Algorithmique et programmation parallèle : DM

William AUFORT et Raphaël CHARRONDIÈRE

7 Décembre 2014

1 Automate cellulaire

Note : Dans toute cette partie, on note p^2 le nombre de processeurs, N la taille de la grille (qui contient donc N^2 éléments). On supposera comme à l'habitude que p divise N.

Question 1

Pour calculer X^{t+1} à partir de X^t il faut appliquer la fonction δ à tous les éléments de X^t, soit N^2 fois. Donc, pour calculer X^t à partir de X^0, il faut répéter cette opération t fois, ce qui donne tN^2 applications de la fonction δ nécessaires.

Question 2

Nous donnons ici l'implémentation de l'automate cellulaire de la partie 3, et non pas l'implémentation de l'automate général (où les neuf voisins sont nécessaires pour la mise à jour). Nous expliquerons les différences après la description de l'algorithme.

L'idée est de disposer les processus selon une grille. processus Chaque dispose d'un bloc de $\left(\frac{N}{p}\right)^2$ éléments.

Considérons un processus P. Celui-ci dispose des données $(\widehat{x_{i,j}})$. Pour effectuer un calcul $\delta(\widehat{x_{i,j}})$, P a besoin en plus de $\widehat{x_{i-1,j}}$, $\widehat{x_{i+1,j}}$, $\widehat{x_{i,j-1}}$ et $\widehat{x_{i,j+1}}$. On peut calculer l'image des éléments de P_i qui ne sont pas à la frontière (les éléments en rouge sur la figure 1), P_i n'a besoin que de valeurs qui lui sont connues. Les éléments à la frontière nécessitent quant à eux une ou deux valeurs dont disposent les processus voisins de P_i sur la grille (les éléments en bleu sur la figure 1) pour pouvoir être mis à jour.

Chaque processus P_i va donc recevoir deux lignes (provenant de ses voisins horizontaux) et deux colonnes (provenant de ses voisins verticaux). Et donc chaque P_i doit également envoyer les lignes (respectivement les colonnes) supérieures et inférieures dont ont besoin les processus voisins horizontaux (repectivement verticaux), qui correspondent exactement aux éléments de la frontière.

Il est intéressant de remarquer que ces envois et réceptions peuvent être fait en parallèle, mais également pendant les calculs pour les éléments qui ne sont pas sur la frontière.

Cas général Dans le cas général, il y a une petite subtilité. Les quatres éléments aux "coins" doivent disposer chacun d'un élément en plus des processus voisins diagonaux. Dans le cas général, il faut donc également envoyer quatre données supplémentaires (les quatre coins) aux quatres processus qui en ont besoin.

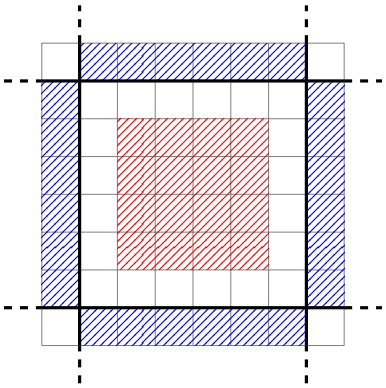

FIGURE 1 – Schéma des données d'un processeur. Les données en rouge peuvent être mises à jour par le processus, alors que les autres nécessitent les données en bleu des processus voisins.

Question 3

L'algorithme final est présenté ci-dessous :

Data: Une matrice X
Result: La matrice $Y = \delta^+(X)$
p $\longleftarrow \sqrt{NbProcs()}$;
q $\longleftarrow$ MyNum();
N $\longleftarrow$ SizeOfGrid();
for *i = 1 to* $(\frac{N}{p} - 2)$ **do**
  for *j = 1 to* $(\frac{N}{p} - 2)$ **do**
    Mettre à jour l'élément $\widehat{x_{i,j}}$ avec les données de P_q.
  end
end
// send(les deux lignes et les deux colonnes) ;
// receive(les deux lignes et les deux colonnes) ;
for *j = 0 to* $(\frac{N}{p} - 1)$ **do**
  Mettre à jour les elements $\widehat{x_{0,j}}$ et $\widehat{x_{N-1,j}}$ avec les données reçues;
end
for *i = 1 to* $(\frac{N}{p} - 2)$ **do**
  Mettre à jour les elements $\widehat{x_{i,0}}$ et $\widehat{x_{i,N-1}}$ avec les données reçues;
end

Algorithm 1: L'algorithme de la question 3

Complexité

Soient ω le temps nécessaire pour effectuer une mise à jour d'un élément $x_{i,j}$, L et b les variables vues en cours pour les temps de communications. La première partie de l'algorithme (premières mises à jour, send et receive) prend un temps $max\left((\frac{N}{p} - 2)^2\omega, 4(L + (\frac{N}{p} - 2)b)\right)$.

La seconde partie prend un temps $(\frac{2N}{p} + 2(\frac{N}{p} - 2))\omega = (\frac{4N}{p} - 4)\omega$.

D'où une complexité temporelle finale : $T(N,p) = max\left(\left(\frac{N}{p} - 2\right)^2\omega, 4\left(L + \left(\frac{(N-2)b}{p}\right)\right)\right) + \left(\frac{4N}{p} - 4\right)\omega$.

Remarque : Quand N tend vers $+\infty$, $T(N,p) \underset{N\to+\infty}{\sim} \left(\frac{N}{p} - 2\right)^2\omega \underset{N\to+\infty}{\sim} \frac{N^2\omega}{p^2}$.

Adaptation sur une grille non torique

Si on conserve avec exactitude le principe de l'algorithme, le seul problème consistera pour les processus en bordure de grille, lors des send/receive. Concrètement, si on note cette fois $P_{i,j}$ les processeurs ($(i,j) \in \{0,\dots,p-1\}$), les processus $P_{0,i}$ ne peuvent pas envoyer leur ligne au processus $P_{p-1,i}$ (même chose de $P_{i,0}$ vers $P_{i,p-1}$). Une solution pourrait être de les faire passer à travers toute une ligne (respectivement toute une colonne), ce qui nécessiterai un temps $(p-1)(L+\frac{(n-2)b}{p})$ supplémentaire pour chaque envoi, soit un temps total de $4p(p-1)(L+\frac{(n-2)b}{p})$

Et sur une topologie en anneau ?

2 Average automata

Question 4

Voir le fichier `average.c`.

Question 5

Pour prouver que δ^+ est linéaire, il suffit de montrer que pour tous $(i,j) \in \mathbb{N}$, pour tout $k \in \mathbb{R}$,

$$(\delta^+(X+Y))_{i,j} = (\delta^+(X))_{i,j} + (\delta^+(Y))_{i,j} \quad \text{et} \quad (\delta^+(kX))_{i,j} = k(\delta^+(X))_{i,j}$$

.

Sachant que $(\delta^+(X))_{i,j} = (1-p)x_{i,j} + p \quad \frac{x_{i,j+1}+x_{i,j-1}+x_{i+1,j}+x_{i-1,j}}{4}$, le résultat est immédiat.