LABORATOIRE NO. 1

LOG640 Introduction au traitement parallèle

Date : 2004-10-25

Auteur : Mohammed ELKANOUNI

I. OBJECTIFS

Ce travail vise à se familiariser avec la conception d'un algorithme parallèle tout en en utilisant les concepts de partitionnement, communication, agglomération et répartition de Ian FOSTER.

Le problème traité consiste à simuler le transfert de chaleur en 2D. Dans cette simulation, on ne s'intéresse qu'à la répartition de la température sur la plaque à la fin de la simulation.

II. ÉNONCÉ DU PROBLÈME

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

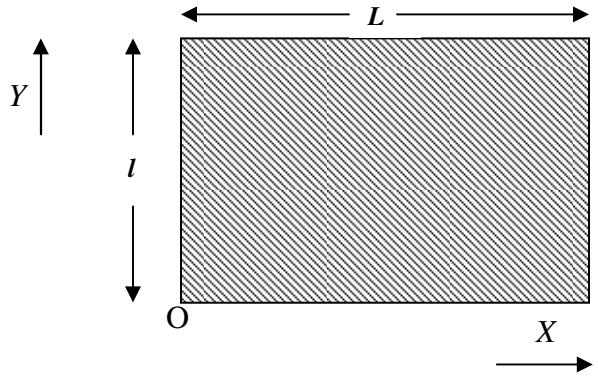

Figure 1

On prend comme origine du repère de la plaque le coin gauche en bas (le point O)

Soit *T(x,y,t)* la température du point de coordonnées *(x,y)* à l'instant *t*. On considère que la conductivité 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 :

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

Les conditions aux limites sont fixées aux frontières de la plaque *(x= 0, x=L, y=0, y=l)* :

- *T(0,y,t)=f1(y)*
- *T(L,y,t)=f2(y)*
- *T(x,0,t)=f3(x)*
- *T(x,l,t)=f4(x)*

Les conditions initiales à l'instant *t=0*

- *T(x,y,0)=f(x,y)*

On partitionne la plaque en petites subdivisions *(m×n)* et on suppose que ces subdivisions sont assez petites pour considérer que la chaleur est uniforme dans chaque subdivision (figure 2). 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×td* (*i* varie de *0* à *m-1* et *j* varie de *0* à *n-1*) :

$$U(i,j,k)=T(i\times L/m,\ j\times l/n,\ k\times t_d)$$

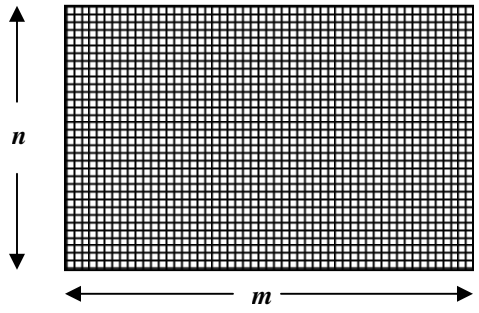

Figure 2

III. TP

Pour simplifier, on prend *L/m=l/n=h* et C=1 et on suppose que *h* et t_d sont assez petites pour avoir une approximation de la dérivée et la dérivée seconde. La formule discrétisée d'Euler est la suivante :

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

IV. Laboratoire

Dans cette partie vous supposez vous avez un seul processeur, vous allez programmer la simulation de l'équation de la chaleur en séquentiel sans tenir compte du parallélisme, les questions de 1 à 4 vont vous aider dans les prochains laboratoires.

1) Écrire le pseudo code de l'équation de la chaleur.

2) Écrire le code séquentiel en C de l'équation de la chaleur, pour cela suivre les consignes suivantes :

- Le programme doit prendre des paramètres lors de son exécution, ces paramètres sont successivement n, m, nombre de pas, td, h, l'instruction d'exécution est la suivante :

  progExe n m nPas td h a

  a = 1 : afficher les résultats initiaux et finaux de la température

  a = 0 : ne pas afficher les résultats initiaux et finaux de la température

- Le programme doit afficher les résultats initiaux et finaux de la température à l'écran (si le paramètre a = 1) ainsi que le temps d'exécution correspondant. 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 suivantes pour t=0:

  Exemple pour *m=10* et *n=5*

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

- 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 tout autre subdivision et plus on s'éloigne du centre, plus la température diminue jusqu'à s'annuler aux frontières :

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

- La température aux frontières est gardée nulle pendant toute la simulation.

  Cette fonction ne sera appelée que pour sauvegarder les températures de plaque au départ et à la fin de la simulation.

§ Insérer un code pour évaluer les performances du programme (temps d'exécution), pour cela utiliser l'exemple de code suivant :

```
// Début d'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 d'exemple
```

Attention, il faut bien placer le code de calcul de temps d'exécution aux bons endroits, sinon il va donner de mauvais résultats.

3) Compiler et exécuter le programme, pour cela on peut utiliser un fichier *makefile* (voir guide d'utilisation dans la page du cours)

4) Tester le programme, exécuter le programme pour $m=n=12$, $t_d= 0.0002s$, nombre de pas de temps = 200, $L=l=1;$

Rapporter les résultats des deux affichages dans le rapport de laboratoire ainsi que le temps d'exécution.

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 ?

5) Tracer la courbe du temps d'exécution en fonction de la taille du problème (pour n=m et pour le pas de temps). Interpréter vos résultats.

6) Faire une interpolation du temps d'exécution en fonction de la taille du problème (pour n=m et pour le pas de temps). Que peut-on dire ?

7) Rapporter vos conclusions

Important :

- Le laboratoire doit se faire dans un environnement POSIX de préférence Unix
- Il faut bien commenter vos codes source
- Il faut respecter les spécifications de l'énoncé du laboratoire
- Le rapport en format papier est exigé, les codes sources et le fichier du rapport doivent être soumis avec l'outil de soumission.
- Respecter les échéances pour ne pas avoir de pénalités de retard (voir le calendrier)