I. OpenMP (bases)

Matériel : poste GNU/Linux, macOS ou Windows

Installation / configuration

GNU/Linux / macOS : ajouter les options de compilation / link : -fopenmp

Windows : dans Visual Studio, aller dans les propriétés du projet "C/C++ -> Langage" et changer "Prise en charge OpenMP" à "Oui (/openmp)"

Hello world

Avec OpenMP, créer un bloc de 4 threads qui affichent "Hello world" suivi de l'identifiant du thread

Somme

Ecrire un programme OpenMP qui effectue la somme d'un tableau de 1 000 000 entiers

Multiplication de matrices

Ecrire un programme *single thread* de multiplication de deux matrices de 1024x1024 éléments

Noter le temps mis pour effectuer le traitement

Parallèliser le code précédent avec OpenMP

Comparer le temps entre le programme *single-thread* et *multi-thread*

Faire varier le nombre de thread avec l'option num_threads dans le *#pragma for* et comparer les temps

Listing 4: algorithme de calcul matriciel

```
const int M = 1024;
const int N = 1024;

int* first = malloc(M * N * sizeof(int));
int* second = malloc(M * N * sizeof(int));
int* result = malloc(M * N * sizeof(int));


for(int i = 0 ; i < M ; i++)

{

  for(int j = 0 ; j < N ; j++)

  {

    int tmp = 0;


    for(int k = 0 ; k < M ; k++)

    {

      tmp += first[i * N + k] * second[k * N + j];

    }
```

II. OpenMP (suite)

Parallel for

Écrire un programme en C qui utilise OpenMP pour paralléliser une boucle **for**. La boucle **for** calcule la somme des éléments d'un tableau. Le programme doit calculer la somme du tableau en utilisant une version séquentielle et une version parallèle, puis afficher les résultats et comparer les temps d'exécution.

Barriere

On souhaite paralléliser le traitement d'un tableau d'entiers en utilisant la librairie OpenMP. Le but est d'utiliser les barrières pour synchroniser l'exécution des threads et garantir la cohérence des résultats.

Le programme doit effectuer les opérations suivantes :

1. Initialiser un tableau d'entiers de taille N avec des valeurs aléatoires.
2. Parcourir le tableau et multiplier chaque élément par 2.
3. Afficher le tableau modifié.

Le programme doit être exécuté avec un nombre variable de threads, passé en paramètre au lancement du programme.

Consignes :

- Utiliser la directive OpenMP "#pragma omp parallel for" pour paralléliser la boucle de traitement du tableau.
- Utiliser une barrière OpenMP pour synchroniser les threads avant l'affichage du tableau modifié.
- Initialiser le générateur de nombres aléatoires avant la création des threads, pour garantir que chaque thread a des valeurs différentes.

Single

Écrire un programme en C qui calcule la somme de tous les éléments d'un tableau à l'aide d'OpenMP et de la directive single. Le programme doit prendre en entrée le nombre d'éléments du tableau et générer un tableau aléatoire de ces éléments. Ensuite, chaque thread doit calculer la somme partielle de ses éléments du tableau. Finalement, un seul thread doit ajouter toutes les sommes partielles pour obtenir la somme totale.

Critical

Écrire un programme en utilisant OpenMP qui calcule la somme des éléments d'un tableau. Chaque thread doit additionner une partie du tableau et stocker sa somme partielle dans une variable partagée. Utilisez la directive critical pour éviter que les threads n'écrasent la variable partagée.

Atomic

Écrire un programme C qui utilise OpenMP pour calculer la somme d'un grand nombre d'entiers positifs. Utilisez des variables atomiques pour garantir l'exactitude des résultats, même lorsque plusieurs threads effectuent des calculs simultanément.

Flush

Écrire un programme en utilisant OpenMP qui calcule la somme des éléments d'un tableau d'entiers. Pour cela, vous allez créer plusieurs threads qui vont chacun effectuer une partie du

calcul. Pour garantir que chaque thread effectue les calculs sur les éléments du tableau de manière cohérente, vous allez utiliser la directive **omp flush**.

Le programme doit prendre en entrée un entier N représentant la taille du tableau, ainsi que les N éléments du tableau. Le résultat doit être la somme de tous les éléments du tableau.

Sections

En utilisant la librairie OpenMP, écrivez un programme en langage C qui calcule le produit matriciel de deux matrices carrées A et B de taille n x n. Pour améliorer la performance de votre programme, vous allez utiliser la directive **omp sections** pour paralléliser les calculs sur les différentes lignes de la matrice C résultante. Votre programme doit demander à l'utilisateur d'entrer la taille n des matrices et les éléments des matrices A et B, puis afficher la matrice C résultante.

OpenMP ordered

Écrire un programme en C qui calcule la somme de tous les éléments d'un tableau en utilisant OpenMP. Utiliser l'option "ordered" pour s'assurer que les threads effectuent les additions dans l'ordre des indices du tableau.