Optimisation d'un code de résolution de l'équation de la chaleur en 2D

Guillaume FUHR

1er février 2019

L'équation de la chaleur avec convection, en 2 dimensions peut se mettre sous la forme :

$$\partial_t u(x,y,t) = D\left[\partial_x^2 + \partial_y^2\right] u(x,y,t) - V\left[\partial_x + \partial_y\right] u(x,y,t) \tag{0.1}$$

En rajoutant la condition aux bords suivante :

$$u(x,0,t) = 0 \tag{0.2}$$
$$u(x,L_x,t) = 0 \tag{0.3}$$
$$u(0,x,t) = 0 \tag{0.4}$$
$$u(0,L_y,t) = 0 \tag{0.5}$$
$$\tag{0.6}$$

Cette équation peut être résolue numériquement par la méthode des différences finies. Si nous considérons un maillage régulier de longueurs L_x et L_y avec les pas de grilles :

$$\Delta x = L_x/N_x, \quad \Delta y = L_y/N_y$$

Nous pouvons ainsi discrétiser l'opérateur de diffusion sous la forme :

$$u_{i,j}^n = u(x_i, y_j, t^n) \tag{0.7}$$
$$D\left[\partial_x^2 + \partial_y^2\right] u(x,y,t) = \frac{u_{i+1,j}^n + u_{i-1,j}^n - 2u_{i,j}^n}{\Delta x^2} + \frac{u_{i,j+1}^n + u_{i,j-1}^n - 2u_{i,j}^n}{\Delta y^2} \tag{0.8}$$
$$\partial_x u(x,y,t) = \frac{u_{i+1,j}^n - u_{i-1,j}^n}{2\Delta x} \tag{0.9}$$
$$\partial_y u(x,y,t) = \frac{u_{i,j+1}^n - u_{i-1,j}^n}{2\Delta y} \tag{0.10}$$
$$\tag{0.11}$$

Concernant les conditions de bords, il est possible de les simuler en utilisant la technique des points fantômes (ghost points). Nous considérons que le domaine de simulation réel n'est pas entre les points 1 et N_x mais entre les points 2 et $N_x - 1$. Par conséquence les points 1 et N_x peuvent être utilisé pour "simuler" les points d'indices -1 et $N_x + 1$ nécessaires au calcul des opérateurs. Dans le cas qui nous intéresse, et afin de vérfier la condition de bords nulle pour notre champs u, nous admettrons que la condition de bords se traduit de la manière suivante (les indices utilisés correspondent aux indices utilisés en Fortran) :

$$u(x, 0, t) = 0 \rightarrow u_{1,j} = -u_{3,j} \tag{0.12}$$
$$u(x, L_x, t) = 0 \rightarrow u_{N_x,j} = -u_{N_x-2,j} \tag{0.13}$$
$$u(0, x, t) = 0 \rightarrow u_{i,1} = -u_{i,3} \tag{0.14}$$
$$u(0, L_y, t) = 0 \rightarrow u_{i,N_y} = -u_{i,N_y-2} \tag{0.15}$$
$$\tag{0.16}$$

La résolution de la partie spatiale se fait via la méthode de Runge-Kutta 4 qui est la suivante :

$$f^{t+1} = f^t + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{0.17}$$
$$\text{with} \tag{0.18}$$
$$k_1 = \Delta t L(t, f^t) \tag{0.19}$$
$$k_2 = \Delta t L(t + \frac{\Delta t}{2}, f^t + \frac{1}{2}k_1) \tag{0.20}$$
$$k_3 = \Delta t L(t + \frac{\Delta t}{2}, f^t + \frac{1}{2}k_2) \tag{0.21}$$
$$k_4 = \Delta t L(t + \Delta t, f^t + k_3) \tag{0.22}$$
$$\tag{0.23}$$

1 Téléchargement et utilisation du code

L'intégralité des fichiers nécessaires sont accessibles à l'adresse suivante : https ://github.com/GFuhr/M2_Nano

2 Mesure des performances monoprocs

Les performances monoprocs peuvent se faire via : une mesure du temps d'exécution et une mesure du nombre de lectures du cache et des données à charger. Attention, pour

les mesures de temps et d'IO sur le cache, les mesures de référence sont différentes sur chaque machine. Vous ne pouvez donc pas comparer vos optimisations sur 2 machines différentes. De même il n'est pas permis de modifier les arguments transmis à gfortran ni d'utiliser un autre compilateur pour cette partie (remarque : gcc et gfortran sont le même compilateur) Afin d'avoir une base commune de compilation et de mesure, tous le processus est automatisé au sein du script profile.py. Ce script se lance, dans un shell, via la commande :
python3 profiler.py. Pour vous donner une idée des optimisations possibles, sur une machine donnée, le fichier compilé correspondant à un cas "idéal" est joint aux fichiers fourni et sera aussi exécuté par le script.

3 Développement d'une version parallèle

Pour cette partie nous supposerons que : Le nombre de points utilisés pour les dimensions x et y sont égaux et sont des puissances de 2. De même le code sera utilisé sur un nombre de coeurs qui sera aussi une puissance de 2. Afin de s'assurer que les mesures de temps sont fiables, toute simulation devra durer au minimum $10s$. Les mesures de temps seront faîtes via l'utilisation de la fonction MPI_WTIME au sein du programme main. Il est à noter que cette fonction est exécutée par chaque processeur par défaut. Afin de pouvoir paralléliser ce problème une approche efficace est de réfléchir à décomposer le domaine suivant la dimension y. Vous pouvez pour cela vous aider des commentaires qui se trouvent dans le fichier main_mpi.f90

4 Connection à la machine de calcul et exécution d'un programme

Premièrement, il faut créer un tunnel crypté pour accéder à la machine via la commande suivante (les logins et IP seront donnés en cours) :

```
ssh -L 1004X:ip_serveur:22 user@ip_gateway
ssh -p1004X user@localhost
```

Remarque :

- les comptes ne seront valides que jusqu'à la fin du mois. De même les actions effectuées sur ces machines sont enregistrées automatiquement.
- X est à remplacer par le chiffre indiqué

L'envoi/réception de fichiers se fait via la commande scp entre la machine locale et le calculateur :

- pour envoyer, par exemple, tous les fichiers f90 d'un dossier vers un dossier du calculateur
  `ssh -P1004X dossier_local/*.f90 user@localhost:/home/user/dossier_distant`
- pour envoyer, un dossier local ainsi que tous les sous dossiers vers un dossier du calculateur
  `ssh -r -P1004X dossier_local user@localhost:/home/user/dossier_distant/`
- pour récupérer, tous les fichiers f90 d'un dossier distant vers un dossier local
  `ssh -P1004X user@localhost:/home/user/dossier_distant/*.f90 dossier_local`

La compilation se passe de manière classique via la commande

```
gfortran -O2 -Wall -Wextra -Werror liste_fichiers_.f90 -o main.exe
mpifort -O2 -Wall -Wextra -Werror liste_fichiers_.f90 -o main.exe
```

5 Utilisation sous windows - via docker

1- télécharger docker CE (https ://hub.docker.com/editions/community/docker-ce-desktop-windows)
2- installer le client docker
3- ouvrir power shell et aller dans le dossier ou les fichiers ont été récupéré sur github
4- aller dans le dossier ./docker (cd ./docker)
5- créer le container, pour cela tapez la commande (et attendre) :
docker build -t mpilinux .
Remarque : une connexion internet est nécessaire 6- pour lancer le systeme linux émulé, toujours dans power shell, allez dans le dossier où se trouvent vos fichiers fortran
7- lancez la commande :
docker run -it -v nom_complet_du_dossier/ :/home/data m /bin/bash
8- vous vous retrouvez alors dans un shell linux complet. Allez dans le dossier /home/data (cd /home/data)
8- vous pouvez alors compiler vos fichiers, éxécuter le programme etc...