Examen machine Calcul parallèle

Macs 3ème

3 novembre 2023

1 Automate cellulaire

Le but de ce programme est d'étudier des automates cellulaires très simples en **une** dimension.

Un automate cellulaire consiste en une grille régulière de cellule (ici en une dimension) pouvant avoir plusieurs "états" (ici deux états : allumée = 1 ou éteinte = 0) qui seront modifiés selon l'état des cellules voisines (immédiates ou non) en suivant des règles fixes à chaque itération.

On appelle **degré de voisinage** le nombre de cellule que l'on doit visiter avant d'atteindre une cellule spécifique. Ainsi, pour une cellule donnée, ses cellules adjacentes sont de degré de voisinage égal à un.

On appelle **rang** le degré maximal de voisinage des cellules prises en compte pour le calcul de l'état suivant.

Ainsi, l'automate le plus simple sera celui où on ne prendra que des cellules adjacentes (et elle-même) pour le calcul de l'état suivant (allumée ou éteinte). Il existe dont huit "motifs" différents (trois cellules donc 2^3 configurations possibles) pour lesquels il faut fixer une règle (à savoir si la cellule sera allumée ou éteinte à la prochaine itération) :

Motif	111	110	101	100	011	010	001	000

Il y aura donc dans ce cas là $2^8 = 256$ configurations (ou règles) possibles qu'on représentera par un nombre binaire variant de 0 à 255.

Par exemple, la règle numéro 30 (en binaire 30 = 00011110) sera :

Motif initial (t)	111	110	101	100	011	010	001	000
Valeur suivante cellule centrale (t+1)	0	0	0	1	1	1	1	0

En prenant les cellules adjacentes et les cellules de degré de voisinage égal à deux, on aura donc cinq cellules prises en compte dans le calcul d'une cellule (la cellule centrale dont on veut modifier la valeur, deux cellules à gauche et deux cellules à droite) ce qui nous fera $2^5 = 32$ motifs possibles pour lesquelles il faut établir une règle. Le nombre de règles possibles sera alors de $2^{32} = 4294967296$ règles possibles !

Dans notre cas, on veut étudier l'influence de ces règles en regardant l'évolution d'une grille de cellule dont au départ toutes sont éteintes sauf une cellule au centre de la grille qui sera allumée à l'initialisation.

On rajoute de plus une "condition limite" consistant à conserver *rang* rangées de cellules à gauche et à droite qui resteront toujours éteintes (et influeront sur le résultat obtenu pour l'automate).

On parcourt ensuite toutes les règles possibles pour générer pour chacune un diagramme "espace–temps" où chaque ligne i du diagramme représente un état de la grille au temps $t+i.\Delta t$.

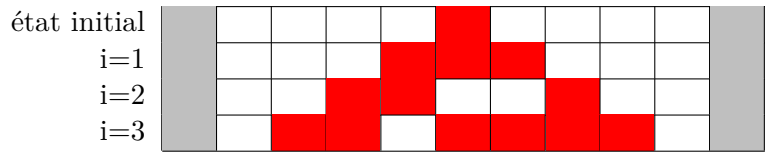

FIGURE 1 – En blanc les cellules éteintes, en rouge les cellules allumées, en gris les conditions aux limites (forcées toujours à éteinte)

Toujours avec la règle n°30, les trois premières itérations d'évolution de l'automate sont :

Une fois que les diagrammes ont été calculés, on les transforme en image png qu'on sauvegarde.

Remarque : Si vous voulez gagner du temps à l'exécution ou bien essayer avec un degré supérieur à un, vous pouvez modifier le code pour ne sauvegarder qu'une ou aucune image (en mettant la variable `SAUVE_IMAGES` à 0 au début du fichier python)…Mais les images servent aussi à voir si vous n'avez pas introduit d'erreurs. À ce propos, pour vérifier que les fichiers images sont les mêmes qu'en séquentiel pour les valeurs de paramètres par défaut :

```
diff configxxxx.png reference/configxxxx.png
```

où `xxxx` est le numéro de la règle.

Enfin, pensez à conserver trois sources bien distincts :

— Un premier fichier contenant la première parallélisation ;

— Un second fichier contenant la seconde parallélisation ;

1.1 Préliminaires

Donnez la configuration de votre ordinateur : nombre de cœurs de calul logiques et physiques.

1.2 Parallélisation des différentes configurations

Paralléliser sur les différentes règles en justifiant (dans un commentaire dans votre code) votre choix de parallélisation : distribution statique ou dynamique ? Granularité choisie ou distribution statique choisie ? …

En désactivant la sauvegarde des images après la mise au point, calculez l'accélération obtenue sur votre ordinateur.

1.3 Parallélisation par partitionnement de la grille de calcul

PAralléliser le code en distribuant la grille sur les différents processus (en utilisant les cellules fantômes).

En désactivant la sauvegarde des images après la mise au point, calculez l'accélération obtenue sur votre ordinateur.

1.4 Analyse des deux accélérations

Laquelle des deux accélérations donne le meilleurs résultat. Pouvions nous le prévoir d'avance (justifiez votre réponse !) ?