

# TP 3. Programmation multi-threads (2)

## Préparation

Récupérer l'archive compressée `TP3.tar.gz` et extraire les fichiers qui sont contenus dans cette archive :

```
cd <repertoire dans votre espace de travail>
cp /home/t/tajchman/AMSI03/2018-12-14/TP3.tar.gz .
tar xvfz TP3.tar.gz
```

Se placer dans le répertoire TP3 :

```
cd TP3
```

et préparer les compilations dans les points suivants avec les commandes ci-dessous :

```
mkdir -p build
cd build
cmake ../src
make install
cd ..
```

## 1 Exemple OpenMP “grain fin”

Le fichier `src/sinus_fine/sinus.cxx` contient la version OpenMP “grain fin” obtenue à la fin du TP précédent.

On s'en servira de base pour les versions dites “grain grossier”.

### Question 1.

Comparer les temps d'exécution de la version séquentielle et de cette version en tapant les commandes

```
./build/sinus_seq/sinus_seq 40000
OMP_NUM_THREADS=3 ./build/sinus_fine/sinus_fine 40000
```

## 2 Exemple OpenMP “grain grossier”

Le fichier `src/sinus_coarse_1/sinus.cxx` contient une version OpenMP “grain grossier”.

### Question 2.

Examiner ce fichier et comparez-le à la version “grain fin”.

Comparer les temps d'exécution de la version “grain fin” et de cette version en tapant les commandes

```
OMP_NUM_THREADS=3 \  
time -p ./build/sinus_fine/sinus_fine 40000  
OMP_NUM_THREADS=3 \  
time -p ./build/sinus_coarse_1/sinus_coarse_1 40000
```

Il y a peu de différence (normalement) entre les deux versions. Expliquer pourquoi.

Le code affiche aussi les temps de calcul des différents threads.

### 3 Exemple OpenMP “grain grossier” avec équilibrage de charge

Le calcul du sinus en utilisant un développement de Taylor a été volontairement ralenti pour accentuer la différence de temps calcul de  $x \mapsto \sin x$  pour différentes valeurs de  $x$ .

Il s’en suit que les threads ne prennent pas le même temps de calcul suivant la plage des valeurs de  $x$  qui leur sont attribuée (et qui est la même que dans le cas “grain fin”), voir le fichier `src/sin.cxx`.

Dans cette version, on utilise un algorithme d’équilibrage de charge entre les différents threads.

#### Question 3.

Examiner le fichier `src/sinus_coarse_2/charge.cxx` qui contient cet algorithme et le fichier `src/sinus_coarse_2/sinus.cxx` qui l’utilise.

Exécuter plusieurs fois la commande

```
OMP_NUM_THREADS=3 \  
time ./build/sinus_coarse_2/sinus_coarse_2 40000
```

Chaque exécution tente d’améliorer les temps calcul en adaptant la répartition de charge de mieux en mieux (si possible).

#### *Remarque*

L’algorithme d’équilibrage de charge utilisé ici n’est pas optimal. Vous êtes encouragés à l’étudier et à l’améliorer.

## 4 Parallélisation du (mini-)code avec le modèle OpenMP “grain grossier”

Le répertoire `code/PoissonOpenMP_CoarseGrain` contient la version du code parallélisé par de l’OpenMP “grain fin”.

### Question 4.

Remplacer la parallélisation “grain fin” par le modèle “gros grain” (sans perte de charge).

Les fichiers source, que vous avez modifiés, seront à fournir avant la fin de la semaine suivante et vous seront renvoyés commentés et notés. La note obtenue pourra améliorer la note finale suivant les modalités précisées au premier cours.

## 5 Parallélisation en utilisant le concept de tâches OpenMP

### Question 5.

Examiner le fichier `src/exemple_tasks/main_seq.cxx`.

Cet exemple est difficile à paralléliser avec OpenMP. Expliquer pourquoi.

Le modèle de programmation par tâches OpenMP a été conçu pour répondre à ce type de situation.

### Question 6.

Examiner le fichier `src/exemple_tasks/main_tasks.cxx`.

Compiler et exécuter. Comparer les temps calcul avec la version séquentielle.

Pensez à vérifier que les résultats sont bien les mêmes dans les deux cas.