DM d'Algo parallèle

Elie Gédéon

30 novembre 2012

Questions théoriques rédigées en binôme avec Hadrien Croubois

Compilation

Pour compiler les sources, lancez `make` depuis le dossier téléchargé. Les fichier produits sont placés dans le même dossie.

Questions théoriques

Question 3

Considérons que nous avons une grille Q de taille $n \times n$ torique sur laquelle nous désirons exécuter le jeu de la vie de conway, et p^2 processeurs placés sur une grille $p \times p$ torique grid. Nous associons, de plus, a chaque processeur P ces coordonnées cartésiennes sur la grille de processeurs P_x, P_y.

La grille de jeu Q sera partagé en attribuant a chaque processeur P une grille Q_P de dimension :

$$Q_P.size\_x = \begin{cases} n/p, \text{ si } P_x < p \\ n - p * (n/p), \text{ sinon} \end{cases}$$

$$Q_P.size\_y = \begin{cases} n/p, \text{ si } P_y < p \\ n - p * (n/p), \text{ sinon} \end{cases}$$

Exemple en figure avec $n = 7$ et $p = 3$

À chaque étape, le processeur P a besoin de connaître les valeurs voisines de ses bord afin de se mettre à jour. Les valeurs voisines sont les limites gauche (resp. droite, haute et basse) du processeur a leur ouest (resp. est, sud, nord) ainsi que le coin supérieure gauche (resp. supérieur droit, inférieur gauche, inférieur droit) du processeur a leur sud-est (resp. sud-ouest, nord-est, nord-ouest).

Une fois ces donnés échangés, chaque processeur peut mettre a jour sa grille et recommencer une nouvelle étape.

La topologie grille empêchant les échanges avec les voisins en diagonal il convient de contourner ce problème. Pour cela, le processeur P stockera sa

0	0	0	1	1	2	2
0	0	0	1	1	2	2
0	0	0	1	1	2	2
3	3	3	4	4	5	5
3	3	3	4	4	5	5
6	6	6	7	7	8	8
6	6	6	7	7	8	8

FIGURE 1 – Répartition de la grille Q sur les processeurs.

2		0	0	0		1
	↘	↓	↓	↓	↙	
5	→	3	3	3	←	4
5	→	3	3	3	←	4
	↗	↑	↑	↑	↖	
8		6	6	6		7

FIGURE 2 – Communication entre le processeur 3 et ses voisins.

grille $G_P.size\_x \times G_P.size\_y$ dans une matrice de taille $(G_P.size\_x + 2) \times (G_P.size\_y + 2)$.

Ainsi on pourra stocker la partie attribué au processeur P dans la partie centrale de cette grille, le contour étant gardé pour le stockage des voisins.

Les communications se divise donc en deux parties :

- Partage avec les voisins est (resp. ouest) des parties centrale (sur une hauteur $G_P.size\_x$) des avant dernières colonnes à droite (resp. gauche) de la grille G_P
- Partage avec les voisins nord (resp. sud) de l'intégralité (sur une largeur $G_P.size\_y + 2$) des avant dernières lignes en bas (resp. haut) de la grille G_P

Une fois les voisins mis à jour, chaque processeur peut facilement calculer les cellules du centre de la grille (correspondant aux cases qui lui sont attribués sans se soucier des bords. La procédure de mise à jour des voisinages doit être répétée au début de chaque étape.

Question 5

Afin d'avoir un automate bien fondé il faut éviter toute dépendance circulaire. On distinguera ces dépendances en deux sous famille :

$$a_{i,j} \Leftrightarrow a_{i+1,j}$$

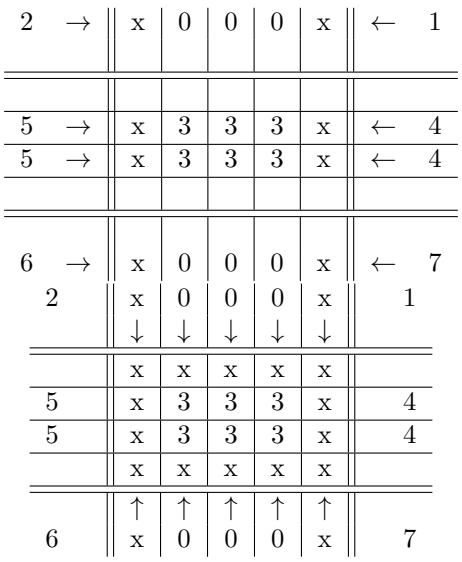

FIGURE 3 – Étapes de communication pour la mise à jour du voisinage

Les dépendances triviale : si $\{N, S\} \subset DEP$ ou $\{E, W\} \subset DEP$ alors on à dépendance réciproque entre deux cases voisines, qui doivent toutes les deux être calculés avant l'autre.

$$a_{i,j} \Rightarrow a_{i+1,j} \Rightarrow \ldots \Rightarrow a_{i,j}$$

Les dépendances circulaire : si la grille considère est circulaire selon un ou plusieurs axes (torique) la présence d'une dépendance selon un axe parallèle à l'un des axe de circularité entraîne une dépendance circulaire. Par exemple sur une grille torique, la présence de $W \in DEP$ fait que toute cellule à la limite gauche dépend de la cellule sur la même ligne sur la colonne de droite qui dépend de son voisin de gauche, qui dépend de son voisin de gauche, ..., qui dépend de la case considéré initialement.

On voit par ailleurs que si deux dépendance selon deux axes perpendiculaire sont présentes, on peut calculé toutes les cellules en commençant par celle dans le coin pointé par les dépendances et en procédant diagonale par diagonale en partant de cette case.

Ainsi sont valides :

– Sans circularité :

$$DEP \in \{u \cup v | u \subsetneq \{N, S\}, v \subsetneq \{E, W\}\}$$

– Avec circularité verticale :

$$DEP \in \{u | u \subsetneq \{E, W\}\}$$

– Avec circularité verticale :

$$DEP \in \{u | u \subsetneq \{N, S\}\}$$

– Avec circularité torique :

$$DEP = \emptyset$$

Question 6

Vu qu'ici, on impose l'ordre de calcul, les dépendances S et E sont relâchées.

– On découpera la grille de cellules en blocs répartis équitablement entre les processeurs de la grille.
Si on n'a pas de dépendance W, on peut utiliser un algorithme simple de type :
Chaque processeur execute pour chaque colonne lui appartenant, dans l'ordre :
  – envoie au processeur du haut l'ancienne valeur (sauf s'il est dans la première ligne)
  – recoit du processeur du bas l'ancienne valeur (sauf s'il est dans la dernière ligne)
  – recoit du processeur du haut la nouvelle valeur (sauf s'il est dans la première ligne)
  – calcule la colonne de haut en bas
  – envoie au processeur du bas la valeur de la case la plus en bas (sauf s'il est dans la dernière ligne)

Le coût est en $O((m * h + 1/b + L) * (k - 1) + m * h * l)$ (m est le coût d'une opération, h et l sont les hauteur et largeur de la grille locale, b le débit, L la latence, k la hauteur de la grille de processeur ; on considère $m * h > L + 1/b$)
On gagne un facteur de l'ordre du nombre de processeurs de la grille
Si on a pas de dépendance en N, on utilise le même algorithme en permutant ligne et colonne.
Si on a une dépendance en N et W, on ne peut pas utiliser ce type d'algorithme, car T(x,y) a besoin que $\forall n \leq x, m \leq y, m \neq y$ ou $n \neq x, T(m, n)$ ait été calculé.
On peut ceci dit avoir un algorithme de type :
– $(0, 0)$ calcule son bloc
– $(0, 1)$ et $(1, 0)$ calculent leur bloc
– $(0, 2)$, $(1, 1)$ et $(2, 0)$ calculent leur bloc
– ...

Le gain sera de l'ordre de $\sqrt{nombredeprocesseurs}$
En fait, le dernier processeur qui commence a calculer est obligé d'attendre tous les autres pour commencer.

– La cyclicité ne change rien aux résultats précédents si la grille est cyclique, car il suffit que les processeurs de la dernière ligne/colonne transmettent au début leur dernière ligne/colonne aux processeurs correspondant de la première ligne/colonne.
– Sur un anneau, on peut s'inspirer des algorithmes de parcours de drapeau. Ici, la dépendance n'a aucune importance, et la cyclicité non plus.

Question 8

Prouvons qu'on ne peut pas trouver de transformation temps espace.

Soit $l(x, y, t)$ la fonction de simulation de deplife (elle vaut 1 si la case (x, y) est vivante à l'instant t et 0 sinon)

Soit $m(x, y, t)$ la fonction de simulation d'un automate cellulaire C, tel qu'il existe une fonction γ, tel que :

$$m(\gamma(x, y, t)) = l(x, y, t)$$

$\forall (x, y, t)$

$$\begin{aligned}
& l(x,y,t) \text{ dépend de } l(x-1,y,t-1), l(x,y,t-1) \\
\Rightarrow\ & \exists \psi, \gamma_t(x,y,t) = \psi(y,t) \\
& l(x,y,t) \text{ dépend de } l(x-1,y,t-1), l(x,y-1,t) \\
\Rightarrow\ & \psi(y,t) = \psi(y+1,t-1) \\
& l(x,y,t) \text{ dépend de } l(x-1,y,t-1), l(x,y+1,t-1) \\
\Rightarrow\ & \psi(y,t) = \psi(y+1,t) \\
\Rightarrow\ & \psi(y,t) = \psi_0 \\
\Rightarrow\ & \gamma(x,y,t) = \gamma'(x,y), \psi_0
\end{aligned}$$

J'avoue ne pas savoir conclure. J'imagine qu'on doit pouvoir trouver une absurdité qui permet de montrer que m ne peut pas être un automate cellulaire.

J'imagine deux choses : ou on montre que la dépendance n'est pas locale, ou on montre que l'on a pas indépendance du comportement en fonction de x,y et t.

Je pense être proche, car j'ai réussi à montrer que l s'injecte entièrement dans une unique étape de m, ce qui est étrange.

Choix implémentation

De manière générale, j'ai fait le choix (mauvais ?) d'éviter les tableaux multidimensionnels, et de juste travailler avec des tableaux unidimensionnels.

J'ai aussi fait le choix que chaque processus accède aux paramètres entrés, y compris le fichier pour la question 9.

Le code est essentiellement écrit en français.

En cas de topologie spécifique, je n'ai pas utilisé `MPI_Cart_Create`. Je me permets ceci dit d'utiliser des fonctions de communications globales (comme reduce), bien qu'elle rompe la topologie.

`run-ca-1d-unbounded`

- La croissance d'une configuration est de au plus 1 à chaque étape. La taille de la zone à calculer est donc de au plus $2 * nbetapes + intial_size$. Celle-ci est majorée afin d'être répartie de manière égale sur chacun des processeur.

`life`

- Ici, je simule la topologie en calculant l'identifiant de chacun de mes quatres voisins.
- Je calcule au préalable le plus grand N tq N^2 soit inférieur au nombre de processeurs. Tous les processeurs ne sont pas utilisés.
- Si la grille de cellules n'est pas un multiple du nombre de processeurs, le dernier processeur aura juste moins de cellules à calculer.

`deplife`

- Il s'agit essentiellement d'une adaptation de life. Cependant, les communications ont été permutées, afin d'avoir les bonnes valeurs au bon endroit. Dans l'idée, on ne transmet la ligne du bas au processeur inférieur après calcul au lieu de le faire avant.

`xlife`

- J'ai malheureusement supprimé le fichier source, je dispose encore d'une version compilée que j'ai inclu. Elle fonctionne mal mais ne plante pas.
- La stratégie ici est d'avoir un nombre arbitraire de grille de taille identique sur chaque processeur. On reprend le code de life, et quand une cellule vivante touche un bord (le gauche par exemple), on demandera au voisin de gauche de créer une grille connectée à la notre. On distinguera les connections en fonction du tag (chaque grille a un tag unique). La difficulté se situe essentiellement dans la demande au voisin. Il faut en effet construire une liste de demandes, pour la transmettre d'un coup.
- l'avantage est que l'agrandissement est quasiment gratuit, et compatible avec des formes bizarres. Il suffit d'une communication. En revanche, il y a un risque d'avoir une surcharge de communication si la taille des grilles locales sont trop petites face à la taille totale (beaucoup de communication intercellule).
- Il y a donc un dimensionnement délicat à faire.

– Je n'ai pas réussi à le faire calculer correctement, le programme est donc incorrect.