# CALCUL PARALLÈLE

Khodor KHADRA

Du 27 juin au 01 juillet 2022

## PREMIERS PAS SUR MCIA

1. Se connecter sur la machine du Mésocentre *Curta* :
   *ssh username@curta.mcia.fr*
2. Créer sous votre racine */home/user* de votre compte un répertoire "Formation-CalculParallele" et copier dedans le fichier suivant à l'aide de la commande :
   *cp /scratch/FormationK-MCIA-CalculParallele/TPs.tar .*
   Puis taper la commande :
   *tar xvf TPs.tar*
3. Dans le répertoire "TEST-CURTA", écrire un petit programme qui calcule pour un entier $n > 1$ donné, la somme $S = \sum_{i=1}^{n} \frac{1}{i}$
4. Compiler le programme et créer l'exécutable à l'aide du Makefile
5. Ecrire un fichier *batch* et soumettez le job de calcul
6. Réserver un noeud en mode interactif et soumettez en mode interactif cette fois le programme.

# EXERCICES CALCUL PARALLÈLE avec la bibilothèque OpenMP

Chaque exercice contient la version séquentielle et la version parallèle OpenMP. Pour chaque exercice, charger le(s) module(s) pour la compilation et exécution, puis compiler. Exécuter pour des jeux de données différents le programme séquentiel et le programme OpenMP avec un nombre de threads allant de 1 à 12, et mesurer les efficacités. Pour chaque jeu de données, tracer la courbe des efficacités en fonction du nombre de threads.

## EXERCICE 1

Cet exercice a pour but de calculer l'approximation de $\pi = \int_0^1 \frac{4}{1+x^2}\ \mathrm{d}x$.

Pour calculer de façon approchée l'intégrale $\int_a^b f(x)\ \mathrm{d}x$, on divise le segment $[a,b]$ en $n$ sous intervalles égaux de pas constant $h = \frac{(b-a)}{n}$. On note $x_i$ l'abscisse au point $i$. On a donc d'après le schéma $x_i = (a + i * h) - (h/2)$

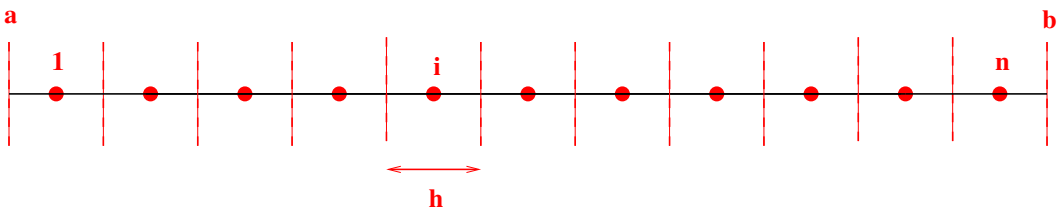


FIGURE 1 – Division du domaine en $n$ sous intervalles réguliers

On a alors la formule : $\int_a^b f(x)\ \mathrm{d}x = \sum_{i=1}^{n} \int_{x_{i-0.5}}^{x_{i+0.5}} f(x)\ \mathrm{d}x$

On peut calculer une approximation de l'intégrale sur chaque sous intervalle par la formule : $\int_{x_{i-0.5}}^{x_{i+0.5}} f(x)\ \mathrm{d}x = h * f(x_i)$

Ce qui donne donc : $\int_a^b f(x)\ \mathrm{d}x\ \simeq h \sum_{i=1}^{n} f(x_i)$

Le programme est complet et prêt à être exécuté et $n$ est une donnée du problème.

## EXERCICE 2

Cet exercice a pour but de manipuler des opérations sur des tableaux tridimensionnels. Mettez les directives OpenMP dans les boucles.

EXERCICE 3
Soit A une matrice carrée de taille $(n, n)$ tridiagonale définie positive, $A = (a_{ij})_{1 \leq i \leq n \quad 1 \leq j \leq n}$.

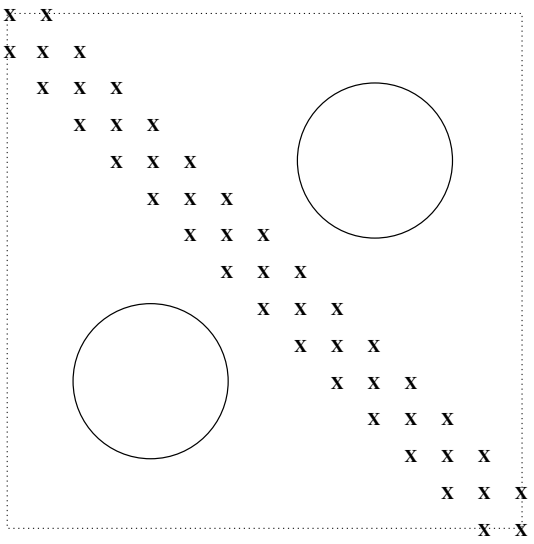

FIGURE 2 – matrice tridiagonale

On souhaite résoudre le système linéaire $Ax = b$. Le vecteur $b$ est calculé dans le code en fonction des coefficients de la matrice de telle sorte que le vecteur solution $x$ soit égal à 1. La méthode de résolution utilisée dans cet exercice est la méthode itérative de Jacobi :

*Initialisation*
$x^0 = 0$

*pour k = 1 à kmax faire*

$$\begin{cases} y^k = x^{k-1} \\ \\ x_i^k = (b_i - \sum_{j=1}^{i-1} a_{ij} * y_j^k - \sum_{j=i+1}^{n} a_{ij} * y_j^k)/a_{ii} \quad 1 \leq i \leq n \\ \\ \textit{si } \|x^k - y^k\| \leq \varepsilon \textit{ alors fin des itérations} \end{cases}$$

*fin des itérations en k*

La taille $n$, les coefficients de la matrice $A$, le nombre d'itérations maximum $kmax$ et le critère d'arrêt $\varepsilon$ sont des données du problème.
Mettez les directives OpenMP dans la boucle principale de calcul de la solution à l'itération courante $x^k$.

EXERCICES CALCUL PARALLÈLE avec la bibilothèque MPI

Pour chaque exercice, charger le(s) module(s) pour la compilation et exécution, puis compiler et exécuter le programme avec un nombre de processus MPI croissant à partir de 1, puis mesurer les efficacités. Réserver un nombre total de coeurs de calcul le plus proche possible du nombre maximum de processus MPI utlisés. Pour les exercices 4, 5 et 6, exécuter le code avec plusieurs jeux de données et pour chaque jeu de données, tracer la courbe des efficacités en fonction du nombre de processus MPI.

EXERCICE 1
Ecrire un programme MPI qui imprime pour chaque processus $p$ parmi $n$ processus, son rang (numéro) $r$ et s'il est pair ou impair, $1 \leq p \leq n$.

EXERCICE 2
Le programme MPI est le suivant : parmi $n$ processus, le processus 0 envoie au processus 1 la valeur 100 qui la reçoit, et le processus 1 envoie au processus 0 la valeur 111 qui la reçoit. Il y a deux erreurs dûs à des phénomènes de "deadlock" dans le code, les corriger.

EXERCICE 3
Il s'agit d'un anneau de communication. Ecrire un programme MPI où parmi $n$ processus :

- le processus de rang $r$ reçoit la valeur $1000+(r-1)$ du processus de rang $r-1$, $1 \leq r \leq n-1$
- le processus de rang 0 reçoit la valeur $1000 + (n-1)$ du processus de rang $n-1$.

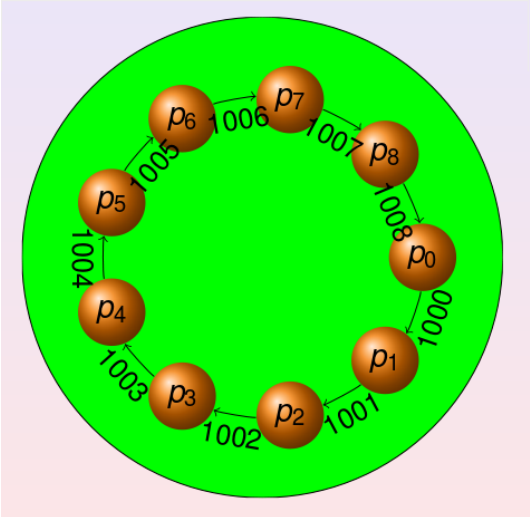


FIGURE 3 – anneau de communication

EXERCICE 4
Reprise de l'EXERCICE 2 avec MPI. Le programme MPI répartit de façon connexe et le plus homogène possible les intervalles sur plusieurs sous domaines et chaque processus associé à un sous domaine calcule localement la valeur de l'intégrale dans le sous domaine. Ecrire l'opération de réduction correspondante par le processus de rang 0 pour le calcul global de l'intégrale sur le domaine $[0, 1]$. Dans cet exercice, il n'y a pas de communication point à point entre les sous domaines.

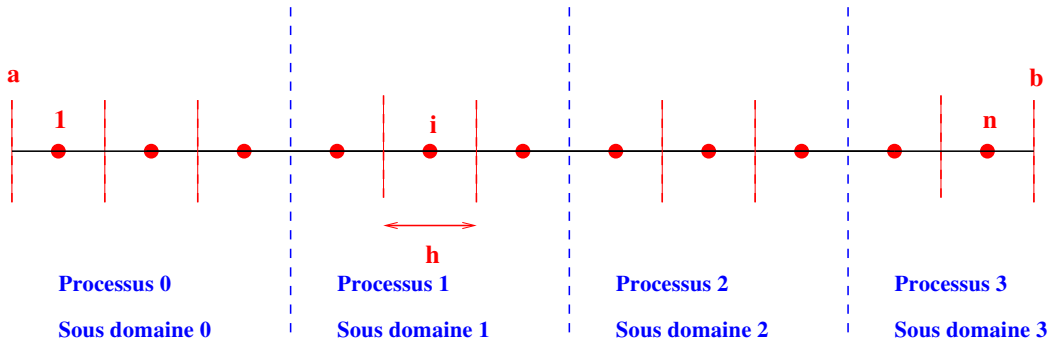


FIGURE 4 – Découpage connexe du domaine global en sous domaines, ici en nombre de 4

EXERCICE 5
Soit $A$ une matrice carrée réelle de taille $(ng, ng)$ diagonale par blocs.

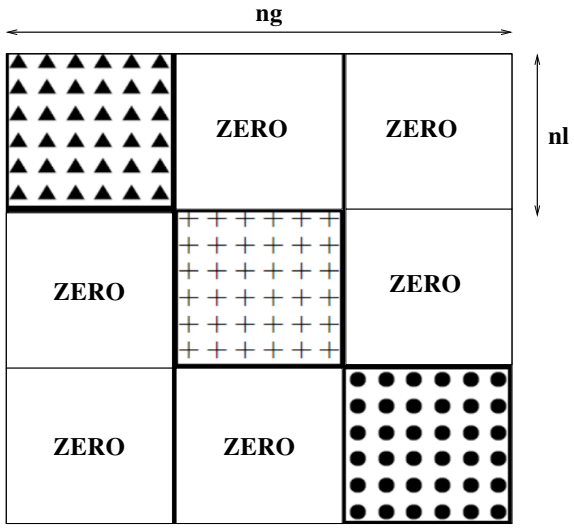


FIGURE 5 – matrice diagonale par blocs

Ecrire un programme MPI où parmi $n$ processus :

- chaque processus de rang $r$ est associé à un bloc diagonal (sous-matrice) de taille $nl$ de coefficients variables ;
- chaque processus de rang $r$ calcule la trace locale de son bloc correspondant ;
- le processus de rang 0 récupère toutes les traces locales pour calculer la trace globale de la matrice.

$ng$ est une donnée du problème.
On rappelle que pour une matrice générale carrée de taille $(m, m)$, $A = (a_{ij})_{1 \leq i \leq m \quad 1 \leq j \leq m}$, la trace de $A$ est définie par $Tr(A) = \sum_{i=1}^{m} a_{ii}$

EXERCICE 6

On souhaite résoudre le problème d'évolution en temps et dans le domaine spatial $[0,1]$, l'équation de la chaleur 1D :

$$\begin{cases} \dfrac{\partial u}{\partial t} - \Delta u = f(x) \quad \text{dans } ]0,T]\times]0,1[, T>0 \\ \\ u(x=0,t) = u(x=1,t) = 0 \quad \forall t \in [0,T] \\ \\ u(x,t=0) = u_0(x) \quad \forall x \in [0,1] \end{cases}$$

La première étape consiste à discrétiser le problème en temps dans le domaine $[0,T]$. On découpe ce domaine en $N$ intervalles réguliers de pas de temps $\Delta t = T/N$ et on choisit ici un schéma d'Euler implicite. Soit $u^n$ la solution que l'on souhaite calculer à l'instant $t_n = n\Delta t$ sur l'intervalle $[0,1]$ à partir de la solution $u^{n-1}$ ($u^0 = 0$), $u^n$ est solution du sytème :

$$\begin{cases} \dfrac{u^n - u^{n-1}}{\Delta t} - \Delta u^n = f(x) \quad \text{dans } ]0,1[ \quad 0 < n \leq N \\ \\ u^n(x=0) = u^n(x=1) = 0 \end{cases}$$

La seconde étape consiste à discrétiser le problème en espace dans le domaine $[0,1]$. On découpe ce domaine en $M+1$ intervalles réguliers de pas d'espace $\Delta x = 1/(M+1)$, ce qui constitue le maillage d'espace, et on choisit ici un schéma centré aux différences finies à 3 points d'ordre 2, $u^n$ est donc solution du sytème linéaire :

$$\begin{cases} \dfrac{u_i^n}{\Delta t} + \dfrac{-u_{i-1}^n + 2u_i^n - u_{i+1}^n}{(\Delta x)^2} = \dfrac{u_i^{n-1}}{\Delta t} + f(x_i) \quad 0 < n \leq N \quad 1 \leq i \leq M \\ \\ u_{i=0}^n = u_{i=M+1}^n = 0 \end{cases}$$

Ainsi $u^n$ est solution d'un système linéaire $Au^n = b(u^{n-1})$ dont la matrice est tridiagonale symétrique définie positive. Les coefficients de la diagonale principale sont égaux à $(1/\Delta t) + 2/(\Delta x)^2$, et les coefficients de la sous-diagonale et sur-diagonale sont égaux à $-1/(\Delta x)^2$ . Ainsi à chaque pas de temps, il faut résoudre en espace ce système linéaire jusqu'à convergence vers la solution du problème stationnaire. La méthode choisie est la méthode itérative du gradient conjugué qu'on rappelle pour un système linéaire $Ax = b$ avec une matrice $A$ symétrique définie positive (ce qui est la cas ici) :

*Initialisation*

$$\begin{cases} x^0 &= 0 \\ r^0 &= b - Ax^0 \\ p^0 &= r^0 \end{cases}$$

*do k = 1, kmax*

$$\begin{cases} \gamma^k &= (r^{k-1}, r^{k-1}) \\ q^k &= Ap^{k-1} \\ \lambda^k &= (p^{k-1}, q^k) \\ \alpha^k &= \gamma^k/\lambda^k \\ x^k &= x^{k-1} + \alpha^k p^{k-1} \\ r^k &= r^{k-1} - \alpha^k q^k \\ \theta^k &= (r^k, r^k) \\ \textit{si } \sqrt{\theta^k} \leq \varepsilon \textit{ alors fin des itérations} \\ \beta^k &= \theta^k/\gamma^k \\ p^k &= r^k + \beta^k p^{k-1} \end{cases}$$

*end do*

Les deux opérations algébriques essentielles dans cet algorithme sont le produit matrice-vecteur et le produit scalaire ( , ) de deux vecteurs.
Le pas de temps $\Delta t$, le nombre global de mailles sur l'intervalle [0,1] (c'est à dire $M+1$), le nombre d'itérations maximum et le critère d'arrêt $kmax$ et $\varepsilon$ sont des données du problème.

Compléter dans le programme MPI :

- le programme principal pour la diffusion des données par le processus de rang 0 aux autres processus ;
- le produit scalaire avec des opérations de réduction ;
- le produit matrice-vecteur avec des communications.

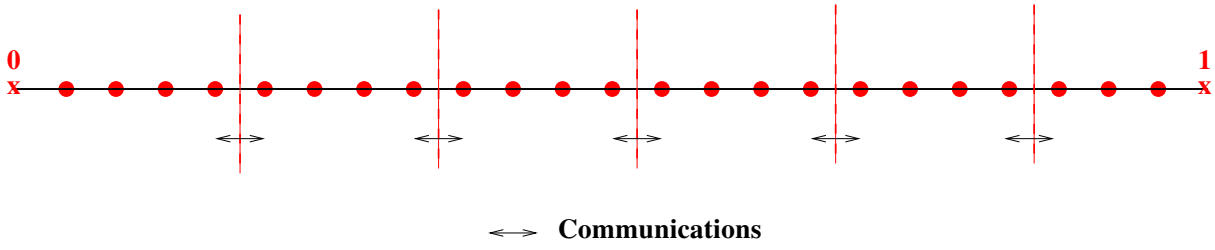


FIGURE 6 – Découpage connexe du domaine global en sous domaines et communications point à point