UNIVERSITÉ D'ORLÉANS **Programmation Parallèle**
M1 Info Année **2019 - 2020**

Série de Travaux Dirigés : **5** *- OpenMP*

Pour faciliter les exercices, vous disposez sous Celene d'une archive *TD5.tgz* contenant la trame de tous les exercices. Vous trouverez un répertoire par exercice avec le programme principal et les indications du code à compléter.

Pour un premier test, le programme **hello word** (répertoire Hello) est disponible. Vous pouvez le compiler et le tester avec la commande `./Hello` .

Exercice 1. Addition de vecteurs (répertoire VecAdd)

Ce répertoire contient le programme `VecAdd` qui prend en argument la taille des vecteurs à additionner et le nombre de threads total à utiliser. Il propose une version séquentielle du calcul dans la fonction `vecadd` ainsi qu'une version parallèle utilisant les threads c++11 (voir TD n°4).

1. Complétez la fonction `vecadd_omp` qui effectue le calcul en parallèle avec `OpenMP`.
2. Vérifiez vos résultats.
3. A l'aide de la macro `BENCHMARK` comparez les temps de calcul en utilisant les trois fonctions pour les tailles de tableaux suivantes : 10, 1000, 100000, 10000000.

Exercice 2. Pi (Monte-Carlo) – répertoire Pi

La méthode de Monte-Carlo permet de calculer des valeurs numériques approchées en utilisant des procédés aléatoires. Les méthodes de Monte-Carlo sont particulièrement utilisées pour calculer des intégrales en dimensions plus grandes que 1 (en particulier pour calculer des surfaces et des volumes). On peut appliquer cette méthode pour approcher la valeur de Π. Soit un carré de côté 2 et d'aire 4 et un disque de rayon 1 et de centre le centre du carré. L'aire du disque est Π. Si l'on choisit aléatoirement un point du carré, la probabilité qu'il soit dans le disque est donc $\frac{\Pi}{4}$. Si l'on tire au hasard un grand nombre de points du carré, on peut espérer que

$$\frac{\text{nb points à l'intérieur du disque}}{\text{nb points tirés}} \simeq \frac{\Pi}{4}.$$

Le répertoire `Pi` contient le programme qui effectue ce calcul séquentiellement. En utilisant `OpenMP` et sans changer une ligne du programme séquentiel, parallélisez ce programme.

Exercice 3. MinMax – répertoire MinMax

Dans cet exercice, il vous est demandé d'écrire une version parallèle (avec OpenMP) des fonctions `min` et `max` qui calculent la valeur minimale (et respectivement maximale) d'un vecteur ainsi que l'indice de cette valeur dans le vecteur.