SXS HPC : BE 2 – Architecture à mémoire distribuée

Auteur : Antoine Brunet <antoine.brunet@onera.fr>
Public : SXS
Date : 2025-2026

Ce BE consiste à paralléliser un code de simulation sur une architecture à mémoire distribuée à l'aide de MPI, et à analyser la performance de cette parallélisation. Le BE se fait en binômes, et sera évalué sur les livrables suivants :

- Un rapport au format `pdf` présentant les réponses aux questions du BE.
- L'ensemble du code source produit.

Ces fichiers seront regroupés dans une archive, qui sera déposée sur LMS avant le 14 décembre 23h59 CET.

Présentation du problème

On s'intéresse à la résolution de l'équation de la chaleur 2D :

$$\frac{\partial T}{\partial t} = \alpha\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \right) \tag{1}$$

On prendra comme domaine le carré unitaire $[0,1]^2$, et comme conditions aux limites :

$$\begin{cases} T(x,0) = 0 & \\ T(x,1) = 0 & \forall x \in [0,1] \\ T(0,y) = 1 & \\ T(1,y) = 1 & \forall y \in [0,1] \end{cases} \tag{2}$$

Pour résoudre cette équation, on va utiliser la méthode des différences finies explicite en temps, sur un maillage cartésien. On désigne par (nx,ny) les dimensions du maillage de l'intérieur du domaine, et dx et dy les pas d'espace correspondants (on a dx = $\frac{1}{\text{nx}+1}$, et idem selon y). On choisit d'exclure les points du bord du domaine pour simplifier le découpage du domaine de calcul par la suite. La figure 1 présente le maillage utilisé.

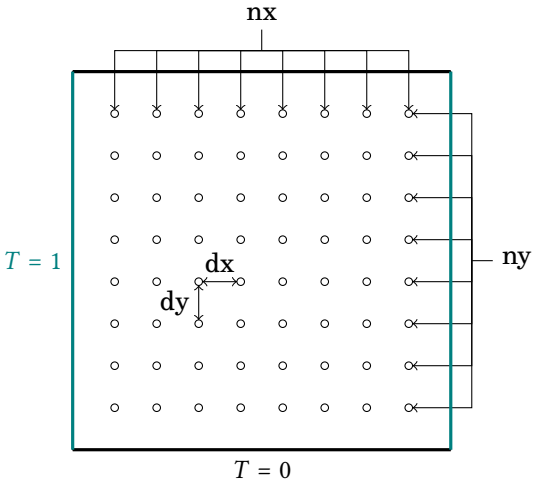

Figure 1 – Schéma du domaine et du maillage de simulation

En notant $T_{i,j}^k$ ($i \in [0, \text{ny}+1]$, $j \in [0, \text{nx}+1]$, $k \geq 0$) les valeurs sur les nœuds du maillage au temps k dt, on obtient le schéma suivant :

$$\begin{cases} T_{i,j}^{k+1} = \alpha\,\text{dt}\left(\frac{T_{i,j+1}^k - 2T_{i,j}^k + T_{i,j-1}^k}{\text{dx}^2} + \frac{T_{i+1,j}^k - 2T_{i,j}^k + T_{i-1,j}^k}{\text{dy}^2}\right) & \forall (i,j) \in [1,\text{ny}] \times [1,\text{nx}] \\ T_{0,j}^k = T_{\text{ny}+1,j}^k = 1 & \forall j \in [1,\text{nx}] \\ T_{i,0}^k = T_{i,\text{nx}+1}^k = 0 & \forall j \in [1,\text{ny}] \end{cases} \tag{3}$$

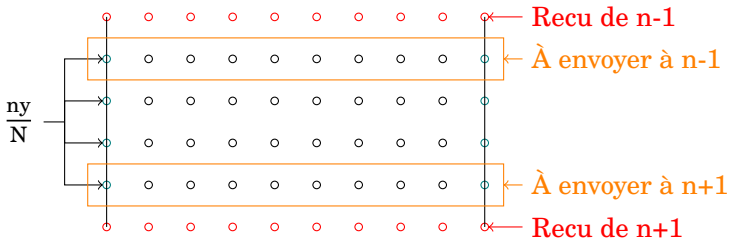

FIGURE 2 – Domaine local à un processus dans la décomposition de domaines

Lors du TP, on prendre une valeur de $\alpha = 1$, et un pas de temps dt $= \frac{0.1}{\text{dx dy}}$, permettant de respecter la condition CFL de stabilité du schéma. On prendra un nombre d'itération fixé à 1000.

L'archive du BE contient un programme implémentant un solveur séquentiel pour ce problème.

Le programme fourni est composé de plusieurs fichiers :

- `src/main.c` contient le programme principal qui lit les arguments du programme et construit la simulation correspondante, réalise le bon nombre d'itérations, et affiche le résultat
- `src/solver.c` contient les fonctions permettant d'initialiser le problème, de réaliser une itération, et d'afficher le résultat
- `include/solver.h` contient les déclaration de structures de données et les fonctions, ainsi que des commentaires utiles sur les différentes données

1 Parallélisation par décomposition de domaines

Dans cette partie, on cherche à paralléliser le code par une décomposition du domaine de calcul entre différents processus. On pourra supposer que ny est divisible par le nombre N de processus. Chaque processus calculera les itérations pour un bloc de lignes du maillage, et transmettra entre chaque itération aux processus chargés de calculer les blocs voisins les valeurs à l'interface des deux blocs, comme présenté dans la figure 2. Les données recues des blocs voisins seront utilisées comme conditions aux limites pour le calcul de l'itération suivante.

Question 1.1 :
Compiler le programme à l'aide de la commande `make solveur`. Lancer le programme avec la commande `./solveur 8 8`. Lire le fichier `include/solver.h`.

Question 1.2 :
Instrumenter le code de la fonction `main` pour qu'il affiche le temps total passé dans les itérations de calcul. Tracer le temps de calcul du solveur séquentiel en fonction de la taille du maillage, et commenter. Dans la suite des questions, on supposera une taille de maillage fixe de 1024×1024. Ne pas hésiter à prendre une taille de maillage plus faible pour vos tests lors du développement.

Question 1.3 :
Implémenter la parallélisation du solveur à l'aide de MPI. On ne modifiera que les fonctions contenues dans le fichier `solver.c`. Pour l'affichage du résultat, on rapatriera en fin de simulation la solution complète sur un des processus qui sera en charge de l'écriture, à l'aide d'une communication collective. On vérifiera que le résultat du calcul est bien indépendant du nombre de processus utilisé. Décrire les principaux problèmes rencontrés et vos solutions.

Question 1.4 :
Évaluer l'accélération obtenue avec votre implémentation parallèle, en fonction du nombre de processus. Tracer et analyser la courbe d'accélération.

Question 1.5 :
À partir de quel raffinement est-il intéressant d'utiliser deux processus ? et quatre processus ?

2 Calcul de la température moyenne

A chaque itération du calcul, on souhaite maintenant calculer et afficher la température moyenne $\tilde{T}_j$ en fonction de x :

$$\tilde{T}_j^k = \frac{1}{\text{ny}} \sum_{i=0}^{\text{ny}+1} T_{i,j}^k \tag{4}$$

Question 2.1 :
Implémenter une nouvelle fonction `print_mean` qui calcule de manière efficace et affiche cette température moyenne. On ajoutera dans la fonction `main` un appel à cette fonction après chaque itération du calcul. Justifier la méthode de calcul retenue.

3 Décomposition de domaines 2D

Dans cet exercice, on cherchera à implémenter une décomposition de domaine selon les dimensions x et y en même temps. On supposera que le domaine est décomposé en N = NX × NY régions.

Question 3.1 :
Ajouter un paramètre à l'exécution du programme permettant de définir la valeur de NX. On supposera que N est bien divisible par NX, et on calculera la valeur de NY en fonction.

Question 3.2 :
À l'aide de communicateurs adaptés, modifier le code pour permettre la décomposition de domaine en 2D. Les échanges de messages selon la dimension x seront réalisés de manière similaire à ceux de l'exercice 1 pour la dimension y. On pourra définir des types dérivés pour ces échanges. Pour l'affichage du résultat, on rapatriera dans un premier temps la solution sur un processus de chaque ligne, puis on réutilisera la fonction développée en partie 1. Décrire les problèmes rencontrés et vos solutions.

Question 3.3 :
Évaluer l'accélération obtenue avec cette nouvelle implémentation, et comparer aux résultats obtenus en question 1.4. Commenter.

License CC BY-NC-SA 3.0