École de technologie supérieure
Département de génie logiciel et des TI

Laboratoire sur OpenMP

N° du laboratoire	1
N° d'équipe	
Étudiants	Samy Lemcelli Nicolas Habak
Codes permanents	LEMS26019103 HABN08118904
Cours	LOG645
Session	Hiver 2015
Groupe	01
Chargé de cours	Carlos Vazquez
Chargé de laboratoire	Kevin Lachance-Coulombe
Date	25 Mars 2015

Introduction

Le but de ce laboratoire est de simuler une plaque à induction en fonction du temps. Étant donné que cette problématique est réaliste, on aura une bonne idée de l'impact du parallèlisme.

Analyse

Afin de calculer le transfère de chaleur, on utilise l'équation d'Euler suivante:

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

Afin de pouvoir implémenter le calcul, nous utilisons la formule discrétisée:

$U(i,j,k+1 = (1 - 4\text{-})$

U(i,j,k+1)=(1-4 td/h2)xU(i,j,k)+(td/h2)x[U(i-1,j,k)+U(i+1,j,k)+U(i,j-1,k)+U(i,j+1,k)].

De plus, nous initialisons la plaque en utilisant: U(i, j, 0)=i(m-i-1)xj(n-j-1). Avec ces formules, nous pouvons voir les dépendances entre les cellules, d'où vient l'importance de bien partitionner et assigner les cellules afin d'avoir une synchronisation avec le moins de barrières possibles.

Conception de l'algorithme

Lors de la conception de l'algorithme, nous avons fait une série de décisions afin de rendre notre solution finale la plus performante possible.

En utilisant les appels *MPI_Get* et *MPI_Put* en combinaison avec nos *MPI_Win*, il est alors possible de faire ces appels au extremités de chaque alteration du thread. Ceci nous permet de chercher les nouvelles données au debut d'une alteration et de pousser les résultats de nos calculs à la fin d'une alteration.

Afin que notre solution soit efficace et que l'on ne manque aucun calculs, il est important de synchroniser nos processus. La méthodologie choisie choisie est l'utilisation de barrières. Ces barrières sont appelés à un moment clef. La copie des données de la matrice vers l'ancienne matrice fait par le processus zero doit être isolée.

Une grande partie de l'efficacité de notre algorithme est due à la division des pixels en fonction des processus. Il y a une grosse section dans notre initialisation qui permet de délimiter les pixels par processus. Les comparaisons qui ne sont pas nécessaires sont évitées. Les bordures en font un bon exemple.

Par la suite, notre matrice n’est rien d’autre qu'un tableau à une dimension. Ceci nous permet de distribuer facilement nos données de manière séquentielle. On évite ainsi de ralentir l'exécution avec un nombre excessif d’accès en mémoire.

Finalement, il est important d’éviter les calculs inutiles dans notre algorithme. Lorsqu’il est possible de précaluler certaines choses, nous avons choisi de les faire lors de l’initialisation.

Temps et Resultats

En executant une matrice 10x15 avec une alteration de 100, un temps discretisé de 0.002 et une subdivision de 0.1 on obtient les resultats suivants :

Initialisation:

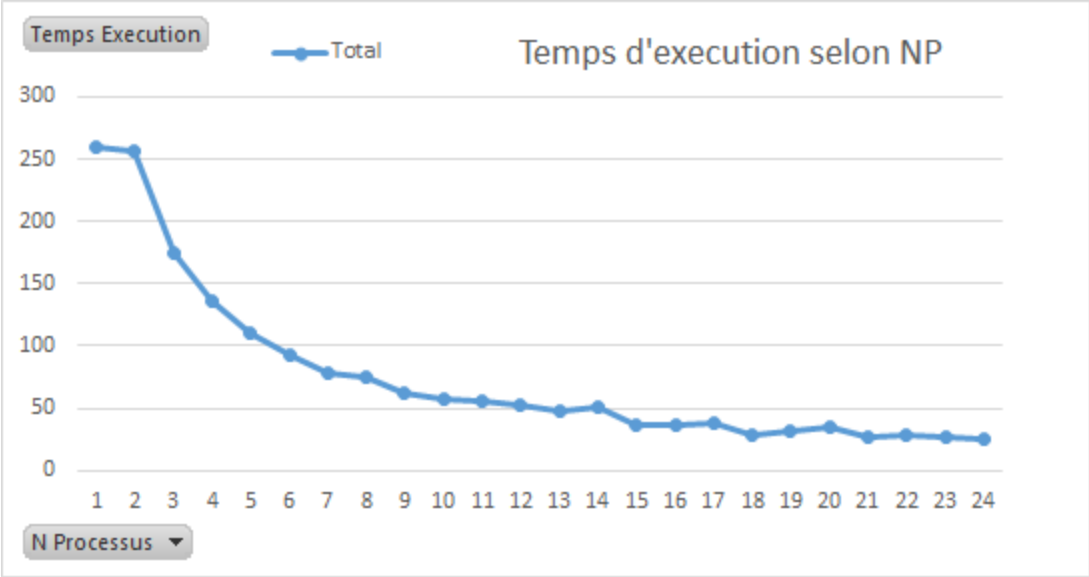
```
|0.00   0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    |
|0.00   104.00  192.00  264.00  320.00  360.00  384.00  392.00  384.00  360.00  320.00  264.00  192.00  104.00  0.00    |
|0.00   182.00  336.00  462.00  560.00  630.00  672.00  686.00  672.00  630.00  560.00  462.00  336.00  182.00  0.00    |
|0.00   234.00  432.00  594.00  720.00  810.00  864.00  882.00  864.00  810.00  720.00  594.00  432.00  234.00  0.00    |
|0.00   260.00  480.00  660.00  800.00  900.00  960.00  980.00  960.00  900.00  800.00  660.00  480.00  260.00  0.00    |
|0.00   260.00  480.00  660.00  800.00  900.00  960.00  980.00  960.00  900.00  800.00  660.00  480.00  260.00  0.00    |
|0.00   234.00  432.00  594.00  720.00  810.00  864.00  882.00  864.00  810.00  720.00  594.00  432.00  234.00  0.00    |
|0.00   182.00  336.00  462.00  560.00  630.00  672.00  686.00  672.00  630.00  560.00  462.00  336.00  182.00  0.00    |
|0.00   104.00  192.00  264.00  320.00  360.00  384.00  392.00  384.00  360.00  320.00  264.00  192.00  104.00  0.00    |
|0.00   0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    |
```

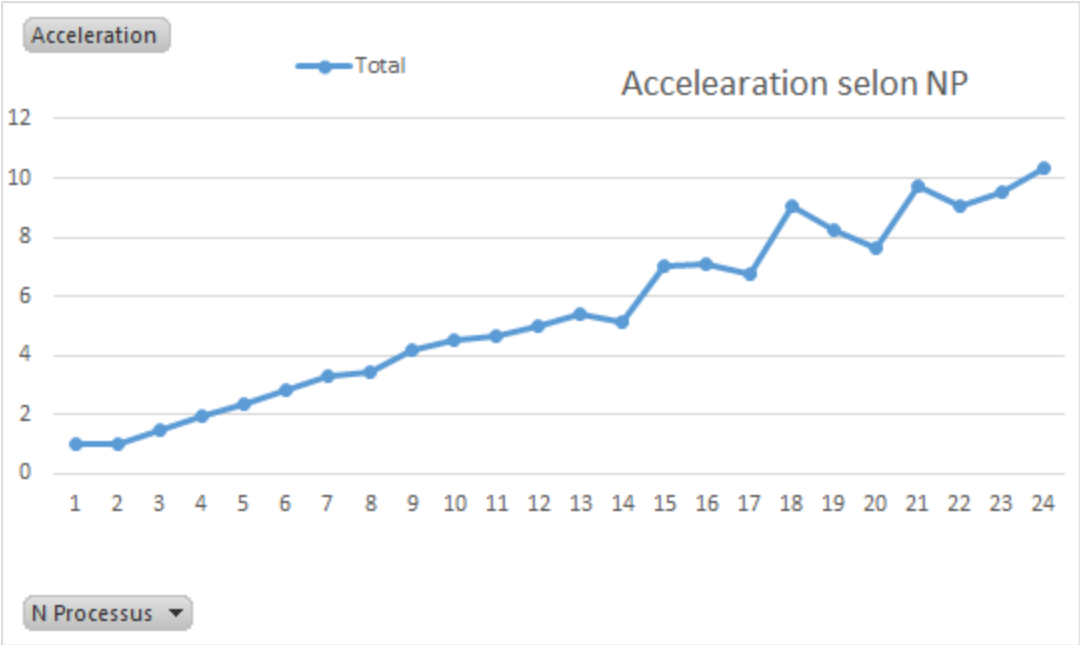
Final:

```
|0.00   0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    |
|0.00   61.02   117.67  166.55  205.62  233.87  250.92  256.61  250.92  233.87  205.62  166.55  117.67  61.02   0.00    |
|0.00   113.83  219.48  310.66  383.53  436.24  468.02  478.64  468.02  436.24  383.53  310.66  219.48  113.83  0.00    |
|0.00   152.08  293.24  415.06  512.43  582.85  625.32  639.50  625.32  582.85  512.43  415.06  293.24  152.08  0.00    |
|0.00   172.02  331.69  469.47  579.60  659.25  707.29  723.33  707.29  659.25  579.60  469.47  331.69  172.02  0.00    |
|0.00   172.02  331.69  469.47  579.60  659.25  707.29  723.33  707.29  659.25  579.60  469.47  331.69  172.02  0.00    |
|0.00   152.08  293.24  415.06  512.43  582.85  625.32  639.50  625.32  582.85  512.43  415.06  293.24  152.08  0.00    |
|0.00   113.83  219.48  310.66  383.53  436.24  468.02  478.64  468.02  436.24  383.53  310.66  219.48  113.83  0.00    |
|0.00   61.02   117.67  166.55  205.62  233.87  250.92  256.61  250.92  233.87  205.62  166.55  117.67  61.02   0.00    |
|0.00   0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    |
```

Les resultats sont constantes et les temps d’execution est similaire. Ce qui indique que notre programme ne possède aucune lacune de synchronisation et de perte de données.

En utilisant une matrice de 10x15, une alteration de 50, un temps discretisé de 0.0002 et une subdivision de 0.1 :





En analysant nos deux graphes, on aperçoit qu'ils sont assez representatif du scaling de notre algorithme. Il est alors bénéfique de rouler l'application sur le plus de processeurs possible afin d'en obtenir tous les gains.

On peut maintenant analyser l'impact de la variation de H sur l'acceleration

Pas de H	Temps	Acceleration
0.15	140.51	1
0.01	140.01	1

0.001	139.95	1

On peut alors conclure que le changement de la valeur de H n'importe peu sur le temps d'execution.

Discussion

L'utilisation du parallélisme confère un net avantage lors des calculs de matrices de taille importante. De plus, les effets des couts de communications sont assez minimes. Ceci est démontré lors de l'acceleration obtenu en bout de ligne.

Conclusion

Grâce à ce laboratoire, nous avons encore démontré l'importance du parallélisme lorsqu'on effectue des calculs sur les matrices. Il est aussi important de bien partitionner la matrice afin d'avoir le moins de dépendances possibles sans, pour autant, compromettre la performance avec un nombre élevé de communications entre les partitions.