Simulation du transfert de chaleur en 2D d'une plaque chauffante

Objectifs

Ce laboratoire vise à mettre en pratique les connaissances acquises au cours de la session en ce qui a trait à la conception d'un algorithme parallèle tout en utilisant les concepts de partitionnement, communication, agglomération et répartition de Foster.

Le problème traité consiste à simuler le transfert de chaleur en 2D d'une plaque chauffante. Dans cette simulation, on s'intéresse à la répartition de la température sur la plaque. Vous devrez implémenter le transfert de chaleur en 2D avec la librairie MPI.

Énoncé du problème

La théorie

Considérons une plaque métallique rectangulaire de longueur L et largeur l (figure 1) :

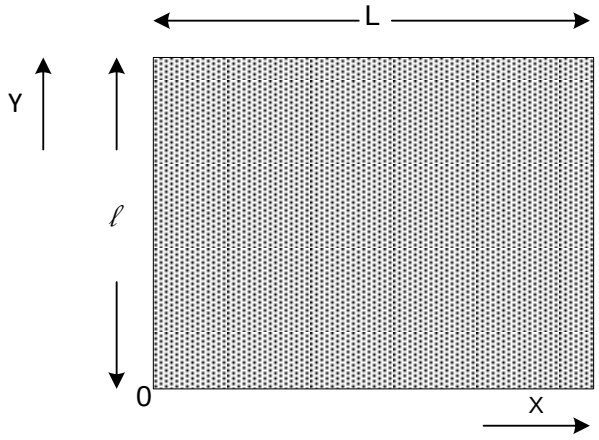


Figure 1. Plaque métallique

On prend comme origine le coin gauche en bas (le point O) de la plaque. Donc, $T(x, y, t)$ est la température du point de coordonnées (x,y) à l'instant t. On considère que la conductivité thermique C de la plaque est uniforme dans toute la plaque métallique.

L'équation de transfert de la chaleur en 2D est donnée par la formule d'Euler suivante :

$$\frac{\delta T(x,y,t)}{\delta t} = C\left(\frac{\delta^2 T(x,y,t)}{\delta x^2} + \frac{\delta^2 T(x,y,t)}{\delta y^2}\right)$$

La simulation

On partitionne la plaque en petites subdivisions (m par n) et on suppose que ces subdivisions sont assez petites pour considérer que la chaleur est uniforme dans chaque subdivision (figure 2).

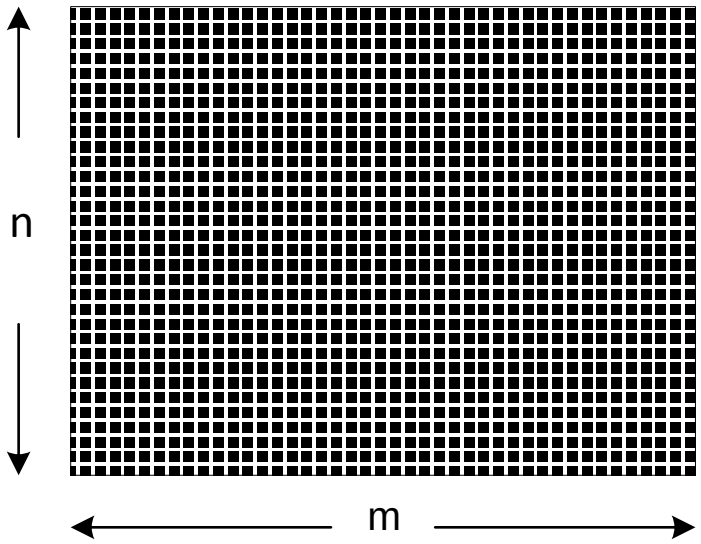


Figure 2. Décomposition de la plaque métallique

On discrétise également le temps et la simulation s'effectue toutes les t_d secondes. On note $U(i,j,k)$ la température de la subdivision (i,j) à l'instant $(k \times t_d)$:

$$U(i,j,k) = T\left(i \times \frac{L}{m}, j \times \frac{l}{n}, k \times t_d\right)$$

Pour simplifier la simulation :

- On suppose une conductivité thermique C égale à 1 partout sur la plaque.
- On subdivise la plaque également : $\frac{L}{m} = \frac{l}{n} = h$
- On suppose que h et t_d sont assez petites pour avoir une bonne approximation de la dérivée première et de la dérivée seconde.

La formule discrétisée d'Euler est la suivante :

$$U(i,j,k+1)=(1-4\ t_d/h^2)xU(i,j,k)+(t_d/h^2)x[U(i-1,j,k)+U(i+1,j,k)+U(i,j-1,k)+U(i,j+1,k)]$$

La formule d'initialisation pour la plaque au temps t = 0 :

$$U(i,j,0) = i(m-i-1) \times j(n-j-1)$$

Laboratoire

Vous devez concevoir un programme qui pourra simuler le transfert de chaleur d'une plaque chauffante en utilisant plusieurs processeurs. De plus, ce programme fera par la suite un traitement séquentiel du même problème pour enfin afficher le temps d'exécution parallèle, le temps d'exécution séquentiel et l'accélération correspondante.

Exigences

E1	Le programme est écrit en langage « C » et utilise la librairie MPI.
E2	Un fichier « Makefile » est utilisé pour compiler le programme et faire les manipulations.
E3	Le programme prend six paramètres de lancement dans cet ordre (« ./prog n m np td h nbproc») : • n = le nombre de lignes • m = le nombre de colonnes • np = le nombre de pas de temps • td = le temps discrétisé • h = la taille d'un côté d'une subdivision • nbproc = le nombre de processus à utiliser

E4	Pour simuler un calcul intensif, avant chaque ligne d'un calcul d'élément vous devez ajouter un temps d'attente artificiel de l'ordre de 5 microsecondes:
	`usleep(TEMPS_ATTENTE);`
	`matrix[i][j] = ...`

E5	Le programme affiche la matrice initiale et finale pour chaque traitement.
E6	Le programme donne des résultats de simulation corrects.
E7	Les résultats doivent être imprimés de sorte qu'ils reflètent les températures de la plaque dans le même ordre que la plaque. L'affichage aura la forme suivante pour t = 0 (m = 10, n = 5) : 0,0 0,0 0,0 0,0 0,0 0,0 0,0 0,0 0,0 0,0 0,0 24,0 42,0 54,0 60,0 60,0 54,0 42,0 24,0 0,0 0,0 32,0 56,0 72,0 80,0 80,0 72,0 56,0 32,0 0,0 0,0 24,0 42,0 54,0 60,0 60,0 54,0 42,0 24,0 0,0 0,0 0,0 0,0 0,0 0,0 0,0 0,0 0,0 0,0 0,0
E8	Le programme affiche les temps d'exécutions parallèle et séquentiel ainsi que l'accélération.
E9	Le programme doit utiliser une fonction d'initialisation qui initialise les températures de la plaque au démarrage de sorte que le centre de la plaque soit plus chaud que toute autre subdivision et plus on s'éloigne du centre, plus la température diminue jusqu'à s'annuler aux frontières.
E10	La température aux frontières est gardée nulle pendant toute la simulation.

Conception parallèle

Le processus patron initialise et agglomère la plaque métallique de sorte à équilibrer la charge du travail sur les processus disponibles. Ensuite, il distribue cette décomposition aux processus employés. Dans chaque pas de temps, les employés doivent échanger les données frontières avec leurs voisins, car la température courante d'un point dépend aussi bien de son ancienne valeur que de celles de ses voisines. Une fois que tous les processus ont terminé le calcul, ils renvoient le résultat de leurs portions de calcul au processus patron.

Voir les diagrammes présentés à la page suivante.

Mesures

Pour évaluer les performances du programme (temps d'exécution) en parallèle et séquentiel, vous pouvez utiliser l'exemple de code suivant :

```
// Début de l'exemple
#include <sys/time.h>
double timeStart, timeEnd, Texec;
struct timeval tp;
gettimeofday (&tp, NULL); // Début du chronomètre
timeStart = (double) (tp.tv_sec) + (double) (tp.tv_usec) / 1e6;
// Insérer votre code ici
gettimeofday (&tp, NULL); // Fin du chronomètre
timeEnd = (double) (tp.tv_sec) + (double) (tp.tv_usec) / 1e6;
Texec = timeEnd - timeStart; //Temps d'exécution en secondes
// Fin de l'exemple
```

Performance

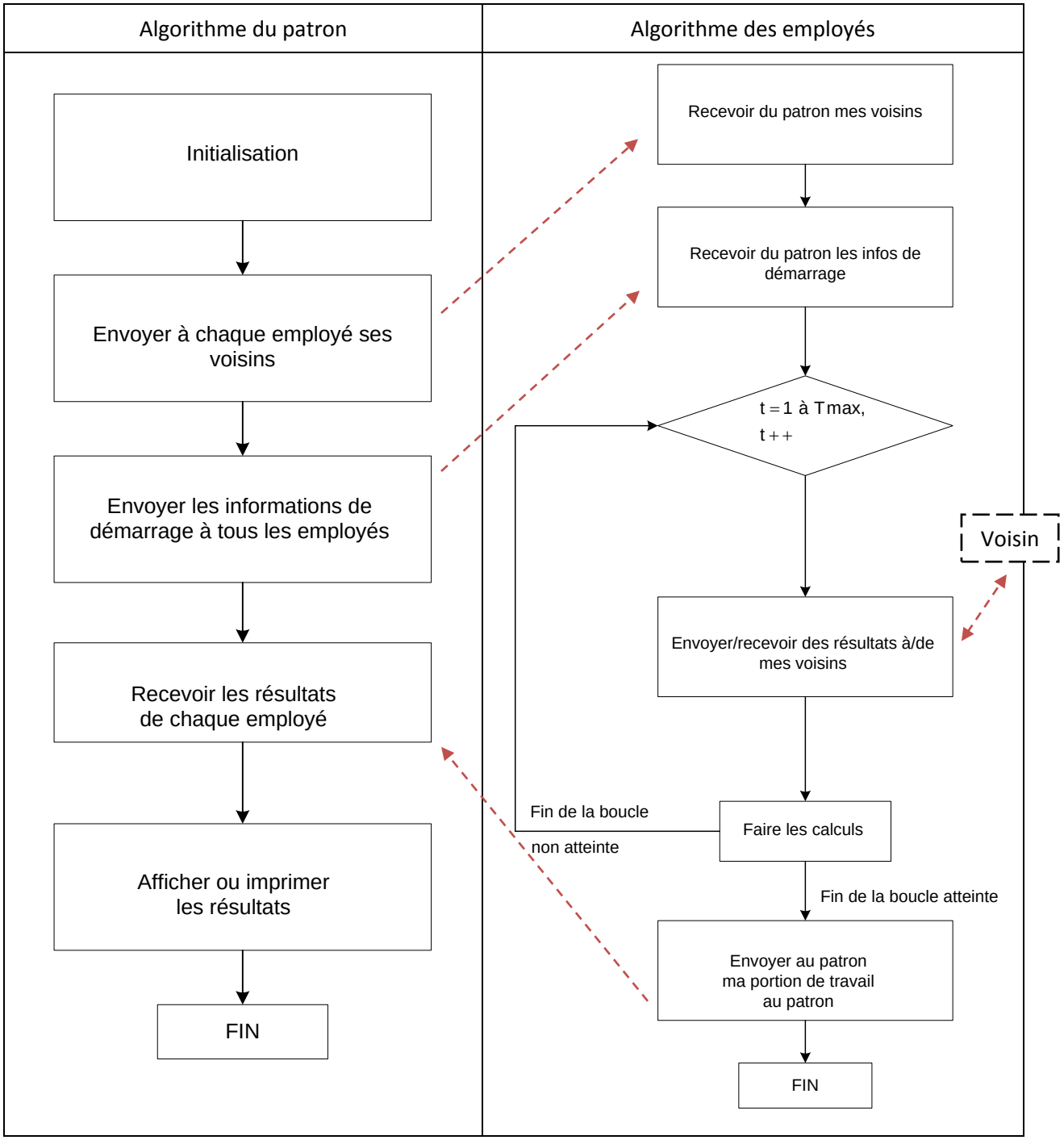
La performance de vos algorithmes sera mesurée selon 4 tests distincts :

- L'utilisation de 48 à 64 processeurs pour une grande matrice relativement carrée.
- L'utilisation de 4 à 16 processeurs pour une petite matrice relativement carrée.
- Le traitement d'une matrice horizontale (plus de 500 colonnes pour moins de 100 lignes).
- Le traitement d'une matrice verticale (plus de 500 lignes pour moins de 100 colonnes).

Chacun de ces tests est noté sur 5% pour un total de 20%. L'équipe dont l'algorithme sera le plus rapide dans un test obtiendra l'ensemble des points pour ce test et celle dont l'algorithme sera le plus lent n'obtiendra aucun point. Les autres équipes seront réparties entre 0% et 5% en fonction de leur classement. Les tests seront exécutés 3 fois et le meilleur temps sera utilisé pour comparaison afin d'éviter les variations incontrôlables du matériel et du système d'exploitation.

Algorithmes

Pour coder l'application, on peut s'inspirer des diagrammes présentés ci-dessous. Ces algorithmes servent comme un guide, on peut les modifier en justifiant la raison des modifications.



Manipulations

On fait la simulation du transfert de chaleur d'une plaque de 1,5 m de largeur par 1 m de longueur.
ATTENTION : À l'exception de la manipulation #4, lorsqu'on demande de modifier m, n ou h, assurez-vous que ces grandeurs sont toujours respectées, faites les ajustements nécessaires.

1) Exécuter le programme avec un h = 0,1, un td = 0,0002 et 100 pas de temps. Rapporter les résultats.

2) Répéter l'exécution du programme plusieurs fois. Est-ce qu'on obtient les mêmes résultats (temps d'exécution et les températures)? Que peut-on conclure?

3) Tracer la courbe du temps d'exécution en fonction de la taille du problème (en faisant varier d'abord le nombre de subdivisions et ensuite le nombre de pas de temps. 1 graphique pour chacun, donc 2 graphiques). Interpréter vos résultats.

4) Faire une régression afin de trouver la formule mathématique permettant d'expliquer le comportement du temps d'exécution en fonction de la taille du problème (d'abord pour m, ensuite pour n et finalement pour np. 1 graphique pour chacun, donc 3 graphiques). Vous devez ignorer les contraintes de taille de plaque pour cet exercice. Que peut-on dire à propos de la « scalability » de votre algorithme selon ces trois facteurs de taille du problème?

5) Fixer les tailles du problème à m = 200, n = 300 et le nombre de pas = 100. Faites varier le nombre de processus P. Tracer l'accélération en fonction de P (1 graphique). Interpréter.

6) Fixer le nombre de processus à 8, le nombre de pas de temps à 100 et faites varier la taille du problème (le nombre de subdivisions). On suggère de faire varier h de 0,15 à 0,00125. Tracer l'accélération en fonction de h (1 graphique). Interpréter.

7) Rapporter vos conclusions.

Remise du laboratoire

Date de remise

- Jeudi 23 mars 2017 avant 23h59 (4 périodes)

Livrables

- Le code source du programme sous format électronique
- Un rapport en format **PDF**
- Compresser tous les fichiers pour n'avoir qu'un seul fichier d'archive

Barème de correction

- Code (50%)
  - Faciliter d'exécution
  - Implémentation d'un programme solide et flexible
  - En-tête, commentaires, explications, noms de variables
  - Performance (20%)
- Rapport (50%)
  - Introduction (5%)
  - Analyse (8%)
  - Conception (15%)
  - Résultats des manipulations et discussion (20%)
  - Conclusion (2%)
- Retard : (-5% par heure)
- Français : (Jusqu'à -10%)
- Rapport de 15 pages de contenu maximum (-1%/page)

Procédure de remise

Envoyer votre fichier compressé à stonkie@gmail.com.

- S.V.P., mettez comme sujet du courriel « **LOG645-Lab3-EquipeX** »
- et le fichier compressé « **LOG645-Lab3-EquipeX.zip** » doit avoir la structure suivante :
  - LOG645-Lab3-EquipeX/
    - Rapport.pdf
    - code/
      - Makefile
      - README
      - run.sh
      - fichier(s).c