Paris Simon
Robin Gregory

Programmation concurrente et parallèle

Rapport - étape 0

L'objectif de ce projet est de simuler la diffusion de chaleur sur une plaque de matière quelconque. Pour modéliser cette plaque nous allons la scinder en plusieurs cellules de dimension égale afin de pouvoir leur attribuer une température. L'objet mathématique choisi sera la matrice. Pour une matrice de taille(NxN) nous créerons un **tableau unidimensionnel** de taille NxN et nous accèderons par exemple à la cellule (2,6) avec l'instruction suivante : tableau[2 * N + 6]. Ce tableau sera constitué d'une **structure** contenant 3 champs : une température courante et la nouvelle suite à la diffusion. Il y aura aussi un champ booléen indiquant si la cellule est constante. En effet, les cellules au centre d'indice i (cf sujet) seront initialement d'une certaine température T et la conserveront tout au long des itérations. De même, pour toutes les cellules sur l'extérieur nous fixons la température à 0. Ces cellules sont rajoutées en plus de notre matrice(NxN) demandée. Toutes les autres cellules seront initialisées à 0 avec le champ constant à 0.

Déroulement de l'algorithme : après l'initialisation détaillée plus haut exécutée grâce à un parcours de notre tableau à l'aide de 2 boucles 'for' imbriquées, nous allons expliquer le déroulement d'une **itération complète**. Cette dernière se répétera autant de fois que nécessaire. Nous répartissons d'abord la température horizontalement, puis verticalement avec la même stratégie; 2 boucles 'for' imbriquées, la première parcourt les lignes, la deuxième les colonnes. Ainsi, pour chaque cellule on stocke dans une variable sa cellule gauche et droite dans le cadre de l'itération horizontale ou alors sa cellule haut et bas pour la verticale. On prend **1/6** de la cellule gauche plus **1/6** de la cellule droite plus **2/3** de la cellule courante, ce résultat sera mis dans le champs de température 'disponible', c.a.d celui qui n'aura pas était utilisé pour le calcul. Le calcul est analogue pour l'itération verticale. Pour les cellules sur l'extérieur, il faut enlever du cal-

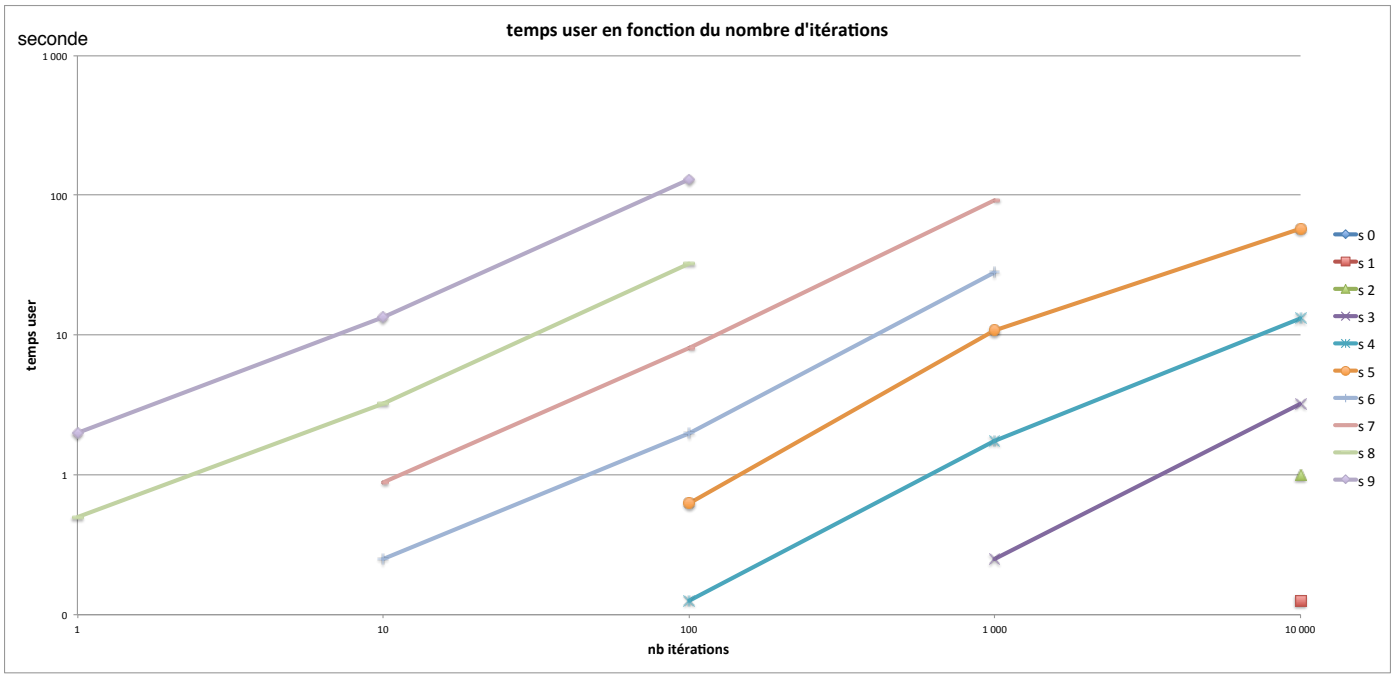
cul les cellules n'existant pas (ex: pour une cellule tout en haut, on ne lui additionnera pas 1/6 de la cellule d'en haut).

Après l'itération horizontale, nous faisons baisser les cellules centrales et extérieurs censées être constantes. C'est à la fin de l'itération verticale que nous rétablissons la température du centre et extérieur avec toujours le même schéma de boucles 'for'. A la suite de l'itération **horizontale**, **verticale** et le **rétablissement des valeurs** constantes nous avons exécuté une **itération complète**.

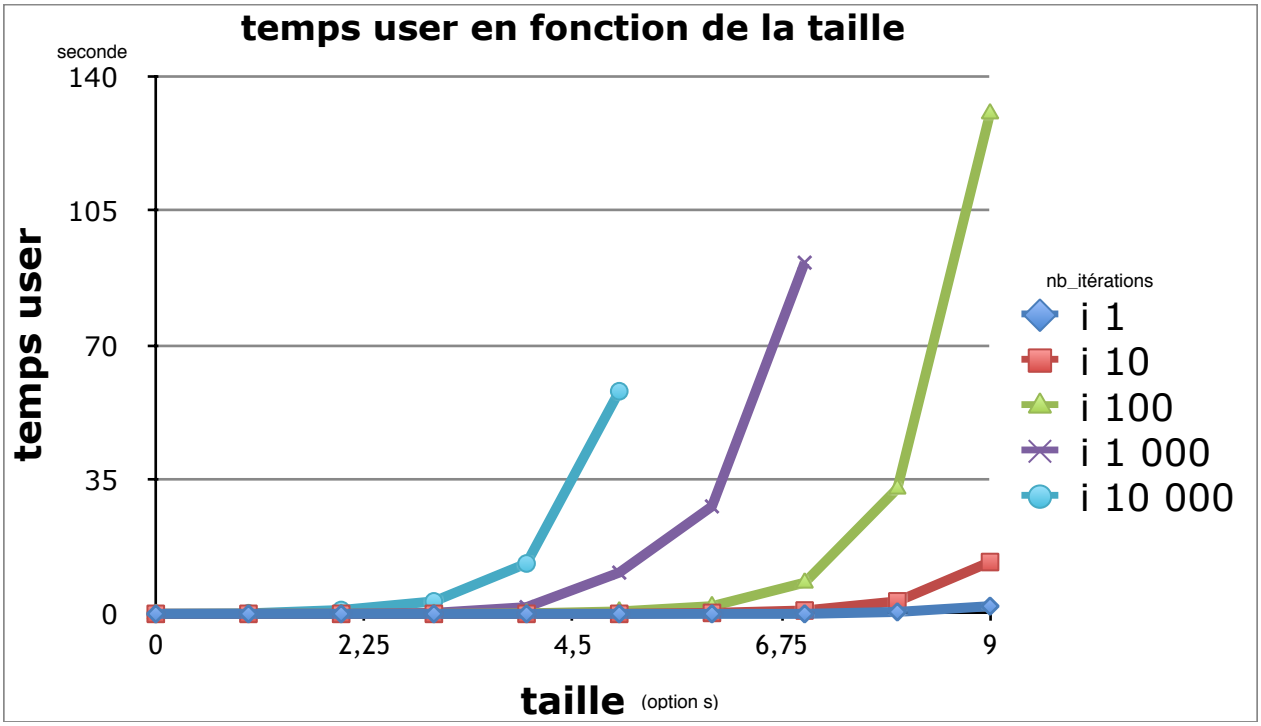
Le **résultat attendu** est une matrice **symétrique** avec les cellules centrales et extérieurs à leurs valeurs initiales. Les autres cellules auront subit les transferts de chaleur successifs et auront donc des valeurs de plus en plus faible tandis qu'elles seront éloignées du centre de la matrice.

Résultats / interprétations

Nous avons fait varier la taille (option s) de notre matrice et le nombre d'itérations. Les résultats que nous allons présenter ont été obtenus à l'aide d'un script bash et notamment de l'option M qui nous a permis d'avoir des résultats 'digne de confiance'.



Les différentes courbes de ce graphique représentent les différentes tailles de matrice possibles. Nous pouvons voir que l'augmentation du temps de calcul en fonction du nombre d'itérations est linéaire. Les courbes 0, 1 et 2 ne sont pas démonstratrices car nous avons utilisé la fonction time() qui a une précision trop faible pour retourner des valeurs exploitables pour ces trois trailles de matrice.



Sur ce graphique en revanche nous pouvons voir que l'augmentation de la taille de la matrice augmente le temps de calcul de manière exponentielle. Cela n'a rien d'anormal compte tenu du fait la taille est une puissance de 2 de l'option 's'.

s	kilobytes	octet memo/cell	cell
0	412	34 816	324
1	612	52 224	1 156
2	612	52 224	4 356
3	664	56 320	16 900
5	4 436	377 856	264 196
6	16 036	1 368 064	1 052 676
7	62 116	5 300 224	4 202 500
8	209 716	17 895 424	16 793 604
9	799 728	68 243 456	67 141 636

Enfin ce tableau où la deuxième colonne correspond à la mémoire utilisée pour le processus. La troisième correspond à cette mémoire divisée par la taille en mémoire d'une cellule de notre matrice et la dernière au nombre de cellule effectif (pour s = 0, on a (16+2)*(16+2) = 324. Le '+2' correspond aux cellules extérieures rajoutées qui seront maintenant à une température de 0). Nous constatons que la mémoire utilisée tend à correspondre à l'espace nécessaire aux stockage de toutes les cellules. Ainsi, on peut conclure que la mémoire utilisée est largement déterminé par le nombre de cellules de notre matrice.