Arnaud GARNIER
Lisa JOANNO

SI4 - 2015/2016 - Info2

Projet de programmation concurrente

Simulateur de transfert de chaleur

Rendu 1 - 21/02/2016

I - Introduction

Le but de ce projet est de simuler la diffusion de la chaleur sur une plaque carrée. On implémente en premier une version itérative de l'algorithme, puis, lors des étapes précédentes, on rendra cet algorithme plus performant en le programmant de manière concurrente.
Le programme qui est implémenté prend en options plusieurs variables qui changent les données en entrées de l'algorithme. On peut ainsi lancer le programme sur plusieurs tailles de matrices, le nombre de threads à créer, les étapes à réaliser et le nombre d'itérations.

II - Hypothèses

Lors du rendu 1, le programme est itératif. Il ne fait donc pas intervenir le nombre de threads, ni le numéro de l'étape (c'est la première). On fait l'hypothèse que la chaleur se diffuse selon l'axe horizontal, puis selon l'axe vertical, cette hypothèse est nécessaire car l'algorithme itératif ne permet pas de propager en même temps la chaleur selon les deux axes, il faut donc faire un choix.

III - Description de l'algorithme

Afin de simuler le phénomène de diffusion de la chaleur sur une surface, cette dernière est modélisée à l'aide d'une matrice. L'algorithme itératif parcourt une à une les cases de la matrice et applique le traitement suivant sur cette dernière :

- Dans un premier temps, on suppose que la diffusion ne s'effectue que selon l'axe des abscisses. La formule à appliquer pour chaque case est la suivante :

$$val_{case_n}[i] = \frac{1}{6} * val_{case_{n-1}}[i-1] + \frac{4}{6} * val_{case_{n-1}}[i] + \frac{1}{6} * val_{case_{n-1}}[i+1]$$

  n étant l'étape actuelle et n-1 l'étape précédente.
  Il est donc nécessaire de conserver une copie de la matrice à l'étape n.

- Dans un second temps, la diffusion s'effectue selon l'axe des ordonnées uniquement. On applique la formule donnée précédemment mais cette fois ci selon les ordonnées (on considère pour chaque case la case située au dessus d'elle et celle située en dessous).

L'algorithme s'effectue donc en deux temps. On répétera ces opérations pour chaque nombre d'itérations donné par l'utilisateur (au moyen de l'option -i).
La formulation de l'algorithme est la suivante :

Algorithme de diffusion de la chaleur :

Entrées :
M_t et M_{t-1} deux matrices égales
taille : taille des matrices
nbIterations : nombre d'itérations

Pour k *allant de* 0 *à* nbIterations *faire*
| *Pour* i *allant de* 0 *à* taille *faire*
| | *Pour* j *allant de* 0 *à* taille *faire*
| | | $M_t[i][j]=(\frac{1}{H}\times M_{t-1}[i][j-1])+(\frac{4}{H}*M_{t-1}[i][j])+(\frac{1}{H}*M_{t-1}[i][j+1])$
| Copier(M_t, M_{t-1})
|
| *Pour* i *allant de* 0 *à* taille *faire*
| | Pour j *allant de* 0 *à* taille *faire*
| | | *Si* i = 0 *faire*
| | | | $M_t[i][j]=(\frac{4}{H}\times M_{t-1}[i][j])+(\frac{1}{H}*M_{t-1}[i+1][j])$
| | | *Ou si* i = 0 taille-1 *faire*
| | | | $M_t[i][j]=(\frac{1}{H}\times M_{t-1}[i-1][j])+(\frac{4}{H}*M_{t-1}[i][j])$
| | | *Sinon faire*
| | | | $M_t[i][j]=(\frac{1}{H}\times M_{t-1}[i-1][j])+(\frac{4}{H}*M_{t-1}[i][j])+(\frac{1}{H}*M_{t-1}[i+1][j])$
| Copier(M_t, M_{t-1})

Avec l'algorithme Copier(M1,M2), qui copie M1 dans M2 :

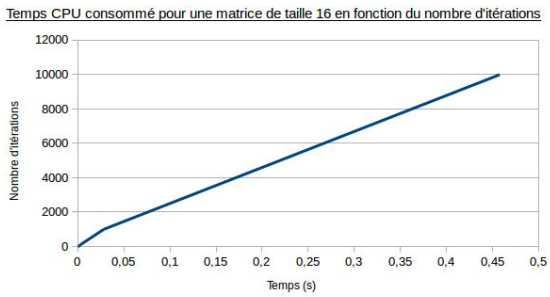
Entrées :
M_1 et M_2 deux matrices
taille : taille des matrices

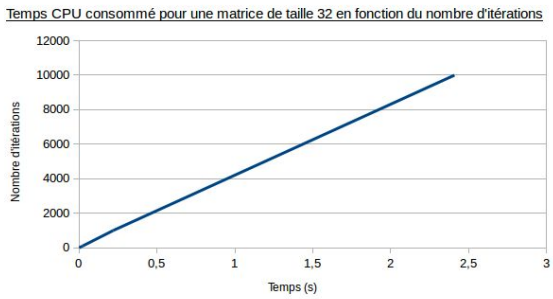
Pour i *allant de* 0 *à* taille *faire*
| *Pour* j *allant de* 0 *à* taille *faire*
| | $M_2[i][j]=M_1[i][j]$

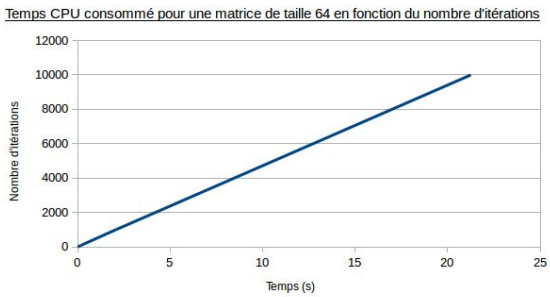
IV - Synthèse des mesures

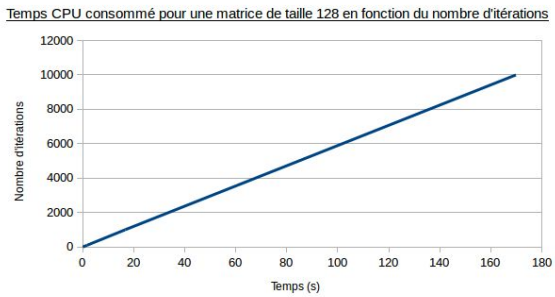
Après plusieurs exécutions et variations sur le nombre d'itérations, on obtient les résultats suivants :

Pour le temps CPU consommé (option -m) :

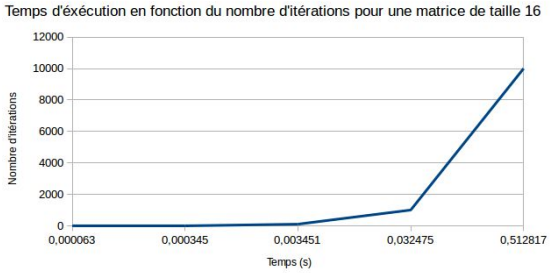


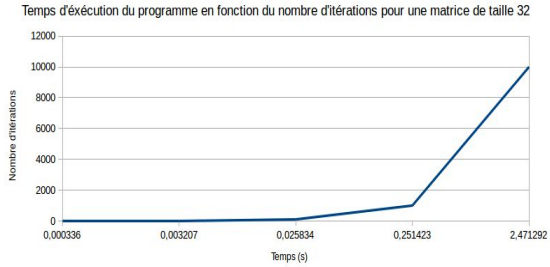


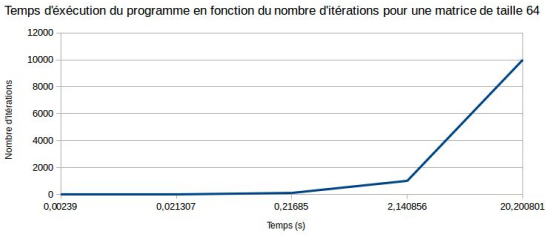


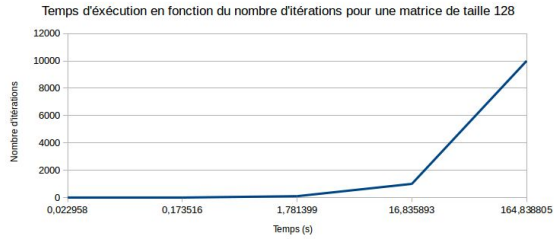


Pour le temps de réponse utilisateur (option -M) :









V - Analyse des résultats

Temps CPU consommé :

Le temps CPU consommé suit une fonction linéaire. Il est proportionnel au nombre d'itérations. Plus la matrice est de taille importante et plus ce temps est conséquent : pour 1000 itérations, le temps CPU consommé pour une matrice de taille 16 est d'environ 0,02 secondes, alors que pour une matrice de taille 128 elle est d'à peu près 17 secondes .

Temps de réponse utilisateur :

On voit que le temps est linéaire, pour un nombre d'itérations doublé, le temps de réponse utilisateur double aussi. Ce temps est très long (164,83 secondes pour 10000 itérations sur une matrice de taille 128).

On peut remarquer que cet algorithme n'est pas très performant, pour des matrices de taille supérieure à 64, avec un nombre d'itérations élevé (supérieur à 1000). Implémenter cet algorithme en diffusant en même temps la chaleur sur l'axe horizontal et l'axe vertical pourrait être une solution pour le rendre plus performant.