DM d'algo parallèle

Baptiste Rozière et Alice Pellet-Mary

24 novembre 2013

Question 3

Question 5

L'automate est bien définit si on n'a pas dans DEP à la fois nord et sud, ou bien à la fois ouest et est. Si on a nord et sud ou est et ouest, on aura une boucle, chaque case devra se mettre à jour après ses voisins qui devront eux même se mettre à jour après elle. En revanche, si on n'a pas à la fois nord et sud ou à la fois est et ouest, l'automate sera bien défini car les lignes et colonnes des bords n'ont pas de voisins, donc elles pourront se mettre à jour sans attendre que d'autres cases se mettent à jour.
Si la première et dernière ligne sont voisines, on ne peut plus faire de dépendances nord ou sud. Les seuls cas où DEP sera bien défini seront $DEP = \{E\}$, $DEP = \{W\}$ ou $DEP = \emptyset$. Si on a N ou S dans DEP, par exemple N, la première ligne aura besoin de la dernière pour se mettre à jour, qui aura besoin de l'avant dernière, ..., qui aura besoin de la première. Donc l'automate n'est plus bien défini.
Dans le cas où la première et la dernière colonne sont voisines aussi, il n'y a plus aucune dépendance possible. Le seul cas où l'automate sera bien défini est le cas $DEP = \emptyset$.

Question 6

On peut trouver un algorithme efficace équivalent à l'algorithme séquentiel si les dépendances sont $DEP = \{N\}$, $DEP = \{W\}$ ou $DEP = \{N, W\}$. Dans les cas où on a S ou E dans les dépendances, on ne peut pas appliquer l'algorithme séquentiel, car comme on va de haut en bas et de gauche à droite, lorsqu'on veut mettre à jour une case, il n'y a que les cases au nord et à l'est qui ont été modifiées. On ne peut donc pas prendre en compte les modifications des cases à l'est et au sud.
Dans le cas où $DEP = \{N\}$, on a un algorithme parallèle efficace en pipelinant les calculs. On distribue la matrice entre les processeurs comme sur le dessin (un bloc de matrice par processeur).
Les blocs du haut mettent à jour la première ligne de leur bloc, et envoient le

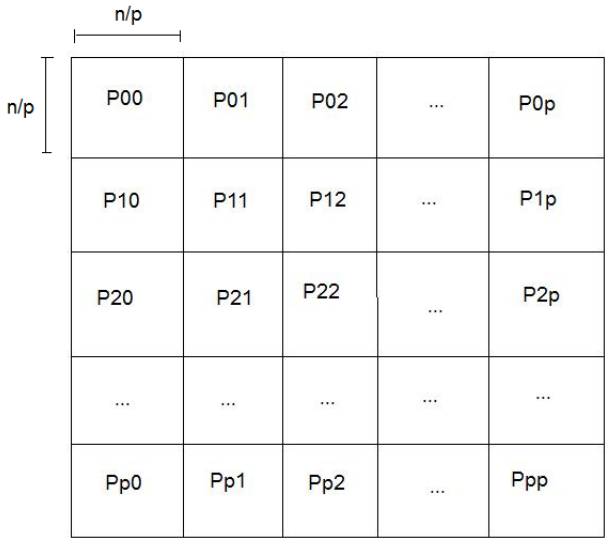

FIGURE 1 – répartition de la matrice sur les processeurs

résultat de la dernière case de leur première ligne au bloc en dessous, qui peuvent alors commencer à calculer les valeurs de leur première ligne pendant que le bloc du dessus calcule les valeurs de sa deuxième ligne et ainsi de suite (cf figure 2).

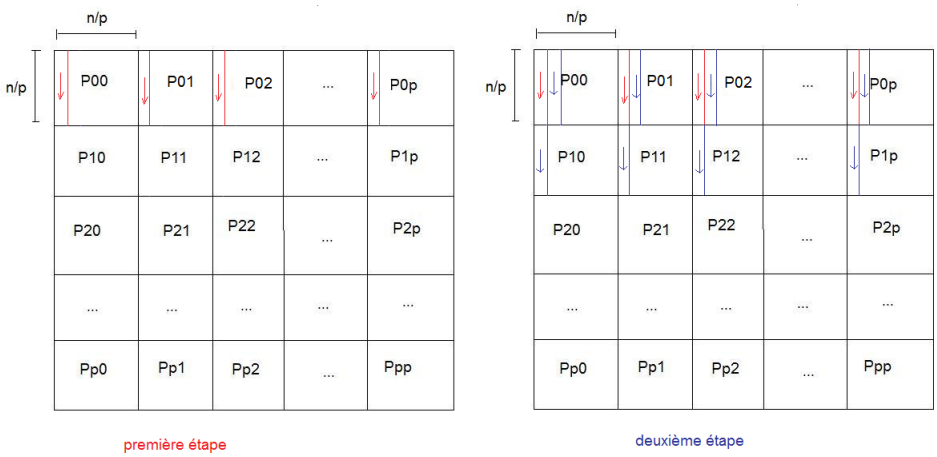

FIGURE 2 – mise à jour des éléments

Si on veut faire plusieurs mises à jours, les blocs du haut peuvent commencer la

deuxième mise à jour une fois qu'ils ont fini de mettre à jour toutes leurs cellules. Comme ça les processeurs n'ont pas de moments oisifs à part au début lorsqu'ils attendent leurs premières données.
Si $DEP = \{E\}$, l'idée est la même.
Si $DEP = \{N, W\}$, on fait les calculs diagonales par diagonales (cf figure 3). On peut optimiser le temps de calcul en modifiant l'ordre de calcul dans les blocs, pour que les blocs voisins puissent commencer à travailler le plus tôt possible.

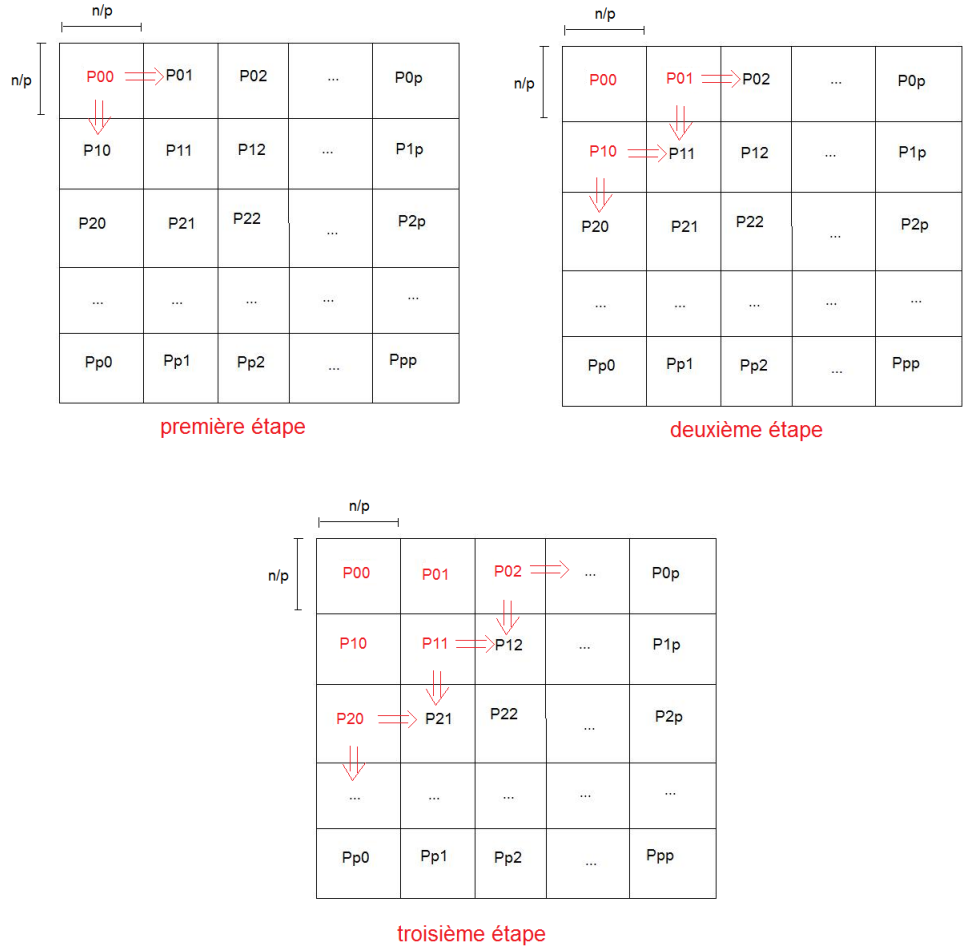

FIGURE 3 – $DEP = \{N, W\}$

Si maintenant on a une topologie d'anneau, on peut toujours trouver des algorithmes efficaces. Par exemple si $DEP = \{N\}$, on attribue à chaque processeur un bloc de colonne, et ils font tous leurs calculs en parallèle sans avoir besoin de communiques (cf figure 4). De même si $DEP = \{W\}$ on attribue des blocs de

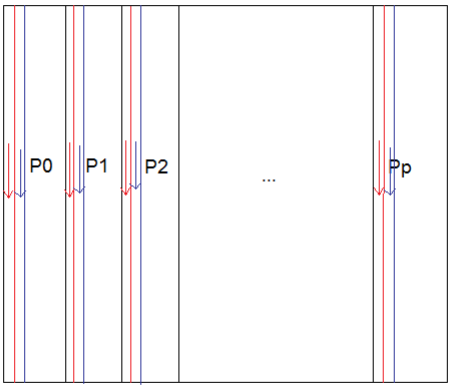

FIGURE 4 – topologie d'anneau et $DEP = \{N\}$

lignes à chaque processeur.
Si $DEP = \{N, W\}$, on attribue des blocs de colonnes à chaque processeur, le premier calcule la première ligne puis envoie son dernier élément au deuxième, qui peut alors calculer sa première ligne, pendant que le première processeur calcule sa deuxième ligne et ainsi de suite (cd figure 5).

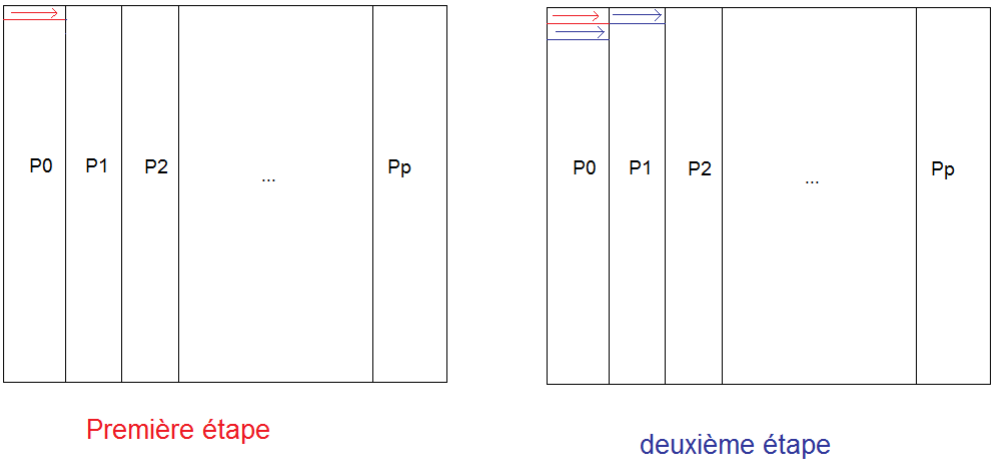

FIGURE 5 – topologie d'anneau et $DEP = \{N, W\}$

Question 8