Simulation par automates cellulaires bi-dimensionnels en géotectonique

Thomas Leduc

Simulation par automates cellulaires bi-dimensionnels en géotectonique

Thomas Leduc

Université Pierre et Marie Curie — Laboratoire LIP6
Thème Algorithmique Numérique et Parallélisme
4, place Jussieu,
F-75252 Paris Cedex 05
Thomas.Leduc@lip6.fr
www.lip6.fr/anp/~

RÉSUMÉ. *Dans cet article, nous présentons la parallélisation d'un automate cellulaire pour la modélisation numérique du processus de subduction-érosion en tectonique des plaques. Après un rappel de tectonique des plaques et de notre modélisation par automate cellulaire unidimensionnel, nous présentons notre modèle bi-dimensionnel puis nous abordons la question de la simulation parallèle et de son implémentation sur un* CRAY T3E *et une* ORIGIN2000. *Nous montrons en particulier l'intérêt des types de données dérivés et des topologies de processus de la bibliothèque d'échange de messages* MPI *dans le cadre d'une méthode de décomposition de domaine sur une architecture parallèle à mémoire distribuée.*

ABSTRACT. *This paper describes an implementation of a parallel cellular automaton on distributed memory machines. After a presentation of the geotectonics context, we briefly describe our one-dimensional discrete computer model ; then we expose our two-dimensional model and the parallel simulation we have developed using the* MPI *library. Our portable code has been implemented on a* CRAY T3E *and an* ORIGIN2000. *We show in particular the interest of the user-defined derived datatypes and the virtual process topologies of this interface.*

MOTS-CLÉS : *Parallélisme, décomposition de domaine, automates cellulaires, systèmes dynamiques discrets, tectonique.*

KEY WORDS : *Parallelism, domain decomposition, cellular automata, discrete dynamical systems, plate tectonics.*

1. Introduction

Le texte qui suit présente une partie de l'étude sur la modélisation informatique du processus de subduction-érosion en tectonique des plaques, que nous menons actuellement en collaboration avec le Laboratoire de Géodynamique Tectonique et Environnement de l'Université Pierre et Marie Curie. Après une présentation du contexte géotectonique, une brève justification de notre choix de procéder par simulation numérique discrète et un rappel de notre simulation par automate cellulaire unidimensionnel, nous exposons notre modèle bi-dimensionnel et la simulation parallèle que nous avons développée avec la bibliothèque d'échange de messages MPI. Pour terminer, nous abordons les problèmes d'implémentation de notre simulation sur machines CRAY T3E et ORIGIN2000 ainsi que les résultats obtenus.

2. Le contexte géotectonique

La tectonique des plaques est une hypothèse bien établie selon laquelle la lithosphère du globe terrestre est formée de plaques rigides d'une centaine de kilomètres d'épaisseur, flottant sur l'asthénosphère visqueuse et animées de mouvements dont la vitesse varie de 1 à 11 cm/an. Les mouvements lithosphériques et asthénosphériques sont couplés. Ainsi, en fonction des sens et/ou des axes de rotations de deux cellules de convection asthénosphériques adjacentes, les plaques lithosphériques correspondantes peuvent converger, diverger ou coulisser. Dans chacun des cas, les déplacements sont de l'ordre du centimètre ou de la dizaine de centimètres par an [1].

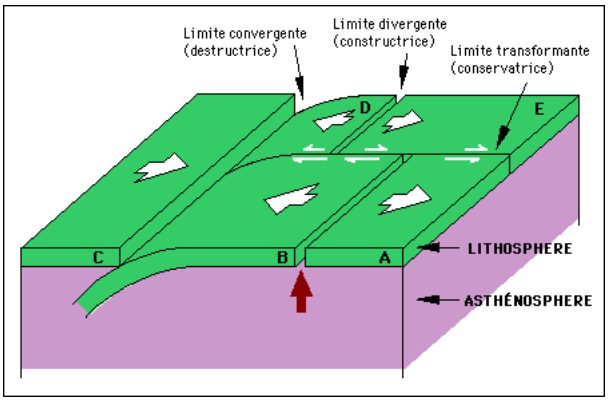

Figure 1. *Les trois principaux types de marges*

1. Pour [CON 97], l'Amérique du Nord se déplace à la vitesse de 1,5-2 cm/an, l'Eurasie à 2,4-2,7 cm/an, l'Afrique à 2,9-3,2 cm/an, l'Australie à 6-7,5 cm/an et le Pacifique à 5-7 cm/an. . .

Nous nous intéressons au cas des plaques (ou marges) convergentes, en subduction, avec extension. L'extension résulte d'un phénomène d'érosion de la base de la plaque chevauchante par hydrofracturation. C'est un processus qui consiste à entraîner dans la subduction, par le jeu des forces tectoniques, du matériel de la marge chevauchante dans l'asthénosphère visqueuse.

Il existe déjà actuellement des simulations analogiques (en « bac à sable », [KIN 87, SHE 92, MAL 93]) et des simulations analytiques et numériques « globales » (méthode des éléments finis, [SCH 94, DUB 95]) du processus de subduction. L'objectif sous-jacent à toutes ces études est clair : il s'agit de fournir une estimation suffisamment précise et détaillée du volume de matière absorbé en subduction, dans le but de procéder à un enfouissement sous-marin de déchets radioactifs [BOU 96]. Mais, même si ces modèles apportent tous leur contribution à la compréhension du phénomène, il nous a paru important de développer une simulation à base d'automates cellulaires, comme nous l'expliquons dans [LED 97].

En effet, comme le remarque [JOL 95], dans une simulation expérimentale, on utilise des matériaux qui ont un comportement proche, a priori, de celui de la lithosphère et on leur applique des conditions aux limites variées. Or, le comportement rhéologique des matériaux utilisés ne peut être totalement contrôlé et une part importante de hasard rentre en jeu. Par ailleurs, dans une simulation numérique on suppose souvent que la lithosphère a un comportement fluide ; le rôle joué par des structures bien définies ne peut absolument pas y être pris en compte. Ainsi, comme le remarque [MÜL 96], l'approche continue « ne s'applique pas à tous les cas d'étude : [il est] difficile en effet d'observer des phénomènes comme la ségrégation [2], où les grains de petites tailles se séparent de ceux de grandes tailles, puisque les grains ne sont pas représentés en tant que tels ».

Bien que les modélisations décrites ci-dessus soient toujours d'actualité et perpétuellement améliorées, nous avons choisi d'adopter une attitude tangente et de profiter ainsi des acquis de l'analogique tout en restant dans un contexte numérique. Plus simplement, partant de la notion de modélisation analogique en « boite à sable » et l'associant avec les travaux déjà menés dans le domaine de la physique de l'état granulaire [BAK 87, TAN 88, JAE 92, GRU 93, DUR 95, MÜL 96, DUR 97], nous avons décidé d'orienter nos travaux sur une modélisation informatique par système dynamique discret du processus géotectonique.

En effet, même s'il peut sembler relativement « osé » d'associer la rhéologie d'une plaque lithosphérique à de la physique de l'état granulaire, nous justifions notre choix de modélisation de la façon suivante :

— les automates cellulaires (par le biais des méthodes dites de « gaz sur réseau » qui sont une version « microscopique » - mouvement de particules - de l'équation « macroscopique » de Navier-Stokes - écoulement d'un fluide) ont déjà permis de modéliser la convection [LAF 95],

— les automates cellulaires ont déjà été utilisés dans des modélisations en sismologie [NAR 92, HUA 92],

2. Nous sommes nous, confrontés à de l'érosion tectonique à la base de plaque supérieure... mais le problème reste entier !

— les automates cellulaires ont déjà été utilisés pour modéliser les mécanismes d'érosion de terrain [SMI 91],

— les automates cellulaires (par le biais du Sand Pile Model, qui est un modèle simple d'écoulement de grains dans un espace modélisé par un graphe) sont utilisés pour modéliser les phénomènes tels que les avalanches [DUR 96][3],

— dans l'univers « réel », les interactions sont locales (écoulements, avalanches, propagation...), limitées par la vitesse de la lumière et uniformes dans l'espace,

— au cours d'un processus géotectonique comme la subduction, plusieurs phénomènes physiques et chimiques concurrents se produisent sur des échelles de temps bien distinctes et portent sur des couches de terrain très hétérogènes qui ne peuvent absolument pas être considérées comme un tout. Adopter une approche où chaque portion de l'espace étudié est considérée dans sa singularité pour une échelle de temps donnée nous semble donc bien être un bon élément de réponse.

Or, comme nous pourrons le constater en recensant les phénomènes à « reproduire » sur la portion « d'espace » que nous devons modéliser (portion limitée aux pourtours de la fosse océanique), il faut essentiellement simuler des effets d'affaissements de terrain, d'avalanches superficielles, d'érosion sous-crustale... Et tous ces phénomènes ont déjà été plus ou moins abordés, individuellement, par une technique de modélisation discrète. Notre approche consiste donc en une tentative d'intégration de ces différents modèles en un seul et unique, afin de simuler un ensemble varié de phénomènes.

Enfin, nous avons constaté que les modèles numériques continus ont été longuement et soigneusement développés et améliorés sans pour autant répondre aux attentes de l'ensemble de la communauté des géologues et géophysiciens. C'est aussi pour cette dernière raison que nous avons choisi de nous démarquer de la tendance générale.

3. Les résultats de la modélisation uni-dimensionnelle

En ce qui concerne notre propre modèle[4], nous avons restreint l'étude de la subduction aux pourtours de la fosse océanique. En fait, nous limitons la portée de notre modèle à une dizaine de kilomètres en amont et en aval de la fosse.

Nous avons décidé de discrétiser (voir [LED 97, LED 98]) la coupe verticale de subduction dans le sens longitudinal. L'élément de base, appelé cellule, est une bande verticale plane, sur toute la hauteur de la coupe, comportant plusieurs couches différentes (eau, sédiments, basaltes...). Ces cellules peuvent prendre un nombre fini d'états et sont influencées par un ensemble fini de cellules « proches ». L'état d'une cellule est déterminé par la donnée des épaisseurs des différentes couches qui la consti-

3. Ces phénomènes sont appelés « 1/f phenomenon » car ils vérifient statistiquement un rapport inverse entre les intensités des perturbations et leurs probabilités. Ils ont aussi une propriété de criticalité auto-organisée, ce qui signifie qu'ils ont la propriété de se retrouver immédiatement après un bouleversement dans un état pseudo-stable susceptible de se re-bouleverser instantanément.

4. Pour le formalisme « réseau d'automates cellulaires » nous invitons le lecteur à lire [LED 97].

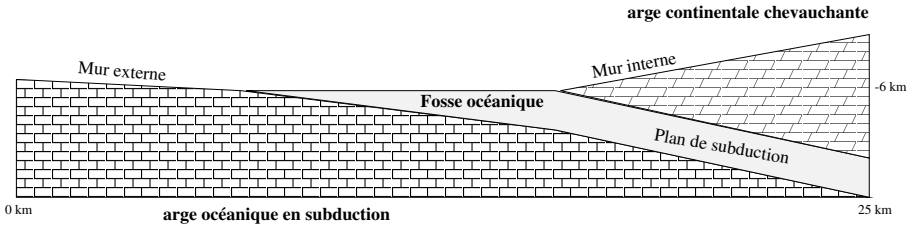

Figure 2. *Plan de coupe schématique de la zone de subduction*

tuent, ces épaisseurs sont des entiers naturels majorés par la hauteur totale de la coupe étudiée ; elles sont au nombre de sept.

Une fois la géométrie et l'état initial du réseau définis, il ne reste plus qu'à laisser le système évoluer de lui-même, par réévaluation parallèle synchrone de l'état de chacune des cellules en fonction de ceux de ses voisines (au sens où elles influencent la cellule courante…), à chaque pas d'horloge.

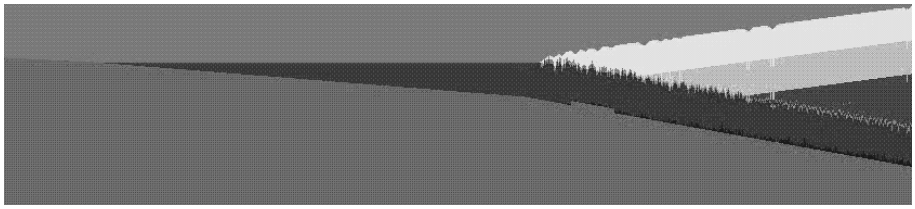

Figure 3. *Copie d'écran obtenue après 1100 itérations de calcul (simulation 1D)*

L'avantage de ce modèle est clair : la quantité d'information à traiter est faible (de l'ordre du millier de cellules, chacune des cellules ne comportant pas plus de 7 champs entiers). Par contre, pour obtenir une représentation acceptable du phénomène physique, il nous a fallu introduire des notions relativement difficiles à mettre en place telles que : trois échelles de temps imbriquées, des phénomènes à « ampleur » locale et d'autres à implications plus globales.

4. La modélisation bi-dimensionnelle

Conscients du fait que le découpage en cellules du modèle unidimensionnel est trop « simplificateur » (7 entiers nous suffisent pour représenter une coupe verticale d'espace de 5 km de haut sur 25 m de large, correspondant au pourtour de la fosse océanique), nous avons cherché à modéliser ce phénomène géotectonique par un automate cellulaire bi-dimensionnel. Une cellule ne représente plus désormais qu'une portion d'espace homogène de 25 m par 25 m. Ainsi, alors que dans le modèle uni-

dimensionnel, nous faisons un découpage en colonnes verticales de la zone de subduction, dans le cas bi-dimensionnel, le découpage est fait par blocs.

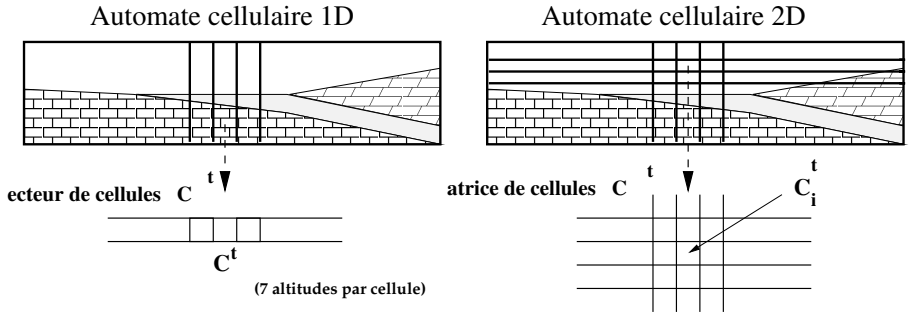

Figure 4. *Discrétisation en colonne contre discrétisation en bloc*

L'état d'une cellule est désormais déterminé par la donnée de six entiers naturels bornés appelés respectivement : `couleur`, `nature`, `vieil`, `courbure`, `memoire`, `altitude`. L'attribut de `couleur` permet de distinguer, à la visualisation, deux cellules de même `nature`. L'attribut `courbure` permet, pour une cellule donnée de la plaque océanique ou du chenal de subduction, de progresser avec la bonne inclinaison (exprimée en %). Son compteur `memoire` est incrémenté d'une unité à chaque "progression horizontale" et remis à zéro lors d'une "progression verticale" (c'est-à-dire lorsque la valeur de l'attribut `courbure` correspondant à la prochaine position de la cellule courante est atteinte). L'attribut `vieil` permet de modéliser un effet de dégradation de la cohésion de la plaque continentale avec transformation de matière. Enfin l'`altitude`, est utilisée pour faciliter l'implémentation des phénomènes d'avalanches superficielles, afin d'éviter qu'ils ne se produisent pour les cellules de la base de la marge continentale. Cet attribut est simplement destiné à améliorer la cohabitation des diverses règles de transition que nous mettons en œuvre.

Ces six entiers sont bornés, l'état d'une cellule du réseau d'automates cellulaires est donc un élément d'un ensemble fini.

Tout comme dans le cas uni-dimensionnel, nous ne pouvons pas considérer un tableau infini de cellules pour des raisons d'ordre technique (par ailleurs, il s'agit de modéliser une portion de coupe finie de zone de subduction de 25 km de long sur 5 km de haut). Notre réseau d'automates cellulaires est donc une matrice finie d'automates. En ce qui concerne les extrémités[5], nous avons choisi de les singulariser par un état "frontière" invariant et de les répliquer symétriquement aux bords de façon à obtenir l'effet d'un tore bi-dimensionnel.

5. Qui dans notre cas pratique représentent environ 2,39 % de l'ensemble des cellules.

Loi de transition locale (début)

```
# comportement par défaut
cell ← CO
# autres comportements
si (CO.nature = NAT_BORD)
  #cas particulier d'une cellule aux bords du domaine
  cell ← CO
sinon si ((numIteration%echelle_temps) = 0) alors
  # phénomènes associés à l'échelle de temps ''lente''
  cell.courbure ← CO.courbure
  cell.memoire ← ((O.memoire≥(CO.courbure-1))?0:O.memoire+1)
  cell.altitude ← CO.altitude
  si (O.nature = NAT_BORD) alors
    # introduction de plaque océanique sur le bord
    # latéral gauche
  sinon si ((CO.nature ≠ NAT_SOCLE_OCEAN) ET
    (O.nature ≠ NAT_SOCLE_OCEAN) ET
    (NO.nature ≠ NAT_SOCLE_OCEAN)) alors
    # cas d'une cellule n'appartenant pas à la plaque océanique
    # ou n'étant pas amenée à être recouverte par une cellule
    # de plaque océanique au pas de temps suivant
    si ((CO.nature = NAT_SOCLE_CONTI) ET
      ((S.nature = NAT_SOCLE_CONTI_ERO)
      OU (S.nature = NAT_COUCHE_INTERSTI))) alors
      # vieillissement de la base de la plaque continentale
      # (par incrémentation aléatoire du coefficient VIEIL)
      # suivie ou non d'une transformation chimique de la
      # cellule (CLR_SOCLE_CONTI_ERO,NAT_SOCLE_CONTI_ERO...)
    sinon si ((N.nature = NAT_EAU) ET
      (CO.altitude = 1) ET
      (CO.nature = NAT_COUCHE_INTERSTI)) alors
      # comblement de la couche supérieure de la fosse
      # océanique
    sinon si ((CO.altitude <1) OU
      ((CO.altitude = 1)
        ET (CO.nature = NAT_SOCLE_CONTI_ERO))) alors
      # plongeon de la couche interstitielle et des
      # cellules érodées
    fin si
  sinon
    # gérer le plongeon de la plaque océanique
  fin si
```

Figure 5. *Principes généraux de la fonction de transition locale (début)*

Loi de transition locale (suite)

```
sinon
  # phénomènes associés à l'échelle de temps ''rapide''
  si (((CO.nature = NAT_SOCLE_CONTI)
       OU (CO.nature = NAT_SOCLE_CONTI_ERO)) ET
     ((N.nature = NAT_SOCLE_CONTI)
       OU (N.nature = NAT_SOCLE_CONTI_ERO)) ET
     ((NO.nature = NAT_SOCLE_CONTI)
       OU (NO.nature = NAT_SOCLE_CONTI_ERO)) ET
     ((O.nature = NAT_SOCLE_CONTI)
       OU (O.nature = NAT_SOCLE_CONTI_ERO)) ET
     ((SO.nature = NAT_SOCLE_CONTI)
       OU (SO.nature = NAT_SOCLE_CONTI_ERO)) ET
     ((SSO.nature = NAT_SOCLE_CONTI)
       OU (SSO.nature = NAT_SOCLE_CONTI_ERO)) ET
     (E.nature ≠ NAT_BORD) ET
     (S.nature = NAT_COUCHE_INTERSTI)) alors
     # affaissement de terrain au sein de la plaque continentale
  sinon si ((CO.altitude >1) ET
     (CO.nature = NAT_SOCLE_CONTI_ERO) ET
     ((S.nature = NAT_COUCHE_INTERSTI)
       OU (S.nature = NAT_EAU))) alors
     # micro affaissement de socle continental érodé
  sinon si ((SS.altitude ≥ 1) ET (CO.nature ≠ NAT_BORD)) alors
     # avalanches superficielles envisageables.
     # Reprendre et adapter l'algorithme du Sand
     # Pile Model bi-dimensionnel
  sinon si ((CO.altitude ≤ 1) ET
     (CO.nature = NAT_EAU) ET
     ((N.nature = NAT_COUCHE_INTERSTI) OU
     (E.nature = NAT_COUCHE_INTERSTI) OU
     (NE.nature = NAT_COUCHE_INTERSTI))) alors
     # comblement de la fosse océanique
```

Figure 6. *Principes généraux de la fonction de transition locale (suite)*

En effet, puisque chaque cellule évolue en fonction de la donnée des états de ses 24 cellules voisines, les cellules des bords du domaine ont elles-aussi besoin de connaître l'état de leurs voisines à chaque pas de temps. Nous avons donc introduit dans le modèle un ensemble de cellules (appelées cellules du bord externe), qui n'a pas à proprement parler "d'existence réelle" vis-à-vis de la modélisation. Ces cellules étendent le réseau d'automates cellulaires et sont stockées sur le pourtour du domaine étudié.

Elles ajoutent ainsi deux rangées ou colonnes au domaine et bordent les cellules qui sont situées sur les bords “intérieurs” (voir fig. 7).

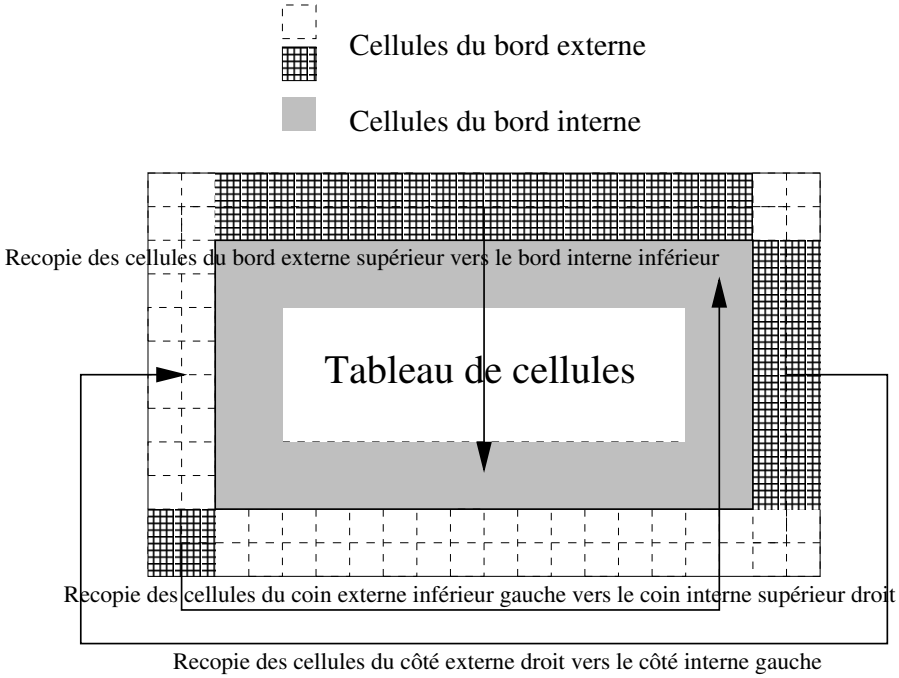

Figure 7. *Traitement des bords du réseau d’automates cellulaires*

Dans l’algorithme de la fig. 8, i désigne l’indice de colonne et j l’indice de ligne. Il n’y a pas exactement “recopie du contenu du tableau c dans le tableau cc” comme il y est indiqué. En effet, c’est un procédé beaucoup trop coûteux en terme de temps de calcul. Nous préférons utiliser une méthode de permutation des adresses des tableaux respectifs qui se résume à de simples affectations de pointeurs.

En pratique, notre automate cellulaire est représenté par un tableau bi-dimensionnel de 200000 cellules. Chacune de ces cellules n’a connaissance, d’un pas de temps à l’autre, que de son état propre et de celui de ses 24 voisines immédiates, puisque les cellules contiguës et les cellules contiguës des cellules contiguës interagissent avec la cellule courante [6] (les valeurs des états étant considérées au pas de temps précédent).

En fait, comme aucune cellule n’a connaissance de sa position (de ses coordonnées) dans la « matrice » des cellules, celles qui sont positionnées sur les bords de la structure doivent aussi échanger des données avec leurs propres voisines. Pour ce faire, nous avons adopté une structure périodique selon les deux dimensions (un tore-2D).

6. Nous pouvons presque parler d’un schéma centré à 24 points pour reprendre la terminologie en usage dans les méthodes par différences finies.

programme
Soient c et cc deux tableaux de NBL $\times$ NBC cellules.
Initialiser c.
compteur_d_iteration $\leftarrow 0$
Itérer le bloc d'instructions suivant :
  Recopier le contenu du tableau c dans le tableau cc
  # Recopie des lignes des bords ‘‘interieurs’’ sur
  # les bords ‘‘exterieurs’’
  Pour $i \leftarrow 0$ à 1, faire :
    Pour $j \leftarrow 2$ à $NBL-3$, faire :
      $cc_{i+NBC-2}^{j} \leftarrow cc_{i+2}^{j}$
      $cc_{i}^{j} \leftarrow cc_{i+NBC-4}^{j}$
  # Recopie des colonnes des bords ‘‘interieurs’’ sur
  # les bords ‘‘exterieurs’’
  Pour $i \leftarrow 2$ à $NBL-3$, faire :
    Pour $j \leftarrow 0$ à 1, faire :
      $cc_{i}^{j+NBL-2} \leftarrow cc_{i}^{j+2}$
      $cc_{i}^{j} \leftarrow cc_{i}^{j+NBL-4}$
  # Recopie des angles des bords ‘‘interieurs’’ sur
  # les bords ‘‘exterieurs’’
  Pour $i \leftarrow 0$ à 1, faire :
    Pour $j \leftarrow 0$ à 1, faire :
      $cc_{i}^{j} \leftarrow cc_{i+NBC-4}^{j+NBL-4}$
      $cc_{i}^{j+NBL-2} \leftarrow cc_{i+NBC-4}^{j+2}$
      $cc_{i+NBC-2}^{j} \leftarrow cc_{i+2}^{j+NBL-4}$
      $cc_{i+NBC-2}^{j+NBL-2} \leftarrow cc_{i+2}^{j+2}$
  # Application de la fonction de transition à
  # toutes les cellules du domaine exceptées celles
  # des bords ‘‘exterieurs’’
  Pour $i \leftarrow 2$ à $NBC-3$, faire :
    Pour $j \leftarrow 2$ à $NBL-3$, faire :
      $c_i^j \leftarrow f(cc_{i+2}^{j-2}, cc_{i+2}^{j-1}, cc_{i+2}^{j}, cc_{i+2}^{j+1}, cc_{i+2}^{j+2},$
      $cc_{i+1}^{j-2}, cc_{i+1}^{j-1}, cc_{i+1}^{j}, cc_{i+1}^{j+1}, cc_{i+1}^{j+2},$
      $cc_{i}^{j-2}, cc_{i}^{j-1}, cc_{i}^{j}, cc_{i}^{j+1}, cc_{i}^{j+2},$
      $cc_{i-1}^{j-2}, cc_{i-1}^{j-1}, cc_{i-1}^{j}, cc_{i-1}^{j+1}, cc_{i-1}^{j+2},$
      $cc_{i-2}^{j-2}, cc_{i-2}^{j-1}, cc_{i-2}^{j}, cc_{i-2}^{j+1}, cc_{i-2}^{j+2},$
      compteur_d_iteration)
  compteur_d_iteration $\leftarrow$ compteur_d_iteration $+1$
Tant que condition de fin de boucle non réalisée

Figure 8. *Algorithme séquentiel simplifié*

Pour reproduire la dimension temporelle, nous utilisons deux tableaux de cellules. A chaque itération (c'est-à-dire à chaque pas de temps), il y a recopie logique du contenu du second tableau dans le premier. Le premier tableau représente donc la dernière configuration de l'automate cellulaire obtenue par application de la loi de transition globale. La fonction de transition globale lui est alors appliquée une nouvelle fois et le résultat est stocké dans le second tableau. Il ne reste plus qu'à reproduire les mêmes opérations au pas de temps suivant et ainsi de suite.

5. Implémentation parallèle du modèle bi-dimensionnel

Dans le but d'optimiser le temps d'exécution de notre simulation, nous avons envisagé d'utiliser une méthode de décomposition de domaine régulière pour le portage de notre code sur une plate-forme parallèle à mémoire distribuée.

En effet, un réseau d'automates cellulaires a, par définition, une nature intrinsèquement parallèle en espace. Ainsi, pour [VIC 84] : « since all the cells "compute" their new state simultaneously, cellular automata are often seen as a paradigm of distributed computation ». Pour établir une stratégie d'optimisation parallèle de notre simulation, nous nous sommes inspirés des techniques de parallélisation en espace en vigueur dans d'autres domaines, comme la convolution en traitement d'images, les schémas explicites en mécanique... Ainsi, implémenter un automate cellulaire sur machine parallèle revient en quelque sorte à mettre en œuvre une résolution par discrétisation d'un problème d'équation aux dérivées partielles par un schéma explicite centré sur un maillage cartésien régulier. Ce schéma doit alors être appliqué sur un domaine de calcul rectangulaire avec un maillage de pas constant en espace comme en temps, et avec des conditions aux limites homogènes. Nous allons donc utiliser une méthode dite de décomposition de domaine et l'appliquer à notre cas particulier de réseau d'automates cellulaires.

Comme le note [ACK 98], l'idée qui est à la base de la méthode par décomposition de domaine est qu'un problème global peut être considéré comme un ensemble de sous problèmes locaux portant sur les partitions du domaine global. Une telle méthode favorise donc l'exploitation de la localité des données et offre la possibilité que tous les sous problèmes soient résolus concurremment. Il importe bien sûr que les traitements à appliquer aux divers sous problèmes soient de même nature mathématique que ceux qui doivent être appliqués au problème global. En raison de la quasi indépendance mutuelle des problèmes locaux, leurs résolutions (les itérations en temps) peuvent être affectées à un processeur de la machine parallèle utilisée. On peut de cette façon espérer un accroissement significatif des performances globales.

On appelle domaine de dépendance d'une cellule du réseau l'ensemble des cellules de son voisinage ayant une influence [7] sur elle à une date quelconque donnée. Si l'on considère l'ensemble des automates cellulaires d'un sous domaine quelconque, on constate que son domaine de dépendance s'étend à des cellules situées dans les huit sous domaines voisins. Or, à chaque itération en temps, chaque processeur doit

7. Au sens où nous l'avons défini dans [LED 97].

disposer de toutes les informations nécessaires à la résolution du problème sur la partie de domaine qui lui est affectée. Il faut donc en déduire qu'il y a nécessité, pour chaque processeur, de communiquer à ses voisins immédiats les cellules situées aux interfaces.

D'un point de vue pratique, nous allons appliquer la méthode de décomposition de domaine avec recouvrement. Le domaine de calcul global est donc partitionné en plusieurs sous domaines avec des zones de recouvrement. Dans le cadre d'une distribution uni-dimensionnelle des données, chaque sous-domaine doit envoyer ses cellules aux frontières aux processeurs voisins et, en retour, recevoir leurs propres cellules. Cette stratégie de sous-structuration implique qu'à chaque itération, chaque processus stocke des cellules n'appartenant pas à son propre sous-domaine (avant de les utiliser pour mettre à jour les cellules aux bords du sous-domaine). Pour résoudre ce problème de stockage en local d'informations non-locales, nous avons introduit dans le modèle une zone de recouvrement de cellules fantômes (encore appelées « ghost cells » ou « guard cells » dans la littérature). Ces cellules qui n'ont pas « d'existence réelle » vis-à-vis de la modélisation sont, d'un point de vue informatique, stockées sur le pourtour du sous-domaine de cellules du processus courant. Elles étendent ainsi de deux unités les cellules qui sont situées sur les « bords intérieurs » du sous domaine courant. Cette stratégie implique aussi que les cellules fantôme soient échangées à chaque pas de temps afin d'être réactualisées simultanément avec l'ensemble des cellules situées à l'intérieur du sous domaine considéré. A chaque itération en temps, chaque processeur dispose ainsi de toutes les informations nécessaires à la résolution du problème sur la partie de domaine qui lui est affectée.

D'un point de vue pratique, nous avons décidé de ne pas réutiliser la bibliothèque d'échange de messages PVM ([GEI 94]) comme pour la mise en œuvre informatique de la simulation uni-dimensionnelle. En effet, la bibliothèque MPI ([SNI 95]) offre plusieurs primitives qui permettent notamment de créer une grille virtuelle de processus, de déterminer les processus voisins... ce qui simplifie notablement la programmation [BRU 96].

5.1. *La topologie cartésienne de processus*

Pour toutes les communications dites « locales », parce qu'elles ne concernent que des échanges de données aux interfaces entre les sous-domaines, il est intéressant de disposer les processus suivant une topologie régulière. MPI, contrairement à PVM, offre la possibilité de définir des topologies virtuelles de type cartésien. Cette possibilité s'avère très intéressante dans la mesure où elle constitue un moyen pratique de dénomination des processus qui correspond, en plus, au schéma de communication et permet donc à MPI d'optimiser les communications.

Il faut tout d'abord rappeler qu'une topologie virtuelle de processus en MPI est une notion qui ne dépend, pour sa description, que de l'application qui la crée. Elle est indépendante de l'architecture de la machine cible et donc très portable.

Au cours d'une simulation bi-dimensionnelle telle que nous l'avons mis en place, il se produit deux types de tâches bien distincts. D'une part l'itération en temps de l'automate cellulaire, et d'autre part la collecte des résultats (c'est-à-dire la collecte des portions d'images de la zone de subduction obtenues après calcul). Pour mettre en œuvre cette division des tâches sans désynchroniser l'ensemble des processus, nous avons décidé de partitionner cet ensemble en deux catégories. Ainsi, nous avons d'une part un groupe (ou plus exactement un communicateur[8], pour reprendre la terminologie de MPI) « d'itérateurs », qui se charge de mettre à jour l'automate cellulaire à chaque pas de temps. Et, d'autre part, nous avons un communicateur de « superviseurs » chargés de la collecte des résultats et de leur stockage sur disque. Dans la pratique ce second groupe de processus se réduit à un seul élément. Cette division des tâches est justifiée par le fait que les travaux correspondants ainsi que les communications entre processus qui leurs sont associés, sont bien différents par nature. Ainsi, alors que les « itérateurs » ne communiquent qu'avec leurs voisins immédiats, le « superviseur », lui, communique avec tous les « itérateurs ».

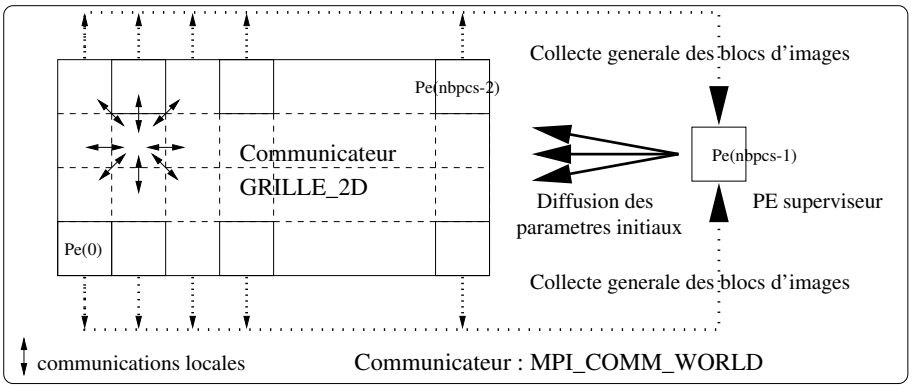

Figure 9. *Décomposition de domaine avec* MPI

En ce qui concerne la décomposition de domaine elle-même, étant donné que le système discret a une structure très régulière, nous avons distribué les données uniformément à l'ensemble des processus du communicateur des « itérateurs ». Ainsi, à chacun des itérateurs est affecté un ensemble de cellules contiguës qui représentent une portion d'image de la zone de subduction à une date donnée.

Pour notre simulation, nous avons partitionné le communicateur global MPI_COMM_WORLD en deux sous-communicateurs (fig. 9) avec la primitive MPI_Comm_split() : le premier constitué du seul « superviseur » et le second constitué du reste des processeurs du communicateur global. Au cours de la simulation, alors que le superviseur est

8. D'après [SNI 95], un communicateur est une « boite noire » dotée d'un certain nombre d'attributs et de règles simples concernant sa création, son utilisation et sa destruction. Un communicateur permet de définir un domaine de communication au sein de l'ensemble des processus, qui pourra être utilisé pour toutes sortes de communications locales.

chargé de gérer les accès aux disques et les diverses entrées-sorties et de collecter les portions d'images, les autres processeurs sont, eux, chargés d'itérer l'automate cellulaire en pas de temps. Ces « itérateurs », parce qu'ils doivent échanger leurs données aux frontières avec leurs processeurs voisins, vont être disposés sur une topologie cartésienne virtuelle régulière à deux dimensions appelée GRILLE_2D. En effet, ces communications locales se renouvelant excessivement fréquemment, justifient la création d'un support de communication spécial (un communicateur, donc). Nous créons cette topologie avec la primitive MPI_Cart_create() et offrons à l'utilisateur de la simulation, la possibilité d'imposer lui-même, dynamiquement, au lancement de l'exécutable, la taille de la grille des processus itérateurs dans chacune des dimensions.

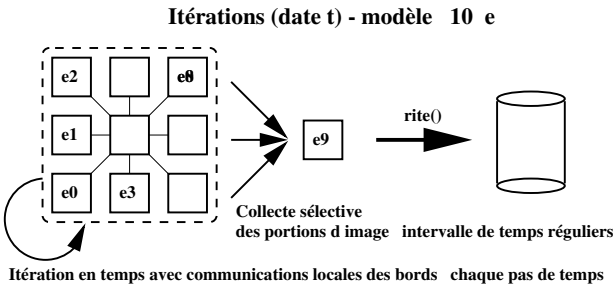

Figure 10. *Partitionnement de l'ensemble des processus en deux communicateurs*

Les primitives MPI_Cart_coords(), MPI_Cart_shift() et MPI_Cart_rank() de la bibliothèque permettent d'obtenir les coordonnées des processeurs (et de leurs voisins) à partir du rang du processeur courant et inversement.

5.2. *Les types dérivés pour les communications entre sous-domaines*

L'envoi de message est une opération coûteuse, à cause du temps de latence. Il faut donc, pour améliorer les performances de la simulation, s'efforcer d'envoyer toutes les cellules aux frontières, à un même processus destinataire en une seule fois.

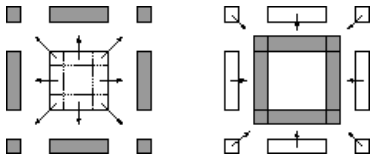

Figure 11. *Echange de messages entre un itérateur et ses voisins*

La bibliothèque d'échange de messages MPI propose trois mécanismes différents de regroupement de données individuelles au sein d'un même message. L'un d'eux consiste en la création de nouveaux types de données dérivés. En effet, dans les communications point à point de notre code, les données échangées sont bien souvent de types plus élaborés que les types "simples" classiques. Ainsi, nos processus de calcul échangent à chaque pas de temps deux lignes de cellules avec leurs voisins N et S, deux colonnes de cellules avec leurs voisins O et E, et des « angles » de quatre cellules avec leurs voisins NO, NE, SO, SE (fig. 11).

En effet, pour pouvoir appliquer la fonction de transition locale à chacune des cellules du sous-domaine courant, il faut que celles-ci aient connaissance des états respectifs des 24 cellules voisines. Or, aux frontières d'un sous-domaine donné, il faut aussi que les cellules aient connaissance des états de leurs voisines, c'est-à-dire des états de cellules situées théoriquement sur le processus voisin. Pour ce faire, il faut donc qu'à chaque itération, les processus voisins se communiquent les états des cellules situées à leurs interfaces respectives.

Avec une bibliothèque d'échanges de message classique (comme PVM), l'envoi et la réception de tels blocs de données auraient nécessité l'écriture d'un certain de nombre de boucles et entraîné, du même coup, des sur-coûts de compactage/décompactage et de recopie des contenus de messages. Avec MPI, nous avons créé des types dérivés et obtenu, avec les primitives MPI_Type_struct() et MPI_Type_vector(), respectivement un type structuré appelé Type_cellule et trois types de données homogènes vectoriels à pas constant (_type_lignes, _type_angles, _type_colonnes, voir fig. 11).

5.3. *Les types dérivés pour la collecte des portions d'image*

Au bout d'un certain nombre d'itérations de la fonction de transition, il faut sauver la configuration de l'automate afin de pouvoir générer, par post-traitement graphique, une animation de l'évolution de l'automate cellulaire. En fait, il suffit juste d'enregistrer la valeur du champ « couleur » de chacune des cellules. Or, du fait de la décomposition de domaine, les valeurs à sauvegarder sur disque sont distribuées sur l'ensemble des itérateurs. Il faut donc les centraliser sur un même processus pour les stocker sur un même disque.

Pour ce faire, et parce que nous ne disposons pas de la possibilité d'accéder concurremment et directement à un même fichier, nous avons adopté la stratégie suivante : regroupement des données distribuées sur un unique processeur (le superviseur) et écriture sur disque après ré-ordonnancement et reconstitution de chacune des images de la simulation. L'important dans cette étape est de ne désynchroniser aucun itérateur, afin de ne pas déséquilibrer la charge d'un processus itérateur par rapport à ses voisins. C'est pour cette raison, mais aussi parce que les entrées-sorties sur disque sont des opérations coûteuses, que le processus superviseur se charge entièrement de cette collecte des « blocs d'images » (ou tableau de ces champs « couleur » mis bout à bout) et du ré-ordonnancement et de l'écriture consécutifs.

Comme nous pouvons le constater sur la fig. 12, nous « inversons verticalement », à la collecte, le tableau des pixels de chaque sous-domaine. En effet, alors que le tableau de cellule est lu de bas en haut et de la gauche vers la droite, ligne par ligne (nous sommes en C), les utilitaires d'affichage lisent , eux, un tableau de pixels de haut en bas et de la gauche vers la droite. C'est aussi pour cette raison que nous « inversons verticalement » la position de chaque bloc de pixels dans l'ensemble de l'image (voir fig. 13).

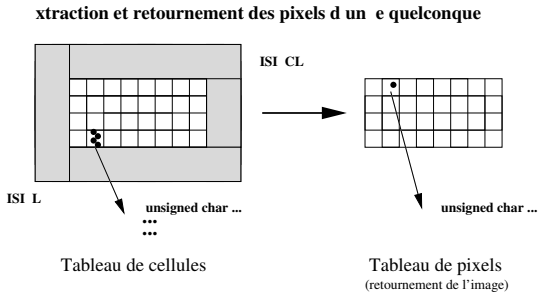

Figure 12. *Formation du tableau des pixels à partir du tableau de cellules*

Nous avons donc créé le type dérivé Type_bloc_image avec la primitive MPI_Type-_indexed() en considérant le tableau des cellules comme un tableau d'octets et en en extrayant l'octet correspondant à la couleur de la cellule avec des décalages à pas non constants (puisqu'il s'agit de ne prendre en compte que les cellules non situées sur les bords, les cellules fantômes n'étant en fait que des cellules redondantes du point de vue de l'information de couleur qui leur est attachée). Pour obtenir des tailles de champs alignées en mémoire, nous avons utilisé la primitive MPI_Type_extent().

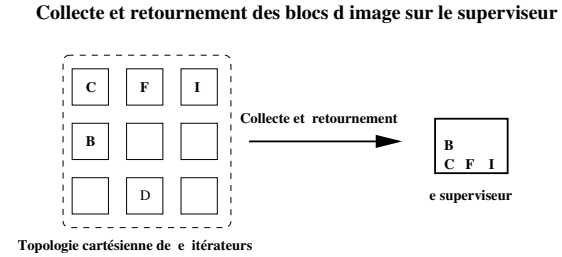

Figure 13. *Collecte et réordonnancement des portions d'images*

Enfin, après émission d'une donnée « Type_bloc_image » par un itérateur quelconque au cours de la simulation, il faut que le superviseur réceptionne le tableau d'octets correspondant et le stocke au bon endroit dans l'image globale. Pour ce faire, nous avons créé le type dérivé Type_image_globale avec la primitive MPI_Type_vector().

Une fois encore, ce dernier type nous permet de réaliser deux opérations en une seule puisque, avec une librairie « conventionnelle », il nous aurait d'abord fallu réceptionner le bloc d'image avant de le scinder en petits blocs à répartir (ou recopier) dans divers endroits de l'image globale.

5.4. *Utilisation de « ghost » cellules aux interfaces des sous-domaines*

Dans le cadre d'une distribution bi-dimensionnelle (ou même uni-dimensionnelle) des données, chaque sous-domaine doit envoyer ses cellules aux frontières aux processus voisins et, en retour, recevoir leurs propres cellules. Il faut donc qu'à chaque itération, chaque processus stocke des cellules n'appartenant pas à son propre sous-domaine (avant de les utiliser pour mettre à jour les cellules aux bords du sous-domaine). Pour résoudre ce problème de stockage en local d'informations non-locales, nous avons introduit dans le modèle une zone de « recouvrement » de « ghost » cellules. Ces cellules qui n'ont pas « d'existence réelle » vis-à-vis de la modélisation sont physiquement stockées sur le pourtour du sous-domaine de cellules du processus courant. Elles étendent ainsi de deux rangées ou colonnes les cellules qui sont situés sur les « bords intérieurs » (voir fig. 14).

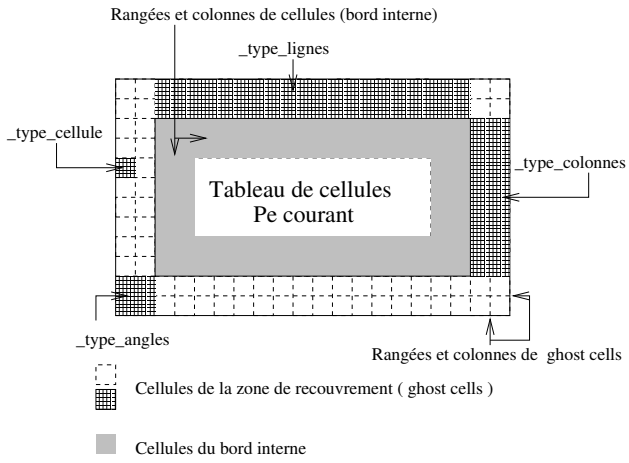

Figure 14. *Les cellules fantômes aux frontières d'un sous-domaine*

Ainsi, à chaque pas de temps, les « ghost » cellules du sous-domaine courant prennent pour nouvelles valeurs celles des cellules correspondantes dans les sous-domaines voisins. Par exemple, les deux colonnes de « ghost » cellules du bord droit prennent les valeurs des deux colonnes de cellules du « bord intérieur » gauche envoyées par le processus situé à l'est du processus courant. Tandis que les deux colonnes de cellules du « bord intérieur » droit du sous-domaine du processus courant

sont envoyées au processus de l'est afin d'alimenter ses propres deux colonnes de « ghost » cellules situées sur le bord gauche du sous-domaine concerné.

5.5. *L'algorithme parallèle*

```
programme
  Initialisation de MPI
  Création de la topologie cartésienne de processus
  Création des types de données dérivés
  Si le processus courant est superviseur alors :
    Itérer
      Pour chaque itérateur faire
        Identifier l'itérateur à l'origine du nouveau message,
        Réceptionner le bloc image correspondant,
        Et le placer simultanément au bon endroit dans
          l'image globale,
      Fin du pour.
      Sauver l'image globale sur disque au format Sun-Raster,
    Tant que la condition de sauvegarde (1) n'est pas réalisée.
  Sinon
    Itérer
      Envoi non bloquant des cellules du « bord interne »
        aux 8 itérateurs voisins,
      Application de la fonction de transition aux cellules
        « propres » (placées à plus de 4 cellules des bords),
      Réception bloquante des « ghost » cellules en provenance
        des itérateurs voisins,
      Application de la fonction de transition aux cellules
        situées sur le bord interne (i.e. aux cellules situées
        à 2 ou 3 cellules des bords),
      Si la condition d'envoi (2) est réalisée alors :
        Envoi du bloc d'image local (« ghost » cellules
          exclues) au superviseur,
      Fin du si.
    Tant que la condition de sauvegarde (1) n'est pas réalisée.
  Fin du si.
  Synchronisation de l'ensemble des processus,
  Sortie de MPI.
```

Figure 15. *L'algorithme parallèle*

Comme nous le montrons dans l'algorithme de la fig. 15, chaque processus itérateur boucle un nombre fini de fois (tant qu'une certaine condition de sauvegarde (1) n'est pas réalisée) sur la production du « bloc image » correspondant à l'état des cellules de son sous-domaine, après application de la fonction de transition. Si, au cours d'une de ces itérations, une condition d'envoi (2) est satisfaite, alors il envoie son « bloc image » au processus superviseur. De son côté, le superviseur sert chaque itérateur tour à tour, en stockant chacun des « blocs images » envoyés au bon endroit. Les deux conditions de sauvegarde et d'envoi précitées sont fonctions de trois paramètres donnés au lancement de la simulation : nombre total d'images à produire, période entre deux images consécutives et numéro de la première image. Une condition importante de la simulation est que le nombre de cellules de l'automate soit un multiple du nombre de processus itérateurs dans chaque dimension !

Dans le but d'améliorer les performances de la simulation parallèle, nous avons fait en sorte qu'il y ait un recouvrement des temps de communication et des temps de calcul dans la portion de code correspondant à l'application de la fonction de transition globale de l'automate. Ainsi, chaque itérateur effectue un envoi non-bloquant des cellules du « bord interne » et enchaîne immédiatement (sans se soucier de leurs réception par les itérateurs voisins) sur la mise à jour des cellules de son sous-domaine propre qui ne nécessitent pas la connaissance d'état de cellules mises à jour au niveau des sous-domaines voisins. A la réception bloquante des cellules de l'interface en provenance des 8 sous-domaines voisins, l'itérateur termine la mise à jour des cellules situées au bord de son sous-domaine.

6. Conclusion et perspectives

Cette simulation bi-dimensionnelle a été écrite en langage C, mise au point sur un réseau de trois PC sous Linux sur lesquels nous avons installé l'environnement LAM 6.1 (Local Area Multicomputer, une implémentation de MPI) et portée ensuite sur le CRAY T3E à 256 processeurs de l'IDRIS [9] et sur l'ORIGIN2000 à 32 processeurs du Pôle Parallélisme Ile-de-France Sud [10].

Nous avons dû veiller à assurer la portabilité du code sur l'ensemble de ces machines utilisées qui sont à la fois « big-endian » (comme le CRAY T3E et l'ORIGIN2000) et « little-endian [11] » (comme les PC) et dont le format de codage des entiers n'est pas homogène (une variable de type « int » est codée sur huit octets sur le CRAY T3E et sur quatre ailleurs). C'est pourquoi, nous avons mis en place des directives à l'usage du pré-compilateur telles que : redéfinition du type entier sur quatre octets (ce qui correspond au type « short » sur le CRAY T3E), inversion de l'orientation des données sur

9. Institut du Développement et des Ressources en Informatique Scientifique, CNRS, Orsay.

10. Ce super-calculateur est situé dans les locaux du Laboratoire de Mécanique et Technologie de l'ENS-Cachan.

11. Une machine little-endian place les octets de poids faible d'un mot en premier, alors qu'une machine big-endian place ceux de poids forts en premier.

les PC afin que les données formatées inscrites dans les fichiers de résultat au format Sun-Raster soient bien compatibles « big-endian »...

Du fait de notre choix d'optimiser les communications en créant des types de données dérivés, nous sommes obligés de décomposer le domaine global en des sous domaines de tailles identiques. En effet, il importe que les types de données homogènes vectoriels à pas constant _type_lignes, _type_angles, _type_colonnes (voir fig. 14) aient la même signification pour tous les itérateurs. Donc, si nous considérons un tableau global de 1000 cellules en ligne pour 200 cellules en colonne, le nombre de processeurs itérateurs en ligne doit être un diviseur de 1000 et le nombre de processus itérateurs en colonne doit être un diviseur de 200. C'est pour cette raison que, dans la fig. 16, nous ne pouvons donner des temps de calcul que pour un certains nombres de processeurs. Les résultats énoncés en fig. 16 correspondent à une simulation sur 2000 pas de temps avec génération d'une image tous les 100 pas de temps.

Nombre de processeurs	1	2	3	5	6	9
CRAY T3E	958.91	960.34	521.10	274.30	219.69	149.16
ORIGIN2000	469.72	474.80	273.40	143.75	117.91	81.23
Nombre de processeurs	11	17	21	26	33	41
CRAY T3E	116.95	77.78	64.29	58.17	43.14	36.88
ORIGIN2000	66.05	44.19	37.10	34.36	/	/
Nombre de processeurs	51	65	81	101	126	161
CRAY T3E	30.87	28.36	17.62	15.04	13.40	11.01

Figure 16. *Comparaison des temps de calcul (en secondes) sur les deux machines*

Comme nous pouvons le constater à la lecture des fig. 16 et 17, à nombre de processeurs de calcul équivalent, l'ORIGIN2000 est plus rapide que le CRAY T3E. Par contre, la fig. 18 montre que l'accélération est plus forte sur le CRAY T3E que sur l'ORIGIN2000. Il semble donc que, pour ce type d'application relativement « communicante », le réseau de communication du CRAY T3E soit le plus adapté.

Par ailleurs, en observant l'accélération de la simulation sur le CRAY T3E en fig. 18, on note une réelle chute de l'efficacité de l'optimisation parallèle pour 65 processeurs. En effet, alors qu'elle est supérieure à six dixièmes pour 51 et 81 processeurs, elle tombe brutalement à la valeur $0{,}52$. Nous supposons que ceci est dû au fait que plus la taille des sous domaines est grande proportionnellement à la taille de leurs interfaces, meilleure est l'efficacité du code parallèle. En d'autre termes, à taille de problème globale constante [12] l'efficacité décroît quand le sous domaine diminue et que la proportion de cellules aux interfaces (et donc à échanger) augmente. Or, dans le cas à 65 processeurs, nous sommes obligés de travailler sur une grille de 8×8 processeurs itérateurs (en effet 16 n'est pas un diviseur de 1000) ce qui implique que chaque itérateur doit traiter un sous domaine de 125×25 cellules, avec 18,6 % de celles-ci

12. C'est notre cas ici, puisque ces mesures ont été faites dans les mêmes conditions initiales et ont générées chacunes 20 images par pas de 100 unités de temps.

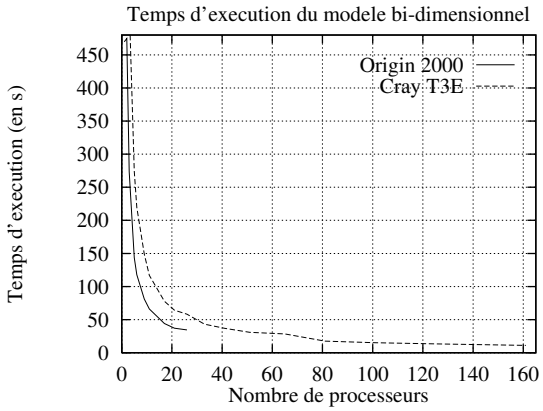

Figure 17. *Temps de traitement comparés selon la machine*

situées aux interfaces. Dans le cas à 81 processeurs par contre, nous travaillons sur une grille de 20×4 processus itérateurs ce qui implique que chaque itérateur doit traiter un sous domaine de 50×50 cellules, avec 15,3 % de celles-ci situées aux interfaces. Cette chute des performances pour 65 processus est donc explicable.

Nous pouvons par conséquent en conclure qu'il importe aussi de disposer la grille des processus de façon adéquate (afin de diminuer la taille des problèmes aux interfaces) si l'on souhaite obtenir un rendement optimal.

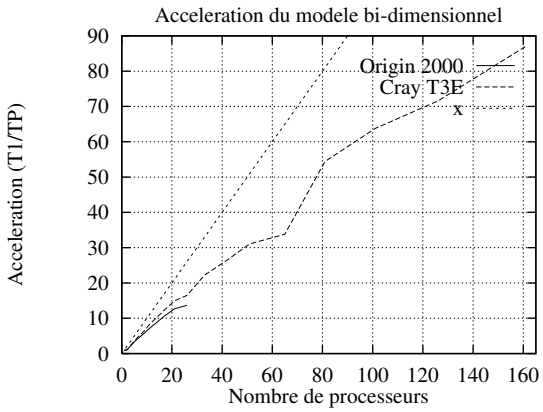

Figure 18. *Accélérations comparées selon la machine*

Des animations graphiques bi-dimensionnelles sous forme de fichiers MPEG sont téléchargeables sur le site :

http://quartz.dgs.jussieu.fr:8080/~

L'accélération de cette simulation bi-dimensionnelle est quasi-linéaire et croissante sur l'intervalle d'étude (fig. 18). L'intérêt d'une solution parallèle est flagrant puisque, alors que le temps de simulation séquentiel est de plus d'un quart d'heure dans le cas (relativement restreint) que nous avons exposé précédemment, il descend à 11 secondes dans le cas parallèle sur 161 processeurs du CRAY T3E ! L'utilisation des nombreuses fonctionnalités de la bibliothèque de communication MPI pour les décompositions de domaine s'est donc avérée rentable dans le cas de notre simulation bi-dimensionnelle.

Une plate-forme logicielle spécifique $\mathcal{LAC}$ (Langage pour Automates Cellulaires) est en cours de création. Elle comportera à terme : un langage propre dédié à la programmation des automates cellulaires bi-dimensionnels, un petit compilateur associé [13] qui génère du code en langage C avec des appels aux primitives de la bibliothèque d'échange de messages MPI (pour le parallélisme) et une interface graphique [14] pour l'initialisation de l'automate cellulaire.

Remerciements

Ces travaux ont été réalisés avec l'assistance et les moyens informatiques de l'Institut du Développement et des Ressources en Informatique Scientifique (IDRIS).

7. Bibliographie

[ACK 98] ACKLAM E. et LANGTANGEN H. P., « Parallelization of Explicit Finite Difference Schemes via Domain Decomposition ». Rapport technique, Oslo Scientific Computing Archive, February 1998. http://www.math.uio.no/OSCA.

[BAK 87] BAK P., TANG C. et WIESENFELD K., « Self-Organized Criticality : an explanation of 1/f noise ». *Physical Review. Letters*, vol. 59, n° 4, p. 381–384, July 1987.

[BOU 96] BOURGOIS J., « Un processus naturel pour éliminer définitivement les déchets nucléaires ultimes ». *Réalités Industrielles - Une série des Annales des Mines*, p. 5–12, janvier 1996.

[BRU 96] BRUGEAS I., Utilisation de MPI en décomposition de domaine. http://www.idris.fr, mars 1996. Publication IDRIS.

[CON 97] CONDIE K. C., *Plate Tectonics and Crustal Evolution*. Butterworth Heinemann, 4^{th} édition, 1997.

[DUB 95] DUBOIS J., *La dynamique non linéaire en physique du globe*. Géophysique interne. Masson, 1995.

[DUR 95] DURAN J., « La physique du tas de sable ». *Revue du Palais de la Découverte*, vol. 23, n° 224, p. 21–39, janvier 1995.

13. Écrit avec Lex et Yacc.
14. Écrite en Perl/Tk.

[DUR 96] DURAND-LOSE J. O., « *Automates Cellulaires, Automates à Partitions et Tas de Sable* ». Thèse de Doctorat, Université de Bordeaux I, Juin 1996.

[DUR 97] DURAN J., *Sables, poudres et grains (Introduction à la physique des milieux granulaires)*. Editions Eyrolles, 1997.

[GEI 94] GEIST A., BEGUELIN A., DONGARRA J., JIANG W., MANCHEK R. et SUNDERAM V., *PVM: Parallel Virtual Machine (A Users' Guide and Tutorial for Networked Parallel Computing)*. Scientific and Engineering Computation series. MIT Press, 1994. http://www.netlib.org/pvm3/book/pvm-book.html.

[GRU 93] GRUMBACHER S. K., MCEWEN K. M., HALVERSON D. A., JACOBS D. T. et LINDNER J., « Self-Organized Criticality : an experiment with sandpiles ». *American Journal of Physics*, vol. 61, n° 4, p. 329–335, April 1993.

[HUA 92] HUANG J., NARKOUNSKAIA G. et TURCOTTE D. L., « A cellular-automata, slider-block model for eathquakes II. Demonstration of self-organized criticality for a 2-D system ». *Geophysical Journal International*, p. 259–269, 1992.

[JAE 92] JAEGER H. M. et NAGEL S. R., « Physics of the Granular State ». *Science*, vol. 255, p. 1481–1612, March 1992.

[JOL 95] JOLIVET L., *La déformation des continents : exemples régionaux*. Enseignement des sciences. Hermann, 1995.

[KIN 87] KINCAID C. et OLSON P., « An experimental study of subduction and slab migration ». *Journal of Geophysical Research*, vol. 92, n° B13, p. 13,832–13,840, december 1987.

[LAF 95] LAFAURIE B., « *Modélisation de la convection par une méthode de gaz sur réseau et technique de suivi d'interface* ». Thèse de Doctorat, Université Paris XI Orsay, mars 1995. N° d'ordre 3622.

[LED 97] LEDUC T., « Modélisation par un système dynamique discret du processus de subduction-érosion en tectonique des plaques : première approche uni-dimensionnelle ». Rapport interne, Laboratoire LIP6, ftp://ftp.lip6.fr/lip6/reports/1997/lip6.1997.008.ps.gz, Mai 1997.

[LED 98] LEDUC T., « A One-Dimensional Discrete Computer Model of the Subduction Erosion Phenomenon ». In *CESA'98 Nabeul-Hammamet*, April 1998. (2e Congrés Mondial IMACS et IEEE/SMC).

[MAL 93] MALAVIEILLE J., LARROQUE C. et CALASSOU S., « Modélisation expérimentale des relations tectonique/sédimentation entre bassin avant-arc et prisme d'accrétion ». *Comptes Rendus de l'Académie des Sciences, Paris*, vol. 316, Série II, p. 1131–1137, 1993.

[MÜL 96] MÜLLER D., « *Techniques informatiques efficaces pour la simulation de milieux granulaires par des méthodes d'éléments distincts* ». Thèse de Doctorat, EPFL, 1996. Dépt de Mathématiques - Ecole Polytechnique Fédérale de Lausanne - thèse numéro 1545.

[NAR 92] NARKOUNSKAIA G. et TURCOTTE D. L., « A cellular-automata, slider-block model for eathquakes I. Demonstration of chaotic behaviour for a low order system ». *Geophysical Journal International*, p. 250–258, 1992.

[SCH 94] SCHNÜRLE P., « *Contribution à la compréhension des mécanismes dérosion tectonique et à la quantification des flux de matière dans les zones de subduction* ». Thèse de Doctorat, Université Pierre et Marie CURIE, Paris VI, mars 1994.

[SHE 92] SHEMENDA A. I. et GROCHOLSKY A. L., « Physical modelling of lithosphere subduction in collision zones ». *Tectonophysics*, vol. 216, p. 273–290, 1992.

[SMI 91] SMITH R., « The application of cellular automata to the erosion of landforms ». *Earth Surface Processes and Landforms*, vol. 16, p. 273–281, 1991.

[SNI 95] SNIR M., OTTO S., HUSS-LEDERMAN S., WALKER D. et DONGARRA J., *MPI : The Complete Reference*. Scientific and Engineering Computation series. MIT Press, 1995. http://www.netlib.org/utk/papers/mpi-book/mpi-book.html.

[TAN 88] TANG C. et BAK P., « Critical exponents ans scaling relations for Self-Organized Critical phenomena ». *Physical Review Letters*, vol. 60, n° 23, p. 2347–2350, June 1988.

[VIC 84] VICHNIAC G. Y., « Simulating physics with cellular automata ». *Physica D*, n° 10, p. 96–116, 1984.

Thomas Leduc *est doctorant en informatique au laboratoire LIP6 de l'Université Paris VI depuis décembre 1995 et Ingénieur de Recherche CNRS aux laboratoires LSV et LMT de l'ENS Cachan depuis le mois de juillet 1998. Au cours de son travail doctoral, Thomas Leduc a travaillé sur la modélisation d'un phénomène géotectonique en étroite collaboration avec le LGTE (Laboratoire de Géodynamique Tectonique et Environnement, CNRS-ESA 7073 et UFR 928 de l'Université Pierre et Marie Curie).*