Master 1 – Devoir d'Algorithmique et Architectures Parallèles

Florent Bouchez & Veronika Sonigo
(basé sur une idée de Loris Marchal et Cédric Tedeschi)

À réaliser en binôme pour le 30 novembre 2008.

Dans ce DM, on souhaite simuler un océan dans lequel cohabitent des requins et des sardines. L'océan sera assimilé à une grille 2D torique. Initialement, chaque point de la grille peut contenir un requin ou une sardine. La population de chacune des espèces évolue avec le temps (discret) selon certaines règles locales.

Pour les sardines :

- À chaque étape, une sardine essaye de se déplacer dans une case voisine choisie alétoirement. Si celle-ci n'est pas libre, la sardine ne bouge pas. Les sardines se déplacent vers le haut, le bas, la droite ou la gauche, mais pas en diagonale.
- À partir d'un certain âge (A_{sardine} nombre d'étapes) et lorsqu'elles se déplacent, les sardines peuvent se reproduire : en quittant une case, une sardine laisse une jeune sardine (d'âge 0) derrière elle, avec une probabilité p_{sardine}. Si une sardine ne se déplace pas, elle ne peut pas se reproduire.
- Les sardines ne meurent pas sauf si elles sont mangées par les requins.

Pour les requins :

- Les requins se déplacent de la même façon que les sardines, en choisissant aléatoirement une case voisine parmi celles du haut, du bas, de la gauche et de la droite. Si cette case est déjà occupée par un requin, le déplacement n'a pas lieu. Si cette case est occupée par une sardine, le requin se déplace et mange la sardine.
- Les requins se reproduisent de la même façon que les sardines : si un requin atteint l'âge de reproduction A_{requin} et se déplace, il laisse un jeune requin d'âge 0 derrière lui avec probabilité p_{requin}.
- Les requins ne mangent que des sardines. Si un requin n'a pas mangé depuis T_{famine}, il meurt.

Les deux parties de ce devoir ont une dépendance temporaire de même nature que leur placement spatial : il est nécessaire d'avoir réfléchi en profondeur à la première partie avant d'entamer la seconde. Lorsque vous donnerez des algorithmes, vous préciserez les structures de données utilisées et expliquerez *leur fonctionnement. Vous devez rendre pour la date indiquée une archive contenant votre code ainsi qu'un rapport (de préférence en LaTeX) pour les questions posées.*

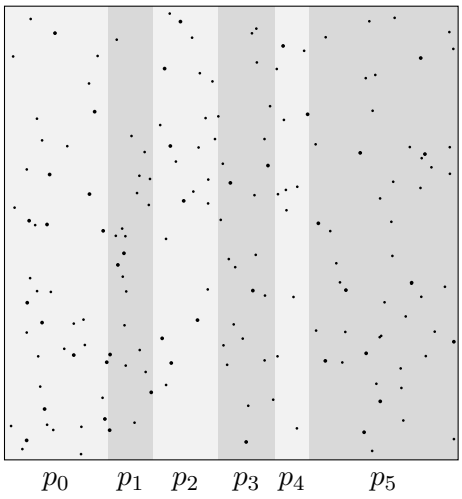

FIG. 1 – Exemple de situation initiale hétérogène, avec une distribution par bandes sur 6 processeurs, avec 43 requins et 129 sardines. (Les gros points représentent les requins, les petits les sardines.)

1 Origamis de poissons

Pour paralléliser la simulation, on se propose de découper l'océan par bandes, comme représenté sur la figure 1. Chaque bande est attribuée à un processeur et il y a autant de bandes que de processeurs. Le plus imporant pour une simulation en parallèle est que le résultat final soit indépendant du nombre de processeurs. En particulier, le résultat doit être identique à celui obtenu par simulation séquentielle sur un unique processeur.

Question 1. *Dans notre simulation, qu'est-ce qui a priori posera problème pour garantir que le comportement en parallèle est bien le même qu'en séquentiel ? Vous prendrez soin par la suite de justifier que vos algorithmes ne modifient pas le comportement en parallèle.*

Question 2. *Des données devront être échangées par les processeurs au niveau des frontières de leur partie d'océan. Proposez un protocole de communication qui minimise le nombre de données échangées. Précisez les conditions de fonctionnement de votre protocole, et les cas d'optimalité.*

Si la distribution des poissons est homogène, les bandes allouées aux processeurs peuvent être choisies de même largeur. En revanche, une distribution hétérogène qui comporte des zones très denses et d'autres peu denses alourdit la tâche de certains processeurs et ralentit la simulation globale.

On suppose pour l'instant qu'un processeur connaît l'océan dans sa totalité, et que les communications sont gratuites.

Question 3. *Proposez un algorithme de découpage optimal pour la répartition de la charge de travail en fonction de la distribution des poissons initiale.*[1] *N'oubliez pas que chaque processeur doit mettre à jour tous les poissons* et *parcourir* toutes *les cases de sa bande d'océan.*

[1] La colonne 0 n'est pas une frontière obligatoire...

On suppose maintenant que les communications sont beaucoup plus chères que le calcul (et on a au moins deux processeurs !).

Question 4. *Proposez un algorithme de découpage qui minimise les données à échanger entre les processeurs sur le plus d'étapes possible (en moyenne). Veillez à donner au moins un poisson à chaque processeur si possible.*

On suppose maintenant que l'on a aucun information *a priori* sur les vitesses de calcul et de communication.

Question 5. *Proposez un algorithme général qui optimise le temps passé à calculer et à communiquer.*

En pratique, on suppose que l'océan est trop gros pour tenir dans la mémoire d'un seul processeur. De plus, le déplacement des poissons modifie la configuration initiale et un découpage initialement très performant peut devenir médiocre. Le découpage initial est en fait choisi arbitrairement (avec chaque bande de largeur égale à un près) ; le but est alors de re-découper au besoin l'océan de manière plus adaptée, avec des informations locales (impossible de regrouper tout l'océan dans un processeur).

Question 6. *Proposez un mécanisme de détection d'un déséquilibre dans le découpage actuel. Proposez également un algorithme distribué de re-découpage de l'océan. Vous prendrez soin de justifier le nouveau découpage choisi ainsi que la mise en place du re-découpage effectif.*

Question 7. *Estimez la perte du qualité dû à votre découpage distribué par rapport à votre algorithme global ? (Illustrez par un exemple.)*

On suppose maintenant $p = k^2$ processeurs, et un découpage de l'océan par blocs rectangulaires, non nécessairement identiques.

Question 8. *Expliquez les différences entre ce découpage et le découpage par bandes : présentez les avantages ou inconvénients en termes de performances pour la simulation (calcul et communications), et en termes d'effort de développement (complexité des algorithmes de communication, de détection de déséquilibre et re-découpage).*

2 MPI (Mettre Plein de sardInes)

Le but de cette partie est d'écrire une implémentation MPI d'un simulateur distribué. Votre implémentation devra prendre pour entrée un fichier de description du type :

```
1000                    (taille de la grille : 1000 × 1000)
1 3 sardine 3           (description des poissons : x y type age)
3 5 shark 12
⋮
```

Vous trouverez sur la page web des TDs[2] deux scripts `gen_homo.c` et `gen_hetero.c` qui génèrent respectivement une distribution homogène et une distribution hétérogène de poissons. Vous y trouverez également un exemple de simulateur séquentiel `simul_seq.c` dont vous pouvez vous inspirer, et que vous pouvez utiliser pour tester différents jeux de paramètres.

On choisit un découpage par bandes (verticales) de dimensions (taille × taille/p).

[2] http://graal.ens-lyon.fr/~vsonigo/teaching/AlgoPar/algopar2008.html

Question 9. *Écrivez en MPI le code d'initialisation de la simulation. Chaque processeur peut lire le fichier de description de l'océan mais ne garde que les informations qui correspondent à sa bande. De plus, à chaque étape, après avoir mis à jour sa zone, chaque processeur devra afficher son numéro et le nombre de requins et sardines contenus dans sa zone. Trouvez un mécanisme pour que les processeurs affichent ces informations dans le bon ordre.*

Question 10. *Implantez maintenant la simulation. Ne vous préoccupez pas pour l'instant du re-découpage. Basez-vous sur vos algorithmes donnés en première partie pour les communications (vous pouvez éventuellement implanter des versions plus simples en justifiant votre choix).*

Testez votre implémentation sur des océans de tailles différentes, avec des distributions homogènes et hétérogènes. Comparez les performances dans les différents cas, en termes de calculs (charges équilibrées ?) et communications.

Après un certain nombre d'étapes de simulation, il se peut que la localisation des poissons change et que la distribution des données sur les processeurs ne soit plus optimale. Ou, plus simplement, la distribution initiale est hétérogène.

Question 11. *Implantez un mécanisme de détection de déséquilibre et de re-découpage de l'océan basé sur vos algorithmes de la partie précédente. Affichez le nouveau découpage (les nouvelles bornes de chaque processeur) à chaque modification.*

Comparez les performances par rapport à un découpage par bandes figé.

Question 12 (Bonus)**.** *Implantez un découpage par carrés de taille* $(taille/k \times taille/k)$ *sur* $p = k^2$ *processeurs. (Sans vous préoccuper du re-découpage.)*

Question 13 (Bonus)**.** *Faites un processeur dédié à un affichage* minimal *de l'océan : nombre total de requins, de sardines, plus quantité par régions et taille des régions. Ces informations ne doivent pas être mises à jour à chaque étape mais plutôt périodiquement (100 étapes, 1 seconde, etc.) Utilisez une fenêtre graphique avec des pixels de couleurs différentes si l'océan est petit.*

Rendez également ce processeur responsable de la lecture du fichier de description de l'océan, et de la répartition des données sur les autres processeurs. Vous devez supposer que ce processeur ne peut pas contenir en mémoire toutes les informations sur l'océan.