Algorithmique et programmation parallèle : DM

William AUFORT et Raphaël CHARRONDIÈRE

7 Décembre 2014

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). On expliquera néanmoins les différences après la description de l'algorithme.

Soient p^2 le nombre de processus disponibles, N la dimension de la grille. On suppose que p divise n. L'idée est de donner à chaque processus de la grille un bloc de $\left(\frac{N}{p}\right)^2$ éléments selon la même topologie que celle de la grille de processus.

Considérons un processus P_i. Pour effectuer un calcul $\delta(x_{i,j})$, on a besoin en plus de $x_{i-1,j}$, $x_{i+1,j}$, $x_{i,j-1}$ et $x_{i,j+1}$. 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 nécessite une ou deux valeurs dont disposent les processus voisins de P_i sur la grille (les éléments en bleu sur la figure 1).

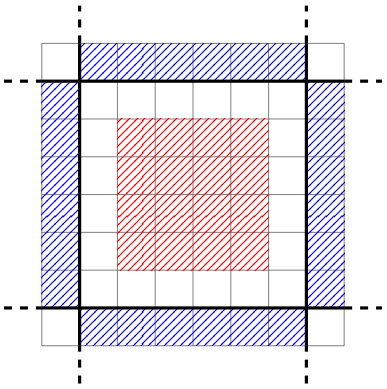

FIGURE 1 – Schéma des données d'un processeur

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 ces lignes et

ces colonnes dont ont besoin les processus voisins, 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.

Question 3

L'algorithme

L'algorithme final est présenté ci-dessous :

Data: Une matrice X
Result: La matrice $Y = \delta^+(X)$
p ⟵ NbProcs();
q ⟵ MyNum();
N ⟵ SizeOfGrid();
for *i = 1 to (n-2)* **do**
  for *j = 1 to (n-2)* **do**
    Mettre à jour l'élément $x_{i,j}$
  end
end
// send(lignes et colonnes) ;
// receive(lignes et colonnes) ;
for *j = 0 to (n-1)* **do**
  Mettre à jour les elements $x_{0,j}$ et $x_{n-1,j}$ avec les données reçues;
end
for *i = 1 to (n-2)* **do**
  Mettre à jour les elements $x_{i,0}$ et $x_{i,n-1}$ 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 $max\left((\frac{n}{\sqrt{p}}-2)^2\omega, 4(L+(\frac{n}{\sqrt{p}}-2)b)\right)$.

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

D'où une complexité temporelle finale : $max\left(\left(\frac{n}{\sqrt{p}}-2\right)^2\omega, 4(L+(\frac{(n-2)b}{\sqrt{p}})\right)+\left(\frac{4n}{\sqrt{p}}-2\right)\omega\right)$.

Adaptation sur une grille non torique

On pose $m = \sqrt{p}$. 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 $((i,j) \in \{0,\ldots,m-1\})$, les processus $P_{0,i}$ ne peuvent pas envoyer leur ligne au processus $P_{m-1,i}$ (même chose de $P_{i,0}$ vers $P_{i,m-1}$). Une solution pourrait être de les faire passer à travers tout le réseau, ce qui nécessiterai un temps :