limité. Pour cette raison, les systèmes à accès non uniforme (NUMA) sont devenus populaires car ils possèdent des temps d'accès moyens moins rapides que les systèmes SMP mais très acceptables. Cette architecture offre une meilleure évolutivité du nombre de processeurs et un prix beaucoup moins élevé. Finalement, les ordinateurs à mémoire partagée (SMP ou NUMA) sont plus simples à gérer et à utiliser que les grappes Beowulf mais le défi demeure de les exploiter convenablement pour en tirer tout leur potentiel (et rentabiliser l'investissement).

Les applications nécessitant la puissance de ces grappes de calcul sont nombreuses. On en retrouve dans des domaines classiques tels la mécanique des fluides numériques mais aussi pour de nouveaux domaines en expansion, par exemple le génie génétique. On peut tout simplement se servir de chaque noeud individuellement et exécuter plusieurs simulations indépendantes (Monte-Carlo). Les grappes de calcul deviennent populaires pour agir en tant que serveurs en « Haute Disponibilité » pour les systèmes téléphoniques par exemple. En fait, les nombreuses unités de traitement peuvent recevoir des milliers de requêtes simultanément et offrir un service rapide à

moindre coût. Cependant, les grappes Beowulf sont destinées principalement au calcul de haute performance faisant intervenir des algorithmes parallèles.

On peut définir la puissance totale d'une grappe en additionnant les statistiques individuelles de chaque processeur. En pratique, il est toutefois difficile d'atteindre ce niveau de performance sauf pour quelques applications dites parallèlement embarrassantes (embarassingly parallel, en anglais). Il existe donc des outils et des bancs d'essai qui permettent d'évaluer la puissance réelle et exploitable des ordinateurs parallèles. (Linpack par exemple)

2.5 Paradigmes de programmation

Il existe plusieurs types de parallélisme opérant à différents niveaux et dépendant de l'architecture matérielle. Lorsqu'une grappe est composée de nœuds distincts ayant chacun leur système d'exploitation, il doit y avoir plusieurs programmes s'affairant à la même tâche simultanément pour pouvoir parler d'exécution parallèle. Deux familles distinctes caractérisent les programmes parallèles, soit SPMD (Single Program Multiple Data) et MPMD (Multiple Program Multiple Data). L'approche MPMD la plus connue est le maître-esclave ou client-serveur. Deux programmes distincts ou plus doivent être conçus. Généralement, les deux entités différentes sont appelées à communiquer ensemble; les esclaves, par exemple, ont rarement la possibilité d'échanger directement entre eux. Ce type d'application est souvent parallèlement embarrassant, c'est-à-dire que les données sont faciles à distribuer et les dépendances entre les unités de traitement sont minimales. Après la distribution des données, les calculs se font en parallèle pendant que le maître attend une réponse d'un esclave. Lorsque ce dernier devient disponible pour le calcul, une nouvelle tâche peut lui être attribuée par le maître.

L'autre paradigme, SPMD, permet la conception d'un programme parallèle basé sur un seul code source. À part le numéro unique qui sert à les différencier à l'exécution,

tous les processus sont égaux et exécutent les mêmes instructions. En s'identifiant, le processus peut définir son domaine de travail unique, donc ses données de départ. Normalement, l'interdépendance entre les données voisines amène un lot de problèmes et la communication entre voisins devient nécessaire.

Par la nature discrète et la localité géométrique de ses calculs, nous verrons que la méthode des éléments discrets se prête bien à la parallélisation SPMD. Cependant, certaines opérations telles la récupération des données sont naturelles pour l'architecture maître-esclave (MPMD). Heureusement, il est toujours possible d'imiter un programme MPMD à l'intérieur d'un programme SPMD en créant des sections distinctes (Gropp, 1999).

2.6 Les communications inter processus

Pour permettre la communication entre des processus s'exécutant sur des ordinateurs indépendants, plusieurs solutions existent. Dans le cas d'une grappe de calcul, les nœuds sont reliés grâce à des cartes réseau et un aiguilleur. Exploiter directement le protocole TCP et/ou l'utilisation de « *sockets* » est possible, mais la gestion des échanges entre plusieurs parties rendrait la programmation assez complexe. L'utilisation d'une bibliothèque de plus haut niveau s'impose. Avec l'émergence des ordinateurs parallèles, plusieurs produits ont vu le jour au début des années 1990. Il s'agissait en fait de bibliothèques développées principalement par les constructeurs et donc compatibles uniquement avec leur architecture matérielle. En 1993, PVM (Geist et al., 1994) en a séduit plus d'un. Portable et efficace pour des communications point à point, il est devenu largement utilisé pour les applications scientifiques parallèles sur des réseaux de stations de travail. Par la suite, un ensemble de membres influents de l'industrie a proposé l'élaboration d'un nouveau standard. En 1995, les spécifications MPI-1 sont devenues officielles. Particularité de MPI, il ne s'agit pas d'une implantation comme PVM mais bien d'un ensemble de fonctions ayant un rôle défini et une syntaxe

précise. Plusieurs implantations du standard ont vu le jour dont les deux principales sont du domaine public: Mpich (www-unix.mcs.anl.gov/mpi/mpich/) et LAM (www.lam-mpi.org). D'autres compagnies ont développé des versions commerciales telle celle d'IBM qui est disponible sur le Réseau Étoile de l'École Polytechnique. Ces versions payantes peuvent offrir des caractéristiques uniques pour le déverminage et/ou de meilleures performances pour certaines architectures. MPI est plus qu'une simple standardisation des opérations de base en calcul parallèle. De nombreuses fonctions permettant la communication collective (broadcast) ont été ajoutées ainsi que diverses fonctions utilitaires. En 1997, la norme MPI-2 a permis de définir d'autres extensions reliées aux fichiers partagés et à la mémoire partagée via MPI. Cependant, ces fonctionnalités ne sont pas toutes officiellement supportées par les implantations MPI. Donc, un programme basé sur ces nouvelles extensions n'est pas nécessairement portable.

2.7 Parallélisation de méthodes numériques

Pour résoudre un problème important, plusieurs algorithmes adoptent l'approche diviser pour régner. Par exemple, pour trier, il est souvent plus rapide d'exécuter une opération sur des plus petits groupes de donnée et ensuite recombiner tous ces sous-ensembles de façon appropriée. Avec les grappes de calcul, il est naturel d'effectuer une telle séparation et de distribuer le travail sur les unités de traitement disponibles. Pour la méthode des éléments discrets, les données sont les particules et les calculs sont la détection et la résolution des collisions. La dynamique moléculaire (Molecular Dynamics, MD) ayant précédé la DEM, nous avons remarqué un décalage au niveau de la parution d'articles sur la parallélisation des deux méthodes. En fait, le calcul parallèle est essentiel en MD parce que le nombre d'atomes est très élevé, encore plus que dans le cas d'écoulements granulaires. La taille des simulations augmentant plus rapidement que la puissance des ordinateurs, une approche passant par la décomposition du travail est

vite devenue la seule solution viable. Comme le précise Plimpton (1995), il existe trois types de division du travail qui peuvent être appliqués à la dynamique moléculaire et donc indirectement à la DEM :

- Distribution des particules;
- Distribution du calcul des forces;
- Décomposition spatiale du domaine.

2.7.1 Distribution des particules

Avec ce paradigme, l'ensemble des processeurs (P) a accès à l'ensemble des particules (N) de la simulation. Indépendamment de leur position, chaque processeur choisit arbitrairement un sous-ensemble d'objets dont il aura la responsabilité. Le défi principal est de recombiner l'information pour qu'elle soit disponible lors de la prochaine itération pour tous les processeurs. La quantité de données à échanger est très grande. Cette technique est applicable en MD parce que les forces ont un rayon d'action infini. Une implantation SPMD est réalisable sans trop de modifications à la version séquentielle. Il suffit de concevoir un mécanisme de communication collective (en $O(log_P N)$) et de remplacer les boucles allant de *1 à N* par des boucles allant de *1 à N/P*.

2.7.2 Distribution du calcul des forces

La procédure ressemble beaucoup à la distribution des particules; elle est toutefois contrainte à l'utilisation d'un nombre carré de processeurs. Inspirée d'algorithmes parallèles pour le calcul matriciel et appliquée pour la première fois par Plimpton (1995) à la dynamique moléculaire, la séparation de la matrice des forces de dimensions (NxN) se fait en blocs au lieu de se faire en rangée. La complexité du patron de communication n'est plus de l'ordre linéaire mais réduit à $O(N/\sqrt{P})$. Il en résulte un gain non négligeable puisque, grâce à la réduction de la taille des messages, on sauve

beaucoup sur les communications qui sont le facteur limitant lors d'une parallélisation par distribution de données.

2.7.3 Décomposition spatiale du domaine

Les deux méthodes précédentes font que chaque processeur a une vue de tout le domaine de simulation, ce qui implique beaucoup de communications pour le réseau d'une grappe de type Beowulf. Pour contourner ce problème, la dynamique moléculaire a recours à une hypothèse réductrice : une approximation limitant le rayon d'action des forces. Procéder ainsi permet de distribuer les particules selon une décomposition géométrique du domaine de simulation (Figure 2-5). On génère de petits sous-domaines qui regroupent des particules spatialement rapprochées.

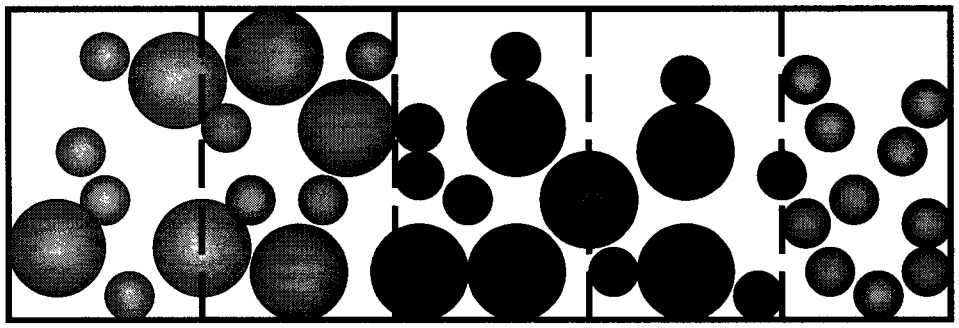


Figure 2-5 Décomposition initiale de domaines sur 5 processeurs

Pour calculer correctement les forces appliquées aux corps en périphérie, des communications avec les sous-domaines voisins directs doivent être effectuées avant le début des calculs. Dans le pire des cas, soit en 3 dimensions, la quantité de données reçues par chacun des processus est équivalente à (Clark, 1994) :

$$Total\ des\ données = (8k^3 + 12\,k^2 + 6k)\,N/P \qquad \textbf{Équation 2.9}$$

où k est le rayon d'action des forces (en unité de sous-domaines), N est le nombre de particules et P le nombre de processeurs. Pour la DEM, $k=1$ est suffisant, on peut donc

simplifier cette expression et la généraliser à toutes les décompositions multidimensionnelles :

$$Total\ des\ données = ((2k+1)^{\text{Nombre de dimensions}} - 1)\ \text{X}\ N/P \qquad \textbf{Équation 2.10}$$

$$Total\ des\ données = (3^{\text{Nombre de dimensions}} - 1)\ \text{X}\ N/P \qquad \textbf{Équation 2.11}$$

Ici, on considère autant les voisins directs que diagonaux et les valeurs possibles sont évidemment 2, 8 et 26. On remarque que plus *P* est grand, moins la quantité de données à transférer par processeur est grande. Pour cette raison, il est tentant de décomposer en 3 dimensions et mettre à contribution le plus de processeurs possibles. Il faut être toutefois très prudent car la quantité totale de messages et de données augmente, ce qui finit par réduire les performances. De plus, comme le transfert de données se fait dans les deux sens, une quantité de données de l'ordre de *52 N/P* est échangée. Cependant il s'agit de la valeur maximale, atteinte seulement lorsque les processus communiquent leur sous-domaine en entier aux voisins. Heureusement avec la DEM, c'est rarement le cas, car les forces sont à court rayon d'action et seulement une partie des données du sous-domaine nécessite d'être transférée. Donc, mis à part cette communication, les processus sont presque indépendants et peuvent exécuter les opérations habituelles sur les particules de leur sous-domaine en parallèle. Pour la décomposition spatiale, le paradigme SPMD est préconisé car seules les données diffèrent entre les processus.

2.8 Parallélisation de la méthode des éléments discrets

La méthode des éléments discrets possède des caractéristiques qui facilitent la parallélisation à l'aide de la décomposition de domaine. Le rayon d'action étant la collision (donc nul), la quantité de données à échanger est réduite au minimum. Ces données sont situées à l'intérieur d'une bande appelée « halo » partagée par des

processeurs voisins. Ainsi, pour une décomposition 1D (Figure 2-6), deux vecteurs halos sont crées pour chaque sous-domaine. La réception et l'envoi de données se font en alternance et les nouvelles particules reçues sont ajoutées à la fin de la liste de particules locales (Henty,2000). Donc, à part les routines de communication, aucun changement majeur à la version séquentielle est requis, un atout non négligeable. Les calculs se font simultanément sur chaque processeur. Il n'y a qu'un léger supplément de travail attribuable aux collisions de particules frontières de chaque halo qui sont calculées par les deux processus.

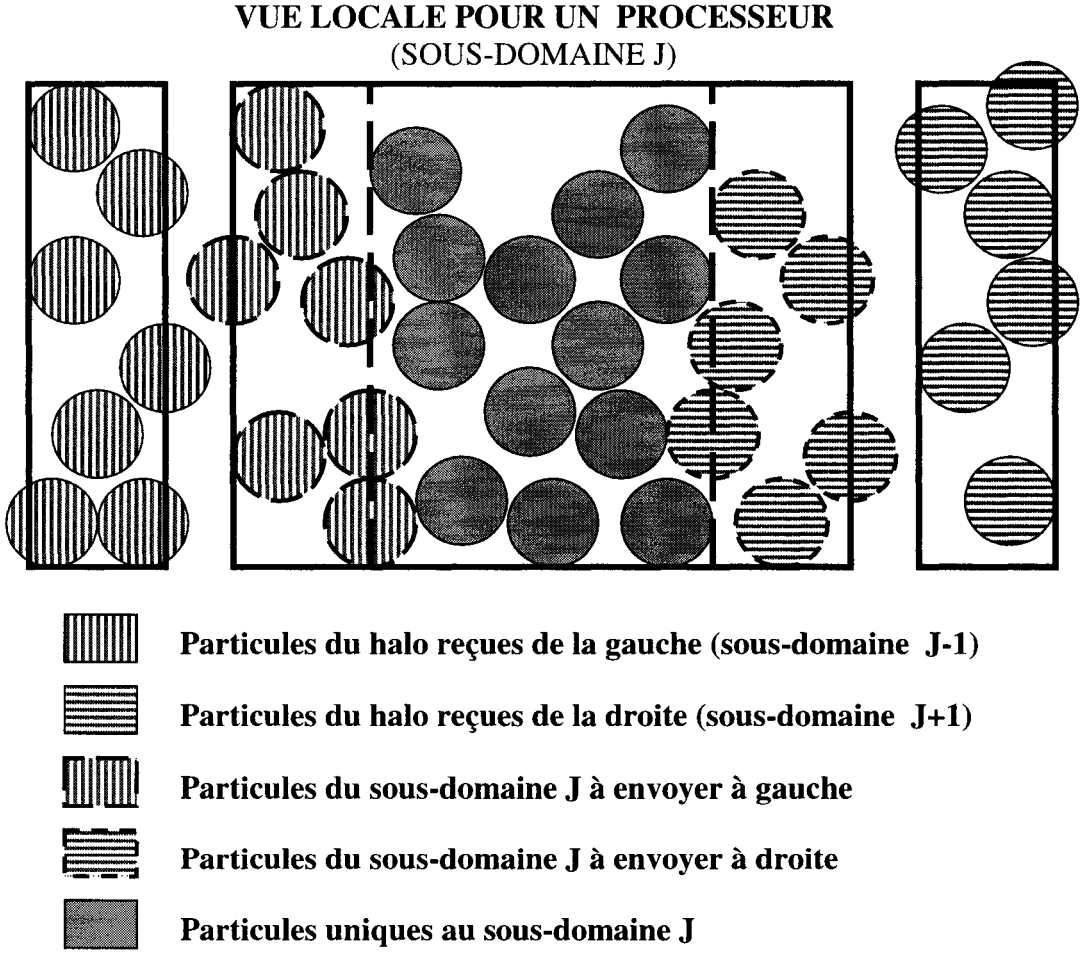


Figure 2-6 Exemple de halos

Les accélérations obtenues par différents groupes de recherche démontrent que la décomposition de domaine est la meilleure technique pour paralléliser sur des machines à mémoire distribuée. Grâce à un CRAY T3D et un pourcentage très faible de code non parallélisable, une équipe suisse a obtenu une accélération de 78 avec 128 processeurs (Ferrez, 1996). Les accélérations sur les grappes de calculs Beowulf sont plus modestes et parfois linéaires jusqu'à 16 processeurs. L'efficacité tombe sous la barre des 50 % lorsque qu'un déséquilibre dans les tâches désynchronise les processus et/ou lorsque le nombre de particules devient insuffisant (ratio temps-calcul/temps-communication trop faible).

2.9 L'équilibrage dynamique de charge

Sur une chaîne de montage dans une usine où le travail est distribué, il est important que la charge de travail soit bien équilibrée pour éviter les accumulations et les goulots d'étranglement. La synchronisation et l'équilibrage des tâches permettront d'optimiser le travail de chaque ouvrier et donc d'obtenir une meilleure production totale. La parallélisation d'applications à l'aide de plusieurs ordinateurs bénéficie directement de l'équilibrage de charge. L'analogie avec la chaîne de montage n'est pas applicable à la DEM parce que la méthode est itérative et que la parallélisation par décomposition fonctionnelle est impossible. La parallélisation de la méthode des éléments discrets étant très sensible au déséquilibre, il est donc normal de tenter de minimiser son impact.

Pour plusieurs méthodes numériques, l'équilibrage statique effectué au début est suffisant parce que les charges restent les mêmes pour la durée de la simulation. Alors qu'avec la méthode des éléments finis, on peut utiliser de librairies automatiques tels *Metis* (www-users.cs.umn.edu/~karypis/metis) qui se charge de découper équitablement le maillage, les méthodes particulaires se débrouille très bien avec une décomposition

cartésienne du domaine. Cependant, rien ne garantit que ces premières estimations seront valides pour toute la durée de la simulation. L'implantation du mécanisme d'équilibrage dynamique des charges s'impose pour tenir compte des changements de sous-domaines des particules en mouvement.

D'après Hendrickson (2000), un bon système de rééquilibrage des charges pour une application scientifique est caractérisé par :

- un équilibre efficace;
- de courts temps de communication pour redistribuer les charges;
- une exécution rapide en parallèle;
- une utilisation de la mémoire modeste;
- une petite variation devrait impliquer seulement un petit changement dans la décomposition;
- une détermination facile du patron de communication pour définir les voisins. Dans ce cas, les décompositions géométriques simples ont un avantage versus les plus complexes.

Une implantation maître-esclave semble séduisante par sa simplicité et son efficacité à équilibrer les charges. Il est fort probable que cette approche combinée aux sous-domaines permettrait de distribuer le travail et de maintenir une charge maximale d'utilisation. Chaque fois qu'un esclave termine ses calculs pour un sous-domaine, il peut en demander un nouveau au serveur. De plus, il est inutile d'attendre la complétion d'une itération sur l'ensemble du domaine. Si des sous-domaines voisins ont tous complété l'itération N, le maître peut commencer à redistribuer ces sous-domaines pour le calcul de l'itération N+1. Le problème de synchronisation est éliminé à condition que le nombre d'esclaves soit inférieur au nombre de sous-domaines. Une vague orientée comme la flèche à la figure 2.7 balayera le domaine et les esclaves travailleront constamment.

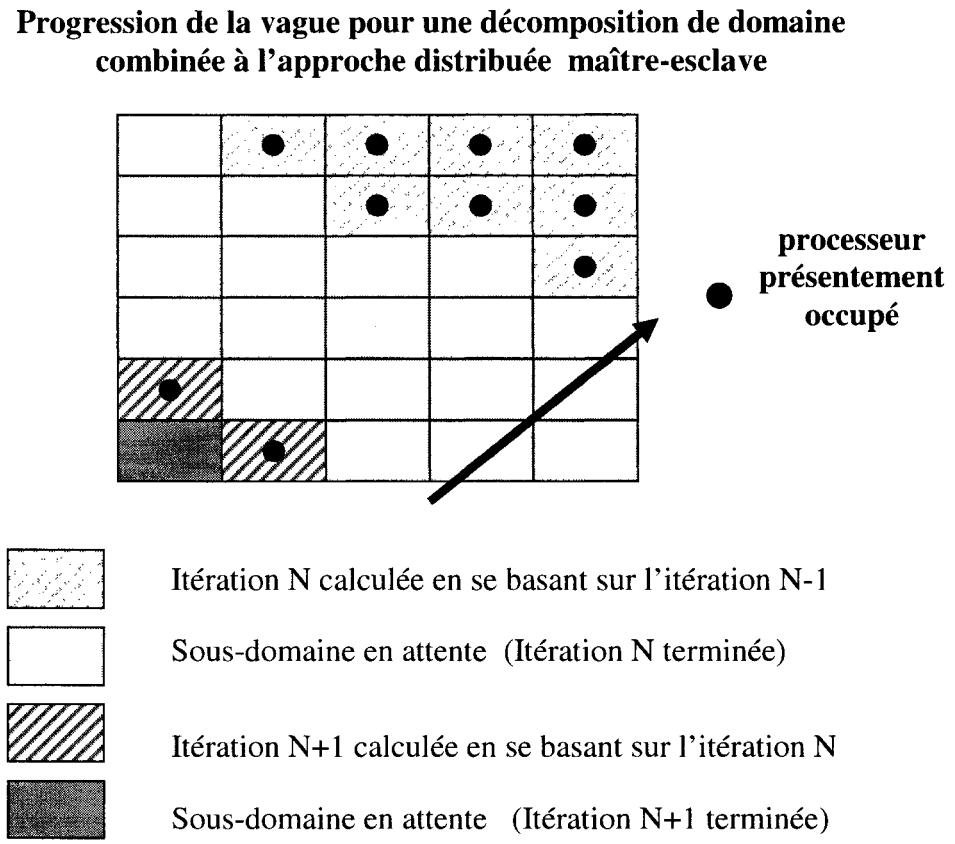


Figure 2-7 Décomposition de domaine couplée à un paradigme maître-esclave

Malheureusement, pour des raisons similaires à celles exprimées précédemment à propos de la distribution des particules en dynamique moléculaire, cette approche souffre d'un inconvénient majeur : l'énorme coût des communications. Pendant que le maître reçoit toutes les informations des particules d'un sous-domaine, un autre esclave peut essayer de communiquer et devient sous-utilisé pendant l'attente. Un système à plusieurs maîtres repousserait temporairement le nombre maximum d'esclaves sans toutefois régler le problème complètement. De plus, le maître doit avoir accès à beaucoup de mémoire : non seulement il doit contenir l'ensemble des particules, il doit aussi conserver au moins 2 itérations de chaque sous-domaine. Enfin, les algorithmes de recherche des voisins doivent être très économiques car ces opérations (initialisation + recherche) doivent être faites à chaque réception contrairement à l'approche SPMD qui

assigne un sous-domaine à un processus et qui suppose que la recherche restera valide pour quelques itérations.

En résumé, pour des applications telles les simulations Monte-Carlo ou les lancers de rayons en infographie, il est souvent pratique d'utiliser le paradigme maître-esclave pour sa simplicité à équilibrer la charge mais pour plusieurs autres applications, il existe des méthodes qui nécessitent moins de communications et qui s'adaptent mieux au paradigme SPMD.

Pour remédier au problème d'équilibrage de tâche dynamique, Kohring (1995) décrit une méthode plus appropriée à la parallélisation de la DEM : la technique du « *diffusion scheme* ». Pour un système comprenant P processeurs, où $W_i(T)$ est la charge sur le processeur i, il suffit d'ajuster la charge au temps $T+1$ de la façon suivante :

$$W_i(T+1) = W_i(T) + \sum_{\forall j \in N(i)} (W_j(T) - W_i(T))/2 \qquad \textbf{Équation 2.12}$$

où $N(i)$ est l'ensemble des domaines voisins du processeur i. Il est reconnu que cet algorithme converge vers une solution équilibrée, mais très lentement. Pour modifier la charge, on déplace la frontière des sous-domaines impliqués. Un certain nombre de particules doivent être transférées, ce qui implique un transfert de données pour chaque déséquilibre détecté. Cleary et Sawley (1999) ont proposé une approche qui permet d'économiser sur le coût de ces communications. Avec cette approche, le processeur surchargé lègue les particules du halo déjà envoyées à son voisin.

Généralement, la décomposition initiale est raisonnable. Le but du système de rééquilibrage est alors de trouver les fluctuations dans les temps de calcul et de modifier les frontières. Dans ce cas, l'algorithme de rééquilibrage arrive à suivre ces mouvements à condition d'être appelé à un rythme plus élevé que les fluctuations elles-mêmes. Une problématique classique est de définir la fréquence idéale de vérification pour détecter le

déséquilibre sans imposer trop de communications. Aucune formule magique ne s'adapte à toutes les situations, ainsi une bonne connaissance du phénomène à simuler est souhaitable. Heureusement, comme les pas de temps sont généralement très petits, il s'écoule plusieurs itérations avant d'observer un déséquilibre. Cela permet donc de considérer de faibles fréquences de rééquilibrage et directement réduire les coûts de vérification.

2.10 La parallélisation et l'utilisation de grappes SMP

Avec l'avènement des grappes de calcul Beowulf qui préconisent le calcul haute performance à moindre coût, un autre phénomène s'est développé ces dernières années, soit celui des grappes composées de nœuds SMP. En effet, il est maintenant commun de rencontrer des serveurs avec 2 ou 4 processeurs Xeon. Ces grappes de calcul offrent la possibilité de développer du code parallèle en utilisant la mémoire partagée d'un noeud et/ou l'approche distribuée avec ses échanges explicites sur le réseau. Dans le contexte de la dynamique particulaire, la combinaison des deux paradigmes peut devenir avantageuse. En effet Smith (2001) dénombre quelques caractéristiques intéressantes de la programmation hybride et certaines situations où il est avantageux de combiner les deux modes :

1. programmes à évolutivité faible avec MPI;
   Quelques applications pour diverses raisons ont de la difficulté à bien utiliser les bibliothèques de communications.
2. Problème d'équilibrage de charge;
   Les codes d'éléments discrets peuvent souffrir de la mauvaise répartition des données alors qu'un programme en mémoire partagée équilibre automatiquement la charge entre les processus légers.

3. Programmes avec des données répliquées;

   Une application typique est la dynamique moléculaire où tous les corps sont nécessaires pour le calcul des forces appliquées sur ceux-ci. Comme il est mentionné plus haut, les communications globales sont très contraignantes. De plus, l'utilisation d'un ordinateur à mémoire distribuée implique beaucoup de dédoublements de données, alors que les fils d'exécution d'un ordinateur à mémoire partagée ont accès à la même mémoire, ce qui permettra de simuler des problèmes de taille plus importantes tout en minimisant les communications.

4. Facilité d'implantation;

   Les applications en mémoire partagée sont souvent plus simples à concevoir que celles en mémoire distribuée. Une application basée uniquement sur MPI avec une bonne évolutivité peut être plus complexe à concevoir qu'un code hybride de performance similaire.

5. Toutes les situations où la programmation en mémoire partagée est plus efficace.

   Une version hybride d'un code d'éléments discrets nécessite la programmation en mémoire partagée. La programmation parallèle sur ces architectures est relativement plus puissante puisque certains échanges se font par le bus mémoire de l'ordinateur.

2.11 La méthode des éléments discrets et la programmation en mémoire partagée

Les avantages d'exploiter les capacités d'un ordinateur multiprocesseur pour un code d'éléments discrets sont nombreux. Chaque processus a accès à toute la mémoire, il n'y a donc pas de dédoublement dû à l'utilisation des halos et aucune communication réseau n'est nécessaire. La granularité de la parallélisation par processus légers est beaucoup plus fine. Au lieu de distribuer des particules, on parallélise les sections ou les boucles du code et l'équilibrage de la charge se fait automatiquement à ce niveau. Donc, l'attente pour le processeur surchargé n'est plus une problématique importante. Il doit

bien sûr avoir une synchronisation des processus légers mais, il en va de même de l'approche de programmation en mémoire répartie.

Sur une machine multiprocesseur, la parallélisation est possible à condition que la communication entre deux processus et la création de zones de mémoire partagée soient permises. Au niveau le plus bas, on retrouve l'utilisation des sémaphores UNIX fournies par le système d'exploitation. Encore une fois, l'utilisation d'un code de haut niveau est souhaitable et les *pthreads* (POSIX) permettent d'y arriver. D'autres bibliothèques publiques ou commerciales facilitent la création de programmes sur processus légers, mais dans le contexte d'applications scientifiques, l'*API* la plus utilisée est OpenMP. Ce standard datant de 1997 est disponible sur les ordinateurs SUN, SGI, IBM et les ordinateurs SMP de type Intel. OpenMP définit un ensemble de directives ajoutées sous forme de commentaires avant les sections à paralléliser. Ensuite, les options de compilations appropriées permettent au compilateur supportant OpenMP de retrouver ces balises et d'ainsi créer un programme ayant la possibilité de fonctionner sur plusieurs fils d'exécution. Les changements au code séquentiel restent minimes et par conséquent l'implantation de la parallélisation est souvent plus facile que la parallélisation en mémoire répartie. Il faut tout de même choisir de façon judicieuse les sections à paralléliser car les opérations de création et de destruction des processus légers sont relativement onéreuses.

En regardant la littérature, on remarque que la plupart des premières implantations parallèles de la méthode des éléments discrets utilisent la décomposition de domaine parce qu'elle est naturelle et possède une bonne évolutivité (Henty, 2000; Ferrez, 2001). En réalité, on retrouve très peu de chercheurs qui exploitent la mémoire partagée de leurs stations de calcul multiprocesseurs. Plusieurs raisons pourraient expliquer cette situation. Le nombre de processeurs reste assez limité (<<64), le prix est exorbitant et les architectures à accès non uniforme (NUMA) ne sont pas assez performantes pour les calculs requis par la DEM. On suppose que les équipes de

recherche préfèrent donc utiliser ces systèmes dispendieux pour lancer plusieurs simulations simultanément plutôt que de faire une simulation en mode parallèle.

Le constat général pour les implantations en mémoire partagée de la DEM est que, peu importe l'architecture, l'efficacité relative diminue trop rapidement avec l'ajout de CPU supplémentaires (Labarta, 2002). Après plusieurs tentatives, Henty (2000), qui a traité à fond le problème, n'a pas réussi à tirer suffisamment profit du potentiel des grappes de calcul SMP pour suggérer son utilisation en mémoire partagée et en mode hybride. Le problème provient du fait que certaines opérations doivent accéder de façon concourante à la mémoire. Des barrières sont alors nécessaires pour garder l'intégrité des données. Ces verrous sont de plus en plus problématiques à mesure que le nombre de processus augmente. Ce goulot d'étranglement au niveau des verrous cause l'effondrement des performances. Henty a conclu qu'il valait mieux utiliser les grappes de calcul composées de nœuds SMP avec la décomposition de domaine où les processus communiquent via une interface comme MPI. En pratique, l'approche hybride pourrait être avantageuse dans les cas où un réseau peu performant serait la principale cause de dégradation. Cette technique permettrait alors de diminuer la quantité de messages à transmettre entre les nœuds SMP.

2.12 Résumé

Les articles concernant l'optimisation de la méthode des éléments discrets sont beaucoup moins nombreux que les rapports expérimentaux qui utilisent cette méthode numérique. Même si la DEM est intense en calculs, la majorité des articles décrivent rapidement les algorithmes utilisés mais ne discutent pas de leur implantation. Parfois, on mentionne la parallélisation. Le peu de données à cet égard provient du fait que souvent c'est le résultat qui importe le plus et non le temps écoulé pour y arriver. Pourtant, la performance de ces algorithmes est cruciale puisque des simulations de plus

en plus importantes devront être effectuées. Aujourd'hui, la DEM permet la simulation de 10^4 particules, mais les besoins sont grandissants et les applications nécessiteront de plus en plus de ressources. Pour l'instant, la parallélisation tente de réduire les temps de calcul mais, à mesure que les simulations se complexifieront, elle sera encore plus attrayante voire inévitable.

3 Objectifs spécifiques et organisation du mémoire

3.1 Objectifs du mémoire

L'objectif général de ce travail est de concevoir une méthode d'éléments discrets parallèle et performante pour la simulation des écoulements granulaires. Plus spécifiquement, cette parallélisation se fera sur un logiciel existant, soit le code séquentiel Powder3D développé à l'U.R.P.E.I. depuis 1 an. Elle sera basée sur la décomposition de domaine et implantée sur une grappe de calcul à mémoire distribuée dont les caractéristiques sont les suivantes :

- Magnum : 12 serveurs biprocesseurs x330 d'IBM (Intel P3-866Mhz)
  - 512 Mb de RAM par noeud
  - Disques durs SCSI
  - Ethernet 100Mbits/s
  - Aiguilleur CISCO catalyst 6500
  - Système d'exploitation : RedHat 7.3
  - Mpich version 1.2.3
  - Compilateur Intel Fortran version 7.1

Puisque chaque nœud de cette grappe comporte deux processeurs en mémoire partagée, nous nous proposons de mettre au point un code hybride qui met à profit les mémoires partagée et distribuée. Enfin nous allons nous appliquer à optimiser la version séquentielle de Powder3D avant de procéder à sa réingénerie logicielle de parallélisation. Pour plus de détails concernant la version séquentielle de Powder3D ainsi que des précisions quant au modèle physique, le lecteur pourra se référer au rapport de Gange (2002) et à Zhou et al. (2001).

3.2 Organisation de la suite de ce mémoire

Nous présenterons au chapitre 4 différents algorithmes de détection de contacts impliquant les particules entre elles ainsi que les particules avec les surfaces solides du domaine de calcul. Le cinquième chapitre, le plus important, fera la description de la méthode de décomposition de domaine. Le chapitre 6 discutera de la problématique du déséquilibre de la charge et de l'équilibrage de charge dynamique. Ensuite, nous allons explorer les possibilités de l'approche hybride. Le paradigme de programmation en mémoire partagée sera en fait testé sur différentes plateformes disponibles à l'U.R.P.E.I. :

- Polaris : superordinateur P690 d'IBM à 16 processeurs (1.1 Ghz)
  - 32 Gb de RAM (Mémoire partagée)
  - Système d'exploitation : AIX 5.1
  - Mpich version 1.2.5
- Réseau Étoile : serveurs quadriprocesseurs P630 d'IBM (1 Ghz)
  - 8 Gb de RAM par noeud
  - Ethernet 100Mbits/s
  - Aiguilleur CISCO catalyst 6500
  - Système d'exploitation : AIX 5.1
- Hamsun: 8 serveurs quadriprocesseurs (Xeon 700Mhz)
  - 1 Gb de RAM par nœud
  - Ethernet 100Mbits/s
  - Système d'exploitation : Adelie Linux (kernel 2.4.20)
  - Mpich version 1.2.5

Le chapitre 7 traitera plus particulièrement de cette implantation sur le nouveau p690 d'IBM. Avant de conclure, nous discuterons des nombreuses améliorations possibles et des corrections à apporter à Powder3D.

3.3 Évaluation des performances

Dans ce travail, nos tests consistent généralement en une sédimentation d'un bloc de particules pendant quelques centaines d'itérations. Dès la première itération, nous imposons que toutes les particules demeurent en contacts avec leurs 6 voisines. Cette contrainte est due à la nécessité d'avoir des temps de résolution en mode séquentiel raisonnablement longs pour que la version parallèle soit efficace. Puisque la disposition des particules est ordonnée, le nombre de contacts est constant dans chaque sous-domaine et, par conséquent, les charges de travail sont équilibrées. Comme on peut le remarquer à la figure 3-1, des domaines de dimensions quelconques sont facilement générés et permettent d'évaluer les performances qui proviennent de nos optimisations.

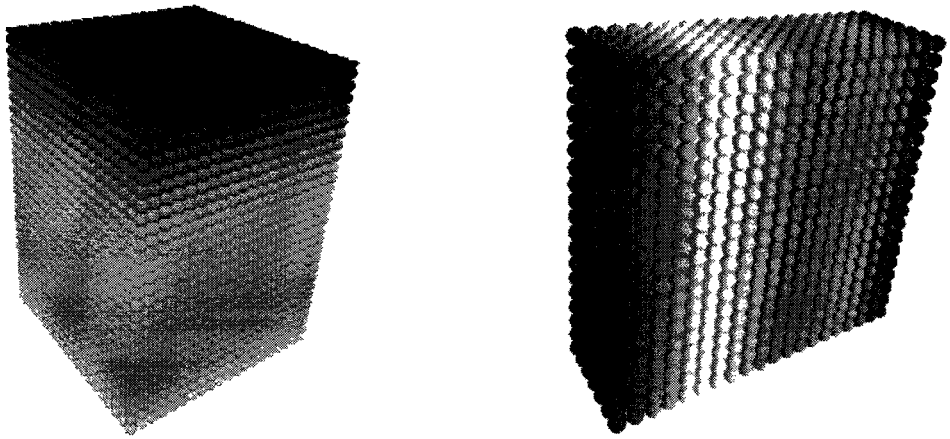


Figure 3-1 Exemple de simulations typiques pour évaluer les performances

4 La détection des contacts

Dans toutes les simulations physiques particulaires, on recherche les interactions entre les différents corps en mouvement. En général, pour les écoulements granulaires, la principale force entre deux particules provient de la force de collision. La détection de ces contacts est une étape importante du processus itératif de la DEM. Non seulement parce que c'est la première et qu'elle est donc nécessaire aux étapes suivantes, mais parce qu'elle peut devenir la plus coûteuse en temps CPU. En effet, une recherche parmi toutes les paires possibles deviendra fastidieuse à mesure que le nombre de particules augmentera puisque la complexité d'une telle vérification est en $O(N^2)$ où N est le nombre de particules. Pour accélérer le traitement, plusieurs méthodes sont disponibles. Nous décrirons et comparerons brièvement les principales tout en expliquant notre choix pour l'implantation dans Powder3D.

4.1 Détection de contacts entre les particules sphériques

Pour l'école des corps déformables, dans le cas de particules sphériques, il y contact lorsque la distance entre les centres de masse est plus petite que la somme des rayons des particules ciblées.

$$\sqrt{(x1-x2)^2+(y1-y2)^2+(z1-z2)^2} \leq R_1 + R_2 \qquad \textbf{Équation 4.1}$$

Toutes les méthodes de détection de contacts présentées plus bas ont un but, définir un voisinage autour de chaque particule. En ayant rapidement accès aux objets les plus proches, on effectue ensuite le test de détection de collision sur un ensemble limité. Toutes ces méthodes contribuent directement à réduire les temps de simulation

puisque le nombre de tests inutiles est réduit. Les techniques décrites seront présentées en 2D, mais habituellement, leur extension est naturelle en trois dimensions.

4.1.1 Méthode de localisation du quadrillage

La recherche des corps voisins est un défi qui passe par la localisation de chaque particule. En quadrillant l'espace, on peut assigner à chacune des cases du quadrillage une référence vers les particules qu'elles contiennent et, par la suite, facilement repérer celles qui sont voisines (Figure 4-1). Cela permet de réduire grandement les comparaisons inutiles avec des particules qui se situent au-delà de ces frontières. Les implantations de la méthode du quadrillage se différencient par la taille de la grille et leurs avantages dépendent des conditions de la simulation. Par exemple, une grille fine, basée sur le plus petit rayon, impliquera beaucoup de recherches à travers des cases vides si la concentration est faible et que les particules sont dispersées. On retrouve l'emploi des méthodes de quadrillage très souvent en dynamique moléculaire, où le rayon maximal est attribué à la largeur des cases de la grille. Pour la méthode des éléments discrets, il peut y avoir un inconvénient à choisir un quadrillage trop grossier basé sur le rayon maximal. Lorsqu'il y a un écart élevé entre les rayons moyen et maximal, les cases contiennent beaucoup de particules et les tests inutiles sont réduits, mais pas de façon optimale. Enfin, peu importe sa taille, le quadrillage doit être fréquemment reconstruit si la vitesse des particules est élevée.

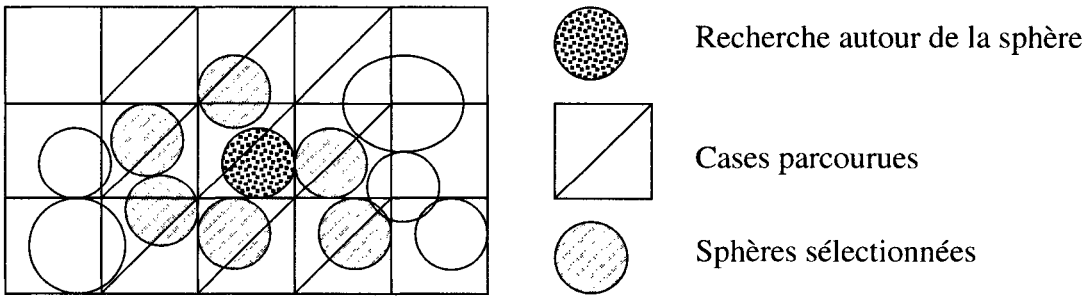


Figure 4-1 Technique du quadrillage régulier statique (grid search)

4.1.2 Quadrillage adaptatif et dynamique

Avec une structure plus économique en mémoire, cette méthode dynamique utilise les arbres quaternaires en 2D pour classer les sphères dans l'espace. Elle s'ajuste mieux aux différentes concentrations que l'on peut retrouver dans le domaine. Elle est implantable en 3 dimensions et s'adapte à la parallélisation (Warren, 1993). Elle s'applique un peu plus mal aux écoulements granulaires qui sont généralement assez denses et semble trouver preneur dans d'autres domaines tels que l'astrophysique. Le plus grand défaut des méthodes dynamiques, c'est qu'elles sont plus complexes à gérer.

4.1.3 Méthode du tri

Les difficultés de la première méthode de quadrillage surviennent lors de simulations en milieu polydispersé où le rapport des rayons maximum et minimum est élevé, puisqu'il est alors plus difficile de définir une largeur de grille optimale. Pour contourner cette problématique, O'connor (1996) a proposé une méthode de localisation basée sur le tri en fonction de la position spatiale. Une deuxième version plus récente, proposée par Williams (2001), se démarque encore plus des méthodes conventionnelles lorsque le ratio entre le rayon des sphères est important. Elle peut s'interfacer avec la parallélisation par décomposition de domaine, mais le nombre de tris pourrait dégrader les performances.

4.1.4 Méthode de proximité (ou halo)

L'idée générale de cette technique est de maintenir une liste des plus proches voisins potentiels autour de chaque objet, tout en tentant de réduire sa taille au minimum. La principale difficulté est de définir la bonne fréquence de rafraîchissement en fonction de la largeur du halo (Figure 4.2). Une fréquence trop faible peut entraîner des oublis de contacts pendant quelques itérations, donc de l'instabilité dans la simulation, alors qu'une largeur de halo inutilement trop grande sera moins performante.

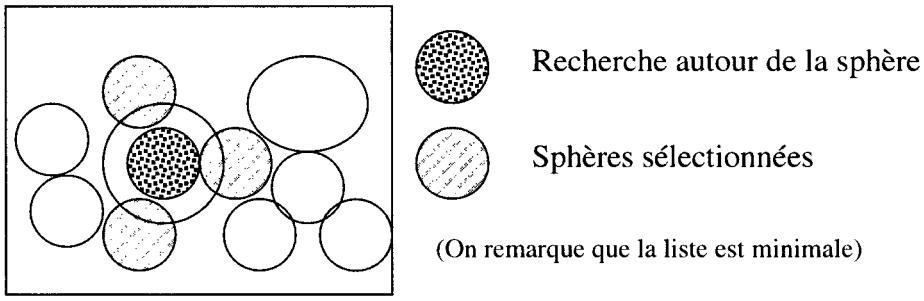


Figure 4-2 Méthode de proximité avec halo

4.1.5 Méthode basée sur la triangulation de Delaunay

Une dernière méthode fort intéressante est basée sur la triangulation de Delaunay. Après avoir généré le diagramme de Voronoï (arêtes reliant tous les corps voisins entre eux), on s'assure que tous les quadrilatères formés de 4 centres de masses rapprochés sont convexes et que la diagonale de ce quadrilatère est de longueur minimum. Le coût de construction du graphe ayant ces propriétés est élevé, mais le maintien reste économique. À intervalles réguliers, on le parcourt et on effectue les changements nécessaires. Les « *flips* » (Figure 4-3) permettent de conserver les propriétés du graphe et d'assurer une détection sans erreur. Cette technique obtient de très bons résultats puisqu'elle permet de limiter de façon optimale le nombre de voisins potentiels. L'implantation en 3D est beaucoup plus contraignante (tétraèdres au lieu de

triangles), mais toujours performante (Ferrez, 2001). De plus, les algorithmes reliés au parcours du graphe sont aussi parallélisables en mémoire partagée. À part la construction du graphe qui demande quelques ajustements, la triangulation de Delaunay peut s'adapter à la décomposition de domaine.

Recherche de collisions basée sur la triangulation de Delaunay

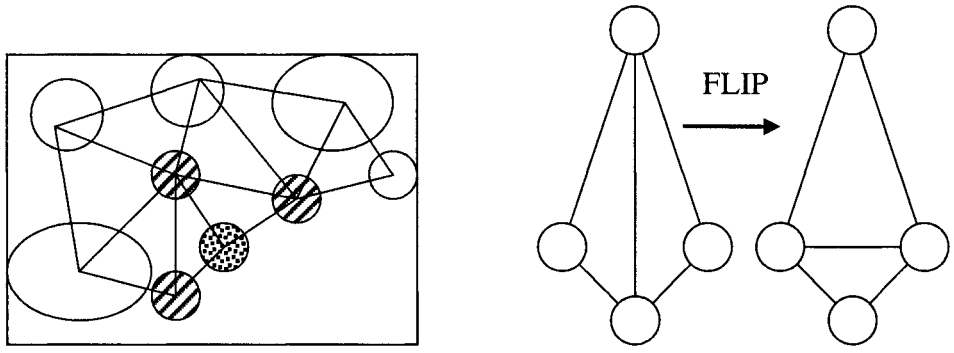


Figure 4-3 Technique de la triangulation de Delaunay et son opération de maintenance
a) Graphe obtenu grâce aux triangulations, le nombre de voisins est réduit au minimum
b) opération de *FLIP* permettant de maintenir le lien entre les particules les plus proches (diagonale la plus courte)

4.2 Le choix d'un algorithme de détection de contacts

Le premier argument pour choisir un algorithme devrait être la performance, ensuite sa flexibilité, sa simplicité et sa facilité d'implantation dans un code parallèle basé sur la décomposition de domaine.

La première technique implantée dans ce travail fut le quadrillage de l'espace avec une taille de cases d'environ 2,5 fois le rayon moyen. Le rafraîchissement de cette grille se faisait à intervalle régulier en fonction de la vitesse des particules. À chaque itération, la boucle sur les particules parcourait les cases voisines afin d'effectuer les tests de collision sur les paires retenues. Cette première implantation devait en théorie réduire la complexité de la recherche à un ordre linéaire en fonction du nombre de

particules. Procéder ainsi permettait d'atteindre cet objectif, mais la détection nécessitait plus de 85% du temps total. Cette fraction était beaucoup trop importante car c'est le calcul des forces qui devrait être dominant (Henty, 2000).

Pour accélérer la détection, il est souvent proposé d'exploiter le fait qu'un contact implique toujours deux particules. Ainsi, seulement une des deux particules doit détecter la collision. On peut alors garder les références aux particules ayant un numéro d'identification strictement plus grand. Cela permet de simplifier la détection et diviser de moitié les coûts associés au calcul des forces. Cependant, cette règle n'est plus applicable dans le cas d'écoulements polydispersés. Pour des raisons pratiques, il est primordial que ce soit la grosse sphère qui détecte ses voisines plus petites. L'algorithme doit donc parcourir l'ensemble des cases et se limiter aux sphères ayant un indice plus élevé et/ou un rayon strictement plus petit.

A la lumière de premières évaluations avec des outils de profilage, le choix d'un nouvel algorithme de détection était souhaitable. La première méthode retenue fut la triangulation de Delaunay. Malgré sa complexité apparente, elle réduit efficacement le nombre de recherches infructueuses. Cependant, le besoin éventuel d'introduire des forces supplémentaires dans le bilan de forces compliquait l'implantation des triangulations. En effet, dans le cas de poudres fines, les forces colloïdales (capillaire ou de Van der Waals) doivent être prises en compte. Lorsque ces forces à grand rayon d'action entrent en jeu, la triangulation de Delaunay pour la recherche des collisions n'est plus adéquate. Elle permet d'identifier seulement les voisins immédiats. Enfin, la difficulté d'utiliser cette méthode pour la détection de collisions entre des sphères et des objets solides de type triangulaire (Muller, 1996, Ferrez, 2001) fait que cette méthode ne correspond pas à nos besoins. Le manque de flexibilité de cette méthode nous aurait obligé à utiliser deux méthodes de localisation, ce qui aurait nuit à la clarté du code ainsi qu'à son uniformité.

Le problème rencontré avec notre première implantation de la méthode de quadrillage est qu'elle était efficace avec peu d'objets, mais quand ce nombre augmentait, le nombre de tests effectués sur des particules n'ayant aucune chance d'entrer en collision devenait trop grand. Sans la conservation en mémoire des numéros des particules dans le voisinage de chaque particule, cette implantation devait parcourir l'espace quadrillé et refaire les mêmes tests négatifs continuellement. Dans son étude sur les algorithmes associés à la dynamique moléculaire, Plimpton (1995) affirme que combiner les techniques du halo et du quadrillage est très efficace à condition que les forces aient un court rayon d'action. Cette technique s'applique donc directement à la DEM avec ou sans forces de cohésion.

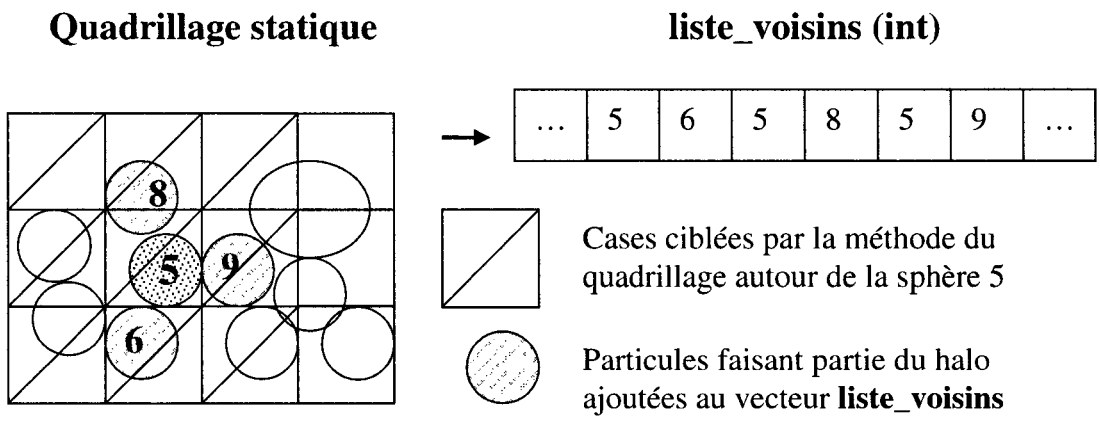


Figure 4-4 Description de la technique implantée (quadrilage + liste de proximité)

L'ajout d'une liste pour contenir les paires de voisins potentiels a permis une accélération de 30%. De plus, cette liste exhaustive de voisins reste bonne pour plusieurs itérations car elle repose sur un quadrillage qui est valide pendant une période déterminée. Il n'est donc pas nécessaire de reconstruire la liste à toutes les itérations. Une autre amélioration consiste à diminuer la taille de la liste de voisins en sélectionnant les particules vraiment rapprochées afin de limiter les tests et l'espace supplémentaire requis par la méthode (Figure 4-4). Ce test de proximité est effectué lors de la construction de la liste des voisins.

$$\sqrt{(x1-x2)^2+(y1-y2)^2+(z1-z2)^2} \leq R_1+(R_2 * facteur) \qquad \textbf{Équation 4.2}$$

Le facteur doit être supérieur à 1 et il influence fortement la fréquence de rafraîchissement des structures. En faisant varier le halo en fonction du rayon de la sphère j (autour de la sphère i), cette astuce permet de minimiser le nombre de liens potentiels à stocker dans la structure *liste_voisins* (Figure 4-5) dans les cas polydispersés. Il faut le choisir avec prudence car s'il est trop petit et combiné à une fréquence de rafraîchissement faible, de nombreux contacts pourraient être alors oubliés et la simulation ne serait plus valide. Avec une fréquence de rafraîchissement assez élevé (moins de 50 itérations, 1,1 (10% du rayon) est une valeur qui n'a jamais posé de problème si la vitesse maximale est raisonnable (plus petite que 5 m/s). Évidemment, dans le cas de force à rayon d'action plus important, le facteur multiplicatif devra être ajusté en conséquence.

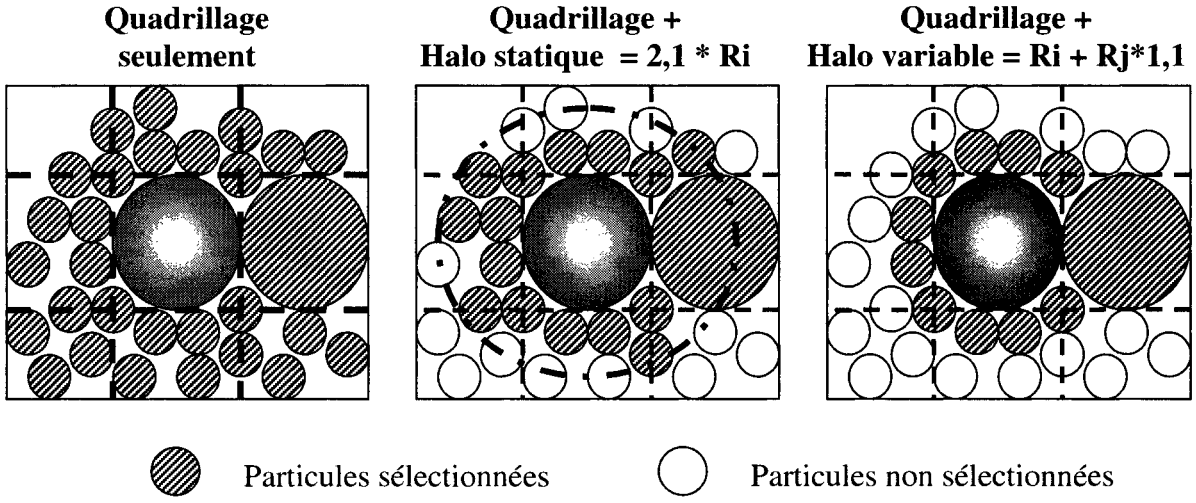


Figure 4-5 Comparaison du nombre de contacts potentiels pour des méthodes de détection
Ri : rayon de la particule centrale qui recherche la particule j
Rj : rayon de la particule sélectionnée dans le voisinage

L'algorithme de détection avec quadrillage et halos variables est très performant. Le coût relatif de la détection a chuté au point que c'est la résolution du contact qui est devenue la plus dispendieuse (Tableau 4-2). La construction des listes est négligeable

(moins de 2% du temps total). Cependant, le tableau 4-2 ne montre pas que la résolution du contact est une fonction comprenant deux étapes : le calcul des forces précédé du test de collision. Grâce au pire cas (Tableau 4-1), correspondant à une simulation sans contact, on estime à 30% le coût maximal de la détection. Pour y arriver, il suffit d'additionner le temps passé dans les routines de détection (*contact_detection*) et de résolution (*contact_resolution*) et une partie de la fonction puissance[ii]. On remarque aussi que les coûts de détection restent faibles par rapport au calcul du mouvement et de la position (*motion_eqns_calc*). Un algorithme de détection idéal pourrait donc au mieux améliorer les temps d'environ 30%. Avec l'implantation de notre nouvel algorithme, des optimisations supplémentaires deviennent difficiles à réaliser, car les opérations à effectuer dans la section de résolution dépendent de la complexité du modèle. En effet, le seul moyen pour accélérer la simulation est d'implanter un modèle moins complexe. Mentionnons que la technique de la grille combinée au halo, en plus d'offrir de bien meilleures performances, reste compatible avec la parallélisation par décomposition de domaine présentée au chapitre 5.

Tableau 4-1 Profilage de l'application pour une simulation sans contact (Sédimentation de particules sur Magnum)

% Time	self seconds	calls	Name
23.70	116.25	2201	**motion_eqns_calc_**
15.25	74.83	2201	**contact_resolution_**
14.36	70.44		**Pow.A**
12.22	59.95	2201	memory_clean_
11.77	57.74	2201	**contact_detection_**
2.55	12.52		FixFree
2.30	11.29		F90_allocate1
1.84	9.01		__mcount_internal

Tableau 4-2 Profilage de l'application pour une simulation avec contacts (Sédimentation de particules sur Magnum)

% Time	self seconds	Calls	Name
26.02	90.83	269	**contact_resolution_**
10.49	36.62		**pow.A**

[ii] En effet, la fonction instantanée pow.A est comptabilisée de façon indépendante et ce sont les routines de résolution et de détection qui lui font le plus souvent appel.

5.43	18.95		FixFree
5.27	18.40	268	memory_clean_
4.61	16.11		__mcount_internal
4.06	14.16		F90_allocate1
3.99	13.94	269	old_contact_lt_
2.74	9.58	268	**motion_eqns_calc_**
2.55	8.92	224535670	Cross_product_
1.42	4.97	269	**contact_detection_**

4.3 Détection de collisions pour objets non-sphériques

La détection des collisions optimisée par la définition d'un voisinage est aussi valide pour une simulation avec des objets polyédriques à condition de la combiner une technique de détection appropriée comme les volumes englobants. Le volume englobant est une forme géométrique simple (boîte, sphère) qui contient tous les sommets d'une forme géométrique plus complexe. Il y a donc une étape de pré-détection où l'on sélectionne les volumes englobants en collision. Ensuite, le véritable calcul de la distance et du point de contact peut avoir lieu comme prévu. Les vérifications inutiles sont non seulement plus nombreuses, mais surtout plus complexes car elles impliquent des objets à plusieurs faces (Hogue, 1998; Muller 1996).

Dans cette situation, il est possible d'utiliser une des bibliothèques de détection commerciales ou « *open-source* » existantes. De nombreuses bibliothèques gratuites et performantes se retrouvent sur le Web (Swift, I–Collide, H–Collide, V–Clip). Ces bibliothèques ont bien sûr leurs limites mais peuvent s'avérer utile. Certaines supportent les objets à plusieurs faces et d'autres ne donnent pas la grandeur du chevauchement lors de collision. Elles ne sont donc pas toujours appropriées à un code d'éléments discrets.

4.4 Algorithmes reliés à la géométrie

Avant de commencer la simulation d'un écoulement granulaire par la méthode des éléments discrets, on doit définir un domaine de simulation pour les particules. Pour des raisons pratiques, dans Powder3D, ce domaine est rectangulaire (boite 3D) et sa

dimension, fournie par l'utilisateur, est utilisée pour la création la grille de détection. Pour l'instant, la taille du domaine demeure statique et aucun objet ne doit en sortir. Pour empêcher un objet de quitter le domaine de simulation, il est possible d'imposer des conditions périodiques. Aucune frontière n'est solide et les sphères franchissant les limites sont déplacées à la frontière opposée. Évidemment, la détection via le quadrillage doit gérer ce cas spécial, c'est-à-dire vérifier les bonnes cases et effectuer une translation temporaire pour le test de collision. Les applications sans frontières sont plutôt limitées. En effet, la plupart des problèmes intéressants telle la sédimentation impliquent au minimum un mur. Une fonctionnalité permettant d'ajouter des surfaces solides en laissant le plus de flexibilité aux utilisateurs de Powder3D était primordiale. Dans la littérature, on retrouve, avant 1995, plusieurs équipes gérant analytiquement les murs. Chaque ligne ou courbe décrivant un mur de la géométrie est définie par une équation. Cette technique semble suffisante pour des géométries simples en 2D mais devient moins pratique en 3D. On retrouve tout de même quelques textes étonnants portant sur l'étude du mélange granulaire à l'aide d'un ruban hélicoïdal défini par une fonction (Kaneko et al., 2000), mais le code semble limité à cet exemple. De la même façon, Ferrez (2001) a défini un ensemble d'objets 3D (cylindre, cône, boîte), représentant soit des obstacles ou contenants, à l'aide de fonctions.

Dans leur article sur la simulation d'applications industrielles comportant des mélangeurs de type « *tumbling mill* » (tambours rotatifs), Sawley et Cleary (1999) ont utilisé *AUTOCAD* pour créer les géométries du domaine de calcul. Un maillage de cette géométrie permet alors de tenir compte des objets dans la simulation avec la DEM. Cette approche permet à la méthode des éléments discrets d'offrir le niveau de flexibilité souhaité pour devenir un outil sérieux pour l'étude de cas industriels.

Le même principe fut adopté pour Powder3D et une interface avec le mailleur commercial IDEAS a été réalisée. Une fois que l'objet solide a été modélisé, un maillage de surfaces (peau) formé de triangles 2D peut alors être généré. Les simulations peuvent ainsi impliquer des objets immobiles pour contenir les particules (un réservoir

par exemple) et d'autres mobiles pour représenter par exemple les pales d'un agitateur. Cette façon de procéder est nommée *MWM* par Favier (2000). La méthode aux murs multiples (*Multi-Wall Method*) est très souple car chaque mur (plat ou courbe) possède ses propres paramètres. Ce dernier mentionne aussi la possibilité de créer des objets solides réagissant aux forces provenant de leurs collisions avec les particules.

4.4.1 Algorithmes de détection de contacts avec la géométrie

Une fois la géométrie importée dans Powder3D, un algorithme doit être utilisé pour la détection de contacts entre les sphères et les triangles des objets solides. Comme pour le cas des collisions entre particules, le calcul des forces résultantes suit cette détection et il y a sommation dans le même vecteur accumulateur. La détection des contacts entre les sphères et les triangles est sans aucun doute l'opération la plus intense. Si la géométrie est complexe et mobile, les temps de calcul peuvent rapidement rejoindre et dépasser ceux de la détection entre les particules, même si le nombre de contacts avec la géométrie est inférieur. Ces algorithmes méritent donc d'être approfondis et optimisés afin de limiter leur impact. Rappelons d'abord qu'il y a plusieurs types de contacts possibles entre les sphères et les triangles :

- contact avec la surface normale du triangle ;
- contact avec une des 3 arêtes du triangle;
- contact avec un des 3 sommets du triangle.

Puisqu'il y a jusqu'à 7 possibilités de contact, la résolution d'un contact est beaucoup plus dispendieuse que la simple comparaison de la distance entre deux centres de masse de particules voisines. Pour empêcher une dégradation des temps de calcul, il faut encore opter pour une approche ayant pour but de limiter les recherches exhaustives infructueuses.

Comme pour la localisation des sphères, le quadrillage dans lequel on place les triangles de l'espace est essentiel. Cependant, la taille des triangles n'étant pas toujours

constante ou proportionnelle à celle des particules, on ne peut pas se fier sur le barycentre du triangle pour la localisation. Des calculs supplémentaires sont requis pour bien identifier toutes les cases de la grille qu'un triangle touche. Cette grille devra elle aussi être rafraîchie, mais cette fois-ci en fonction de la vitesse des objets mobiles. Heureusement, dans le cas de géométries fixes, l'opération peut être effectuée une seule fois.

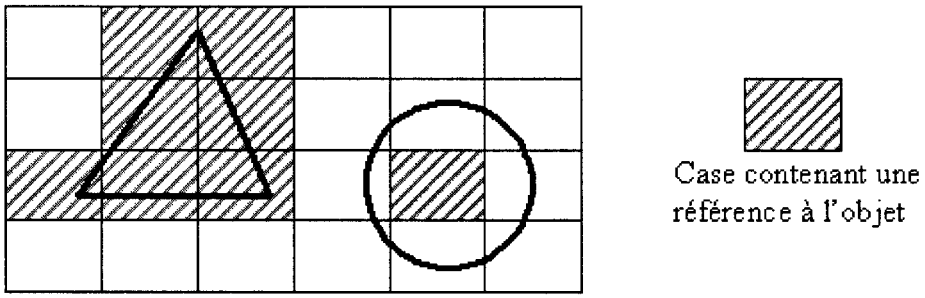


Figure 4-6 Comparaison entre l'espace utilisé dans une grille par une sphère et un triangle

L'étape de détection de contacts avec les murs de triangles se déroule comme pour le cas des sphères. On parcourt les particules et on recherche les triangles du voisinage tout en s'assurant de ne pas dédoubler les références puisqu'on peut les retrouver dans plus d'une case à la fois (Figure 4-6). Une recherche de type halo est encore applicable et essentielle pour réduire les cycles CPU qui seraient sinon gaspillés à vérifier des contacts entre des objets très éloignés. Lorsque deux objets sont assez proches, il ne reste plus qu'à vérifier qu'ils se touchent vraiment en appelant une fonction plus spécifique.

4.4.2 Détection du contact entre une sphère et un triangle

Cette partie traite d'algorithmes géométriques utilisés pour la détection des contacts entre les sphères et les triangles. Les informations connues sont la normale du

triangle et la position des deux objets. On peut retrouver les détails des algorithmes dans le premier rapport sur Powder3D (Gange, 2002).

D'abord et avant tout, on doit évaluer la distance entre le plan du triangle et la sphère. Pour trouver les deux points les plus rapprochés dans l'espace tridimensionnel, il suffit de projeter un des sommets du triangle et le centre de masse de la sphère sur la normale du triangle. La distance entre ces deux points doit être inférieure au rayon pour continuer. Une fois la première condition remplie, il reste à savoir s'il y a vraiment contact, plus précisément vérifier si le point de la sphère le plus proche du triangle est à l'intérieur de celui-ci (Figure 4-7). Il s'agit de la routine la plus souvent appelée et cumulativement une des plus coûteuses.

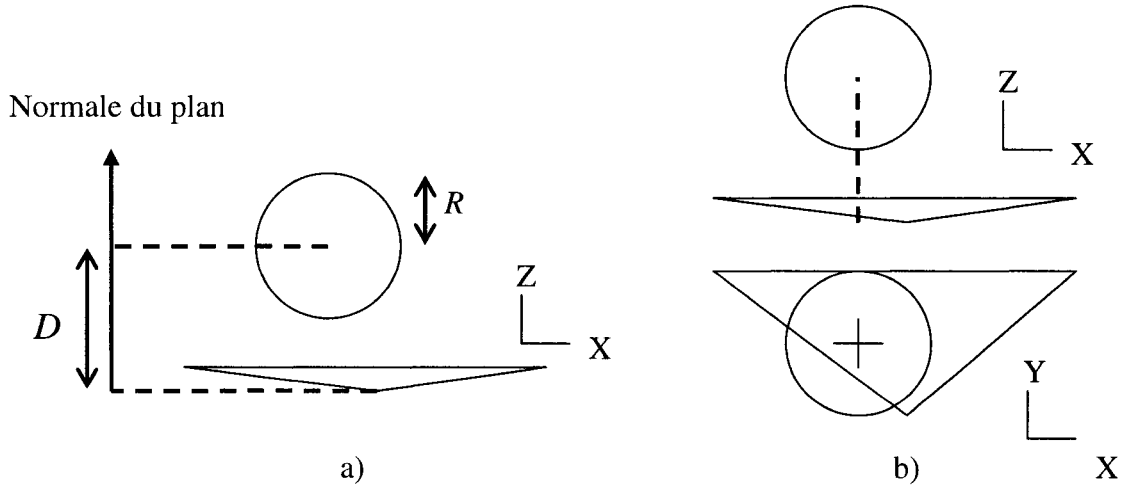


Figure 4-7 Opérations pour la détection de contacts entre les sphères et les murs

a) Projection sur la normale du triangle et vérification de la distance entre le plan et la sphère. La vérification continue au point b) si $D < R$

b) Vérification qui confirme si le point de contact est à l'intérieur du triangle

Finalement, lorsque le point de contact n'est pas inclus dans le triangle, on doit vérifier les contacts possibles avec les arêtes et les sommets. La détection de ces contacts étant plus simple, ces algorithmes ne seront pas approfondis dans cette section mais doivent être toutefois implantés avec attention. Pour optimiser la détection des contacts avec les murs, il faut donc réduire le temps passer dans la section la plus critique, où on

doit vérifier si le point de contact est à l'intérieur du triangle. Nous présentons trois algorithmes qui ont été implantés et testés afin d'obtenir les meilleures performances possibles.

4.4.2.1 Méthode de l'aire

La méthode de l'aire repose sur la sommation des aires des trois triangles formés à partir des sommets du triangle et du point de contact avec le plan de ce triangle. Il y a collision si la somme de ces aires est égale à l'aire totale de ce triangle. C'est la première technique implantée dans Powder3D. Elle nécessite principalement 4 produits vectoriels.

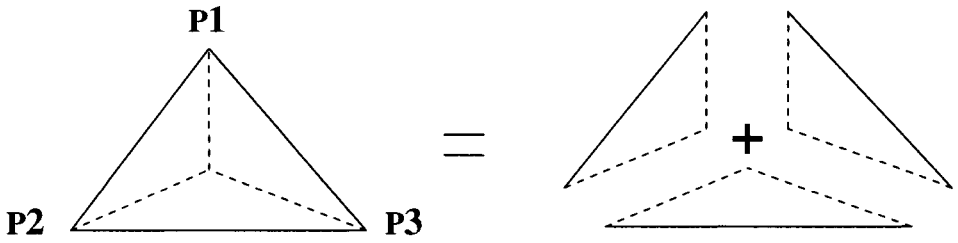


Figure 4-8 Méthode de l'aire pour la détection de contact avec des triangles

4.4.2.2 Méthode des angles

De la même façon, la méthode des angles valide un contact lorsque la somme des angles internes donne 180 degrés. Cette technique est classique mais nécessite des appels à des fonctions trigonométriques (arccos). Ces opérations sont dispendieuses, surtout pour une application qui l'est déjà beaucoup. N'offrant pas de meilleurs temps de résolution qu'avec la méthode de l'aire, elle fut mise de côté rapidement.

4.4.2.3 Méthode CCW

Les deux premières méthodes sont précises mais peu performantes parce qu'elles impliquent trop d'opérations. Une recherche sur les algorithmes géométriques nous a permis de découvrir une nouvelle technique de fonctionnement assez simple. Elle est basée sur le fait qu'en parcourant les arêtes d'un objet convexe, il y a contact seulement si ce point est toujours du même coté de chaque arête (Figure 4-10). L'algorithme CCW (Counter Clock Wise), présenté dans un ouvrage classique de l'algorithmique de Sedgewick (1990), a pour but premier de vérifier si deux droites se croisent sur un plan 2D. Pour pouvoir utiliser cet algorithme, les coordonnées du point de contact et des sommets doivent être projetés sur le plan 2D le plus approprié, selon la normale du triangle (Figure 4-9).

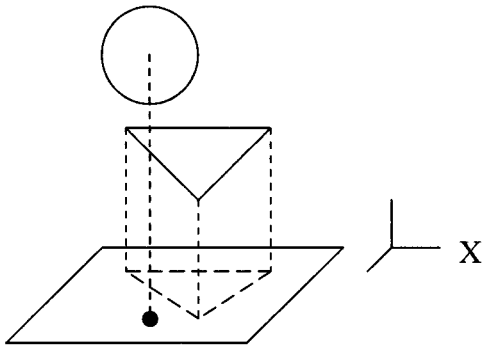
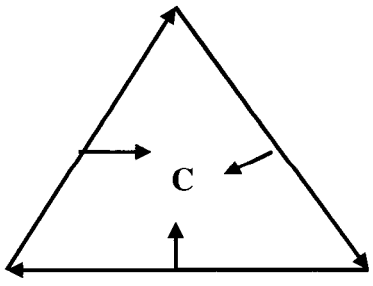


Figure 4-9 Projection du triangle sur un plan 2D



Fonction **ccw**

Retourne -1 si C est à gauche
Retourne 0 si C est sur l'arête
Retourne 1 si C est à droite

Contact **si et seulement si**
les appels retournent tous ≤ 0
ou tous ≥ 0

Figure 4-10 Méthode CCW pour détection de contact avec des triangles

Ensuite, en parcourant les côtés de l'objet convexe, il reste à vérifier si la projection du point se trouve toujours du même côté de chaque arête. En s'inspirant de cet algorithme, on a obtenu des résultats remarquables. L'algorithme CCW arrive à la même conclusion que la méthode des aires ou la méthode des angles, mais plus rapidement (Tableau 4-3). Précisons que sauver sur la détection sphère-triangle rend aussi le remplissage de la grille plus économique. La méthode CCW est plus performante que la méthode de l'aire parce qu'on fait d'abord une simplification (projection 2D).

Tableau 4-3 Comparaison des méthodes de détection avec un triangle (Sédimentation de particules dans une cuve sur Magnum)

Méthode	Temps total (s)	Temps de détection (s)	Temps pour placer dans le quadrillage (s)
Aire	437	265	11
Angles	449	273	12
CCW	260	112	6

4.4.3 Extension du modèle pour la génération des murs

À la lumière des derniers résultats, la détection des contacts entre les triangles de la géométrie et les particules sphériques est coûteuse. Évidemment, d'autres tentatives d'optimisation pourraient peut-être réduire les temps de calcul davantage mais une chose est claire : ce problème ne sera jamais aussi simple que celui de la détection des collisions entre sphères. Et c'est en essayant de réduire la complexité de la détection à celle des sphères que l'idée de combiner ces dernières pour créer des murs nous est venue. Cette approche n'est pas nouvelle; pour Gallas et Sokolowski (1993) et Mattutis et al. (2000), la définition de la géométrie à l'aide de cercles est mise à contribution pour

générer des murs (plus rugueux, certes), dans leur simulation 2D. Nous avons donc repris cette idée en 3D, en l'appliquant à nos géométries définies par un maillage de surface.

Le défi principal était de remplacer tous les triangles par un ensemble de sphères sans créer de trou et ce, peu importe la taille de maille. La façon de procéder est fastidieuse mais, heureusement, ce travail ne doit être fait qu'une seule fois. On remplit récursivement l'aire du triangle en se basant sur son barycentre et ses sommets. Le niveau de rugosité est inversement proportionnel au temps et à l'espace requis par l'ajout de ces sphères. Les particules générées sont immuables à moins qu'elles appartiennent à un objet solide mobile dont elles reçoivent les propriétés. Évidemment, aucune inter-détection n'est nécessaire car elles ont un statut spécial et une structure distincte des vraies particules.

Le remplissage de chaque triangle du maillage est réalisé grâce à un algorithme simple paramétrable selon trois variables :

1. Ratio : Rayon des sphères des murs / Rayon de la plus petite sphère
2. Écart1 : Écart entre les sphères alignées entre deux sommets
3. Écart2 : Écart entre deux sphères alignées selon l'axe sommet-barycentre

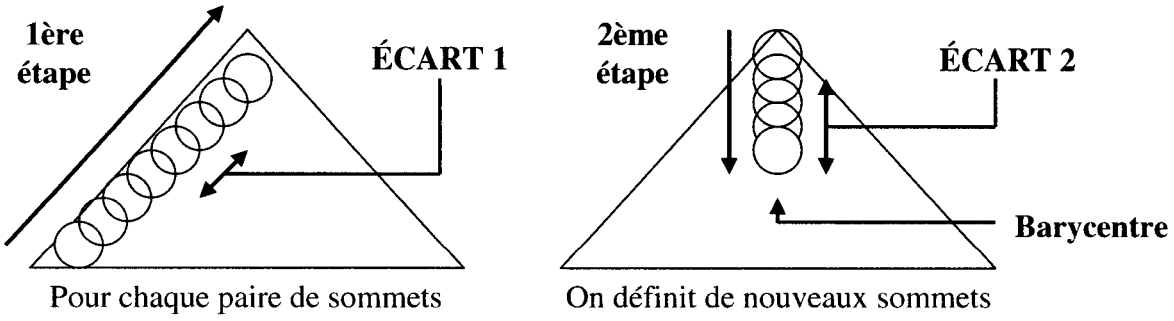


Figure 4-11 Technique implantée pour remplir la surface des triangles

Le remplissage se fait parallèlement aux arêtes comme il est représenté à la figure 4-11. Ensuite, on répète cette procédure en se déplaçant vers le barycentre. On boucle de cette façon tant que le barycentre n'est pas atteint. À la fin, le triangle est rempli de sphères (Figure 4-12).

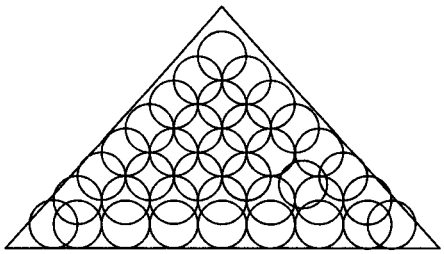


Figure 4-12 Exemple du remplissage d'un triangle à l'aide de sphères

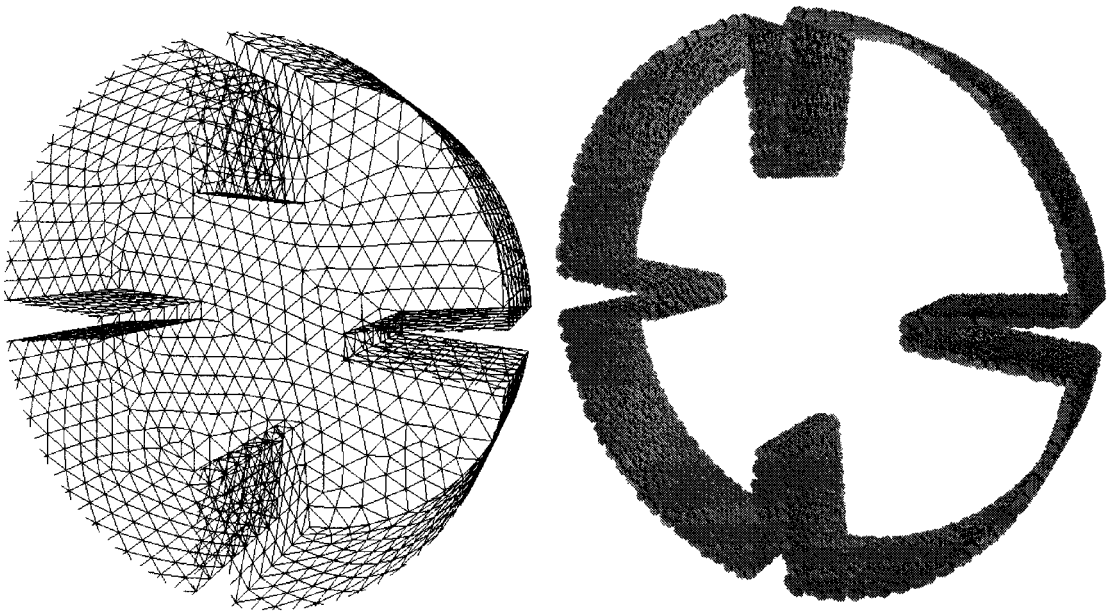


Figure 4-13 Exemple du remplissage des triangles pour un « Tumbling mill »

En regardant la figure 4-13 et le nombre de sphères généré pour chaque triangle, il est normal de se demander si cette façon de procéder est avantageuse. Quels sont les réels avantages d'une telle représentation des géométries? Voici les principaux :

- La simplicité avec laquelle on peut placer les sphères dans la grille de localisation;
- La possibilité d'utiliser des algorithmes de détection de contacts entre sphères qui sont déjà implantés;
- Un seul test de collision (au lieu de 7).

Les principaux inconvénients sont :

- Le nombre de particules supplémentaires à l'intérieur de chaque triangle;
- L'espace mémoire requis pour contenir toute cette information.

4.4.4 Évaluation et comparaison

Nous avons évalué les techniques présentées pour représenter un mur à l'aide d'un test comportant tous les attributs d'une simulation complexe. Ce test consiste à déposer 1900 sphères dans une cuve qui tourne. Ce test est idéal puisqu'il comporte :

- Peu de contacts inter-particulaires;
- Une fréquence de rafraîchissement très élevée;
- Un objet mobile.

a) b)

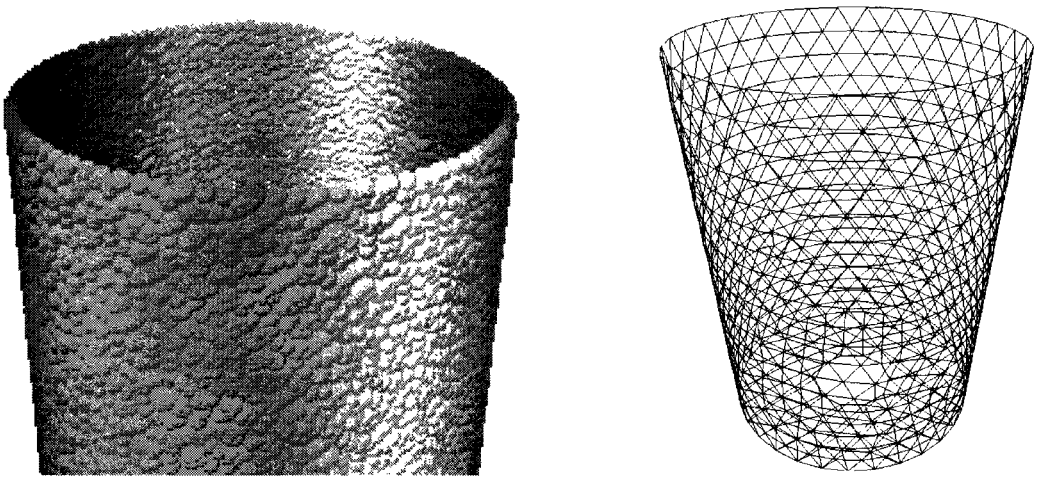


Figure 4-14 Représentation de la cuve à l'aide de a) sphères et b) d'un maillage de triangles

Tableau 4-4 Comparaison des méthodes de représentation des murs
Conditions de simulation : 1900 particules, 950 contacts particules-mur (4750 itérations)

Méthode		Contacts inter-particulaires	Temps (s)
Aire	(2700 triangles)	1011	704
CCW	(2700 triangles)	1011	425
Sphères	(28000 sphères)	2006	286

Malgré le nombre de sphères supplémentaires nécessaire pour composer les murs, les résultats sont plutôt surprenants (Tableau 4-4), si on considère qu'il y a deux fois plus de contacts dans ce cas (2006) comparativement à la représentation à l'aide d'un maillage de triangles. On en conclut que placer des triangles dans la grille est trop coûteux et tout simplement trop de temps est passé à vérifier les contacts probables avec les arêtes et les sommets de ces triangles. Donc, les temps de calcul sont réduits lorsqu'on utilise les sphères pour représenter les murs et ce, suffisamment pour conclure

qu'elle devrait être plus rapide dans la plupart des systèmes particulaires. Le Tableau 4-5 montre les coûts en fonction de la rugosité définie par les trois paramètres pour le test de la cuve qui tourne.

Ces résultats mettent en lumière quelques propriétés intéressantes de la nouvelle méthode. Si on désire diminuer les aspérités, il vaut mieux augmenter le chevauchement que d'augmenter le rapport de taille des rayons. La réduction du rayon fait ressortir les défauts de l'algorithme de détection qui semble moins performant avec un mélange polydispersé. Remarquons que si le nombre de sphères est quintuplé, il n'y a pas nécessairement plus de contacts avec les murs (~2000); il y a seulement plus de tests inutiles. Générer une surface beaucoup plus lisse est évidemment plus coûteux, toutefois son niveau d'efficacité est comparable aux méthodes basées sur les triangles.

Tableau 4-5 Influence du niveau de rugosité sur les temps de calculs

Ratio Rmur/Rmin	Écart (1 et 2)	Nombres de sphères pour les murs	Temps (s)
4/5	2.0	28000	286
4/5	1.0	76200	351
3/5	2.0	46425	340
3/5	1.5	62000	360
3/5	1.0	122000	414
2/5	1.5	129000	455

4.5 *Résumé des performances des nouveaux algorithmes de détection de contacts*

L'accélération des calculs de la méthode des éléments discrets devait d'abord passer par l'accélération du temps d'exécution en séquentiel. La détection peut être mise aux oubliettes car la section la plus contraignante est la résolution de contacts. Le

Tableau 4-6 compare pour des cas pratiques notre version optimisée au cours de l'été 2003 et le code hérité un an auparavant (Gange, 2002). On remarque les importantes diminutions de temps de calcul réalisées grâce aux optimisations apportées aux algorithmes de détection. Il est impossible de faire ressortir un facteur commun d'accélération, toutefois le constat est assez évident : la détection des collisions n'est plus problématique.

Tableau 4-6 Comparaison des performances des nouveaux algorithmes de détection

Description de la simulation	Nombre de Particules	Été 2002 Temps (s)	Été 2003 Temps (s)
Sédimentation de pigments monodispersés	300	29	0,3
Sédimentation de pigments bidispersés (ratio 5)	3000	103	7
Sédimentation de pigments bidispersés (ratio 5)	9000	261	20
Objet solide (ruban hélicoïdal + cuve)	9000	1380	31

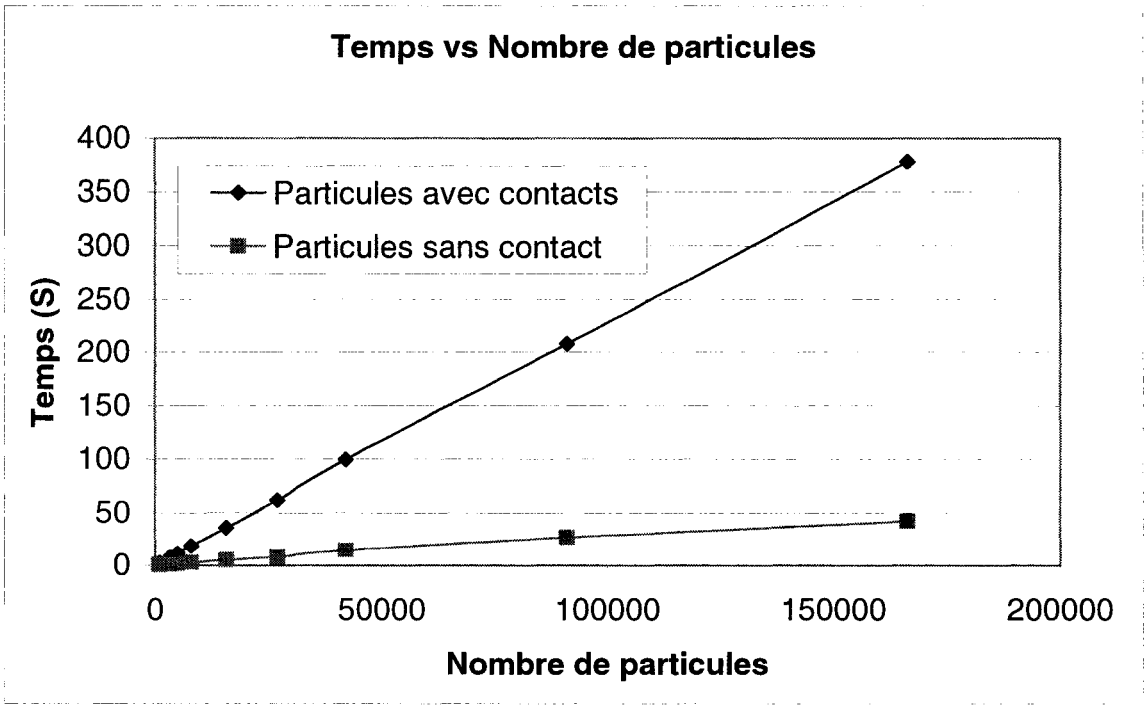


Figure 4-15 Évolutivité des temps de calcul en fonction du nombre de particules

Le graphique de la figure 4-15 montre que le calcul des forces est devenu la section dominante. Les temps de simulation varient linéairement en fonction du nombre de collisions. Malheureusement, même après toutes les optimisations présentées dans ce chapitre, les temps requis sont toujours astronomiques pour des problèmes le moindrement complexes. Par conséquent, les techniques reliées au calcul de haute performance, plus précisément la parallélisation, seront indispensables pour diminuer ces temps de calcul.

5 Parallélisation de la DEM

Ce chapitre ainsi que les prochains seront consacrés à la parallélisation de la DEM et ses sujets connexes. La parallélisation est séduisante mais son implantation l'est moins. Pour obtenir des performances hors du commun, il est préférable de compter au préalable sur un code séquentiel optimal et c'est ce que nous avons fait dans un premier temps et présenté dans le chapitre précédent. Ensuite, pour espérer une diminution de temps supplémentaire, le travail en parallèle devient inévitable. Notre support informatique étant les grappes de calcul, la décomposition de domaine permet de distribuer le calcul inhérent aux sous-domaines sur plusieurs processeurs en mémoire distribuée.

Précisons que l'accélération générée en créant ces sous-ensembles ne peut plus provenir d'algorithmes non performants qui pourraient tirer avantage de cette décomposition. Dans le cas de Powder3D, tous les algorithmes implantés sont de complexité linéaire; l'accélération sera donc, dans le meilleur des cas, linéaire avec le nombre de processeurs utilisés.

Pour garantir la validité de la simulation, la parallélisation en mémoire distribuée requiert à toutes les itérations l'échange de bandes de particules fantômes communément appelées les halos (*ghost cells*). Puisque les processus doivent communiquer fréquemment, l'utilisation d'une bibliothèque de communications est requise. Dans ce travail, le standard MPI a été choisi comme outil de parallélisation.

L'implantation choisie pour le développement de Powder3D est Mpich. Gratuit, il peut être couplé facilement au compilateur Fortran 90 d'Intel, le compilateur de Powder3D sur Linux (www.intel.com/developper). Même si le standard MPI propose plus d'une centaine de routines, un adage dit que seulement 6 fonctions sont nécessaires pour paralléliser une application. La description des fonctions MPI utilisées débute cet

important chapitre. Ensuite, nous aborderons la technique de décomposition de domaine appliquée à la méthode des éléments discrets. Suivront l'analyse des performances de notre implantation et quelques tentatives d'optimisation.

5.1 Fonctions basiques du standard MPI

Pour les programmes SPMD et plus particulièrement ceux qui sont basés sur la bibliothèque MPI, on doit initialiser une zone parallèle dans laquelle les échanges auront lieu. La création du communicateur principal *MPI_COMM_WORLD* permet d'abord d'informer les processus du groupe de travail et de leur attribuer un identificateur personnel unique. Pour la décomposition de domaine, le nombre de sous-domaines correspond au nombre de processeurs dédiés pour la simulation, soit la variable *numprocs*. Les fonctions requises sont :

MPI_INIT() / Initialisation du communicateur (MPI_COMM_WORLD) */*
MPI_COMM_RANK(communicateur, id)
MPI_COMM_SIZE(communicateur , numprocs)
MPI_FINALIZE() / Destruction du communicateur à la fin du programme */*

Les communications « point à point » peuvent alors avoir lieu. Évidemment, elles impliquent deux processus : un qui envoie et l'autre qui doit s'attendre à recevoir. Les fonctions de base décrites plus bas sont *MPI_SEND* et *MPI_RECV*. Leurs versions non bloquantes sont similaires et ont l'avantage d'éviter les interblocages :

MPI_SEND(vecteur d'envoi, quantité, TYPE, destinataire, TAG, communicateur)
MPI_RECV(vecteur de réception, quantité, TYPE, source, TAG, communicateur, status)

Ces fonctions MPI supportent par défaut les types de données conventionnels. Cependant, il est possible d'ajouter des types plus spécifiques à une application. Dans le cas de Powder3D, *MPI_DATA_TYPE* a permis de créer des vecteurs d'envoi et de réception de type *particule*. Les éléments du vecteur contiennent environ une douzaine de nombres réels, selon la complexité du modèle. Il n'est pas nécessaire de prévoir l'empaquetage et le dépaquetage des données et on diminue le nombre de messages à envoyer. Plusieurs autres types ont été créés de la même façon afin de transférer plus facilement par exemple les données concernant les forces et les contacts :

MPI_DATA_TYPE(nom du nouveau type, nombre d'éléments, type d'éléments)

En plus des communications directes, Mpich offre de nombreuses fonctions de communications collectives. Ces dernières sont très efficaces pour répandre une information rapidement ou mettre les processeurs en accord. Par exemple, lorsqu'on veut obtenir la vitesse maximale des particules afin de vérifier la stabilité de la simulation, il suffit de faire une réduction :

MPI_ALL_REDUCE(valeur locale, valeur globale, opération (MAX), communicateur)

Enfin, MPI offre un ensemble d'outils pratiques permettant de résoudre des problèmes classiques relatifs à la topologie de la décomposition de domaine (Figure 5-1) :

MPI_CART_CREATE(communicateur, nb de dimension, topologie,
condition périodique, nouveau communicateur)
MPI_CART_GET(nouveau communicateur de topologie, nb de dimension,
condition périodique, coordonnées du sous-domaine)
MPI_SHIFT(nouveau communicateur de topologie,1, axe, voisin1, voisin2)

Tous les processus font appel à *MPI_CART_CREATE* pour la création d'une topologie fondée sur la décomposition cartésienne. Par la suite, ils font référence à une

image globale de la décomposition du domaine en trois dimensions via le nouveau communicateur. Avant de définir les frontières de son sous-domaine, chaque processus doit trouver sa position dans la grille de processeurs grâce à la fonction *MPI_CART_GET*. De la même façon, l'identification des six voisins potentiels se fait grâce à *MPI_SHIFT* qui supporte les conditions périodiques. Le lecteur pourra consulter *Using MPI (Gropp,1999)* pour plus de détails.

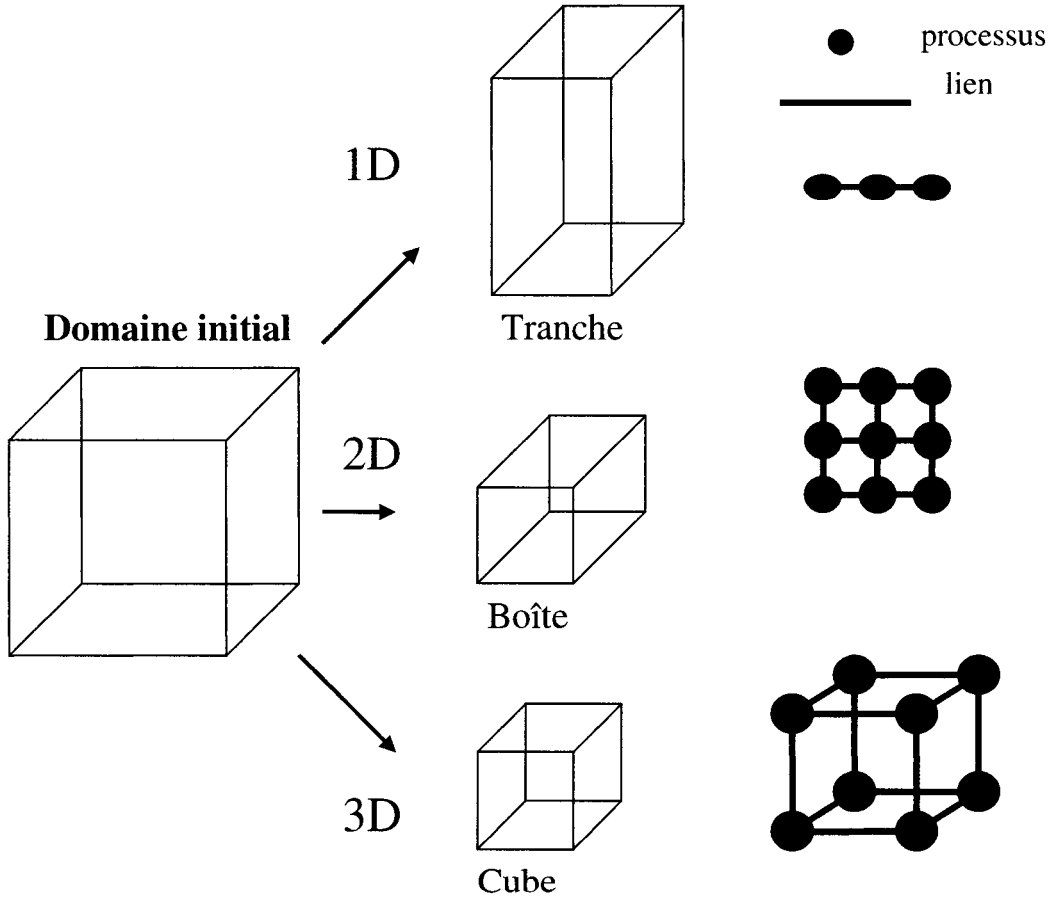


Figure 5-1 Décompositions possibles et liens entre les processus voisins

5.2 La décomposition du domaine

La parallélisation de la méthode des éléments discrets par décomposition de domaine débute par la décomposition initiale du domaine de calcul. Cette dernière pourra être conservée durant toute la simulation ou être modifiée suite à un rééquilibrage dynamique de la charge (chapitre 6). Pour les méthodes particulaires, une décomposition cartésienne de l'espace est recommandée. On divise tout simplement le domaine en régions rectangulaires en fonction du nombre de processeurs disponibles. Il n'est pas obligatoire que le quadrillage de l'espace concorde avec le quadrillage utilisé pour la détection des contacts; cela dépend de l'implantation. Les sous-domaines créés contiennent toutes les particules situées à l'intérieur de leurs frontières respectives. Un mécanisme est toutefois nécessaire pour recréer les véritables conditions de simulation.

5.3 Halos et transferts de particules

La technique du halo permet de simuler correctement le comportement des particules d'un sous-domaine. Pour calculer les forces agissant sur une particule à un temps donné, il faut connaître la position de ses voisines à l'itération précédente. Si la particule voisine appartient à un sous-domaine voisin, c'est le processus correspondant à ce sous-domaine voisin qui est chargé de transmettre l'information. Les particules reçues dites « fantômes » correspondent aux particules frontalières des sous-domaines voisins. À chaque itération, tous les processeurs procèdent à l'envoi de leurs particules frontalières et à la réception des particules fantômes. Afin que tous les contacts soient pris en compte pour garder la cohérence avec la méthode de détection, la largeur de la région partagée est proportionnelle au rayon maximal des particules, plus précisément 2,1*Rmax (Figure 5-2).

Les particules fantômes sont ajoutées à la fin du vecteur de particules réelles qui contient les particules exclusives et les particules frontalières. On fait référence à une particule en se basant sur sa position dans ce vecteur, qui correspond à la numérotation locale. Une particule frontalière se retrouve sur plusieurs processus et son numéro local peut ainsi varier. Pour différencier chaque particule, une numérotation globale est aussi utilisée; chaque particule traîne son numéro unique d'identification.

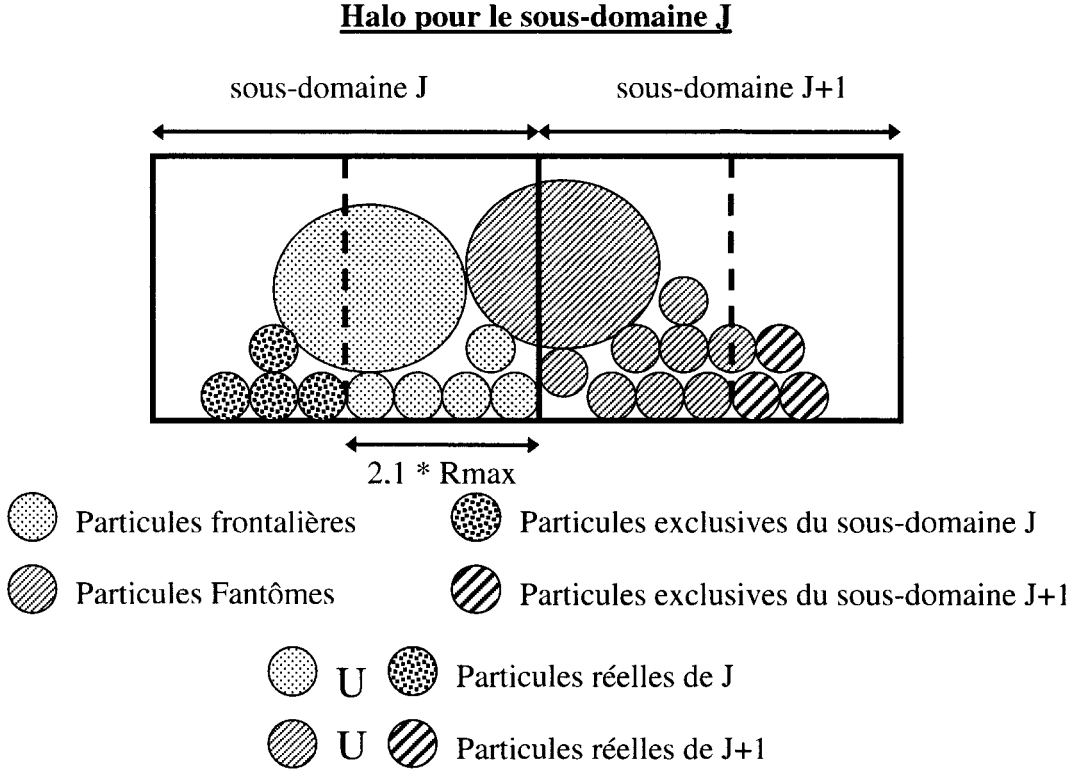


Figure 5-2 Halo du point de vue d'un sous-domaine

Au cours d'une simulation, une sphère en mouvement peut dépasser les frontières de son sous-domaine. Une vérification est donc nécessaire pour détecter ce déplacement et procéder au changement de sous-domaine. Précisons ici que c'est toujours le processus propriétaire qui décide de se départir d'une particule et c'est lui qui en informe le processus voisin.

5.4 Description de l'implantation

Tel que mentionné dans la section précédente, un processus doit envoyer à ses voisins deux informations :

- les particules quittant définitivement le sous-domaine (les particules léguées);
- l'information des particules frontalières.

Les premières tentatives de parallélisation par décomposition de domaine ont permis de confirmer que l'envoi de deux messages ralentissait l'exécution. En effet, la latence réseau inhérente à chaque message étant relativement grande, il est avantageux de limiter le nombre de messages en augmentant la taille des vecteurs transmis. Une version plus complète de la loi d'Amdahl précise d'ailleurs que les communications peuvent nuire à l'accélération :

$$A = \frac{T1}{T(P)} = \frac{T1}{\left[\frac{(T1 \times \alpha)}{P} + (1-\alpha) \times T1\right] + TComm + TCasParticuliers} \qquad \textbf{Équation 5.1}$$

où *TComm* équivaut au temps requis pour les communications comprenant la latence et le transfert des données. Nous incluons aussi le temps écoulé pendant la construction du message. Le traitement des cas particuliers (*TCasParticuliers*) regroupent toutes les autres opérations supplémentaires qu'un programme parallèle exécute mais que le programme séquentiel ne requiert pas. Par exemple, le mécanisme de rééquilibrage de la tâche (chapitre 6) comporte des coûts que l'on retrouve seulement lors de l'exécution parallèle. Les communications sont en fait un cas particulier unique à la parallélisation, mais nous les comptabilisons à part.

5.5 Définition du patron de communication

Pour obtenir les meilleures performances, la parallélisation doit être efficace. Les cartes Ethernet 100Mbits/s n'ayant pas des caractéristiques exceptionnelles, il est impératif de réduire le temps des communications afin d'améliorer l'efficacité. Les prochaines sous-sections feront la description de notre implantation et présenteront diverses tentatives d'optimisation.

5.5.1 Fusion des messages transmis

En réalisant que le contenu des 2 messages décrits précédemment demeurait de même nature, des structures de type *particule*, nous avons eu l'idée de fusionner les deux messages. Nous pouvions alors espérer un gain relatif à la latence et la synchronisation du deuxième message qui était éliminé. Les résultats obtenus avec cette première version parallèle montre que la parallélisation de la DEM est naturelle et efficace. Comme on peut le remarquer à la figure 5-3, une accélération acceptable jusqu'à 6 processeurs a été obtenue lorsque les algorithmes de détection n'étaient pas optimaux.

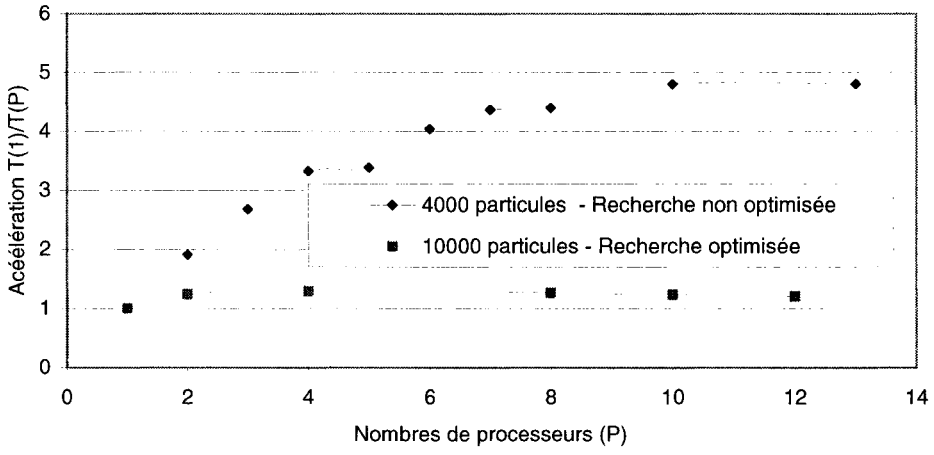


Figure 5-3 Courbes d'accélération pour la première implantation avec messages fusionnés (Sédimentation, Magnum)

Les raisons pour lesquelles la première courbe fléchit sont d'abord, le coût grandissant des communications par rapport aux calculs. Ensuite, avec l'utilisation de 10 processeurs, il y a presque deux fois plus de particules à gérer qu'en séquentiel, conséquence directe de l'information dédoublée par les halos. Toutes les particules des sous-domaines sont frontalières; chaque processeur envoie tout son sous-domaine à ses voisins. Il est donc impossible d'espérer de meilleures performances quand les problèmes sont trop petits.

Malheureusement, le temps de résolution en parallèle avec cette version était encore trop long. Nous sommes retournés à l'élaboration de nouveaux algorithmes de recherche tels que ceux présentés au chapitre 4. Une fois les algorithmes de détection optimisés, les performances de l'exécution parallèle sont devenues particulièrement décevantes (Figure 5-3). Ces résultats ont amené à la remise en question de notre politique d'échange. Éliminer une communication aurait pu être efficace mais, après expérimentation, nous avons réalisé que la technique souffre d'un inconvénient majeur : la construction du message à envoyer. En effet, il faut rebâtir complètement un nouveau message à chaque itération puisqu'on ne peut léguer deux fois la même particule. Il faut alors porter une attention particulière à l'ordre des particules qui peut changer et créer des conflits au niveau de la numérotation locale et les grilles de localisation.

Pour limiter le coût des communications et accélérer les performances, nous avons tenté de réduire la fréquence des envois. Cela peut paraître absurde car l'information des halos n'est plus transférée systématiquement à chaque itération par les processus propriétaires. En procédant ainsi, dès la deuxième itération, il y aura à la frontière de petites variations par rapport à ce qui se passe lorsque les halos sont rafraîchis à toutes les itérations. Ces variations se propageront lentement vers l'intérieur du sous-domaine et les coordonnées de toutes les particules divergeront de celles obtenues en séquentiel. Les problèmes de différences entre les versions séquentielle et parallèle seront discutés plus en détail à la section 7.9.

5.5.2 Deuxième tentative

Certains articles sur la décomposition de domaine mentionnent que le legs de particules réelles est un phénomène fréquent mais pas assez pour s'en préoccuper à chaque itération (Henty, 2000). Par conséquent, il est souvent proposé de reporter l'envoi des messages concernant les particules qui changent de sous-domaine. Le but est de minimiser le coût de transmission de ce message. Transférer des particules réelles modifie la numérotation locale donc, on doit en profiter pour synchroniser l'opération de transfert avec celle du rafraîchissement des grilles de localisation. De plus, grâce à cette technique, on évite de nombreux tests répétitifs lors de la construction des messages. Puisque la numérotation locale doit demeurer cohérente avec les grilles de localisation, les messages contenant les particules fantômes doivent rester les mêmes pendant plusieurs itérations. Au lieu de parcourir le vecteur de particules pour rechercher à chaque fois les candidats faisant partie du halo, on peut stocker les numéros locaux dans les tables prévues à cet effet. Tant que ces tables sont valides, on peut les utiliser pour accélérer la construction des messages de particules fantômes.

Au besoin, c'est-à-dire lors du rafraîchissement des structures de localisation, chaque processeur boucle sur l'ensemble de ses particules réelles, rebâtit une nouvelle table et communique (voir pseudo-code).

Pseudo code pour la politique d'échange (effectué au début de chaque itération)

// 1er message : traitement des particules changeant de sous-domaine
Si requis (taux de rafraîchissement)
    Pour chaque frontière avec un sous-domaine voisin
        Détermination des particules à léguer
        Envoi des particules transférées au sous-domaine voisin
        Réception des particules léguées par le sous-domaine voisin

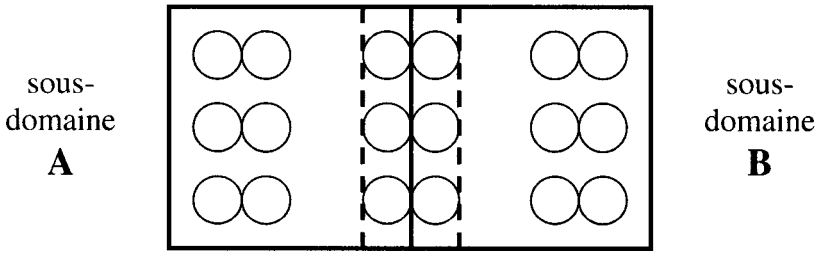
// 2ème message : envoi et réception des particules faisant partie des halos
Pour chaque frontière avec un sous-domaine voisin
    Si requis (taux de rafraîchissement)
        Mise à jour de la table de transmission des particules frontalières
        Construction du message (en fonction de la table des particules frontalières)
        Envoi des particules frontalières
        Réception des particules fantômes
    Sinon
        Construction du message (en fonction de la table des particules frontalières)
        Envoi des particules frontalières
        Réception des particules fantômes

Comme on le remarque plus haut, ce nouvel algorithme permet une gestion plus aisée des particules léguées et son extension pour les décompositions 3D est évidente. Enfin, grâce à ces meilleures performances, il n'est plus nécessaire de diminuer la fréquence des communications, et on évite de générer des différences entre l'exécution de la version séquentielle et la version parallèle.

5.5.3 Décomposition de domaine multidimensionnelle

La décomposition de domaine et l'utilisation des halos impliquent nécessairement un dédoublement de certaines données. Nous allons montrer qu'une décomposition de domaine selon plusieurs axes peut être envisagée et profitable. L'objectif principal est de limiter la quantité de données partagée et la taille des messages à transmettre. De plus, il y aura moins d'opérations relatives aux particules fantômes et les dédoublements du calcul des forces seront limités. En effet, pour réduire

de moitié le temps consacré aux calculs des forces, le principe d'action-réaction est exploité. Cependant, avec la décomposition de domaine, le calcul des forces de contact entre les particules fantômes et particules frontalières ne peut tirer avantage de la troisième loi de Newton. Les opérations sont effectuées en double sur chaque processeur. Lorsque le rapport entre les particules frontalières et le nombre total de particules réelles est trop élevé, l'efficacité de la parallélisation demeure très faible (Figure 5-4).



Nombre de contacts pour l'exécution séquentielle : **9**

Nombre de contacts pour le sous-domaine A: **6**

Nombre de contacts pour le sous-domaine B: **6**

Efficacité maximale de la parallélisation pour cet exemple : (9/6)/2 = 75%

Figure 5-4 Influence du dédoublement des calculs sur la parallélisation

Pour des décompositions rectangulaires cartésiennes, la forme du sous-domaine optimale est le carré en 2D et le cube en 3D. En effet, ce sont des formes géométriques offrant le meilleur rapport entre l'aire (volume) et le périmètre (surface). Il s'agit donc du meilleur rapport entre le nombre de particules locales et celui des particules frontalières. S'il y a moins de particules fantômes, c'est moins de tests pour les placer dans la grille de localisation ou pour vérifier les contacts potentiels. Le dédoublement des calculs peut être réduit significativement avec la décomposition de domaine sur plus d'une dimension à condition que le domaine s'y prête bien. Dans le pire des cas, une

décomposition d'un domaine en forme de cube, la quantité de données à transmettre peut prendre diverses valeurs (Figure 5-5).

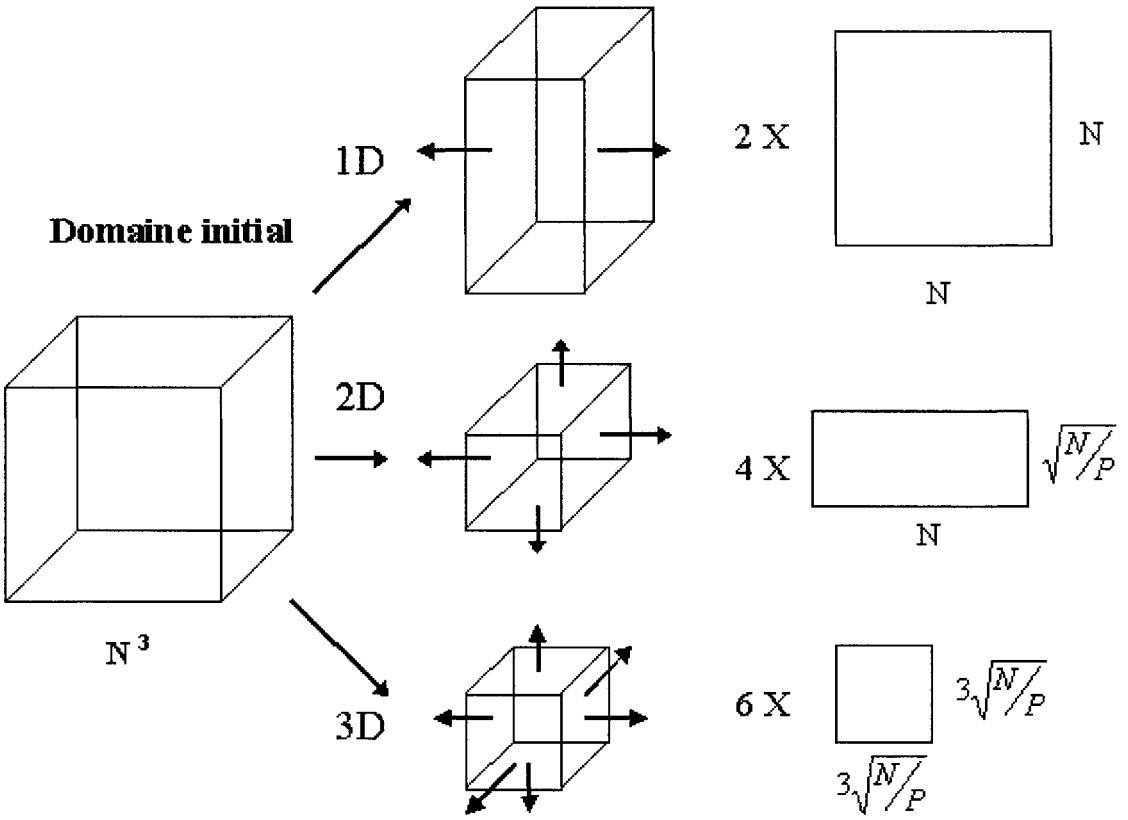


Figure 5-5 Comparaison de la quantité de données à tranferérer selon la décomposition avec *P* processeurs

Il s'agit ici de nombres indicateurs proches de la borne maximale puisque la quantité exacte est variable car elle dépend de la disposition, de la concentration et de la taille des particules.

Dans la littérature, bien qu'on ait essayé de décomposer selon plusieurs axes, on conseille souvent de demeurer avec la décomposition 1D (Knecht, 1995), appelée décomposition par bandes (strip decomposition). Souvent le patron de communication

devient trop complexe lorsqu'on ajoute des voisins, ce qui dégrade les performances et rend la décomposition multidimensionnelle à peine plus efficace que la décomposition par bandes.

Nous savons que les temps de communication dépendent principalement du nombre de messages et de la quantité de données à transférer. Pour une décomposition 3D, un processeur possède trois fois plus de voisins comparativement à une décomposition 1D. Si on fait abstraction de l'optimisation des communications point-à-point, le temps de communication est donné par (équation 5.3) :

$$Tcomm = Tlatence + Taille * Tdébit \quad \textbf{Équation 5.2}$$

$$Tcomm\ Total = Nb\ de\ voisins * (\ Tlatence + Taille\ moyenne * Tdébit) \quad \textbf{Équation 5.3}$$

On voit donc que la latence a intérêt à être négligeable par rapport au temps de transfert. La parallélisation par décomposition de domaine étant caractérisée par une granularité assez grossière, la latence a donc relativement peu d'impact sur les performances (Dimitrov et Skjellum, 2003). Ainsi, la quantité de données (*Taille moyenne*) doit être suffisamment réduite pour remarquer une amélioration inhérente à la décomposition multidimensionnelle. Nous devons donc compter sur un patron de communication peu complexe et efficace.

La transmission des halos peut être implantée de deux façons : de point à point avec des communications non bloquantes ou en bloc selon la méthode *SHIFT*. Ces deux approches seront maintenant présentées avant d'analyser les performances de la décomposition multidimensionnelle.

5.5.4 Communications non bloquantes

La stratégie de communication la plus simple est la communication directe entre un sous-domaine et ses voisins directs et diagonaux. Malheureusement, dans une telle

situation, il y a beaucoup de voisins et beaucoup de messages; les risques d'interblocage sont donc très élevés et un patron de communication utilisant seulement les communications non bloquantes est privilégié. Un interblocage se produit lorsque la politique d'échange est mal définie et en mode bloquant. Si deux processus en communication tentent d'effectuer la même opération, par exemple de recevoir de l'autre, leur attente ne trouvera jamais preneur. La simulation se terminera car il n'y a pas de moyen de sortir de cette impasse.

Grâce aux routines *MPI_ISEND* et *MPI_IRECV,* il n'est plus nécessaire de spécifier l'ordre d'envoi. En terminant l'étape de communication par une barrière d'attente *MPI_WAIT*, on s'assure que tous les messages ont été transmis. De plus, on peut espérer de meilleurs temps de transfert. Les communications non bloquantes permettent de prendre de l'avance et d'effectuer les échanges de données en rafale pendant que le processus le plus lent termine ses calculs. Enfin, en mode non bloquant, les cartes réseau qui supportent la transmission *full duplex* peuvent théoriquement atteindre un débit de 200Mbits/s.

Dans notre version non bloquante, on doit allouer autant de vecteurs de réception et d'envoi que le nombre de voisins car les échanges sont tous indépendants. En 2D (Figure 5-6), à chaque itération, {2X8}16 messages sont prévus et peuvent être transmis au même moment. Ainsi, le niveau de performance pourrait être rehaussé car aucune barrière implicite ne synchronise les envois. Malheureusement, cela ne s'est jamais concrétisé en pratique. Les messages diagonaux ne font qu'empirer le trafic et augmentent les temps de communication de 60%. L'implantation 3D avec ses 26 voisins et ses 52 vecteurs d'échanges par processus devrait empirer ce phénomène. Une méthode plus performante en 2D et en 3D était requise.

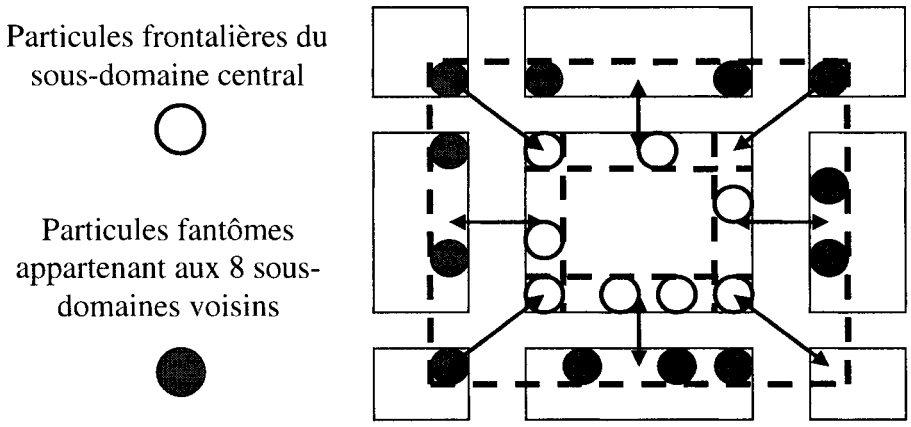


Figure 5-6 Échanges nécessaires pour une décomposition 2D

5.5.5 Le patron de communication *SHIFT*

Le patron de communication SHIFT est une stratégie mentionnée par Clark (1994) pour transférer efficacement les données entre plusieurs sous-domaines. Les communications se font par blocs selon chaque axe de décomposition. Son avantage principal est de rendre la construction des messages moins complexe. Le *SHIFT* permet de transmettre l'information chez le deuxième voisin dans une direction de façon indirecte et en limitant le nombre de messages. À la figure 5-6 qui schématise une décomposition 2D, les particules frontalières doivent être transmises aux 8 voisins pour que les contacts aux coins soient bien pris en compte. Avec le *SHIFT,* seulement 4 messages sont nécessaires :

- 2 messages dans la première direction
- 2 messages dans la deuxième direction

Le processus débute par la sélection d'un axe de décomposition afin de construire les deux messages de particules frontalières à envoyer. Le processus peut alors recevoir de la gauche et de la droite et concaténer les particules fantômes à la fin de

son vecteur de particules. La construction du halo dans la deuxième dimension peut commencer mais il faut boucler jusqu'à la fin du vecteur pour tenir compte des particules fantômes du premier halo. Cela permet de faire des déplacements vers les sous-domaines diagonaux en deux étapes. L'implantation en 3D est évidente; il faut construire les halos en tenant compte des 4 messages précédemment reçus. Comme mentionné plus haut, il est conseillé de conserver les particules sélectionnées dans les tables (6) prévues à cet effet pour éviter de reconstruire les messages à chaque itération. Pour que ces tables demeurent valides, chaque transfert doit s'effectuer dans l'ordre initial de construction.

Les routines MPI pour l'envoi et la réception de messages peuvent être bloquantes ou non. Avec l'implantation de communications bloquantes, il faut être plus prudent afin d'éviter les interblocages. Pour éviter un interblocage avec deux voisins, une politique « *pair-impair* » a été implantée. Généralement dans une décomposition 1D, les processeurs ayant un numéro pair commencent à recevoir pendant que les autres effectuent leur envoi. Cependant, comme la décomposition est multidimensionnelle (Figure 5-7), deux processeurs avec des numéros d'identification de parité identiques peuvent se retrouver côte à côte. Grâce aux outils fournis par MPI_Cart, on peut donc effectuer un test sur les groupes colonnes et lignes pour définir les paires.

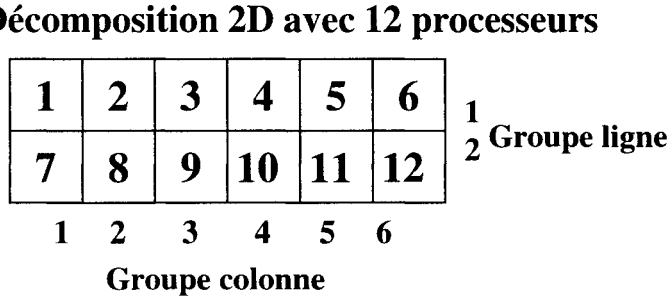


Figure 5-7 Exemple de numérotation des processus pour une topologie 2D

Cette méthode permet d'alterner les opérations d'envoi et de réception des processus selon leur position dans la topologie. Aucun conflit n'est possible et les communications se font les une à la suite des autres.

L'autre approche consiste à utiliser les communications non bloquantes. Cela permet de combiner les avantages des envois non bloquants et la simplicité de la méthode SHIFT. Cependant, puisque que l'ordre d'envoi des messages doit être conservé pour assurer la consistance des messages diagonaux, les échanges sont contraints à trois barrières de synchronisation implicite. Le seul avantage est donc la transmission de données en mode *full duplex*.

5.5.6 Résultats et choix du patron de communication

Après quelques tests, nous avons décidé d'effectuer les communications à l'aide de la méthode SHIFT en mode bloquant. Simple d'implantation, elle est plus performante que toutes les autres approches qui utilisent les routines de transmission en mode non bloquant.

Nous allons donc tenter d'expliquer cette différence notable entre les deux modes de transmission. Pour y arriver, il faut comprendre comment l'implantation de MPI fait la gestion des communications non bloquantes. Pour éviter l'interblocage, le système doit alterner entre les liens, regarder les tampons de lecture et d'écriture et lire ou écrire au besoin. Cette gestion est une dépense supplémentaire et son coût augmente avec le nombre de messages simultanés.

Dans le cas du *SHIFT* non bloquant, nous avons été surpris de remarquer une légère dégradation des performances. En réalité, la possibilité de communiquer en mode *full duplex* aurait dû compenser pour les coûts de gestion supplémentaires. Il faut alors se demander si l'implantation MPI exploite vraiment les 200Mbits/s disponibles.

L'information à ce sujet est rarissime et pourtant les communications sont la clé d'une implantation parallèle efficace et il y a intérêt à les minimiser par tous les moyens. Si les communications non bloquantes sont gérées par un mécanisme qui alterne entre les messages à la manière d'un *Round Robin*, il ne peut y avoir deux messages à travers le fil au même moment. Le mode bloquant effectue les échanges séquentiellement mais au moins, quand ils ont lieu, la carte est dédiée, il n'y a aucune interruption et le débit est maximal pour la durée de l'opération.

Afin de mieux comprendre les performances des communications, nous avons mis au point un petit test (indépendant de Powder3D) pour vérifier la bande passante réelle. Le débit théorique des cartes Ethernet *full duplex* ne peut être atteint en mode bloquant. En réalité, il faudrait que 2 processus légers soient créés et se chargent de la réception et de l'envoi séparément. Cependant, la fonction MPI doit être « *thread safe* », pour qu'aucun conflit ne survienne entre ces processus légers. Ce qui est le cas de la plupart des fonctions de MPI-1 et MPI-2, mais pas nécessairement de leur implantation. Ces tests ont permis de faire ressortir quelques particularités au niveau des différentes architectures. Pour pouvoir exploiter le mode *full duplex* sur Magnum, il est nécessaire de lancer quatre processus pour effectuer les deux échanges simultanément (Tableau 5-1). Ce tableau confirme clairement que le mode non bloquant, peu importe le nombre de processus, ralentit légèrement le taux de transfert. Cependant, les communications non bloquantes de l'implantation fournie par IBM permettent la transmission et la réception à plein débit avec seulement deux processus, ce qui serait en pratique avantageux pour Powder3D (Tableau 5-2).

Tableau 5-1 Taux de transfert pour Magnum

	nombre de nœuds	nombre de processus	bande passante réelle (Mbits/s)	bande passante théorique (Mbits/s)
Bloquant	2	2	87	100
non-bloquant	2	2	87	200
bloquant	2	4	165,5	200
non-bloquant	2	4	164	200
bloquant	2	8	162	400
non-bloquant	2	8	158	400

Tableau 5-2 Taux de transfert pour les serveurs de calcul du réseau Étoile

	nombre de nœuds	nombre de processus	bande passante réelle (Mbits/s)	bande passante théorique (Mbits/s)
Bloquant	2	2	94	200
non-bloquant	2	2	182	200
bloquant	2	4	179	200
non-bloquant	2	4	179	200

Enfin, nous avons repris ce test lorsque les processus font partie du même serveur et les taux de transfert sont excellents. Encore une fois, l'architecture IBM, plus élaborée détecte si les processus sont physiquement situés sur le même noeud et utilise le bus mémoire de la machine à sa pleine capacité; les p630 d'IBM atteignent le niveau très respectable des 3.3 Gbits/s (Tableau 5-3). Dans le cas de Mpich sur Magnum, il ne semble pas y avoir de transfert sur le réseau mais le débit maximal n'atteint pas la norme pour un bus à 133Mhz, soit 1,1 Gbits/s. Évidemment, ces valeurs ne sont jamais atteintes en pratique, mais nous sommes persuadés qu'il serait possible de faire mieux avec une bibliothèque optimisée pour les systèmes SMP. À ce sujet, une version pour des systèmes à mémoire partagée de Mpich, installée par moi-même sur le Regatta

(Polaris), a permis d'obtenir des performances similaires à celles du POWER4, et nous espérons faire encore mieux lorsque les bibliothèques officielles d'IBM seront installées.

Tableau 5-3 Performances des communications MPI entre processus d'un même noeud

	bande passante réelle (Mbits/s)	bande passante idéale (Mbits/s)
MAGNUM SMP (Mpich chp4)	468	1100
IBM POWER4	3300	4000
IBM Regatta (Mpich shmem)	3400	35 000

Nous avons discuté des raisons pour lesquelles la transmission de messages par des routines non bloquantes doit être mise de côté, du moins tant qu'elle ne sera pas plus performante. Dans ce travail, en ce qui concerne Powder3D, les communications bloquantes en mode *full duplex* ont été utilisées, mais grâce à la modularité de la méthode *SHIFT*, il est toujours possible d'activer le mode non bloquant. Avant de clore le débat sur les communications *full duplex*, précisons qu'il est toujours impossible d'exploiter convenablement les 2 cartes Ethernet disponibles sur chacun des nœuds de Magnum. La création de 8 processus n'est d'aucune aide et le débit théorique de 400 Mbits/s n'est pas atteint (Tableau 5-1). Il faudra trouver une autre utilité à cette deuxième carte, autre que de servir de pièce de rechange en cas de panne.

Dans cet ordre d'idée, le projet MP_lite (www.scl.ameslab.gov/Projects/MP_Lite) propose une petite bibliothèque de communication qui peut exploiter le « *channel bonding* ». Cette technique permet de décomposer un message en blocs, de le transmettre en parallèle sur plusieurs fils à l'aide de plusieurs cartes réseau et de le recomposer chez le destinataire. Dans le cas de la parallélisation de la DEM, nous croyons que la quantité de données à échanger serait suffisamment grande pour noter une amélioration des temps de communication due au « *channel bonding* ». Cependant, mous croyons qu'il serait mieux de pouvoir envoyer et recevoir des deux voisins simultanément.

5.6 Influence de la forme du domaine sur l'accélération

Une fois la parallélisation de notre code de simulation d'éléments discrets réalisée et optimisée grâce à la méthode SHIFT et les communications en mode bloquant, l'étape suivante a consisté à comprendre les phénomènes reliés au parallélisme qui causent des dégradations. Les premières courbes présentées au début du chapitre faisaient piètre figure, mais maintenant, nous pouvons simuler sur plus de 20 processeurs lorsque le problème s'y prête. Cette section traitera de l'influence de la forme du domaine de simulation pour une décomposition unidimensionnelle.

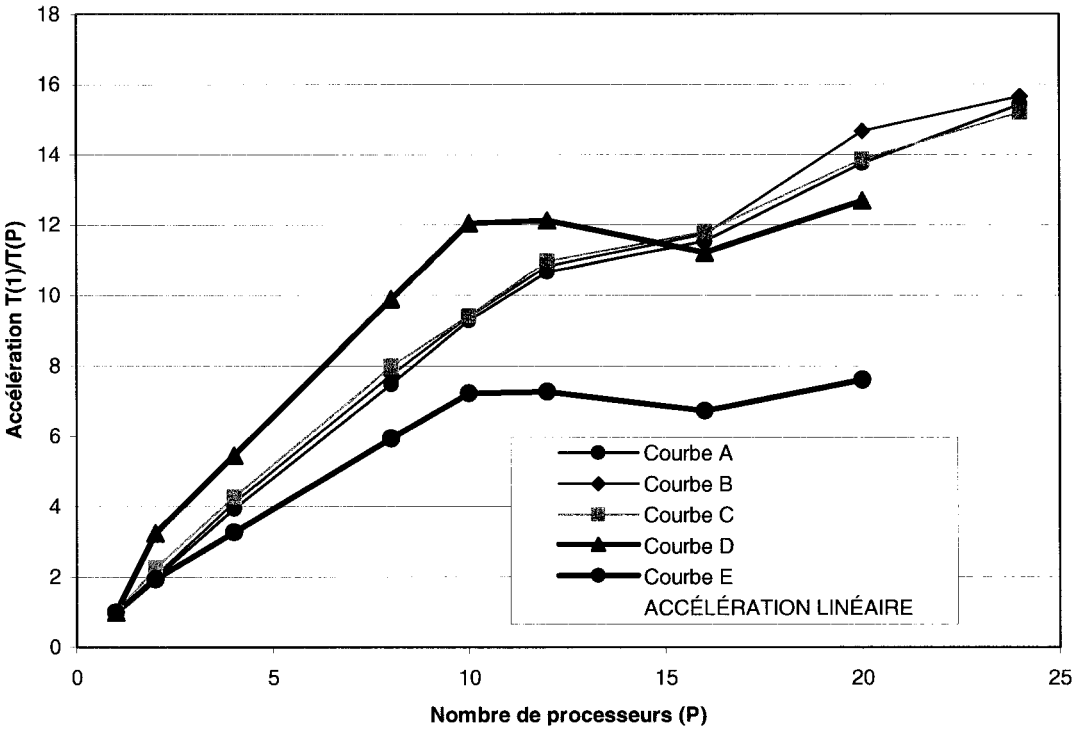


Figure 5-8 Accélération pour différentes formes de domaine sur Magnum

La figure 5-8 montre l'influence de la forme du domaine sur les accélérations obtenues avec la version parallèle de Powder3D. Ci-dessous, on retrouve les conditions de simulation pour les courbes du graphique (Tableau 5-4).

Tableau 5-4 Description des arrangements présentés à la Figure 5-8

Courbe	Nb de particules	Configuration	Nb de particules fantômes par frontière
A	80000	800x10x10	100
B	160000	400x20x20	400
C	320000	200x40x40	1600
D	512000	80x80x80	6400
E	512000	80x80x80	6400

Ce graphique montre deux tendances : les domaines allongés ont le même profil tandis que le domaine cubique est plus difficile à paralléliser. Les courbes (A,B,C) correspondent à des situations théoriques (A : un bâton de 10cm par 8m). La décomposition se fait toujours dans le sens de la longueur mais la quantité de particules transférée change; en fait, elle quadruple à chaque cas (A,B,C). Ces courbes démontrent que pour compenser l'influence de messages plus volumineux, on doit augmenter la charge de travail en doublant le nombre de particules. Entre d'autres mots, il est primordial d'avoir un problème suffisamment imposant pour que le ratio entre les calculs et les communications reste élevé afin d'obtenir de bonnes accélérations.

La courbe la moins intéressante (E) provient d'un domaine cubique avec un nombre de particules insuffisamment élevé par rapport au nombre de particules frontalières pour espérer pour rejoindre les trois autres. En fait, au moins 250 000 particules supplémentaires auraient été nécessaires pour y arriver. Sa jumelle (D) a cependant une bien meilleure allure. Il s'agit en fait d'un autre phénomène; un effet de super linéarité causé par le manque d'espace mémoire sur un nœud de la station de travail. Étant à court de mémoire vive, le processus en mode séquentiel doit utiliser la

mémoire virtuelle sur le disque dur et tombe alors en état de « trashing » et le processeur travaille avec une efficacité de 60%. À la section 5.10, nous reviendrons sur la relation entre la mémoire vive et la décomposition de domaine. La courbe E est en réalité une extrapolation de ce qui serait obtenu si les nœuds de Magnum possédaient assez de ressources pour cette simulation, soit environ 800 Mo de RAM.

Enfin, les courbes fléchissent à partir de 12 processeurs car, dans les conditions actuelles, il y a alors deux processus par noeud qui partagent la même carte réseau. Les temps de communication sont doublés, l'efficacité de la parallélisation diminue et les courbes d'accélération écopent. Comme nous l'avons dit à la section précédente, l'utilisation de la deuxième carte Ethernet serait avantageuse et permettrait d'obtenir de meilleurs résultats.

5.7 Accélération pour décompositions multidimensionnelles

Les géométries des problèmes à résoudre avec la méthode des éléments discrets sont variées et elles sont rarement de la forme idéale pour une décomposition unidimensionnelle. Avec seulement quelques processeurs, on peut se permettre une décomposition dans le sens de la longueur, mais rapidement les tranches deviennent trop minces et l'information est exagérément dédoublée.

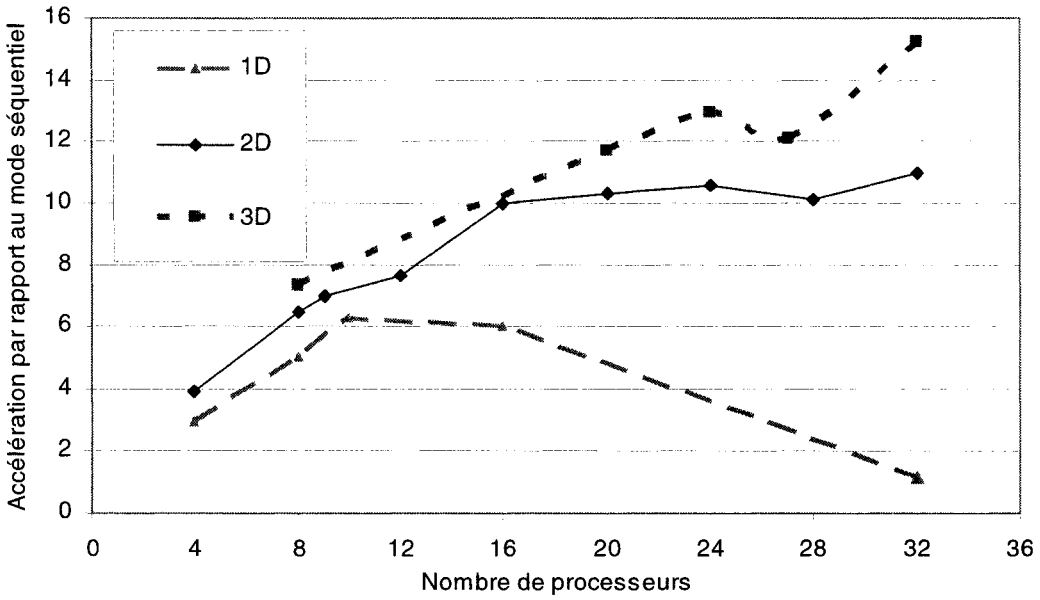


Figure 5-9 Facteur d'accélération pour une décomposition multidimensionnelle sur la grappe Hamsun (sédimentation de 500000 particules)

La figure 5-9 confirme que décomposer selon plusieurs axes est la meilleure chose à faire quand le domaine est compact. La quantité de données à échanger est alors minimisée et, par conséquent, les calculs relatifs au dédoublement le sont aussi. La courbe de la décomposition 3D fait bonne figure dans tous les cas sauf un. La grappe Hamsun contient 32 processeurs, il est difficile de respecter les conditions d'une véritable décomposition 3D. En fait, seul l'arrangement avec 27 processeurs (3x3x3) possède un sous-domaine central avec 6 voisins donc 6 messages à transmettre. Dans ce cas, la politique d'échange *SHIFT*, qui impose trois barrières de synchronisation implicites, ralentit les calculs car les processus sont fréquemment en attente. À ce propos, comme l'a précisé Plimpton (1995), la décomposition de domaine selon plusieurs axes permet de réduire le nombre de sphères transférées, mais cette synchronisation entre les étapes du *SHIFT* vient, dans certaines situations, annuler les gains obtenus par la décomposition multidimensionnelle. Pour une décomposition 3D avec 8 sous-domaines (2x2x2), les processeurs au coin du cube n'ont que 3 voisins, donc

6 messages à échanger. Ces conditions sont presque idéales et cela explique pourquoi l'accélération l'est tout autant. Le cube est une topologie de choix car, même si le nombre de particules est faible et le domaine de forme compacte, il permet généralement des accélérations raisonnables.

Finalement, le partage de la bande passante par les nombreux processus (4) d'un même nœud accentue la dégradation des performances. Une optimisation possible à notre application consisterait à limiter l'influence de ces communications en tentant de réduire la quantité de donnée transférée. C'est le sujet de la prochaine section.

5.8 Optimisations complémentaires pour la transmission des données

5.8.1 Retour sur la distribution du calcul des forces

L'envoi et la réception de l'ensemble des propriétés de chaque particule frontalière fait en sorte que la transmission des messages devient une opération d'envergure. Une astuce visant un compromis entre performance du réseau et puissance de calcul pourrait offrir de meilleures accélérations lorsque le réseau est considéré comme une source importante de ralentissement. Lors de la présentation de la décomposition spatiale, nous avons mentionné qu'il existait aussi une technique de distribution de la charge de travail en fonction des forces. Nous croyons qu'une telle approche permettrait d'économiser sur la longueur des messages en envoyant seulement les vecteurs de forces aux voisins. Au lieu des 16 mots double précision avec l'approche standard, cette stratégie ne nécessiterait que des messages de 6 mots double précision.

Malheureusement, la technique de distribution des forces appliquée à la DEM a un inconvénient majeur : elle requiert plusieurs changements à l'implantation actuelle basée sur les halos. Cependant, on peut reprendre l'idée générale en l'adaptant à notre

version; au lieu d'un échange de particules frontalières, nous allons effectuer un transfert des forces.

Dans cette approche, chaque processeur échange uniquement les forces accumulées par les particules frontalières. Cette méthode suppose qu'aucune paire de contacts entre particules fantômes n'est calculée. Elle reste valide jusqu'au prochain rafraîchissement des structures (qui nécessite un nouvel envoi du halo)(Figure 5-10). Cette économie réseau doit en contrepartie être compensée par un travail supplémentaire pour le calcul du mouvement car chaque processus doit alors calculer la nouvelle vitesse et la position de ses particules réelles ainsi que des particules fantômes.

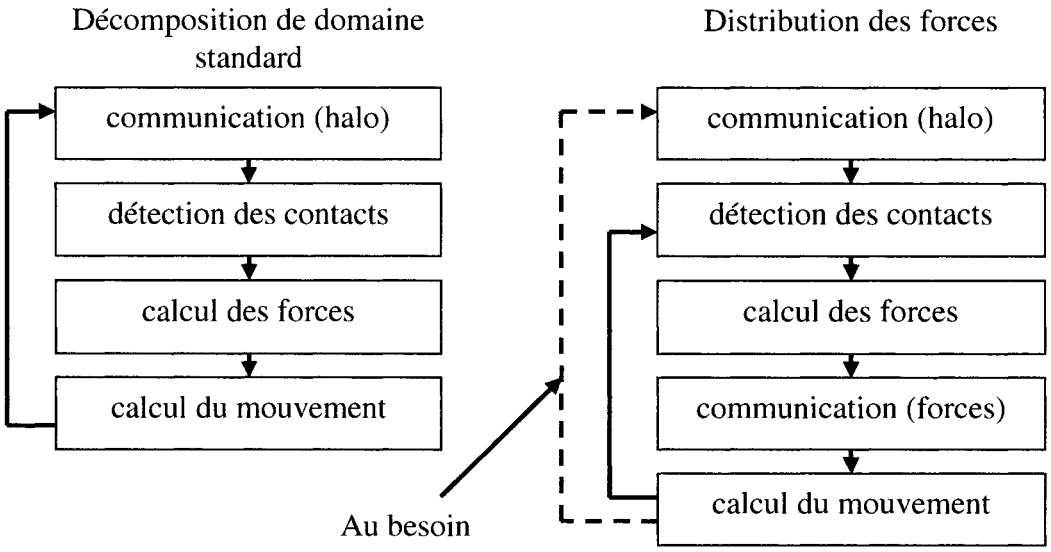


Figure 5-10 Décomposition de domaine avec communication des forces

Nos tests nous ont montré que pour une grappe comme Magnum, l'idée de réduire les temps de communication de cette façon est séduisante mais le coût du calcul du mouvement des particules fantômes est encore trop élevé par rapport à l'économie réelle (Tableau 5-5). De plus, ces opérations supplémentaires contribuent à agrandir le déséquilibre et augmenter les temps de synchronisation. Donc, l'implantation du transfert des forces n'est pas concluante. Son utilisation pourrait offrir un gain seulement

si le déséquilibre entre le débit et fréquence CPU est grand, c'est-à-dire quand la grappe de calcul est composée d'unités de calcul dernier cri et que le réseau est encombré. Puisque cette condition est rarement rencontrée en pratique, l'approche de la distribution des forces, dans sa forme actuelle, perd littéralement tout intérêt.

Tableau 5-5 Comparaisons des temps de transfert

	Original (16 réels)	Distribution des forces (6 réels)	Mode allégé (10 réels)
Temps calcul (s)	200	215	200
Temps communication (s)	23	14	17
Temps synchronisation (s)	11	15	12
Temps total (s)	234	244	229

5.8.2 Transferts allégés

Même si la dernière technique présentée n'a pas été retenue, il n'en demeure pas moins qu'on a quand même intérêt à réduire la quantité de données à transmettre, à condition de ne pas augmenter le travail requis. Présentement, dans Powder3D, on dénombre 16 champs d'information par particule, mais il n'y en a que 6 dont la valeur ne change pas entre deux itérations (par exemple le rayon et la masse). Donc, un nouveau type de données (MPI_DATA_TYPE) a été implanté dans Powder3D. Ce type de données est utilisé seulement pour la transmission des particules frontalières et concerne uniquement les données qui peuvent changer entre deux itérations successives.

L'économie réelle est faible mais les temps de communication sont réduits de 33%, ce qui est proportionnel à la nouvelle taille du message (Tableau 5-5). On en conclut que la latence est négligeable parce que la quantité de données est suffisamment grande. Ici, c'est la synchronisation implicite qui est néfaste. Comme nous l'avons dit plus tôt dans ce chapitre, le test utilisé consiste en un bloc rectangulaire de densité

constante dont la décomposition crée des sous-domaines de charge équivalente. Nous sommes plutôt déçus de constater que même si la charge est équilibrée, les temps de synchronisation sont comparables aux temps de communication.

5.9 Validation et traitement des cas particuliers

Cette section traite de certains cas particuliers reliés à la parallélisation de la méthode des éléments discrets. Une attention particulière doit être portée à ces petits détails afin de rester le plus fidèle à la version séquentielle.

5.9.1 Périodicité et décomposition de domaine

Lorsque des conditions aux limites périodiques sont imposées sur les frontières du domaine, un mécanisme doit gérer différemment les communications et legs de particules. Le standard MPI gère les communications entre voisins de façon appropriée toutefois, nous devons effectuer la translation des particules hors limites. Dans notre première version parallèle, cette opération était effectuée immédiatement après le calcul de la position, avant les communications, ce qui causait quelques problèmes.

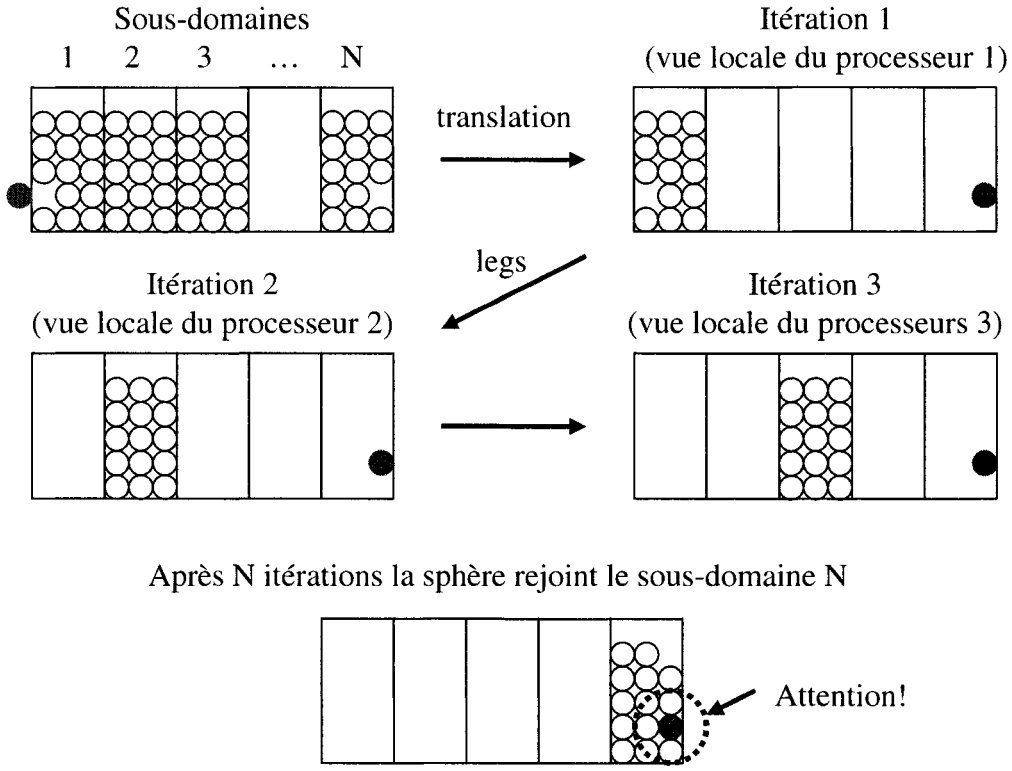


Figure 5-11 Exemple d'exécution parallèle erronée avec des conditions périodiques

À la figure 5-11, lorsque le changement des coordonnées (translation) pour respecter les conditions périodiques est effectué avant l'opération du legs, la particule à l'extrême gauche du domaine est transférée au domaine de droite car elle se trouve à droite de la frontière maximale au moment du legs de particules. Une erreur de cette nature devient difficile à détecter car la sphère assignée à un mauvais sous-domaine finit généralement par retrouver son véritable propriétaire après quelques itérations sans perturber la stabilité du système de façon significative. Cependant, lorsque le nombre de processeurs est élevé, beaucoup plus d'itérations sont requises et l'espace réservé à cette particule peut être occupé lorsque cette dernière retrouve son véritable propriétaire. Un chevauchement excessif sera détecté et la simulation devra être arrêtée. Heureusement, au lieu de complexifier la boucle de construction du message dédié au legs de particules, il suffit de reporter l'opération de translation des particules qui dépassent les limites périodiques après celle du legs. Grâce à cette modification qui n'implique aucun coût

supplémentaire de traitement, la version parallèle ne diffère plus de la version séquentielle.

5.9.2 Communication des contacts

Dans plusieurs modèles physiques et plus particulièrement celui de *Zhou et al.* (2001), duquel notre modèle est inspiré, il existe parfois un lien entre les calculs de l'itération courante et ceux de l'itération précédente. Dans la méthode des éléments discrets, la force de frottement provient du déplacement tangentiel entre deux particules. Un contact pouvant durer une dizaine d'itérations, il faut donc conserver l'information pertinente reliée à chaque contact. Afin d'accélérer les recherches relatives aux paires de vieux contacts, deux structures de données ont été définies afin de retrouver rapidement les anciennes collisions ainsi que la valeur des forces de frottement cumulatives. Ces informations sont conservées localement par les processus qui gèrent les particules impliquées dans une collision.

Évidemment, lorsqu'une sphère doit changer de sous-domaine, son nouveau processus ignore tout des anciens contacts de cette particule avec les autres particules de l'ancien sous-domaine. Pour être cohérent avec la version séquentielle, nous avons réalisé qu'il faut accompagner chaque particule transférée de ces détails essentiels. Les données relatives aux contacts entre cette particule et ses voisines et ceux entre les triangles de la géométrie doivent faire partie des communications. Heureusement, le coût de ces communications est minimal car les changements de sous-domaines ne sont pas fréquents, ayant lieu soit lors du rafraîchissement des sous-domaines ou lors de l'équilibrage de tâche dynamique (chapitre 6).

Enfin, un changement de sous-domaine pour une particule veut aussi dire un changement de numérotation locale pour tous les processus impliqués. Les structures de données où sont stockées les références vers les particules en contact ne sont plus

exactes. Comment retracer rapidement deux particules qui étaient réellement en contact à l'itération précédente afin de trouver leurs nouvelles positions? Lorsqu'il y a changement de sous-domaine, le seul moyen de suivre le déplacement d'une particule à travers plusieurs sous-domaines est d'utiliser une numérotation globale. De cette façon, un processus peut gérer une particule fantôme qui est devenue sa propriété en cours de route. Pour accélérer les opérations reliées à la numérotation des particules, une table de références a été créée. Cette structure de données permet de trouver la position d'une particule dans le vecteur local à partir de son numéro unique (Tableau 5-6). Si cette particule ne fait pas partie d'un sous-domaine ou de ses halos, cette table retourne -1.

Tableau 5-6 Exemple de table de références

Numéro unique de la particule	1	2	3	4	5	6	...
Position dans le vecteur local	1	556	2	-1	4	-1	...

5.9.3 Validation des résultats

Un long processus itératif couplé à une intégration avec un pas de temps très court est une situation idéale pour faire diverger deux simulations. En effet, une petite différence au niveau de la troncature des derniers chiffres peut s'accumuler pendant les milliers d'itérations et se propager jusqu'aux chiffres dits « significatifs ». Pour ces mêmes raisons, la version séquentielle et la version parallèle de Powder3D peuvent donner des résultats différents.

Le test pour valider la parallélisation par la décomposition de domaines est de simplement comparer les fichiers des résultats finaux après une durée raisonnable. Avec la concaténation des halos, la numérotation locale de chaque processus est différente de la numérotation locale de la version séquentielle. Les opérations ne se font plus dans le même ordre et les valeurs après arrondissement diffèrent; un écart est créé et s'amplifie par la suite.

Pour obtenir le même résultat avec les deux versions, une dernière modification a été nécessaire. Il a fallu forcer l'ordre des opérations pour qu'il soit le même en parallèle et en séquentiel. En utilisant la table de numérotation inverse (Tableau 5-6), chaque processus impose un ordre précis pour toutes les opérations, que ce soit pour placer les particules dans le quadrillage de localisation ou détecter les collisions entre elles. Heureusement, cette modification a pu être implantée sans altérer les performances de façon significative.

Les trois modifications présentées ci-dessus illustrent que si la parallélisation par décomposition de domaine de la méthode des éléments discrets semble *a priori* plutôt simple, elle doit être faite avec minutie afin de garantir la cohérence avec le code séquentiel.

5.10 Parallélisation et mémoire vive

La parallélisation de la méthode des éléments discrets par décomposition de domaine a comme but premier d'accélérer les temps de résolution des simulations. Un autre objectif est de permettre l'exécution de problèmes nécessitant une mémoire plus grande que celle disponible sur un seul ordinateur. Essentielle pour stocker toutes les données et le programme lors de l'exécution, elle était jusqu'à tout récemment très dispendieuse. La parallélisation par décomposition de domaine permet de faire des simulations qui ne peuvent être faites sur un seul nœud. Par exemple, dans le cas de Magnum et ses 12 nœuds de 512 Mo, il est possible de faire une simulation nécessitant 6 Go, ce qui dépasse largement les limites de l'adressage 32-bits conventionnel (4 Go).

Une utilisation judicieuse de la mémoire est donc primordiale pour pouvoir traiter des problèmes d'envergure. Trois modifications au code ont permis de réduire de façon significative la mémoire requise lors de simulations.

5.10.1 Allocation et réallocation dynamique

L'allocation dynamique des vecteurs contribue à l'évolutivité aisée d'un code de calcul. Pour n'utiliser que les méga-octets nécessaires, chaque processus doit maintenir des vecteurs de taille minimale. Cependant, plusieurs tests doivent être réalisés pour éviter les dépassements lors d'un ajout. On peut bien sûr créer un nouveau vecteur et y copier l'ancien. On parle alors d'un compromis où on économise de la mémoire en échange de cycles CPU supplémentaires pour la gestion des structures de données.

En effet, lors de la communication du halo, le nombre de particules reçues et envoyées est variable, donc les vecteurs de réception doivent l'être aussi. Au début, ces vecteurs étaient de longueur proportionnelle au nombre total de particules. Pour identifier le nombre d'éléments reçus, on pouvait utiliser les appels MPI suivants :

MPI_RECV(vecteur de réception , nombre_max, type, voisin1, tag, status)
MPI_GET_COUNT(status , type, nombre véritablement reçu)

Si on alloue dynamiquement l'espace mémoire pour toutes les structures de données, il faut pouvoir prévoir le nombre de données à recevoir et réajuster le vecteur en conséquence. *MPI_PROBE* permet d'obtenir cette information avant la réception. On peut alors réallouer l'espace mémoire au besoin :

MPI_PROBE (status , voisin1)
MPI_GET_COUNT (status , type, nb à recevoir)
/ RÉALLOCATION SI NÉCESSAIRE */*
MPI_RECV(vecteur de réception , nb à recevoir, type, voisins1, tag, status)

5.10.2 Réingénierie de structures de données

Dans le cas de Fortran 90, quelques types de données semblent plus coûteux que d'autres. Par exemple, les pointeurs NULL pour créer une liste chaînée consomment beaucoup trop d'espace. Au lieu des 8 octets prévus, 48 octets sont nécessaires même si la liste est vide. Dans notre première implantation de la DEM, le quadrillage de l'espace était principalement basé sur un tableau 3D couplé à une liste chaînée. À mesure que l'on diminuait le rayon des particules, la taille de la grille devait s'ajuster et cela avait pour effet d'augmenter de façon cubique le nombre de cases du quadrillage. Rapidement, cette grille pouvait nécessiter à elle seule jusqu'à 400 Mo de mémoire. Les pointeurs ont été remplacés par une structure de données plus économique et tout aussi efficace.

5.10.3 Décomposition de domaine appliquée aux données

La décomposition de domaine implique une distribution des données. Elle s'applique principalement aux particules et indirectement aux contacts dont le nombre par sous-domaine est réduit lorsque le nombre de processeurs augmente. Des économies additionnelles sont possibles si on tient compte de cette décomposition de domaine au niveau des structures de données relatives aux particules. Par exemple, le quadrillage de l'espace peut devenir encore moins coûteux même après les changements précisés à la section précédente. En effet, le quadrillage n'a pas besoin de contenir tout le domaine, mais seulement le sous-domaine correspondant ainsi que l'espace occupé par ses halos. Toutes ces structures doivent être dynamiques car lors d'un rééquilibrage par le déplacement d'une frontière, la taille des sous-domaines est modifiée.

De la même façon, il est possible d'appliquer la décomposition de domaine aux données relatives aux obstacles. Un processus n'a pas besoin du maillage de triangles en entier, il peut donc sélectionner le « sous-maillage » correspondant à son sous-domaine. De plus, cette décomposition du maillage permet de réduire le temps consacré à placer les triangles dans le quadrillage des objets solides.

Dans le cas où les frontières sont représentées à l'aide de particules, la décomposition permet de réduire significativement le nombre de sphères et la mémoire requise pour les stocker (Figure 5-12).

Décomposer la géométrie peut toutefois comporter des risques si la géométrie est mobile. Pour éviter une erreur de l'utilisateur, le système permet seulement une décomposition unidimensionnelle selon l'axe de rotation de l'objet mobile. Si cette condition n'est pas respectée, un objet mobile peut changer de sous-domaine. Pour l'instant, aucun mécanisme n'est prévu pour traiter ce cas spécial, ce qui peut entraîner des erreurs lors de la détection de contacts avec la géométrie. Enfin, pour des raisons similaires, le rééquilibrage par le déplacement de frontières requiert quelques ajustements supplémentaires pour demeurer compatible avec la décomposition de la géométrie; une solution serait de garder une copie de la géométrie originale afin de la redécomposer selon la nouvelle position des frontières.

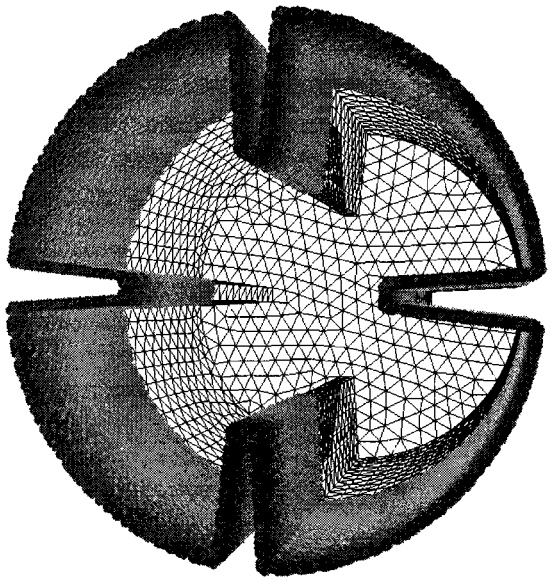


Figure 5-12 Décomposition de domaine sur deux processus appliquée à une géométrie définie par des murs particulaires (au fond, c'est le sous-domaine du 2^{e} processeur)

5.10.4 Économies réalisées

Pour une simulation d'écoulements granulaires à 250 000 particules, la version parallèle de Powder3D basée sur l'allocation statique des structures de données nécessitait typiquement 400 Mo par processus. Avec à peine 512Mo de mémoire sur chaque nœud, cette implantation ne pouvait ainsi tirer profit des deux processeurs disponibles sur chacun de ceux-ci. Cette fâcheuse situation a été résolue à l'aide des modifications discutées plus haut. Pour le même type de problème, l'espace requis est maintenant de 100 Mo. Le graphique de la figure 5-13 montre la relation entre l'espace mémoire par processus et le nombre de particules et de processeurs. Il permet évaluer l'espace nécessaire pour une simulation.

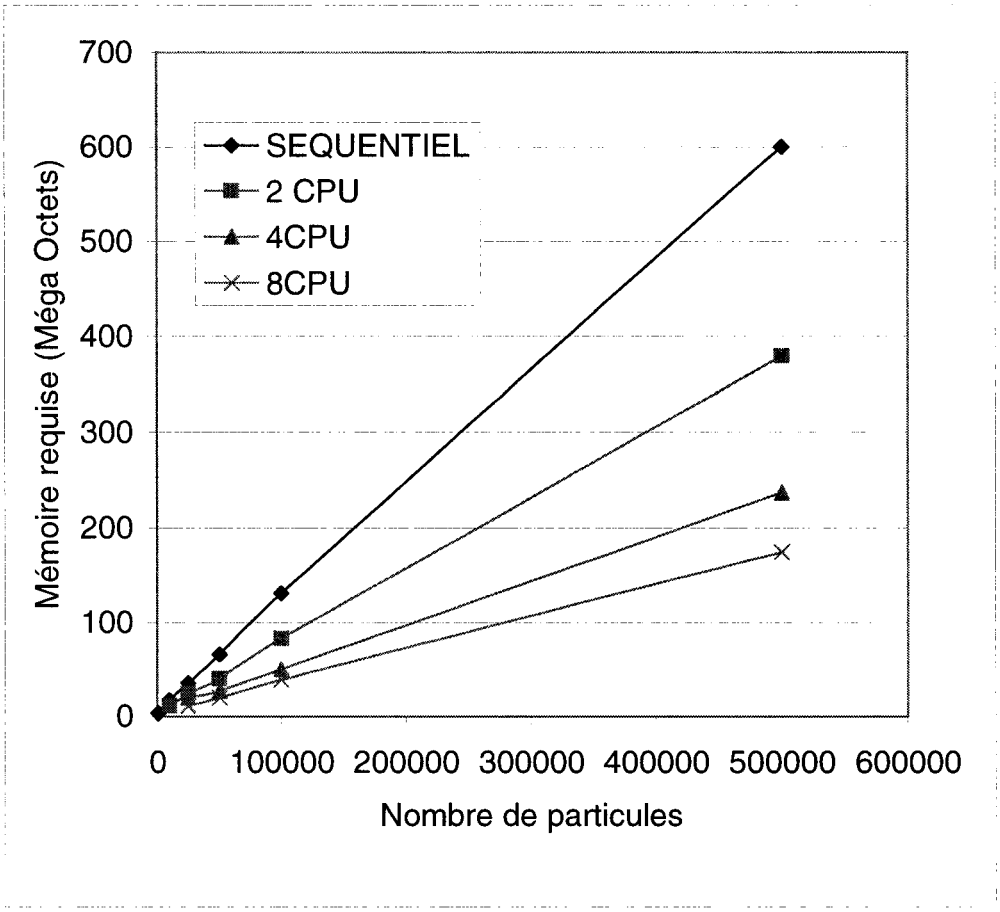


Figure 5-13 Graphique de la mémoire requise en fonction du nombre de particules et de processeurs

Avec la décomposition de domaine et son coût supplémentaire relatif au dédoublement des données, deux processus requièrent environ 380Mo pour un problème avec 500 000 particules au lieu des 600 Mo en mode séquentiel. Des simulations de taille plus importante sont maintenant possibles et non contraintes par les limites individuelles des nœuds de la grappe. Cependant, comme le montre la figure 5-13, cette réduction n'est pas linéaire. L'ajout de processeurs permet de réduire les ressources requises sur chaque nœud, mais les dédoublements atténuent la réduction. Une évaluation du nombre maximum de particules pouvant être traitées sur Magnum a été réalisée. Nous considérons que le système d'exploitation prend 60 Mo et nous supposons qu'il n'y a pas de permutation de mémoire sur les disques (*memory swapping*). Avec 225 Mo par processeur, la limite est donc de {24 X 165 000} ~ **4,0** millions de

particules. Pour en faire autant en séquentiel, un processeur 64-bits avec 4,6 Go serait nécessaire. Aucune simulation de cette taille n'a encore été exécutée avec Powder3D mais de telles simulations sont envisagées.

5.11 Macro-particules et parallélisation

La plupart des travaux de la littérature concernant la DEM ont porté jusqu'à maintenant sur des écoulements de particules sphériques. Certaines équipes ont simulé des écoulements de particules 2D non circulaires, mais l'extension à la troisième dimension n'est pas triviale. Comme il a été mentionné au chapitre 4, la détection de contacts entre les sphères et les triangles de la géométrie est une opération complexe et très coûteuse. La détection de contacts entre deux objets non sphériques peut être plus complexe (Hogue, 1998). Malgré les résultats encourageants de ce chapitre obtenus avec notre version parallèle, des simulations de plusieurs dizaines de milliers de particules non sphériques seraient irréalisables dans des laps de temps raisonnables. Remarquons que, grâce à tous ces calculs supplémentaires, les modèles d'objets non sphériques auraient au moins l'avantage de présenter des courbes d'accélération exceptionnelles et ce même pour des simulations comportant peu d'éléments.

Pour remédier au problème de la détection entre particules non sphériques, une solution plus abordable a été proposée par Gallas et Sokolowski (1993). Ces auteurs proposent de combiner plusieurs sphères pour créer une macro-particule qui représente un objet de la forme souhaitée. Ces sphères peuvent s'interpénétrer ou non. Le modèle physique implanté doit bien sûr gérer différemment les interactions entre ces sphères. En particulier, il force les sphères composant une macro-particule à se déplacer en bloc (solid-body motion) en utilisant une force d'attraction interparticulaire dont la valeur est suffisamment élevée. Les avantages de cette technique sont que la détection de contact n'a pas besoin d'être modifiée selon la forme des particules et que sa complexité est

linéaire en fonction du nombre de particules. L'inconvénient majeur de cette technique est le nombre élevé de particules sphériques requis.

La création de grains composés de particules élémentaires a un autre avantage qui n'est jamais mentionné. Au début de ce chapitre, nous avons dit que la largeur du halo est directement proportionnelle au rayon de la plus grosse particule (2,1 x Rmax). De cette façon, on s'assure que pour un mélange polydispersé, une collision entre deux grosses sphères chevauchant deux sous-domaines voisins pourra être détectée normalement. Si on pousse cette situation à l'extrême (boules de billard entourées de grains de sable), la parallélisation par décomposition de domaine n'est plus appropriée (Figure 5-14). Gérer ces grosses particules avec une mémoire partagée virtuelle serait alors une option, mais apporter des modifications majeures à un code uniquement pour ces cas spéciaux n'est pas une solution idéale. L'utilisation de macro-particules permet de contourner ce problème. Des petites particules liées peuvent générer de plus grandes tout en étant distribuables sur plusieurs sous-domaines. En effet, s'il n'y a aucune interaction autre qu'une grande force d'attraction entre deux particules voisines faisant partie de la même macro-particule, rien n'empêche de fractionner une macro-particule sur plusieurs processus tant que les sous-domaines voisins envoient dans leur halo les particules adéquates.

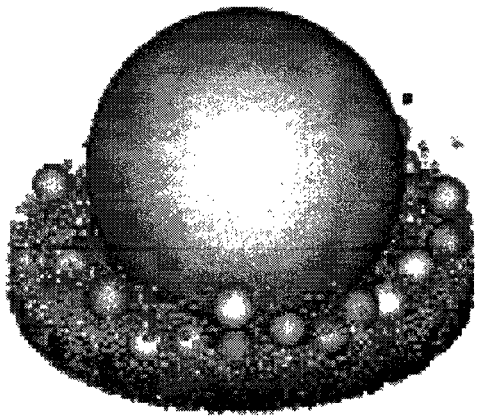


Figure 5-14 Sédimentation de grains autour d'une sphère (ratio 35) (Ferrez, 2001)

5.11.1 Implantation

Nous présentons ici le modèle que nous utilisons pour modéliser l'écoulement de macro-particules et qui permet une parallélisation efficace par décomposition de domaine.

Le modèle physique est très simple. Au départ, il a été question de prendre directement le modèle proposé par Ferrez (2001) et qui s'inspire de la dynamique moléculaire et du potentiel V de Lennard-Jones :

$$V = Ax^{\alpha}Bx^{\beta},\ \alpha et\ \beta > 0 \qquad \textbf{Équation 5.4}$$

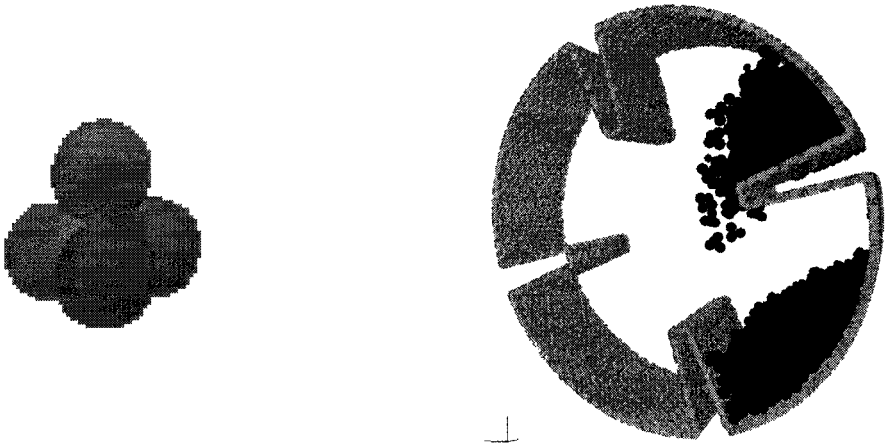
où x est la distance entre les centres de masses des deux particules voisines composant la même macro-particule. Malheureusement, trouver des valeurs intéressantes pour les variables A et B a été plus difficile que prévu, les macro-particules créées étant très instables; une macro-particule pouvait s'effondrer ou se déformer au moindre choc.

Comme solution de rechange, un modèle artificiel, moins « physique » a été esquissé pour générer des macro-particules stables. Nous avons pris le modèle pour les collisions et lui avons apporté quelques modifications pour que les sphères se déplacent en bloc. D'abord, pour les sphères d'une même macro-particule, il n'y a plus de frottement interne et il n'y a plus de vitesse de rotation. Comme d'habitude, une force de répulsion est appliquée lorsque deux particules s'interpénètrent et une force de rappel les retient aussitôt. Les forces découlent de la même formule que celle utilisée pour les contacts ordinaires mais elles ne sont comptabilisées que lorsque la distance est importante. Précisions que cette implantation n'a pas été validée mais qu'elle donne un comportement qualitativement acceptable. Des tests de validation sont prévus et portent sur le débit de particules à travers un orifice et les corrélations de Beverloo (1961).

Pour faire la gestion des contacts interparticulaires au sein d'une macro-particule, deux propriétés ont été ajoutées à chaque particule :

1. Groupe (type de liens)
    - 0 : sphère ne faisant partie d'aucune macro-particule
    - 1 : sphère faisant partie d'une macro-particule
2. Objet_id : numéro de la macro-particule

La disposition des sphères qui forment les macro-particules est très importante et cruciale pour leur stabilité. Puisque les sphères sont indépendantes entre elles, il est impossible de garder une forme structurée et rigide si celles-ci ne sont pas empilées en bloc. Une ligne de sphères n'est pas considérée comme stable car des forces extérieures peuvent tordre la macro-particule correspondante.



a) b)

Figure 5-15 (a) Tétraèdre de 4 sphères et (b) exemple de mélange de macro-particules

Le tétraèdre de sphères possède les bonnes propriétés (Figure 5-15). En ajoutant une sphère au centre de trois autres, on génère une structure stable. Rien ne limite cette technique à 4 particules (le losange n'est d'ailleurs pas vraiment plus difficile à générer), mais construire des formes complexes nécessiterait l'utilisation d'un outil approprié.

Pour cette raison, un autre agencement a été développé, soit le cube (Figure 5-16). Composé de 9 sphères, ce cube reste stable parce que la petite sphère empêche l'écrasement de la structure. À partir de ce cube, on peut générer des boîtes de dimension désirée. Il faut préciser que, contrairement à la figure 5-16, le rapport des diamètres entre la sphère centrale et les autres est en réalité beaucoup plus faible en 3D, soit à peine 18%.

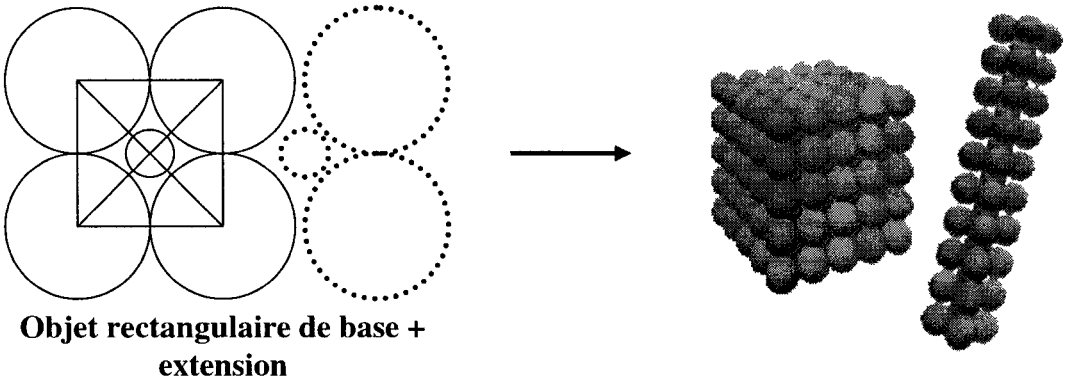


Figure 5-16 Cube de 5 sphères

Toutes ces contraintes ne seraient pas nécessaires si notre algorithme de détection était basé sur les triangulations de Delaunay. Ferrez (2001) combine sans difficulté sa méthode de détection aux objets non sphériques. Le graphe reliant les particules est toujours le même mais certaines arêtes du graphe sont immuables. Les opérations de maintenance (*FLIPS)* ne sont plus nécessaires (Figure 5-17) et il s'agit en fait d'une « *constrained Delaunay triangulation* ». L'expérience nous permet de croire que l'utilisation d'une structure de données semblable serait bénéfique. En plus d'aider à conserver la stabilité des macro-particules en empêchant les déformations irréversibles, un autre avantage de l'utilisation des triangulations de Delaunay est d'économiser sur la détection de contacts entre les particules de l'objet. Son inconvénient majeur est que même si elle implantable en 3D, elle demeure toutefois complexe. Au lieu d'implanter les triangulations, nous avons donc préféré conserver dans une liste toutes les interactions entre les particules composant les macro-particules.

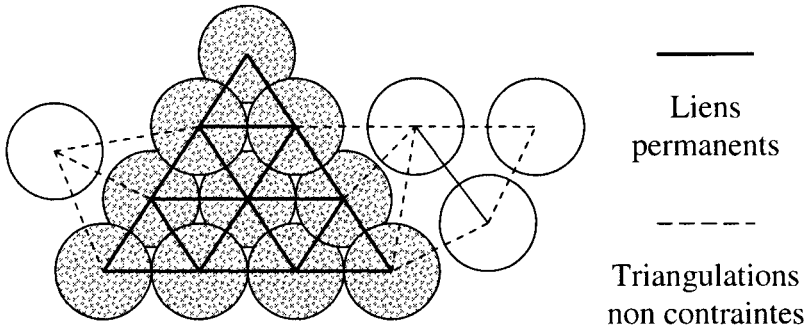


Figure 5-17 Utilisation d'un graphe pour la création d'une macro-particule

5.11.2 Performances du modèle de macro-particules

Afin de démontrer que la création de particules non sphériques à l'aide de particules élémentaires est une stratégie performante, un test spécifique a été développé. L'expérience consiste à laisser tomber 2 gros objets sur un lit de 50000 petites billes de 2.5 mm de rayon (Figure 5-18a). La première partie de l'expérience a comme but de vérifier l'accélération maximale de la parallélisation en fonction du diamètre des deux objets. Les deux objets sont représentés par des sphères dont le rayon est un multiple du rayon des billes qu couvrent le plancher.

La deuxième partie de l'expérience consiste à mesurer l'accélération maximale lorsqu'on varie la taille des objets. Cependant ces objets sont maintenant des macro-particules créées avec des particules élémentaires de 2.5mm de rayon. Pour simplifier la création des macro-particules, nous avons préféré utiliser des cubes même s'ils contiennent plus de particules qu'une macro-particule sphérique de taille équivalente (Figure 5-18b).

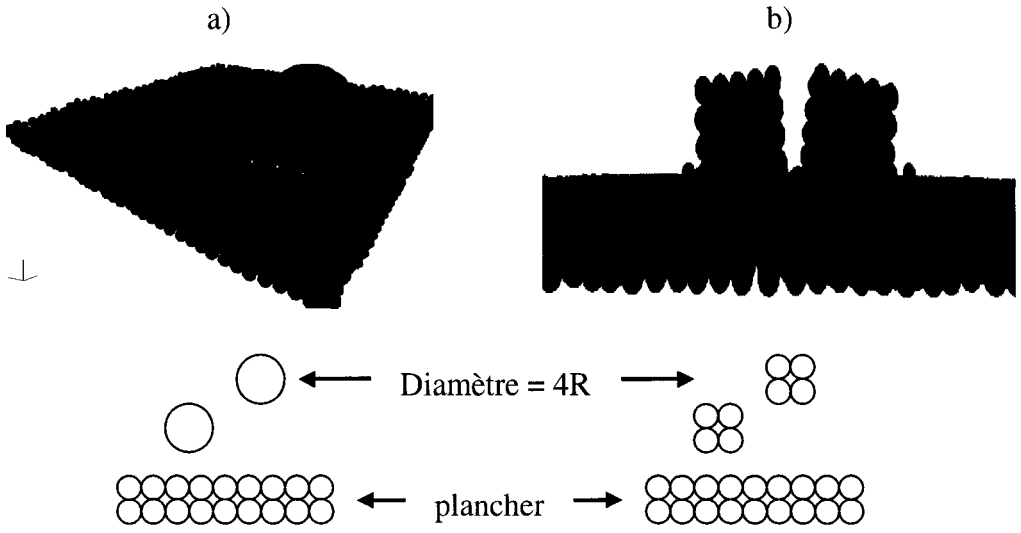


Figure 5-18 Test pour vérifier l'efficacité de la parallélisation avec des macro-particules

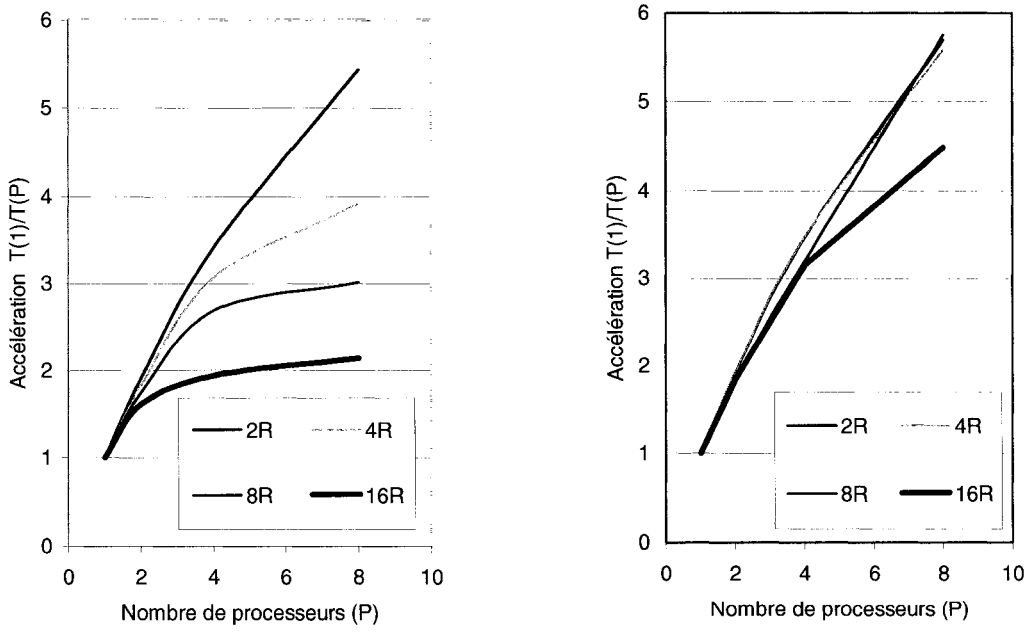


Figure 5-19 Accélération en fonction du modèle de représentation des gros objets (Magnum, 50000+ particules)
a) Accélération en fonction du rapport de rayon elevé
b) Accélération en fonction de la dimension des macroparticules

Lorsque plusieurs processeurs sont utilisés, les accélérations sont décevantes pour la première partie de l'expérience. Cette dégradation est causée par l'obligation d'élargir les halos pour permettre la détection de collisions impliquant les plus grosses sphères. La quantité de données à échanger à chaque itération augmente et les temps transfert diminue l'accélération. Si ces sphères étaient encore plus volumineuses, la largeur des halos limiterait aussi le nombre de processeurs utilisables.

L'utilisation de macro-particules rend la parallélisation par décomposition de domaine beaucoup plus performante. L'économie au niveau de la quantité de données à échanger est significative et l'accélération est excellente. Cependant, ces deux graphiques sur l'accélération ne donne aucune information à propos des temps réels de résolution. En fait, quand le diamètre de l'objet est petit, les temps séquentiels (non présentés ici) sont semblables peu importe le modèle de représentation des gros objets. Cependant, avec un cube de dimension 16R, il y a beaucoup plus de sphères à gérer : 25000 pour les objets et 75000 au total. C'est 50% de plus que les 50002 particules du mode de représentation standard. Le coût de traitement pour les sphères supplémentaires de l'objet est élevé (Figure 5-20). L'exécution séquentielle est beaucoup plus rapide sans macro-particules. Toutefois, la meilleure évolutivité permet de compenser pour le travail supplémentaire occasionné par les 25000 particules du cube. En résumé, les macro-particules n'offrent aucune contrainte à la parallélisation et permettent de simuler des phénomènes plus complexes impliquant des particules non-sphériques.

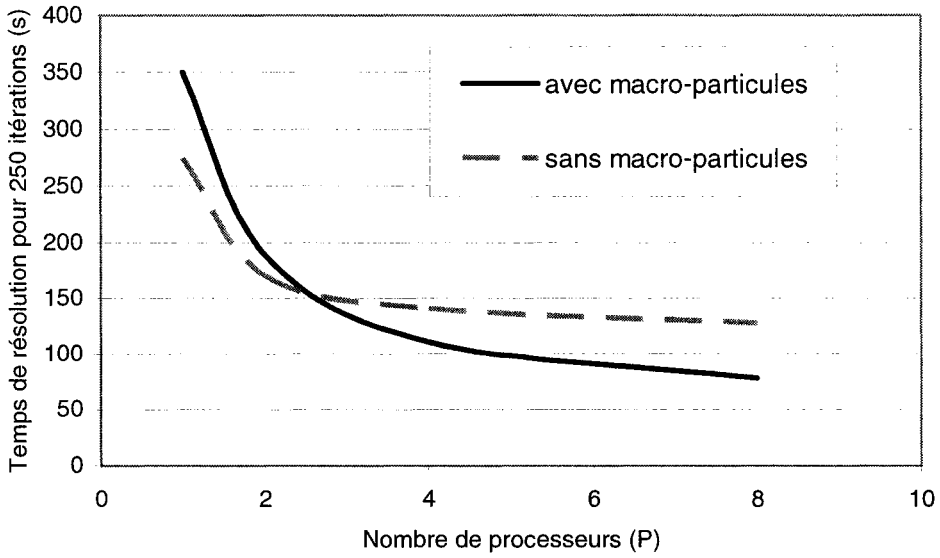


Figure 5-20 Temps de calcul avec et sans macro-particules (rapport de rayon elevé Rmax /Rmin = 16)

5.12 Conclusion

La décomposition de domaine a été étudiée et présentée sous plusieurs aspects. De nombreuses optimisations ont permis une implantation parallèle efficace de la méthode des éléments discrets. Nous avons abordé des aspects reliés à la parallélisation comme la stratégie d'échange SHIFT et la transmission *full duplex*. Après avoir discuté de ce mode de transmission, nous n'avons trouvé aucune solution pour l'exploiter convenablement sur toutes les plateformes. Nous avons aussi constaté que la complexité du modèle physique implanté peut nuire aux accélérations parce qu'il oblige à communiquer plus souvent; il faut toujours tenter de réduire en taille des messages au maximum. Enfin, nous avons montré que l'utilisation de macro-particules faites de sphères plus petites ne nuisait pas à la parallélisation et qu'il était très avantageux d'utiliser cette technique.

La décomposition multidimensionnelle et les communications non bloquantes ont été abordées dans le but de réduire les coûts de communication, mais en pratique le déséquilibre de la charge entre les processeurs est souvent plus néfaste. Pour cette raison, le prochain chapitre y sera consacré entièrement.

6 Équilibrage dynamique de la charge de travail

Pour simplifier, nous avons supposé au chapitre précédent que la décomposition initiale du domaine était parfaite, que la concentration était uniforme dans tout le domaine, donc que la charge de travail de chaque processus était équivalente. Cependant, les deux remarques suivantes sont importantes :

- Si on n'effectue que quelques milliers d'itérations pour vérifier l'efficacité de la parallélisation, il est normal que la concentration ne varie pas assez rapidement pour créer un déséquilibre majeur. Mais, dans le cas général, la concentration peut varier. Il est possible qu'une décomposition initiale adéquate se détériore et que la dernière itération en temps soit exécutée par un seul processeur.

- Lors de son achat, une grappe de calcul est souvent homogène. Pour un travail donné et distribué équitablement, tous les processus devraient terminer au même moment. Toutefois, lorsque des nœuds sont ajoutés à la grappe, celle-ci devient la plupart du temps hétérogène par ce que ces nœuds sont différents de ceux de la grappe originale. Si les sous-domaines de calcul sont de même taille, les nouveaux processeurs plus rapides sont alors sous-utilisés, le goulot d'étranglement provenant des nœuds les plus lents.

6.1 Introduction à l'équilibrage de charge pour la DEM

Toutes les courbes d'accélération sont directement touchées par les inégalités de charge. On peut formuler la loi d'Amdahl pour en tenir compte (Horoi et Enbody, 2001) :

$$A = \frac{T(1)}{T(P)} = \frac{T(1)}{\left[\frac{(T1 \times \alpha)}{\eta \times P} + (1-\alpha) \times T(1)\right] + TComm + TCasParticulier} \qquad \textbf{Équation 6.1}$$

où η représente la charge moyenne des processus. Sa valeur est comprise entre 1 et 1/P. Au chapitre précédent, en supposant un équilibre parfait, η valait 1. Dans le pire des cas, la valeur de η peut atteindre *1/P*, ce qui veut dire qu'un seul processeur effectue tout le travail.

Le rôle d'un mécanisme d'équilibrage de charge est de garantir que la charge de travail de chaque processus soit le plus près possible de la charge moyenne. Ce contrôle devient essentiel pour plusieurs types de programmes parallèles, peu importe l'architecture. Il pose quelques problèmes supplémentaires pour une grappe de calcul en mémoire distribuée. Pour les applications parallèles, il y a deux situations embarrassantes à corriger :

- les processeurs sous-utilisés, qui sont des ressources perdues qui ne contribuent pas à accélérer les temps de calcul;
- le processeur le plus lent (ou le plus occupé) fait attendre tout le groupe de travail car la complétude d'une itération sur ce sous-domaine est tout simplement plus longue.

Au cours de ce chapitre, des techniques permettant l'équilibrage dynamique de charge pour la décomposition de domaine appliquée à la méthode des éléments discrets seront présentées. Ces méthodes se distinguent par leur granularité. La première est

basée sur la modification dans le temps de la décomposition de domaine, où chaque processus se voit attribuer un nouveau sous-domaine dont la taille dépend de la puissance de ce processus ainsi que du nombre de particules. Cette méthode équilibre la charge au niveau des particules, mais nous verrons qu'il est possible de procéder à cet équilibrage selon une granularité plus grossière, ce qui nous conduira à une deuxième méthode. Enfin, le prochain chapitre sur la mémoire partagée introduira l'équilibrage implicite au niveau des boucles.

6.2 Implantation de l'équilibrage dynamique de la charge

Avec le paradigme SPMD, où chaque processeur possède un sous-ensemble de particules en fonction de la décomposition de domaine, il existe deux façons d'augmenter la charge moyenne de travail : l'équilibrage global ou local. Effectuer ce type de travail globalement est complexe et implique beaucoup trop de communications. Il faut redéfinir toutes les frontières et échanger des données avec tous les voisins. À la fin de l'opération, la charge de travail de tous les processus est équivalente à la charge moyenne. Il y a un grand intérêt à effectuer cette vérification le moins souvent possible puisqu'elle coûte très cher. Heureusement, des milliers d'itérations sont souvent nécessaires avant qu'un véritable déséquilibre ne survienne pour un système où la charge a été bien distribuée au départ.

Dans le cas d'une décomposition géométrique de domaines pour des systèmes particulaires, l'équilibrage local est plus approprié. D'abord parce que, de par la nature discrète du problème, il est très difficile d'obtenir un parfait équilibre en utilisant la technique de l'équilibrage global. De plus, le coût de l'équilibrage ne dépend pas du nombre de processeurs parce que la vérification ne se fait qu'avec les voisins directs. Le but de cette politique est d'améliorer l'utilisation des ressources dans l'immédiat. Plus précisément, réduire l'importance des cas extrêmes : en soulageant les processus plus

occupés et en donnant progressivement plus de travail aux moins occupés. En pratique, minimiser la charge de travail du processeur le plus occupé est suffisant (à court terme) et devrait réduire le temps calcul lors des prochaines itérations. Quelques appels seulement sont nécessaires pour réduire le déséquilibre à un niveau acceptable. De plus, il est possible d'imiter un rééquilibrage global avec plusieurs appels successifs à l'équilibrage local. Généralement, cette technique est efficace pour gérer de petits déséquilibres, mais nécessite des vérifications fréquentes.

6.2.1 Vérification 3D de la charge

L'équilibrage dynamique de la charge de travail débute par une vérification, celle de s'assurer, à une certaine fréquence, que la charge de travail individuelle de chaque processeur est équivalente à la charge de travail moyenne. Comparer les temps d'exécution moyen pour un nombre d'itérations reste la meilleure technique pour comparer le taux d'utilisation de chaque processeur, même si elle ne tient pas compte de facteur externes tels que les entrées/sorties réseaux. Si cette information n'est pas disponible, le nombre de contacts peut donner une approximation acceptable de la charge à condition que la grappe soit homogène.

L'équilibrage local minimise les communications puisque chaque processeur compare sa charge avec celle de ses voisins immédiats. Les décisions relatives au déplacement des frontières sont évaluées pour chaque sous-domaine et doivent être compatibles avec les décisions des voisins immédiats. Toutefois, dans le cas d'une décomposition multidimensionnelle, tous les processeurs alignés sur un même axe doivent se concerter, sinon la communication des halos pour la nouvelle décomposition deviendrait très complexe (Figure 6-1).

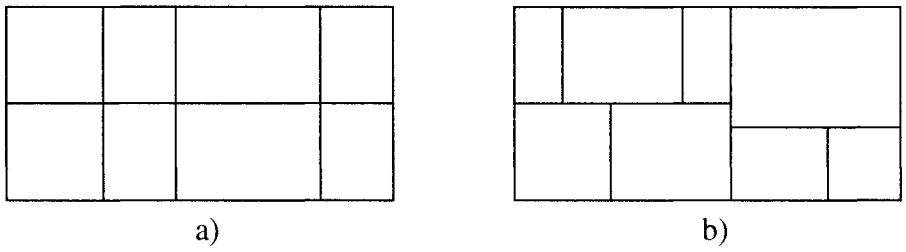


Figure 6-1 Exemple de décompositions 2D
a) compatible avec la stratégie d'échange SHIFT
b) incompatible avec la stratégie d'échange SHIFT

Tous les processeurs de deux tranches voisines doivent donc arriver à la même conclusion : réduire la charge du processeur le plus occupé des deux sous-groupes. Des opérations collectives permettent ensuite d'échanger rapidement l'information pertinente, peu importe le nombre de processeurs par tranches.

Enfin, nous devons spécifier un écart acceptable par rapport à la charge moyenne, sur lequel le mécanisme de vérification se base pour détecter un déséquilibre. Puisque l'équilibrage est local, on ne connaît pas directement la situation par rapport à la charge moyenne. Pour éviter une situation en escalier (Figure 6-2), on doit ajuster dynamiquement la valeur de l'écart acceptable en fonction de la charge locale et de la charge moyenne.

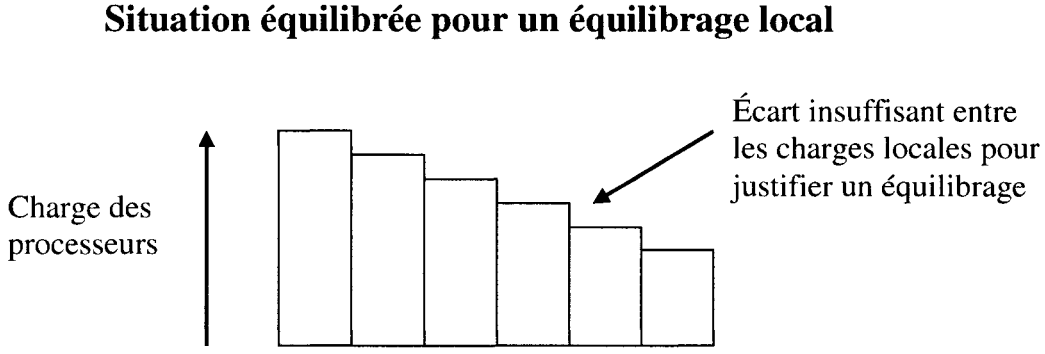


Figure 6-2 Problématique de la définition d'un écart par rapport à la charge moyenne

6.2.2 Information échangée lors d'un déplacement de frontières

Lors d'une vérification de la charge, la décision d'un processeur doit être l'inverse de celle de son voisin. En cas de déséquilibre, les processeurs ayant décidé de modifier leurs frontières doivent communiquer. Puisqu'il s'agit de particules changeant de sous-domaine (5.9.2), chaque transfert implique l'envoi de trois messages :

- Les particules changeant de sous-domaine;
- Les contacts inter-particulaires;
- Les contacts particules-triangles.

6.3 Efficacité du mécanisme de rééquilibrage de la charge

Le rééquilibrage basé sur le déplacement de frontières est une fonctionnalité essentielle pour un code de simulation d'écoulements granulaires à l'aide la DEM. Il est à noter que son utilisation n'est pas toujours bénéfique puisque tout rééquilibrage implique un coût qui peut nuire aux performances de façon plus ou moins significative selon les cas.

La vérification fait partie des opérations relatives à la parallélisation et est inexistante dans la version séquentielle (*TCasParticulier*). De plus, il faut choisir la fréquence de vérification qui dépend du problème à résoudre. Un mécanisme automatique permettant d'ajuster la fréquence de vérification serait souhaitable mais n'est pas du tout facile à mettre au point. Notre approche dans ce travail a prôné la flexibilité. Par exemple, pour un système où le mouvement général se fait vers le bas selon l'axe des z (par exemple, une sédimentation de particules), l'algorithme vérifiera de moins en moins souvent la charge dans les autres directions car un déséquilibre dans ces directions est peu probable.

L'expérience a démontré qu'il ne fallait pas déplacer les frontières de façon drastique. Cette opération doit se faire graduellement pour éviter des situations fâcheuses. La figure 6-3 illustre un événement courant qui arrivait lors des premiers tests.

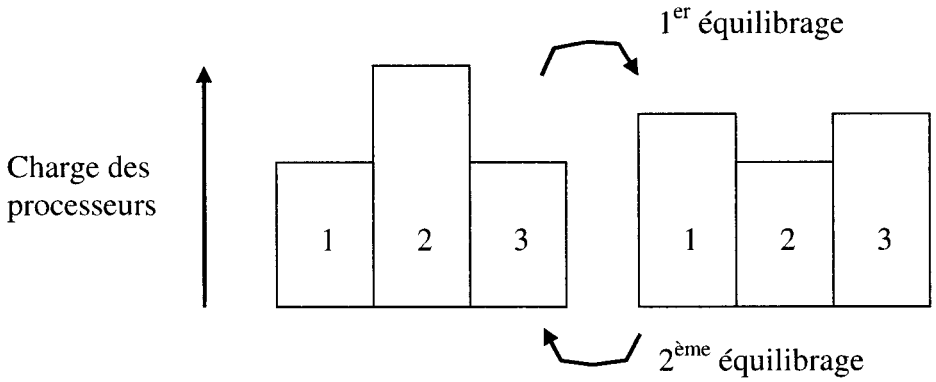


Figure 6-3 Exemple de cas pathologique pour l'équilibrage de charge

Il s'agit du cas d'un processeur (#2) ayant une plus grande charge de travail que ces deux voisins immédiats (#1 et #3). Réalisant qu'il est surchargé, il modifie ses frontières et communique de façon appropriée avec ses voisins. Malheureusement, après l'opération, les deux voisins se retrouvent avec significativement plus de particules. À la prochaine occasion, ils retourneront les particules au processeur (#2) qui se retrouvera à nouveau surchargé.

Ce dernier exemple montre comment peuvent survenir les oscillations néfastes qui contribuent à réduire l'efficacité de la parallélisation. Mentionnons que ces oscillations sont inévitables si tous les échanges se font simultanément. La solution implantée consiste à alterner les opérations d'équilibrage. Au besoin, on peut faire jusqu'à 6 appels à la fonction *load-balancing*. Procéder ainsi augmente évidement les coûts de vérification, mais il s'agit du prix à payer pour éviter ces oscillations. Dans cet exemple, modifier la largeur de la zone transférée serait une solution efficace. Toutefois, à cause de la nature discrète des données, il est difficile d'ajuster automatiquement cette bande et d'extrapoler une largeur à partir de temps de calculs des itérations précédentes.

Trouver le banc d'essai idéal pour vérifier l'efficacité de nos algorithmes d'équilibrage de charge ne fut pas simple. Dans les cas pratiques où le déséquilibre est néfaste, l'équilibrage dynamique ne réussit pas toujours à améliorer les performances de façon significative. La sédimentation de particules est une simulation idéale parce que la charge varie rapidement. Il est évident que le processeur dédié au sous-domaine le plus bas deviendra surchargé si aucun mécanisme ne se s'occupe de déplacer les frontières. Cet exemple comporte quand même quelques défauts.

Si on fait disparaître le mur du bas qui supporte l'ensemble de particules sédimentées, celles-ci débutent leur descente. Avec l'équilibrage dynamique, la frontière monte vers le haut! Ce phénomène est dû à l'entassement de particules qui se dilate. Le nombre de contacts interparticulaires et les temps de calcul sur le processeur du bas diminuent; il demande alors des particules à son voisin plus haut. Bien sûr, après un certain temps, le déplacement des frontières finit par s'effectuer vers le bas pour suivre le mouvement général des particules. Malheureusement pendant cette période, les particules sont en chute libre et ne se percutent que rarement. Ainsi, les temps de calculs sont minuscules et par conséquent, l'efficacité de la parallélisation chute elle aussi. Pour pallier à ce problème, les particules sont collées et ne peuvent bouger qu'en direction du bas. De cette façon, la détection et la résolution des contacts ont un coût constant pendant toute la durée de la simulation, ce qui permet de mieux évaluer notre mécanisme de rééquilibrage dynamique.

Nous avons tenté de définir la fréquence idéale de vérification pour la chute d'un bloc de 32000 particules (20x20x80). Nos résultats montrent clairement qu'il vaut mieux un système de rééquilibrage qui vérifie trop souvent que pas assez (Figure 6-4). Avec 4 processeurs, nous obtenons une accélération maximale de 3,44 avec une fréquence de 20 (rééquilibrage à toutes les 20 itérations). Le coût de vérification pour une décomposition 1D est faible car il est à peine plus coûteux de vérifier à chaque fois que l'occasion se présente. À partir d'un multiple supérieur à 30, le système ne peut toutefois rattraper les fluctuations et le déséquilibre affecte considérablement l'efficacité. La deuxième courbe

reprend le même test pour une décomposition 2D. Dans cet exemple, la décomposition multidimensionnelle souffre d'un inconvénient supplémentaire : la forme du bloc est trop allongée. De plus, avec une fréquence élevé (<10) cette perte de performance provient de la vérification de la charge en 2D qui est plus coûteuse car il doit y avoir un accord entre les processus de chaque tranche, donc plus de communications.

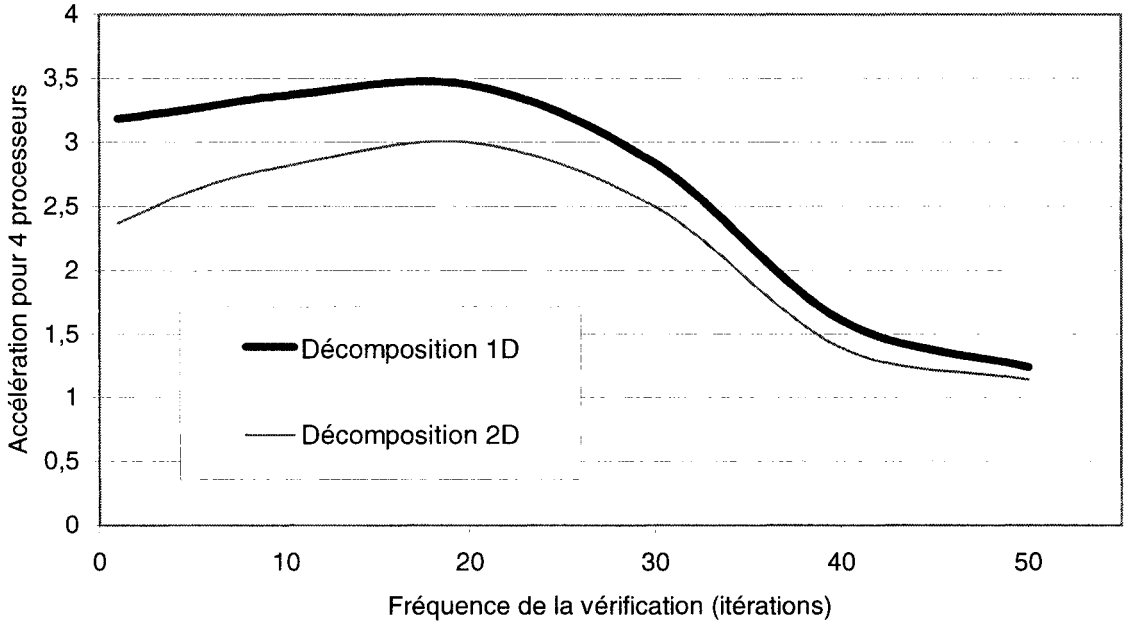


Figure 6-4 Accélération en fonction de la fréquence de vérification du rééquilibrage de charge (bloc de 32000 particules en chute libre, dans un domaine de hauteur équivalente à 4 fois la hauteur du bloc (figure 6.5))

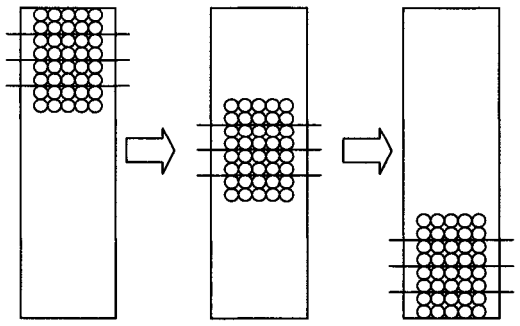


Figure 6-5 Représentation du déplacement de frontières pour le problème de la chute d'un bloc

6.4 Équilibrage de charge et décomposition multidimensionnelle

Une autre situation moins courante peut se produire et rendre l'équilibrage dynamique de charge complètement inutile. Imaginons un carré où les particules sont réparties dans les deux coins opposés. Essayons maintenant de décomposer ce domaine en 4 sous-domaines (Figure 6-6). Le taux d'utilisation des processeurs sera nul pour deux processeurs tandis que les deux autres feront tout le travail (a). La charge moyenne tout comme l'efficacité de la parallélisation seront de 50%. Dans un tel cas, l'équilibrage dynamique basé sur la vérification locale aboutirait dans une impasse car le système semble équilibré dans toutes les directions. Un système plus « intelligent », capable de déplacer deux frontières à la fois, pourrait au mieux réduire le nombre de processeurs inutilisés à 1 (b).

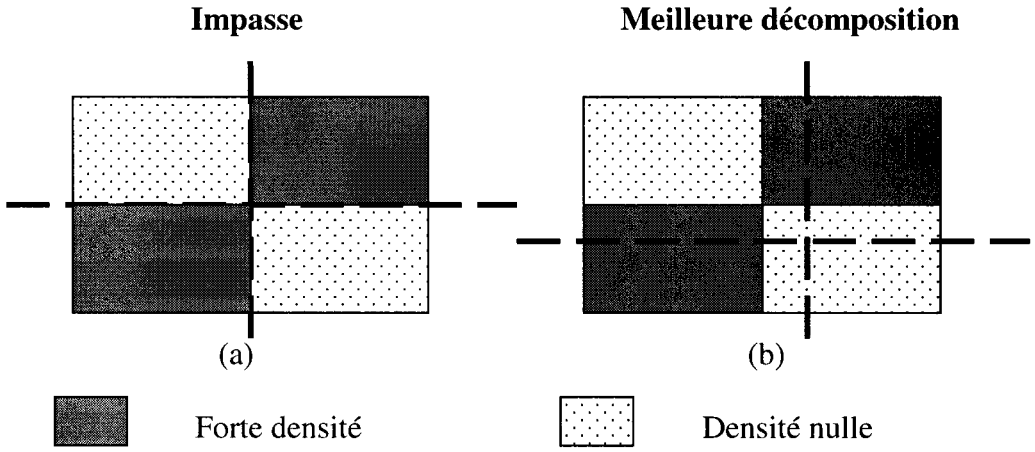


Figure 6-6 Exemple où un système d'équilibrage de charge demeure dans une impasse

Évidemment, dans cette situation artificielle, une décomposition 1D aurait suffit. Ce cas illustre tout de même un problème qui pourrait survenir dans le cas de géométries irrégulières. Nous avons démontré au chapitre 5 qu'il est souvent avantageux d'utiliser

une décomposition multidimensionnelle. Cependant, on doit tenir compte de la perte d'efficacité occasionnée par des processus en « chômage forcé ».

L'équilibrage de la charge par déplacement de frontière peut donc être inefficace dans le cas d'une décomposition de domaine multidimensionnelle. Alors qu'elle semble évidente et efficace pour une décomposition 1D, nous venons de démontrer qu'elle pose quelques difficultés pour les domaines avec des concentrations irrégulières de particules. Heureusement, il existe d'autres mécanismes pour rééquilibrer la charge.

6.5 Stratégie d'équilibrage pour des domaines irréguliers

L'exemple de la section précédente a illustré que le rééquilibrage dynamique de charge peut être compliqué pour des domaines de calcul irréguliers. Alors, que faire par exemple dans le cas de géométries concaves où des processeurs seront éventuellement assignés à des zones vides? La littérature reste assez muette sur cette problématique appliquée à la méthode des éléments discrets. Cependant, Henty (2000) mentionne une autre façon de réaliser le rééquilibrage dynamique de charge sans modifier les frontières initiales. L'idée générale est de décomposer le domaine afin que le nombre de processus soit plus élevé que le nombre de processeurs. On espère alors qu'en générant assez de petits blocs à forte concentration, ils se retrouveront sur des processeurs qui étaient inutilisés auparavant. Les sous-domaines vides seront eux aussi plus nombreux mais leur charge pratiquement nulle ne devrait pas nuire de façon significative à la performance. Cette création de processus supplémentaires a toutefois plusieurs inconvénients :

- Mémoire requise plus grande, d'où l'importance de « rationner »;
- Plus de communications (heureusement, certaines sont nulles);
- Gestion d'un nombre plus élevé de processus par le système d'exploitation.

Tous ces inconvénients seront toutefois rapidement oubliés si cette stratégie permet une redistribution efficace de la charge et que le gain en performance compense pour l'effort supplémentaire dû à l'augmentation du nombre de processus. Il ne faut pas oublier que sans cette stratégie, un pourcentage relativement important des processeurs peut être sous-utilisé (jusqu'à 50%). On peut donc envisager une meilleure utilisation de tous les processeurs s'il y a un grand nombre de petits processus qui s'exécutent au même moment. La prochaine section discute de l'implantation et des conditions à respecter pour rendre cette technique efficace.

6.5.1 Décomposition excessive du domaine

La décomposition cyclique du domaine « *Block cyclic decomposition* » de Henty (2000) reprend indirectement l'idée de maître-esclave avec de petits domaines telle que présentée à la figure 2-7. Toutefois, le rôle du processus maître est joué par le système d'exploitation qui, via l'ordonnanceur, distribue les tâches sur les processeurs libres. La plate-forme idéale pour un tel travail est l'ordinateur multiprocesseur puisqu'un seul système d'exploitation doit gérer les processus sur l'ensemble des processeurs.

Le premier test de performance relatif à cette technique a été de vérifier si la création de processus MPI supplémentaires dégradait la performance du code de calcul Powder3D. À titre de banc d'essai, nous avons utilisé une simulation parfaitement équilibrée. L'ordinateur utilisé (Hamsun) est un ordinateur multiprocesseur à 4 processeurs Xeon. Magnum peut être aussi utilisé car la charge est équilibrée, mais à condition d'occuper pleinement et équitablement les processeurs de chaque noeud. Le graphique de la figure 6-7 nous indique que la dégradation est de plus en plus importante à mesure que le nombre de blocs (sous-domaines) augmente par rapport au cas standard

de 1 bloc par processeur. Le dédoublement du calcul des forces est de plus en plus important et le nombre croissant de messages sur le réseau amplifie le phénomène.

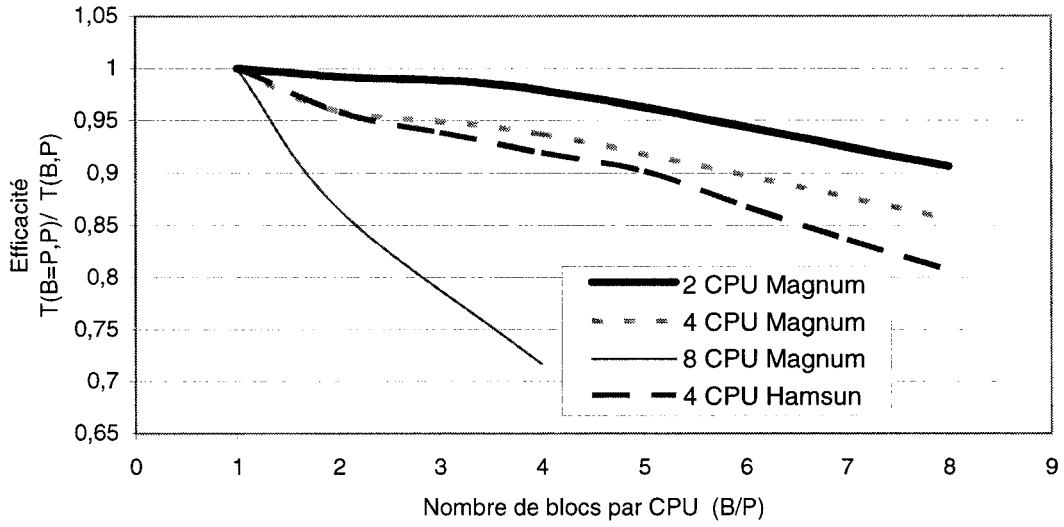


Figure 6-7 Graphique de l'efficacité en fonction du nombre de processus

Il peut paraître étrange que la courbe soit moins intéressante sur le Xeon que sur deux nœuds de Magnum alors que le serveur Xeon a accès à des communications quasi-instantanées entre les processus MPI. Nous attribuons la faute au nombre exagéré de processus qui encombrent un seul système d'exploitation.

Notre deuxième test, pratique cette fois-ci, reprend le problème de la chute libre sur un seul nœud quadriprocesseur de Hamsun. En doublant le nombre de processus, on crée 8 sous-domaines qui décomposent équitablement l'espace total. De cette façon, à tout moment, il y a 4 processus toujours très occupés et 4 autres qui ne le sont pas du tout (Figure 6-8). Le temps total devrait alors diminué d'un facteur 2 par rapport à une

simulation sans équilibrage de la charge. À la section 6.5.4, on comparera cette approche à la méthode classique du déplacement de frontières.

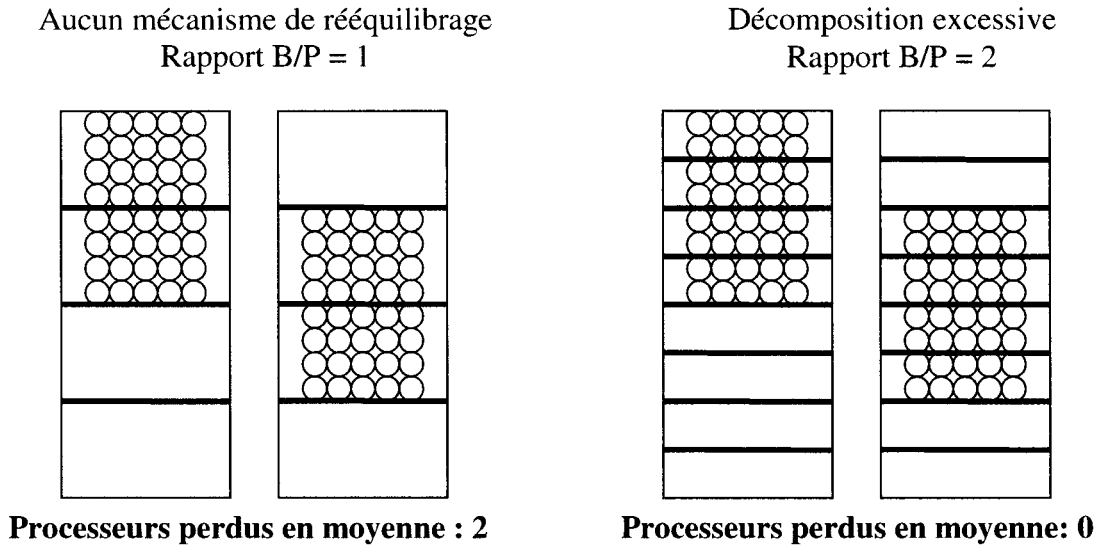


Figure 6-8 Exemple de décomposition excessive pour un cas pratique

Evidemment, dans cette situation, tous les processeurs sélectionnés ont un nombre identique de processus supplémentaires. Cependant, sur une grappe d'ordinateurs à mémoire distribuée, rien n'assure a priori que tous les nœuds sont bien occupés. Par exemple sur Magnum, imaginons un déséquilibre majeur sur 4 processeurs (2 nœuds SMP). Si les deux processus occupés sont sur le même noeud, la stratégie de décomposition excessive sera totalement inutile. Elle séparera les deux processus occupés à 100% en 4 processus à 50% et il n'en résultera aucun gain.

La granularité inhérente à la décomposition excessive est très différente de celle inhérente au rééquilibrage par déplacement de frontière. Au lieu d'équilibrer la charge des processus en jouant sur la taille des sous-domaines, la décomposition excessive vise à équilibrer la charge des noeuds en leur associant un nombre de processus de façon à obtenir un taux d'utilisation le plus près possible de 100%. Deux approches sont alors

disponibles pour effectuer le partage des processus. La première méthode, dite statique, sera présentée mais n'a pas été implantée. Elle nécessiterait beaucoup trop de changements et resterait limitée à quelques types de simulations. L'autre méthode, dite dynamique, est plus naturelle et beaucoup plus flexible, mais un peu moins performante.

6.5.2 Décomposition statique du domaine

La décomposition statique du domaine est possible si la géométrie est connue et ne change pas. Cette méthode a comme premier inconvénient d'être basée sur une décomposition définie par l'utilisateur. Une grille de dimension N par M doit être spécifiée dans un fichier de paramètres dédié à un groupe de travail de X*Y processus. Une matrice définie manuellement par l'utilisateur permet d'associer automatiquement plusieurs sous-domaines vides à un ordinateur et de distribuer équitablement les processus occupés sur les processeurs libres.

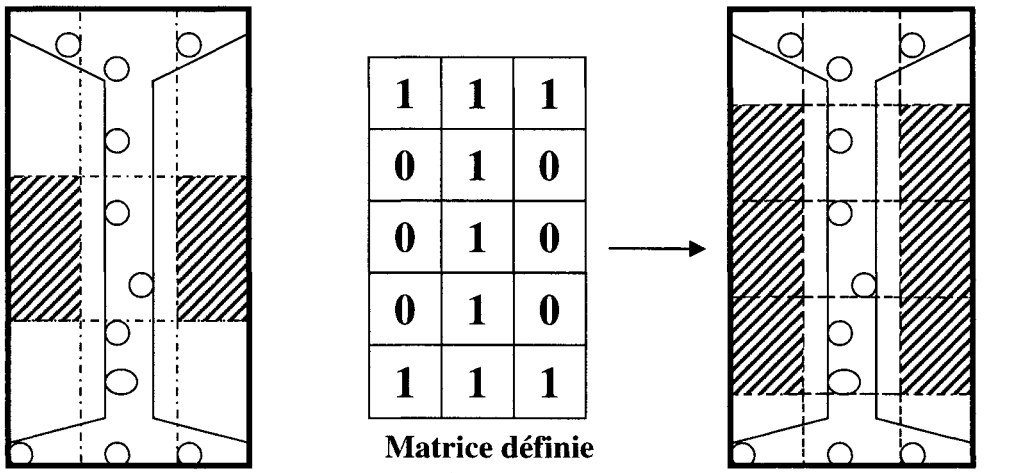


Figure 6-9 Exemple de décompositions pour 9 processeurs

a) Décomposition classique avec autant de processus que de processeurs
b) Utilisation de la décomposition excessive pour s'adapter aux géométries

La figure 6-9 montre que l'utilisation de la décomposition excessive statique peut être bénéfique. Dans cet exemple, les 9 processeurs sont utilisés pleinement alors que 2 sont perdus avec une décomposition standard à un processus par processeur. Cette technique est intéressante pour des géométries non mobiles de forme quelconque, mais beaucoup moins dans le cas de problèmes où la géométrie est variable ou la concentration de particules varie beaucoup à l'intérieur du domaine de calcul.

6.5.3 Gestion dynamique de processus sur une architecture à mémoire distribuée

Pour analyser la performance de la stratégie d'équilibrage basée sur la décomposition excessive du domaine, nous avons effectué les premiers tests sur un seul noeud SMP Xeon. La raison est simple : sur une grappe Beowulf, rien n'assure que les processus occupés et nuls seront en tout temps bien répartis sur l'ensemble des noeuds. En fait, la décomposition excessive est possible sur les systèmes à mémoire distribuée grâce à des outils spécialisés permettant d'émuler un ordinateur à mémoire partagée. Les deux produits évalués dans le cadre de ce travail sont BProc et OpenMosix, et c'est ce dernier qui fut choisi pour les tests.

OpenMosix (www.openmosix.org) est un ensemble logiciel gratuit permettant de créer une super machine virtuelle à partir d'une grappe Beowulf. Initialement, il était la version à code source libre de l'application *Mosix*. Maintenant, il est développé de façon indépendante et intègre de nouvelles fonctionnalités uniques. OpenMosix est une modification au noyau du système d'exploitation Linux permettant la migration des processus à travers la grappe. Par exemple, sur un « Magnum *OpenMosix* », on peut lancer 24 processus sur le nœud principal. Le système intégré détectera la surcharge et 22 d'entre eux seront migrés vers les autres noeuds non utilisés. Cette opération comporte évidemment des coûts d'entretien et de vérification mais permet, du moins en théorie, une utilisation optimale des nœuds de la grappe.

Le principal avantage d'OpenMosix est qu'il n'implique aucune modification au code et est compatible avec MPI. De plus, Openmosix définit son propre système de fichiers partagés, *omFS* qui permet la gestion de fichiers et garantit la cohérence des données lorsqu'il y a migration de processus.

Pour maintenir des charges équilibrées, un ordonnanceur permet la migration de processus à travers la machine virtuelle. Les critères pour le rééquilibrage sont paramétrables par l'administrateur. Selon le type d'application, OpenMosix peut soit tenter de minimiser les entrées/sorties ou soit maximiser le taux d'utilisation des processeurs.

6.5.4 Analyse de performance

Pour vérifier concrètement l'efficacité d'OpenMosix dans le contexte de notre stratégie de rééquilibrage par décomposition excessive, le test de la chute unidirectionnelle de particules présenté à la section 6.3 a été effectué sur Magnum. Deux nœuds ont été reconfigurés pour supporter OpenMosix; quatre processeurs entrent en jeu tout comme sur le serveur Xeon. Nous avons comparé la technique de la décomposition excessive avec celle basée sur le déplacement de frontière (Figure 6-10). Les tests ont donc été effectués sur un vrai système SMP (Hamsun) et un système SMP virtuel sous OpenMosix (Magnum).

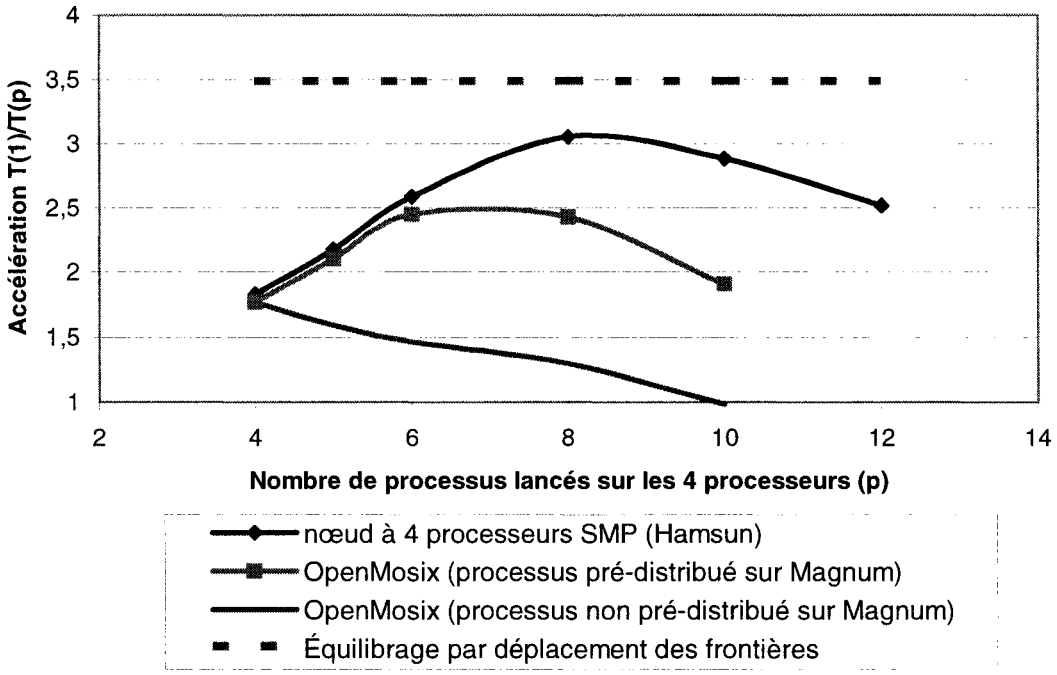


Figure 6-10 Équilibrage de charge par les techniques de la décomposition excessive et de déplacement de frontières

Avec 4 processus, on sait que, cumulativement, seulement 2 processus seront actifs au même moment (Figure 6-8); pas de surprise puisque nous obtenons une accélération se situant autour de 1,8. Nous constatons que la décomposition excessive avec 8 processus sur 4 processeurs Xeon de Hamsun est relativement performante. Comme prévu, l'accélération est inférieure à celle obtenue avec le rééquilibrage par le déplacement de frontière en raison de la nature du problème simulé (sédimentation de particules). Ce résultat est très encourageant car nous savons que la stratégie de la décomposition excessive peut aussi être utilisée avec des décompositions multidimensionnelles qui sont mieux adaptées aux géométries complexes.

L'analyse des courbes de la figure 6-10 confirme que ce système de rééquilibrage par décomposition excessive n'est pas encore tout à fait au point. Lorsque les processus sont pré-distribués sur les deux machines, le hasard fait bien les choses et l'accélération

ne subit qu'une petite perte d'efficacité inhérente à un système plus complexe. Cependant, lorsque tous les processus sont démarrés sur le nœud principal (cas non-distribué), l'efficacité chute de façon importante à mesure que le nombre de processus augmente. Enfin, une tendance commune à toutes les courbes est leur fléchissement expliqué par le coût croissant du dédoublement des calculs et des communications.

Pour mieux comprendre ces résultats, il faut regarder en détail comment OpenMosix équilibre et migre les processus sur les machines de la grappe de calcul. Le problème vient du but premier d'OpenMosix, c'est-à-dire migrer des processus de façon totalement transparente. Le système se charge de tout et il n'y a aucune coopération avec les processus migrés. De cette façon, le programme n'a pas besoin d'être recompilé et lorsqu'un processus se déplace, tout doit se faire de façon transparente. Pour y arriver, OpenMosix doit gérer toutes les entrées/sorties du processus migré. Le système de fichier omFS est d'ailleurs là pour ça. Par contre, les communications réseaux doivent être gérées de façon transparente. Pour chaque envoi de messages par un processus migré, les messages sont acheminés au nœud de départ pour ensuite être réacheminées au processus receveur qui lui aussi peut avoir migré. Deux intermédiaires peuvent être nécessaires avant d'atteindre le véritable destinataire, ce qui ajoute au temps système (overhead) de l'application. Puisque Powder3D est une application intensive au niveau des communications, l'utilisation d'OpenMosix a un impact significatif sur les performances obtenues. Pour ces raisons, même si la décomposition excessive semble prometteuse, nous ne conseillons pas son utilisation sur les grappes de calcul de type Beowulf gérées par OpenMosix.

6.5.5 BProc : outil de gestion de processus sur des grappes de calculs à mémoire distribuée

BProc (Beowulf Process space) est un autre système permettant la migration de processus. Même si son rôle semble similaire à OpenMosix, son mode d'opération ne

l'est pas. BProc permet la gestion d'une grappe de calcul avec une seule image contenue sur le noeud principal. Lors du démarrage, les nœuds de calcul sont appelés à obtenir le système d'exploitation via le réseau. BProc permet de créer un espace de travail unique où l'on peut observer tous les processus actifs sur la grappe. La grande différence avec OpenMosix, c'est que la migration n'est pas transparente. BProc redéfinit quelques fonctions systèmes (fork par exemple) et le processus migré connaît son emplacement dans la grappe. Il peut alors communiquer directement avec son voisin dans la topologie. C'est évidemment très avantageux pour toutes les application qui doivent communiquer constamment. Les temps systèmes sont presque négligeables et par conséquent, il s'agit d'une plate-forme idéale pour les applications de calcul de haute performance. Grâce à son *démon* unique (ce qui limite les conflits probables), BProc est extrêmement rapide : la création de 15000 processus se fait en moins de 3 secondes (Hendricks, 2002). BProc propose une API pour la migration de processus, mais aucun système n'est responsable de l'équilibrage des processus. Pour y arriver, on peut éventuellement utiliser des outils-logiciels du domaine public. ClubMask (clubmask.sourceforge.net) combine l'ordonnanceur MAUI (supercluster.org/maui) à BProc et pourrait créer, du moins en théorie, un système approprié pour l'exécution en mode parallèle de Powder3D.

Après toutes ces louanges, il peut être surprenant qu'aucun test n'ait été effectué avec BProc. À l'instar d'OpenMosix, BProc ne requiert pas de modification au niveau du code source et seulement une recompilation est requise. Malheureusement, Mpich doit être aussi recompilée pour utiliser les fonctions de BProc. Mpich fournit théoriquement une option de compilation pour supporter BProc, mais cette option demeure introuvable (version 1.2.5). Nous devions donc nous en remettre à des systèmes basés sur BProc incluant des versions pré-compilées de MPI (Scyld (www.scyld.com) et Clustermatic (www.clustermatic.org)). Clustermatic, étant gratuit, a été installé, mais l'impossibilité de compiler avec Mpich et le compilateur Fortran d'Intel nous ont forcés à abandonner cette idée.

7 Programmation parallèle en mémoire partagée

L'équilibrage dynamique de charge et des temps de transmission courts pour des communications interprocessus à l'intérieur d'un même nœud sont deux raisons pour lesquelles l'utilisation d'un ordinateur à mémoire partagée SMP peut être avantageux lors de simulations avec la DEM.

Comme Magnum possède 2 processeurs SMP par nœud, le développement d'une paralélllisation hybride de la DEM, basée sur MPI et OpenMP, a été envisagé dans ce travail. Le compilateur Fortran 90 d'Intel supporte les directives d'OpenMP et c'est ce compilateur qui a été utilisé sur Magnum. Ce chapitre est consacré à l'exploitation efficace des ordinateurs multiprocesseurs de différentes architectures (Intel SMP, IBM P630 et P690) afin de vérifier si ce mode peut devenir avantageux pour la parallélisation de la DEM.

Avant de spécifier les détails de l'implantation OpenMP de la DEM, voici un bref rappel de quelques directives qui doivent être suivies pour effectuer une parallélisation en mémoire partagée efficace d'une application :

1. Déclarer les variables de travail temporaires privées;
2. Faire attention aux modifications de variables partagées, deux accès concourants pouvant entraîner la perte d'information. Il faut prévoir un mécanisme qui préserve l'atomicité des opérations;
3. Paralléliser uniquement les sections qui valent la peine. La distribution d'une boucle trop simple ou trop courte n'est généralement pas rentable. Il faut penser que la création d'un nouveau processus léger (thread) et l'allocation des variables privées entraînent des coûts.

7.1 Implantation de OpenMP

Plusieurs modifications ont été requises pour exploiter OpenMP mais comme pour le cas de la parallélisation par décomposition de domaine, le programme peut toujours s'exécuter en mode séquentiel. Pour déterminer où il fallait ajouter les directives OpenMP compilées, il fallait identifier les sections les plus prometteuses. Puisque la plupart des boucles dépendent du nombre de particules, cette version de Powder3D sera d'autant plus performante que la taille du problème sera grande.

La version de Powder3D avec détection optimisée (voir section 4.2) contient les boucles suivantes :

- Boucle de création de la liste des voisins;
- Boucle de détection des contacts en fonction de la liste des voisins;
- Boucle de calcul des forces;
- Boucle de calcul du mouvement des particules.

Ces boucles ont été initialement sélectionnées pour la parallélisation avec OpenMP parce qu'elles ont été jugées suffisamment complexes. La boucle de création de la liste des voisins a finalement été soustraite à l'ensemble car elle n'est pas cumulativement assez importante. Les autres boucles ont été parallélisées normalement, en faisant attention de déclarer les variables privées (environ une vingtaine) et de forcer certaines opérations critiques à s'exécuter en mode d'exclusion mutuelle.

Les premiers tests sur deux processeurs n'ont pas offert des taux d'efficacité raisonnables. Une revue de littérature nous a permis de découvrir un texte fort intéressant sur la parallélisation d'un code d'éléments discrets avec OpenMP (Labarta et Henty, 1999). Cet article décrit l'implantation d'un code parallèle et les difficultés reliées au paradigme de mémoire partagée. Il est mentionné que le goulot d'étranglement de la méthode des éléments discrets se situe au niveau de la sommation des forces de collision. En effet, il peut y avoir conflit lorsque deux processus veulent modifier la

même entrée du vecteur des forces (voir pseudo-code ci-dessous). Nous présentons ici trois approches qui garantissent la cohérence de ce vecteur lors de la sommation des forces. La boucle en question a la forme suivante :

Pour 1 à N (Nombre de contacts)

Particle1 = contact.particle1

Particle2 = contact.particle2

Force = Calcul_force(Particle1, Particle2)

Force(Particle1) = Force(Particle1) + Force

Force(Particle2) = Force(Particle2) - Force

Puisque des références à la *Particule2* pourraient se retrouver sur plus d'un processus parallèle, il faut empêcher la perte d'information qu'occasionneraient deux accès simultanés au vecteur *Force*. La première méthode, la plus simple, est de créer une zone d'exclusion mutuelle à l'aide des directives *CRITICAL* ou *ATOMIC*. La cohérence est assurée mais les processeurs ne font qu'attendre que le verrou soit levé. Les tableaux 7.1 et 7.2 ci-dessous montrent les données recueillies par l'outil de profilage (gprof) pour les exécutions en séquentiel et en parallèle sur un nœud de Magnum. Même si cet outil n'est pas tout à fait approprié pour étudier un programme parallèle (à cause du nombre d'appels différent dans chaque cas), il est clair que l'implantation OpenMP sur Magnum souffre sérieusement du problème des verrous. Le temps perdu à attendre la levée du verrou (fast_lock) annule la réduction des temps de calcul de *pow.A* et *contact_resolution* obtenue grâce à la parallélisation.

Tableau 7-1 Profil de l'exécution séquentielle de Powder3D (Sédimentation 5000 particules)

%	Self		
time	Seconds	calls	Name
27.91	**54.28**	800	contact_resolution_
15.09	**29.35**		pow.A
6.33	12.31		FixFree
5.88	11.44		Alloc64
5.63	10.95		__mcount_internal

Tableau 7-2 Profil de l'exécution parallèle en mémoire partagée de Powder3D (Sédimentation 5000 particules)

%	Self		
time	Seconds	Calls	Name
42.92	**142.45**		**__fast_lock**
11.64	**38.63**	1293	_contact_resolution__103__par_region0
6.42	21.29		__mcount_internal
5.84	**19.39**		pow.A
5.19	17.23		Alloc64

Une deuxième méthode, la réduction de vecteurs, permet de faire mieux. En déclarant un vecteur d'accumulation de forces privé pour chaque fil d'exécution, aucune barrière pour gérer la section critique n'est requise. Cependant, il faut prévoir un mécanisme pour combiner les vecteurs à la fin de la boucle. La réduction de vecteurs devait être standardisée dans la version 2.0 de OpenMP, mais aucune des implantations à notre disposition ne la supporte. Nous l'avons donc implantée manuellement.

Une troisième méthode, qui constitue la contribution principale de Labarta et Henty(2002), est l'élaboration d'un agent *inspecteur/exécuteur*. Lors de la première itération, l'inspecteur stocke les conflits potentiels et définit des intervalles où aucune section critique n'est nécessaire. Par la suite, à l'exécution, les verrous *CRITICAL* sont utilisés selon les besoins. Cette méthode dynamique permet de gérer les quelques cas problématiques sur les millions d'opérations reliées à la sommation des forces de contact. Les résultats obtenus sont étonnants, l'*inspecteur/exécuteur* offre un taux d'efficacité supérieur, indépendamment de l'architecture (CRAY T3D, SUN

UltraSparc2 et COMPAQ SMP). Il serait tentant d'implanter une technique semblable pour accélérer la sommation des forces. Cependant, comme nous le verrons à la section 7.3, les gains de performances escomptés en fonction de sa complexité d'implantation sont plutôt faibles et nous avons préféré utiliser seulement la méthode de la réduction et du verrou.

7.2 Optimisation de la version OpenMP de Powder3D

Une fois l'implantation de base OpenMP réalisée, on doit généralement optimiser plus finement le code. En effet, la programmation dite « naïve » par la parallélisation des boucles donne rarement des résultats exceptionnels (Krawezik et Cappello, 2003). L'utilisation de directives spécialisées est conseillée. Certains changements, parfois très simples, peuvent permettre de diminuer le coût des verrous. Powder3D a été restructuré en fonction de OpenMP pour permettre l'exécution parallèle de processus légers. Puisque les opérations à distribuer doivent être suffisamment complexes, il a été nécessaire de créer des boucles plus longues. Par exemple, la détection des contacts et le calcul des forces ont été fusionnés. Désormais, le code comprend trois boucles principales :

- Boucle de création de la liste des voisins;
- Boucle de résolution des contacts;
- Boucle de calcul du mouvement;

et le pseudo-code de la boucle de résolution des contacts est donné par :

```
Pour 1 à N (Contacts Potentiels)
        Particle1 = contact_potentiel.particle1
        Particle2 = contact_potentiel.particle2
    Si ( collision) alors

        Force = Calcul_force(Particle1, Particle2)

        /* MECANISME D'EXCLUSION */
        Force(Particle1) = Force(Particle1) + Force
        Force(Particle2) = Force(Particle2) – Force
        /* MECANISME D'EXCLUSION */
```

Par exemple, voici quelques chiffres typiques pour ces opérations. La recherche par quadrillage identifie premièrement 200 000 paires de particules rapprochées. Ensuite, environ 60 000 sont sélectionnées pour créer une liste temporaire de contacts potentiels, parmi lesquels il peut y avoir jusqu'à 20 000 contacts réels. Procéder ainsi est très avantageux pour un programme parallèle en mémoire partagée. Pendant que certains processus parcourent la liste de collisions potentielles, un autre processus peut écrire dans le vecteur de forces sans avoir à attendre la levée du verrou. Le nombre de conflits en écriture est diminué.

Pour optimiser davantage, on doit aller jusqu'à restructurer l'ordre d'exécution de certaines fonctions. Grâce aux balises de sections (SECTION, SECTIONS), il a été possible d'effectuer la détection de contacts particule/particule et la détection de contacts particule/mur à l'aide de deux processus légers indépendants. Cette astuce est évidemment limitée à deux processus, une situation idéale pour Magnum. Malheureusement, les temps de calcul pour ces deux opérations sont rarement identiques, ce qui fait en sorte qu'un des 2 processeurs peut demeurer inutilisé pendant une longue période. On peut faire mieux en créant une seule région parallèle composée de plusieurs boucles consécutives. Ce travail a été effectué en collant l'étape de

résolution des contacts avec celle du calcul du mouvement. Cette optimisation a contribué à réduire le coût associé à la création de processus parallèles.

Enfin, un autre point important consiste à s'assurer que le travail est bien distribué et que la synchronisation des processus légers n'est pas trop coûteuse. En mémoire partagée, le rééquilibrage ne se fait pas par le déplacement de frontières. Certaines directives permettent de mieux distribuer le travail effectué dans les boucles. La parallélisation consiste à distribuer par blocs la liste de contacts potentiels aux processus légers. La directive *SCHEDULE* permet d'optimiser la répartition afin de sauver quelques millisecondes perdues lors de la synchronisation. Les politiques de répartition *DYNAMIC* et *GUIDED* ont été comparées mais aucune ne s'est démarquée suffisamment pour la conseiller. Cependant, la politique GUIDED ne demande pas d'intervention de la part de l'utilisateur. Elle s'adapte donc mieux à toutes les situations. Par contre, précisons que, puisqu'il s'agit de contacts potentiels, la distribution statique (STATIC) est proscrite car la charge de travail n'est pas forcément équilibrée entre deux blocs de longueur identique.

7.3 Résultats

L'efficacité d'une implantation OpenMP doit être évaluée lorsque le nombre de contacts est grand. En effet, pour espérer un gain, les boucles doivent être suffisamment longues. Pour des fins de comparaison, nous avons aussi effectué quelques tests sans protection (verrou) du vecteur des forces afin d'observer l'accélération maximale dans le cas utopique d'un mécanisme de verrou idéal.

En ce qui concerne Magnum, nos changements n'ont pas eu d'impact. Tout comme nos résultats précédents (Tableau 7-2), ces nouveaux tests ont confirmé que la

plateforme Intel SMP ne permet pas d'obtenir de résultats intéressants, peu importe la taille des boucles. En effet, lorsqu'il y a peu de contacts (Tableau 7-3 et 7-4), on constate que, comme prévu, le coût des verrous (_fast_lock) est relativement minime. C'est plutôt l'influence de la synchronisation des processus qui est importante (*kmp_wait_sleep*). Pour mieux comprendre les difficultés de la parallélisation OpenMP sur Magnum, il faudrait utiliser un outil tel le *Vtune Parallel Optimiser*^{TM} d'Intel. Cet outil pourrait fournir des informations plus précises sur l'exécution parallèle.

Tableau 7-3 Profil de l'exécution séquentielle pour un petit problème (5000 particules, 0 contact)

% time	Self seconds	Calls	Name
23.64	4.31	800	contact_resolution_
17.88	3.26		pow.A
6.36	1.16		Free64
5.81	1.06		Alloc64
5.54	1.01		FixFree
5.27	0.96		__mcount_internal

Tableau 7-4 Profil de l'exécution OpenMP pour un petit problème (5000 particules, 0 contact)

% time	Self seconds	Calls	Name
32.29	6.47		__kmp_wait_sleep
11.33	2.27	1252	_contact_resolution__103__par_region0
10.48	2.10		pow.A
7.63	1.53		__kmp_static_yield
5.99	1.20		__fast_lock
3.44	0.69		__kmp_x86_pause

Suite à ces résultats décevants, nous avons décidé de ne pas utiliser OpenMP sur les plateformes Intel. Heureusement, la portabilité de la version OpenMP de Powder3D nous a permis de faire des tests sur Polaris. Avec ses connections ultra-rapides et sa bande passante uniforme pour les 16 processeurs, nous nous attendions à une réduction des temps de synchronisation et une meilleure efficacité. Nous présentons au tableau 7.5

un résultat de profilage où nous pouvons constater que, contrairement à Magnum, le coût des verrous est très faible (pthread_*mutex_lock)* pour les sections critiques.

Tableau 7-5 Profil d'exécution sur Polaris en mode openMP (verrous CRITICAL, 500 000 particules)

%time	Seconds	Calls	Name
22.1	115.45	996193762	._pow [4]
18.9	98.50		.__mcount [6]
15.7	81.75	780641723	.loginner2 [7]
15.1	78.93	780641723	.expinner2 [8]
9.7	50.52	80	.contact_resolution [3]
2.9	15.03	184352000	._sin [9]
2.5	12.93	523456000	.modulus [5]
1.9	9.96	184352000	._acos [12]
1.9	9.91		.qincrement [13]
1.8	9.19	676608000	.cross_product [15]
1.1	5.53	80	.motion_eqns_calc [10]
0.4	1.96		.__stack_pointer [22]
0.3	1.78	3120008	.cvtloop [23]
0.2	1.16		**.call_pthread_init [25]**
0	0.02	3194476	**.pthread_mutex_lock[47]**

La courbe d'accélération pour une sédimentation de 390000 Particules, 1.2 millions de contacts est qualitativement acceptable (Figure 7-1). Cette implantation OpenMP permet d'obtenir une accélération de 2,6 pour 4 processeurs. C'est évidemment plus faible que la décomposition de domaine mais cette implantation en mémoire partagée avec OpenMP devrait mieux se comporter dans des situations réelles où la charge est déséquilibrée.

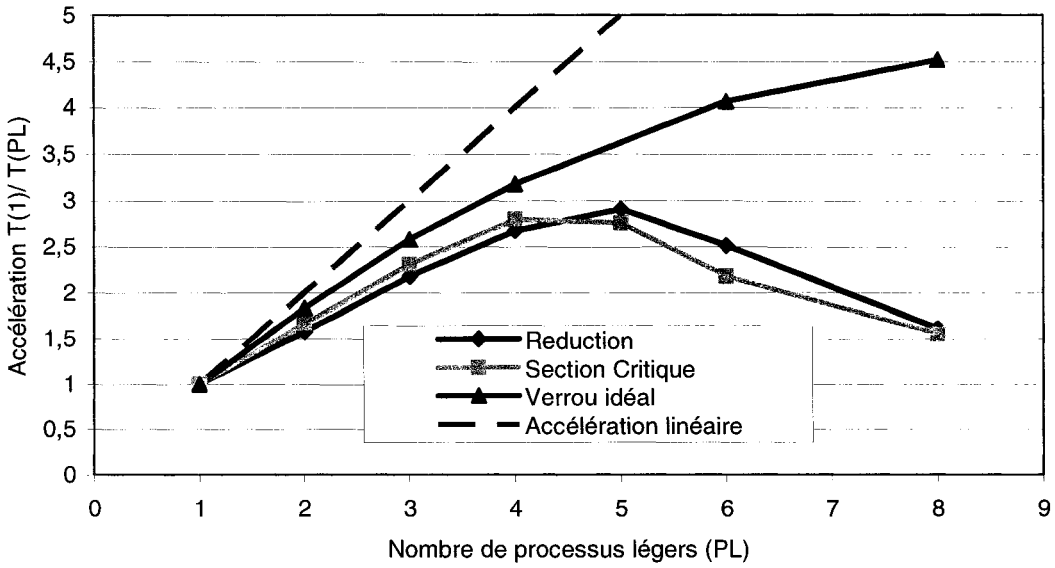


Figure 7-1 Accélération pour l'exécution en parallèle sur Polaris

Nous remarquons ainsi que l'utilisation de la réduction est plus performante que la section critique seulement lorsque le nombre de processus est élevé car il y a plus de conflits pour l'unique verrou. Avec peu de processeurs, la différence entre les versions avec ou sans verrous est petite. Pour cette raison, l'implantation d'un agent *inspecteur/exécuteur* ne nous semble plus primordiale. Sur la figure 7-2, nous avons comparé nous résultats à ceux obtenus par Henty (2000) pour les architectures Compaq et SUN. On voit que le P690 se comporte bien puisque l'efficacité de notre implantation se situe entre les deux meilleures courbes de ce dernier.

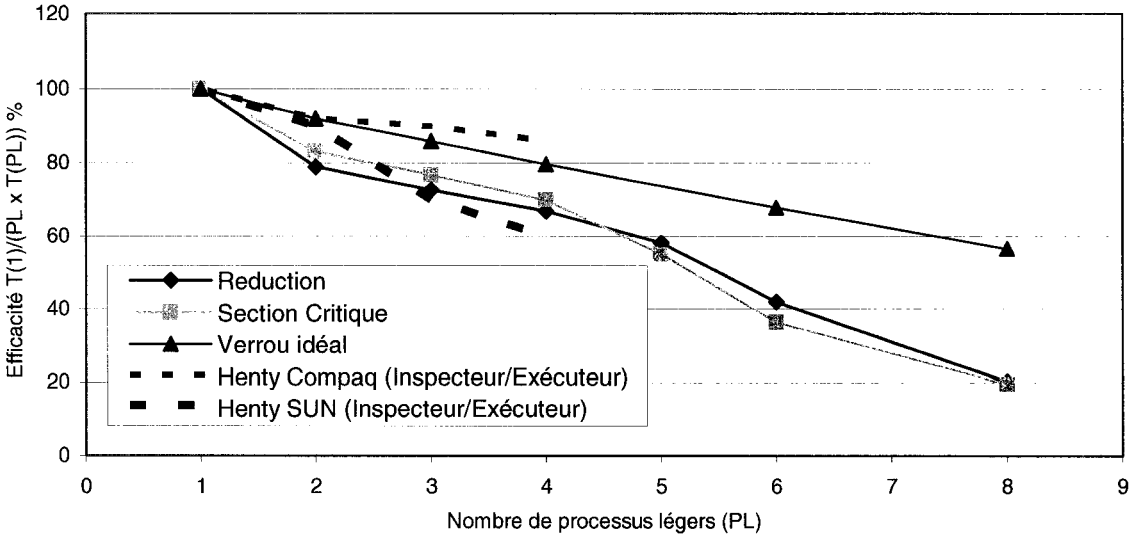


Figure 7-2 Efficacité de la parallélisation OpenMP sur Polaris

Comme pour la méthode de parallélisation en mémoire distribuée, nous avons constaté qu'augmenter le nombre de particules fait mieux paraître la parallélisation car on modifie indirectement la longueur des boucles. Nous pensions que l'utilisation des 32 Gigaoctets de Polaris permettrait d'exploiter les 16 processeurs et d'obtenir des taux d'efficacité satisfaisants. Cependant, lorsque les besoins en mémoire dépassent 1,5 Gigaoctets, le nombre important d'accès à cette mémoire dégrade les performances et semble être la cause du ralentissement de l'exécution parallèle.

La parallélisation en mémoire partagée d'une application basée sur la méthode des éléments discrets est somme toute assez triviale, très portable, et son efficacité avec peu de processeurs est parfois comparable à l'approche distribuée. Mais pour combiner tous ces avantages, il faut mettre le prix. Les défis techniques pour construire des architectures à mémoire partagée sont plus grands que dans le cas d'architectures à mémoire distribuée. Le coût de ces superordinateurs varie de façon non linéaire avec le nombre de processeurs. Généralement, les grappes de serveurs SMP sont avantageuses car elles offrent le meilleur des deux paradigmes à un coût plus raisonnable. Une étude succincte de ces systèmes hybrides (supportant le modèle mémoire distribuée entre les

noeuds via le réseau et le bus mémoire entre les processeurs de ces derniers) complétera ce chapitre sur la méthode des éléments discrets et l'architecture à mémoire partagée.

7.4 Parallélisation hybride appliquée à la méthode des éléments discrets

L'utilisation des deux paradigmes de programmation parallèles permet, du moins en théorie, de combiner le meilleur des deux mondes. On peut alors profiter des avantages des ordinateurs multiprocesseurs et de l'évolutivité des grappes de calcul. Le mode hybride ne requiert aucun changement puisque les deux techniques de parallélisation sont indépendantes et facultatives (Figure 7-3). En effet, la décomposition de domaine nécessite des communications au moyen de MPI qui sont réalisées avant d'effectuer le calcul des forces et du mouvement, les deux boucles parallélisées avec OpenMP.

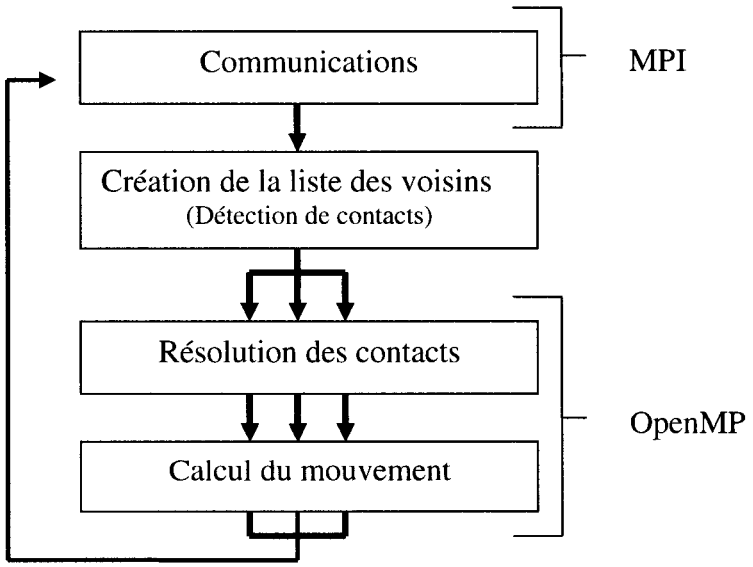


Figure 7-3 Modèle hybride pour un code DEM

Les performances avec OpenMP sur les systèmes Intel SMP étant trop faibles, le modèle hybride a été évalué sur les serveurs de calcul du Réseau Étoile (P630) et sur le P690. Cependant, nous avons rapidement réalisé que sur ces plateformes, l'approche hybride n'obtiendrait jamais de meilleurs temps que ceux obtenus avec l'approche distribuée. Pour le P690, inutile de comparer : une implantation de Mpich optimisée pour la mémoire partagée, rend ce dernier très efficace pour la décomposition de domaine. L'efficacité de la version OpenMP est toujours surclassée, peu importe le nombre de processus. Les résultats sur les serveurs POWER4 du Réseau Étoile sont un peu mieux à condition bien sûr d'utiliser des nœuds différents pour chaque processus MPI afin que les communications entre les processus ne reposent pas sur des communications quasi-instantanées intra-serveurs mais passent par le réseau. L'implantation hybride est dans le meilleur des cas 10% moins performante (Figure 7-4).

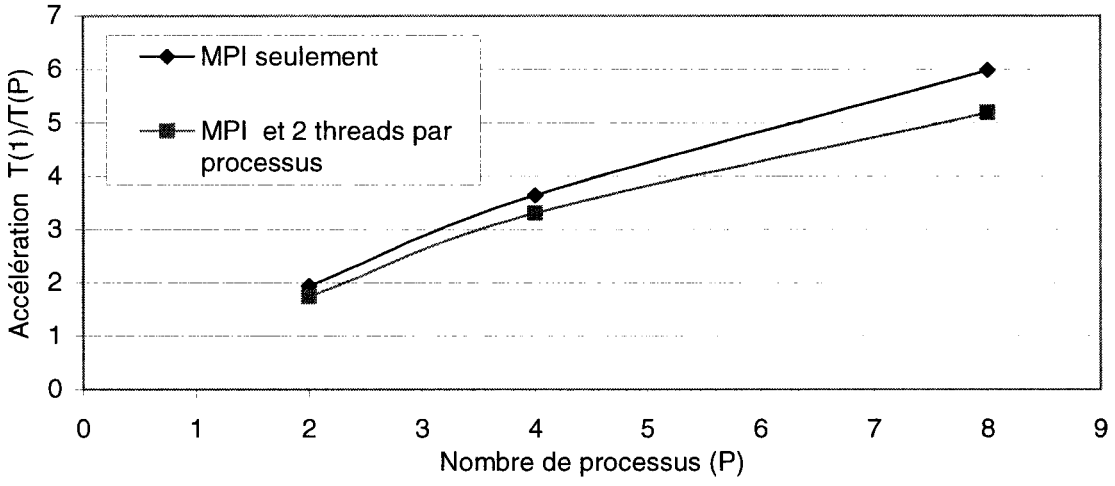


Figure 7-4 Efficacité du mode hyride sur le Réseau Étoile (P630) (sédimentation de 125000 particules (50x50x50))

Il y a selon nous deux situations prometteuses pour la parallélisation hybride. La première application concerne les problèmes où l'utilisation de décompositions multidimensionnelles devient intéressante quand les sous-domaines générés par une

décomposition unidimensionnelle sont trop minces. En effet, grâce à l'implantation hybride, il n'est plus nécessaire de décomposer les tranches obtenues par cette dernière dans les autres directions. La décomposition peut être faite selon un axe, générant donc moins de messages. L'équilibrage de tâche est implicite au niveau des boucles et un mécanisme de déplacement de frontières très efficace en 1D pourrait être appelé au besoin. En reprenant l'exemple de la section 6.4 qui avait introduit la décomposition excessive, nous montrons à la figure 7-5 que l'approche hybride peut profiter d'un déséquilibre dans une direction secondaire.

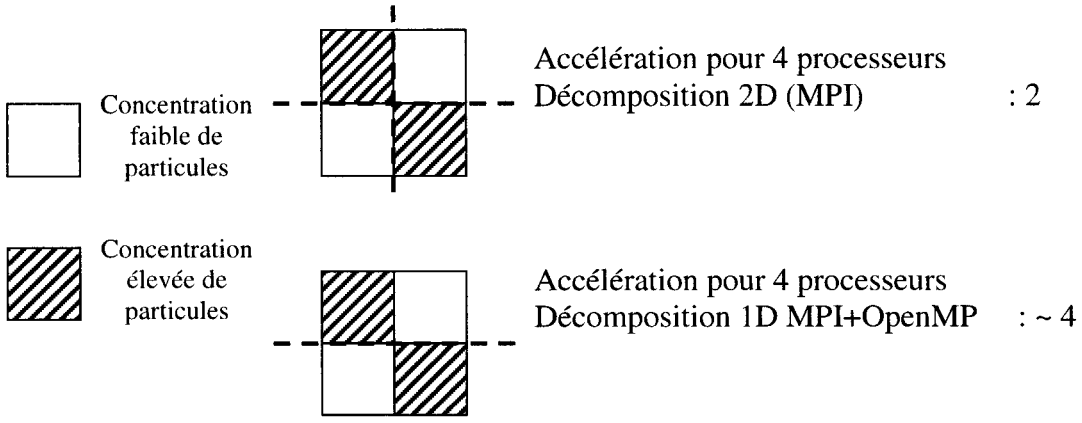


Figure 7-5 Situation artificielle où le mode hybride serait plus performant

Le deuxième champ d'application prometteur pour la parallélisation hybride concerne les programmes s'apparentant aux données répliquées. Nous ciblons plus particulièrement la parallélisation de modèles particulaires se situant entre la DEM et la dynamique moléculaire. Lorsque les forces ont une plus grande portée et ne nécessitent pas de contacts, les halos doivent être élargis. Le graphique de la figure 7-6 montre que l'efficacité de la décomposition de domaine diminue en fonction de la portée pour des forces de type colloïdal. Le gain est faible mais mesurable.

L'approche hybride limite les inconvénients des problèmes de données répliquées car elle requiert moins de communications. En plus de limiter la quantité de

données à transmettre, la mémoire requise par la duplication des données est réduite par l'utilisation de la même information par les processus légers. Le gain par rapport à une exécution MPI pure reste toutefois limité. Par contre, si l'efficacité de l'implantation en mémoire partagée est améliorée, le mode hybride pourrait devenir exploitable pour des cas pratiques réels.

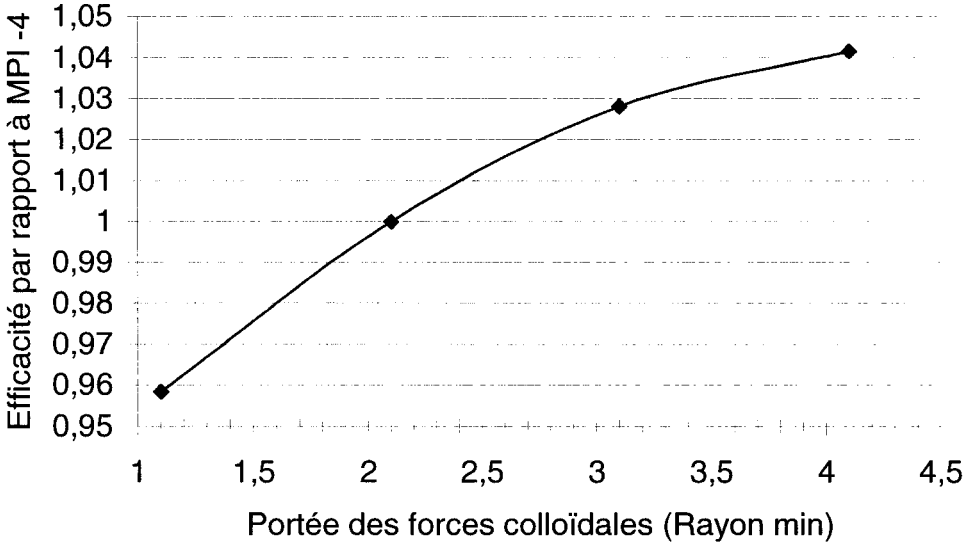


Figure 7-6 Efficacité du mode hybride en fonction du rayon d'action par rapport à 4 processus MPI (Réseau Étoile)

Les résultats précédents ne nous permettent pas de recommander l'utilisation de la version OpenMP. Son évolutivité ne peut se comparer à l'implantation en mémoire distribuée. Néanmoins, cette section a soulevé une problématique importante. Le développement d'un mécanisme pour mieux gérer les accès irréguliers à une structure en mémoire partagée est un défi qu'on peut retrouver dans plusieurs applications scientifiques parallélisables en mémoire partagée. Il est donc primordial de continuer à rechercher une solution efficace à ce problème. Dans le cas de Powder3D, la parallélisation hybride doit être explorée plus sérieusement. Nos tests démontrent qu'une

implantation OpenMP plus efficace sur les plateformes SMP bon marché pourrait être avantageuse lors de l'ajout des forces à long rayon d'action à notre modèle.

8 Conclusion et perspectives

8.1 Conclusion

Dans ce travail, la parallélisation par décomposition de domaine de la méthode des éléments discrets a été présentée et évaluée. La méthode des éléments discrets comme plusieurs autres méthodes numériques a grand intérêt à profiter de la puissance des grappes de calcul. Nous avons démontré que la décomposition spatiale, en créant de petits sous-domaines indépendants, est idéale pour paralléliser efficacement un code d'éléments discrets. De plus, plusieurs points importants ont été abordés. Parmi eux, l'équilibrage de tâche dynamique. Ce mécanisme est essentiel pour maximiser l'utilisation d'un ordinateur parallèle. Au lieu de procéder uniquement par le déplacement de la charge entre les sous-domaines, nous avons proposé une méthode dite de décomposition excessive, par laquelle le nombre de sous-domaines est supérieur au nombre de processeurs, ce qui permet au système de s'ajuster même dans le cas de géométries très complexes. Cependant, cette technique a besoin d'un ordinateur en mémoire partagée réelle (SMP) ou virtuelle (openMosix) pour fonctionner. L'utilisation de plateformes multiprocesseurs a permis la programmation d'un code d'éléments discrets à mode mémoire partagée et hybride. Toutefois, même en oubliant le coût exorbitant des architectures à mémoire partagée, ce paradigme appliqué à la DEM n'est pas assez efficace pour être recommandé; l'approche en mémoire distribuée avec les grappes de calcul Beowulf est préférable. Enfin, la création d'objets non sphériques composés de particules élémentaires a été abordée. Ces agglomérations statiques pourront permettre la simulation d'écoulements granulaires complexes.

8.2 Forces à long rayon d'action

La parallélisation a permis de diminuer les temps de calcul d'un processus itératif très coûteux. Une des extensions prévues dans un futur rapproché sera d'intégrer dans le code les forces de cohésion, forces à long rayon d'action qui deviennent de plus en plus importantes à mesure que la taille des particules diminue. Comme pour certaines applications en dynamique moléculaire, des hypothèses simplificatrices aideront à limiter les coûts de communication supplémentaires et offrir des performances acceptables. La largeur des halos sera directement touchée par ce nouveau modèle et la parallélisation devra s'adapter à cette nouvelle situation pour rester efficace.

8.3 Extension des modèles de programmation hybride

Plusieurs versions de programmation parallèle de la DEM ont été conçues dans ce travail et présentées dans ce mémoire, y compris des modèles hybrides basés sur MPI et OpenMP. La figure 8-1 inspirée de Rabenseifner (2003) résume les différents modèles de programmation hybride possibles sur les grappes de calcul SMP.

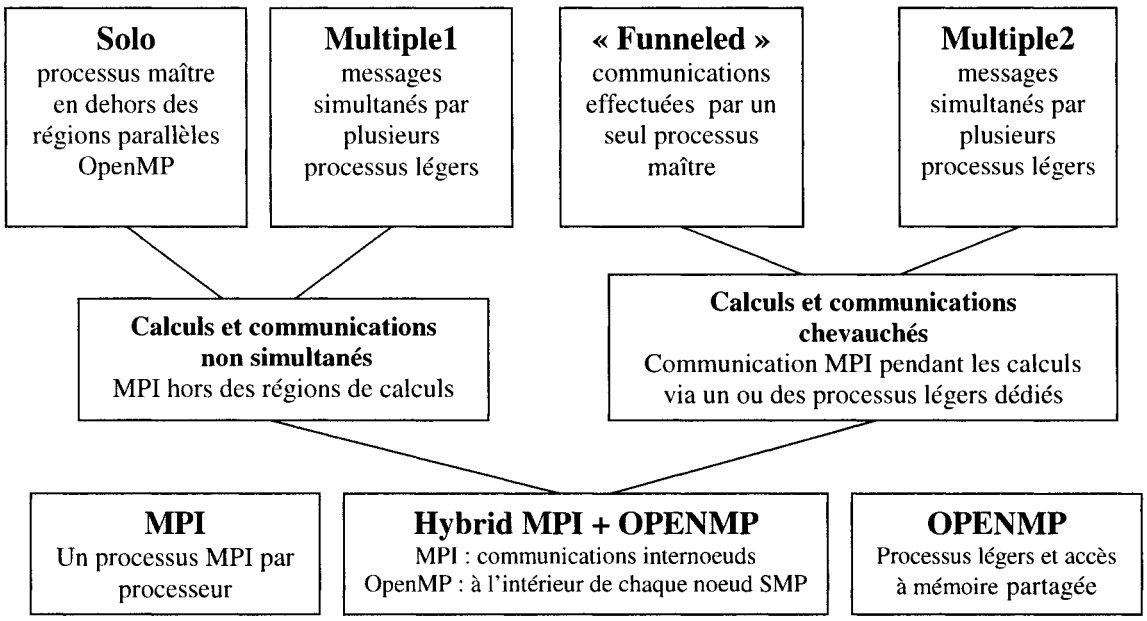


Figure 8-1 Modèles de programmation hybride

Dans la version hybride de Powder3D que nous avons développée, les communications sont effectuées par un seul processus, le maître, et les calculs sont distribués sur plusieurs processus légers. En d'autres mots, notre travail s'est limité à un mode d'exécution où les communications et les calculs ne sont pas exécutés simultanément (Figure 7-3). Lors de notre discussion sur les communications *full duplex*, nous avons montré que l'utilisation de plusieurs processus et d'une implantation thread-safe pourrait permettre de maximiser l'utilisation de la bande passante. Combiner cette technique à notre approche hybride donnerait une application tombant dans la catégorie **Multiple1** (Figure 8-2).

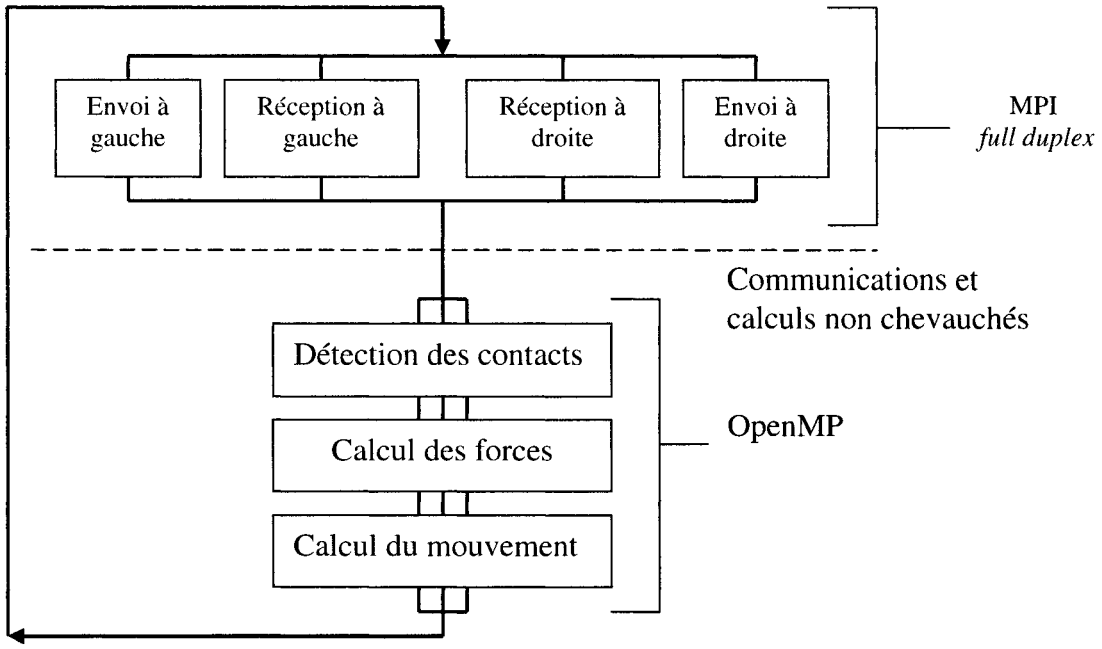


Figure 8-2 Programmation hybride avec communication *full duplex* (mode Multiple1)

Le problème de l'approche distribuée conventionnelle sera toujours le même : la communication doit se faire après la mise à jour des nouvelles coordonnées dans une étape à part. Peu importe les astuces employées pour transmettre plus rapidement et peu importe la qualité de l'architecture matérielle, les communications viendront toujours nuire à l'évolutivité de l'accélération en parallèle. Si les échanges de particules frontalières pouvaient commencer avant et/ou pendant les calculs (**calculs et communications chevauchés**), nous pourrions obtenir avec notre programme parallèle de meilleures courbes d'accélération.

Pour y arriver, une implantation MPI *thread-safe* serait requise pour permettre à certains processus MPI de communiquer pendant que les autres calculent(**Multiple2**). En réalité, les implantations non sécuritaires, par exemple MPICH, permettent quand même la programmation sur plusieurs processus légers. Cependant, un seul processus doit être dédié à la communication et on parle alors de « **MPI funneled** ».

Une fois les détails de l'implantation réglés, il suffirait d'exploiter la localité des calculs de la DEM. Dans notre cas, la décomposition spatiale du domaine utilise déjà la localité des collisions dans la définition du halo afin d'économiser sur la quantité de données à transmettre (largeur minimum). Pour faire mieux, il faudrait exploiter le concept de localité au maximum. Il suffirait de compléter les calculs attribués aux particules frontalières avant d'effectuer les calculs attribués aux particules centrales (Figure 8-3). Pendant que ces derniers sont complétés, un autre processus pourrait débuter les communications. Il s'agit d'une modification importante qui aurait un impact sur tous les algorithmes (détection, calcul des forces, communications) mais qui pourrait permettre de masquer le coût des communications.

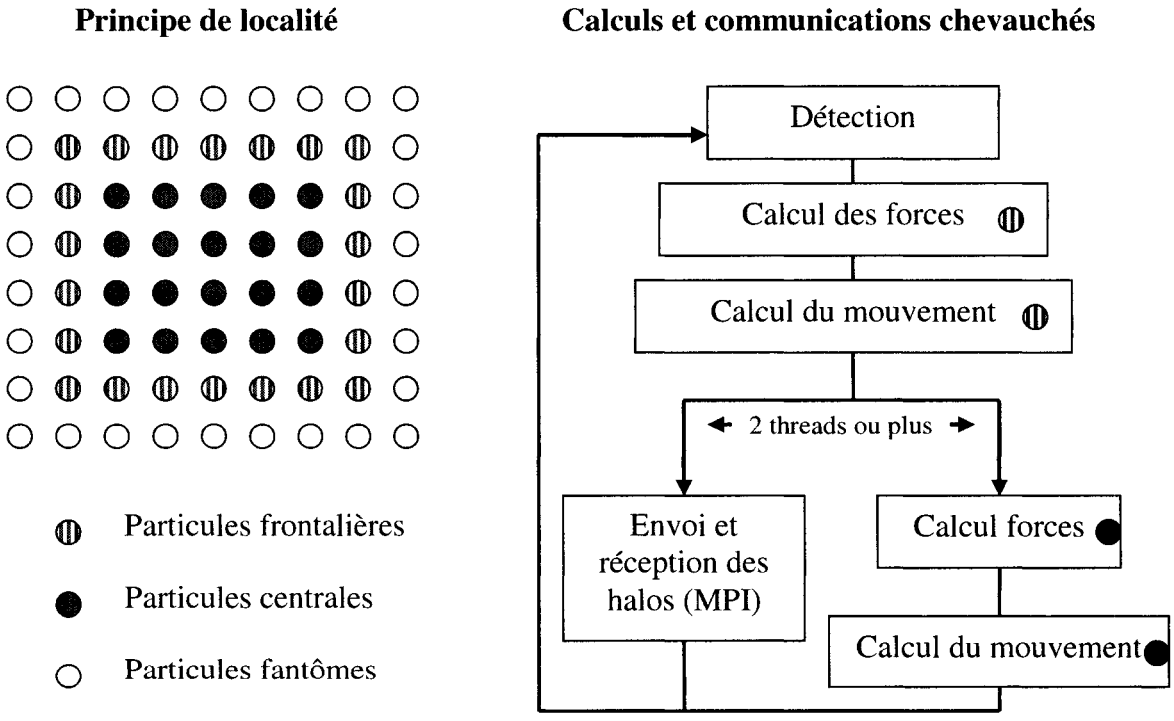


Figure 8-3 Calculs et communications chevauchés dans le cas de la parallélisation de la DEM (MPI Funneled ou Multiple2)

8.4 Importance de la qualité de la définition du pas de temps sur la parallélisation

Le pas de temps est une des constantes les plus importantes de la DEM. Il est primordial pour la stabilité du système et c'est lui qui définit le nombre d'itérations à effectuer pour atteindre le temps réel voulu. En fait, nous estimons que définir sa valeur maximale est un défi ayant autant d'impact sur les temps de résolution que la parallélisation ou l'optimisation de la détection des contacts.

Pour l'instant, dans Powder3D, le pas de temps est estimé au départ et demeure fixe. Quelques tests nous ont montré qu'il est profitable de pouvoir le modifier dynamiquement en fonction de la vitesse maximale courante. Mais les inconvénients d'un pas de temps dynamique sont dissuasifs.

La problématique de la fréquence idéale pour la vérification est encore une fois soulevée. De plus, même si on arrive dans la plupart des cas à estimer correctement le pas de temps, un mécanisme automatique de retour en arrière doit être implanté dans le cas où un chevauchement excessif a eu lieu. En effet, ce mécanisme permet, lors de la détection d'un chevauchement trop important, la récupération de toutes les valeurs conservées lors du dernier point de sauvegarde. Un tel mécanisme pourrait être implanté dans un programme parallèle, mais ce changement devrait être effectué avec prudence afin de ne pas augmenter les coûts de communication. L'utilisation d'un *broadcast* non bloquant pourrait être envisagé.

La parallélisation et la définition dynamique du pas de temps ne sont pas des sujets totalement déconnectés. La décomposition de domaine pourrait même être adaptée pour exploiter la localité de la vitesse maximale. En effet, dans plusieurs procédés, il y a souvent des régions mortes où le mouvement moyen est presque nul. Malgré tout, c'est la région la plus active qui définit le pas de temps pour le domaine en entier. Un sablier est un bon exemple où la vitesse maximale est atteinte par seulement quelques grains.

Effectuer la décomposition de domaine en fonction du pas de temps dynamique est une idée qui mériterait d'être approfondie (Figure 8-4). Toutefois, cela impliquerait un nouveau lot de problèmes dont la désynchronisation des communications. En effet, si deux sous-domaines voisins ont des pas de temps différents, il est essentiel que la valeur de ces derniers soit transmise afin de déterminer la fréquence des communications entre chaque processus.

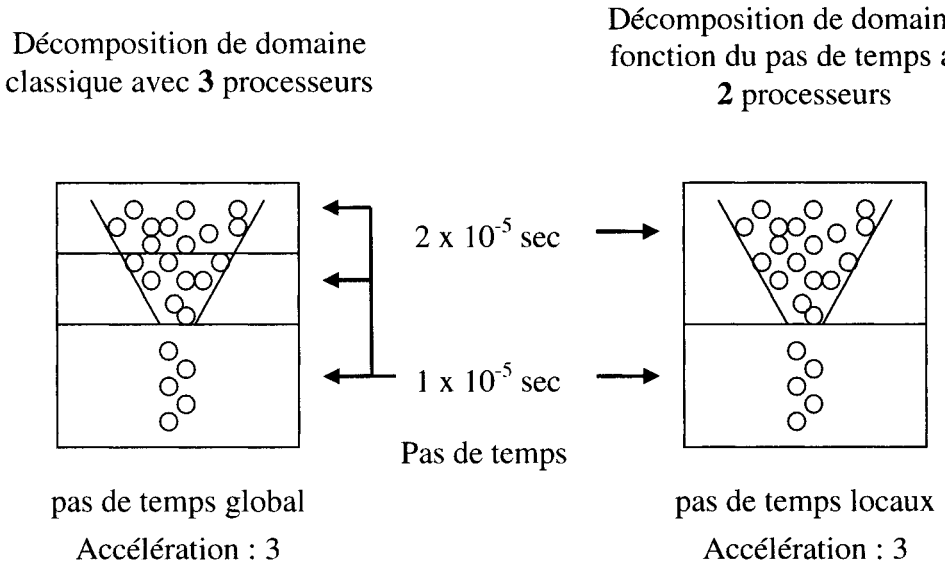


Figure 8-4 Exemple de décomposition de domaine en fonction du pas de temps

9 Références

BEVERLOO, W.A., LENGER, H.A., VAN de VELDE, J. 1961. « The Flow of Granular Solids Through Orifices ». *Chem. Eng. Sci.* 15. 260-269.

CLARK, T.W., HANXLEDEN, R., McCAMMON, J.A., SCOTT, L.R. 1994. « Parallelizing Molecular Dynamics using spatial Decomposition ». *Proceedings of the Scalable High Performance Computing Conference*. Knoxville. TN. 95-102.

CUNDALL, P.A, STRACK, O.D.L. 1979. « A Discrete Numerical Model For Granular Assemblies ». *Géotechnique*. 29:1. 47-65.

DIMITROV, Rossen. SKJELLUM, Anthony. 2003. « Impact of Latency on Applications' Performance ». [En ligne]. www.mpi-softtech.com (Page consultée le 2 novembre 2003).

ENBODY, R.J., HOROI, M. 2001. « Using Amdahl's Law as a Metric to Drive Code Parallelization: Two Case Studies ». *International Journal of High Performance Computing Applications*. 15:1. 75-80.

FAVIER, John, KREMMER, Martin. 2000. « Discrete element modeling of the flow of irregular, smooth-surfaced particles through an environment of arbitrary geometry ». *14th Engineering Mechanics Conference*. Texas: American Society conference (MSCE).

FERREZ Jean-Albert, MULLER, Didier, LIEBLING Thomas. 1996. « Parallel Implementation of a Distinct Element Method for Granular Media Simulation on the Cray T3D ». EPFL-SCR. 8.

FERREZ, Jean-Albert. 2001. *Dynamic Triangulations for efficient 3D simulation of Granular Materials.* 171p. Thèse de doctorat en mathématique. Ecole Polytechnique fédérale de Lausanne.

GALLAS, J.A., SOKOLOWSKI, S. 1993. « Grain Non-Sphericity Effects on the Angle of Repose of Granular Material ». *International Journal of Modern Physics B.* 7:9. 2037-2047.

GANGE, Timothy. « Discrete element simulations of pigment packings and blade mixer ». École polytechnique de Montréal. U.R.P.E.I. 2002.

GEIST, Al, BEGUELIN, A., DONGARRA, J., JIANG, W., MANCHEK, R. 1994. *PVM : Parallel Virtual Machine A Users' Guide and Tutorial for Networked Parallel Computing*. Cambridge: MIT Press. MA. 279p. 0-262-57108-0.

GROPP, W. LUSK, E., SKJELLUM A. 1999. *Using MPI*. Second Edition. Boston: MIT Press. 372p. 0-262-57132-3.

HENDRICKS, Erik. 2002. « BProc : The Beowulf Distributed Process Space ». *ICS'02* Juin 22-26. New York. NY.

HENDRICKSON, Bruce, DEVINE, Karen. 2000. « Dynamic load balancing in computational Mechanics ». *Comput. Methods Appl. Mech. Eng*. 184. 485-500.

HENTY, D.S. 2000. « Performance of Hybrid Message Passing and Shared Memory Parallelism for Discrete Element Modeling ». *Proceeding of the IEEE/ACM SC2000 Conference*. Nov. 04-10. Dallas. Texas.

HOGUE, Caroline. 1998. « Shape representation and contact detection for discrete element simulations of arbitrary geometry ». *Engineering Computations*. 15:3. 374-390.

KANEKO, Y., SHIOJIMA, T. et HORIO, M. 2000. « Numerical analysis of particle mixing characteristics in a single helical ribbon agitator using DEM simulation ». *Powder Technology*. 108. 55-64.

KNECHT, R., KOHRING, G.A. 1995. « Dynamic Load balancing for the Simulation of Granular Materials ». Proceeding *of 9th international conference on supercomputing*. Barcelona.

KRAVESIK, Géraud, CAPPELLO, Franck. 2003. « Performance Comparaison of MPI and three OpenMP Programming Styles on Shared Memory Multiprocessors ». *SPAA'03*. San Diego. California.

LABARTA, J., AYGUADE, E., OLIVER, J., HENTY, D. 2002. « New OpenMP Directives for Irregular Data Access Loops ». *Scientific Programming*. Vol. 9. no. 2-3. IOS Press.

MATUTTIS, H.G., LUDING, S., HERMANN, H.J. 2000. « Discrete element simulations of dense packings and heaps made of spherical and non spherical particles, *Powder Technologies*. 109. 278-292.

MULLER, Didier. 1996. *Techniques informatiques efficaces pour la simulation de milieux granulaires par des méthodes d'éléments distincts*. 187p. Thèse de doctorat en mathématique, École polytechnique de Lausanne.

MUZZIO, F.J., SHINBROT, T., GLASSER B.J. 2002. « Powder Technology in the Pharmaceutical Industry: The Need to Catch Up Fast ». *Powder Technology*. 124. 1-7.

O'CONNOR, R. M. 1996. *A distributed discrete element modeling environment - algorithms, implementation and applications*. Thèse de doctorat en informatique, Massachussetts Institute of Technology.

OWEN, D.R.J., FENG, Y.T. 2001. « Parallelised finite/discrete element simulation of multi-fracturing solids and discrete systems ». *Engineering Computations*. 18 :3. 557-576.

PELESSONE, D. 2003. « Discrete element simulations using macro-particles ». *Computational Fluid and Solid Mechanics*. K.J. Bathe (Ed.). 2089-2092.

PLIMPTON, Steve. 1995. « Fast Parallel Algorithms for Short-Range Molecular Dynamics ». *Journal of Computational Physics*. 117. 1-19.

SAWLEY, Mark L., CLEARY, Paul W. A. 1999. « Parallel Discrete Element Method for Industrial Granular Flow Simulations ». EPFL-SCR. 11.

RABENSEIFNER, Rolf, WELLEIN, Gerhard. 2003. « Communication and Optimization Aspects of Parallel Programming Models on Hybrid Architectures ». *International Journal of High Performance Computing Applications*. 17:1. 49-62.

SEDGEWICK, Robert. 1990. *Algorithms in C*. 2nd edition. Adison-Wesley. 657p. 0-201-06673-4.

SMITH, Lorna, BULL, Mark. 2001. « Development of mixed MPI/OpenMP applications », *Scientific Programming*. 9:1. 83-98.

VIDAL, David, ZOU Xuejun, UESAKA Tetsu. 2001. « Modeling coating structure development using a Monte-Carlo deposition method ». *TAPPI 2001 Advanced coating fundamentals symposium proceedings*. San Diego. 329.

WALTON O.R., BRAUN R.L. 1986. « Viscosity, granular-temperature and stress calculations for shearing assemblies of inelastic, frictional disks ». *Journal of Rheology*. 30. 949-980.

WARREN, M., SALMOM, J.A. 1992. « Astrophysical n-body simulations using hierarchical tree data structures ». *Proceeding of Supercomputing '92*. Minneapolis, Minnesota. 570-576.

WARREN, M., SALMOM, J.A. 1993. « A Parallel Hashed Oct-Tree N-Body Algorithm », *Proceeding of Supercomputing '93*. Portland, Oregon. 12-21.

WILLIAMS, John R., PERKINS, Eric. 2001. « A fast detection algorithm insensitive to object sizes ». *Engineering Computations*. 18:1. 48-61.

YANG, R.Y., YU, A.B., ZOU, R.P. 2000. « Computer simulation of the packing of fine particles ». *Physical Review E*. 62. 3900-3908.

ZHOU, Y.C., XU, B.H., YU, A.B., ZULLI, P. 2002. « An experimental and numerical study of the angle of repose of coarse spheres ». *Powder Technology*. 125. 45-54.