**Exercice 1 :** Open MP

Soit le code suivant

1 #**include** <omp.h>

2 #**include** <stdio.h>

3 #**include** <stdlib.h>

4

5 **int** a, b, tid;

6 **float** x;

7 #**pragma omp** **threadprivate**(a, x)

8

9 main () {

10

11 omp\_set\_dynamic(0);

12

13 #**pragma omp** **parallel** **num\_threads(4) private(tid)**

14 {

15 tid = omp\_get\_thread\_num();

16 a = tid;

17 b = tid;

18 x = 1.1 \* tid +1.0;

19 printf("Thread %d: a,b,x= %d %d %f\n",tid,a,b,x);

20 }

21 **#pragma omp parallel private(tid)**

22 {

23 tid = omp\_get\_thread\_num();

24 printf("Thread %d: a,b,x= %d %d %f\n",tid,a,b,x);

25 }

26 }

1. Expliquer la ligne 7
2. Expliquer la ligne 11
3. Expliquer la ligne 13
4. Soit la sortie suivante de la première zone parallèle

Thread 0: a,b,x= 0 1 1.000000

Thread 2: a,b,x= 2 3 3.200000

Thread 3: a,b,x= 3 3 4.300000

Thread 1: a,b,x= 1 2 2.100000

Le résultat vous semble-t-il correct ? Commenter

1. Si on veut avoir la sortie suivante a-t-on besoin de modifier quelque chose, et si c’est oui Quoi ?

Thread 0: a,b,x= 0 0 1.000000

Thread 2: a,b,x= 2 2 3.200000

Thread 3: a,b,x= 3 3 4.300000

Thread 1: a,b,x= 1 1 2.100000

1. Après correction s’il y en a, donner le résultat de la ligne 24

**Exercice 2 : produit matrice vecteur parallèle distribué en communication collective**

On considère une machine parallèle à mémoire distribuée à p processeurs. On veut réaliser le produit d’une matrice pleine *A* de dimension n par un vecteur *x*.

 *y* = A\**x*

Le vecteur *x* sera initialement connu dans tous les processus, et à l’issu du produit, le vecteur résulta *y* devra être reconstitué totalement dans chaque processeur. Par contre, les coefficients de la matrice *A* ne devront pas être dupliqués. On cherchera d’abord à équilibrer la charge de travail et la taille mémoire occupée dans chaque processeur.

1. Décrire la parallélisation du produit matrice-vecteur pour une répartition de la matrice par bloc de lignes ou bloc de colonnes.
2. Décrire la parallélisation du produit matrice-vecteur pour un découpage de la matrice en bloc carrés..
3. Donner le code MPI pour la première question.

**Exercice 3 : calcul d’une transposée d’une matrice en communication collective**

Ecrire un programme qui calcule la transpose d’une matrice carrée A de dimension quelconque (multiple du nombre de processeurs), B = AT.

La matrice est divisée entre les tâches et initialisée de telle sorte que chaque élément est unique (utiliser le numéro ligne-colonne et le rang des taches).

Une représentation visuelle des matrices A et B (dans l'exemple 8x8):



Résoudre le problème distribuant A et B par lignes (ou colonnes) sur les tâches en utilisant la communication collective MPI\_ALLtoall. Noter que cette fonction collective fonctionne avec des données stockées dans la mémoire contiguë donc choisir avec soin comment il faut stocker la matrice par lignes ou colonnes.

On vous demande de :

* Expliquer la primitive MPI\_Alltoall par un schéma et donner ses paramètres
* Initialiser la matrice A telle qu’elle est donnée sur le schéma
* Evaluer B = AT