TP 2. Programmation multi-threads

Préparation

Récupérer l'archive compressée `TP2.tar.gz` et extraire les fichiers qui sont contenus dans cette archive :

```
cd <repertoire dans votre espace de travail>
cp /home/t/tajchman/AMSI03/2018-12-07/TP2.tar.gz .
tar xvfz TP2.tar.gz
```

Se placer dans le répertoire `TP2` :

```
cd TP2
```

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

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

1 Code séquentiel

Le fichier `src/sinus_seq/sinus.cxx` calcule une approximation par série de Taylor de la fonction $x \mapsto \sin(x)$ pour un ensemble de valeur de $x \in [0, 2\pi]$. Les résultats sont sauvegardés dans un fichier.
Un script de commande unix `trace.sh` est fourni pour visualiser les résultats (graphes du sinus calculé par la machine et par la formule approchée, en utilisant `gnuplot`).

Question 1.

Se placer dans le répertoire TP2.

Compiler le code en tapant

```
make -C build sinus_seq
```

Exécuter le code en tapant

```
./build/sinus_seq/sinus_seq 500
```

Tracer le graphe des résultats et le visualiser en tapant

```
./trace.sh
```

Les courbes sur le graphe représentent les valeurs calculées par le sinus de la librairie standard et celles calculées par la formule de Taylor du programme

2 Première version multi-threads avec OpenMP

On peut choisir a priori le nombre maximum de threads qui seront utilisés dans l'exécution d'un code, ce choix peut se faire de plusieurs façons.
Ici, l'utilisateur définira une variable d'environnement `OMP_NUM_THREADS` avec une valeur entière (entre 1 et le nombre de cœurs disponibles dans le processeur).
On peut aussi utiliser la fonction `omp_set_num_threads()` dans le code.

Question 2.

Compiler, exécuter le code en utilisant 3 threads et tracer les résultats en tapant

```
make -C build sinus_openmp_1
OMP_NUM_THREADS=3 ./build/sinus_openmp_1/sinus_openmp_1
./trace.sh
```

Question 3.

Comparer les temps de calcul entre
— la version séquentielle
— la version multi-threads en utilisant 1 thread
— la version multi-threads en utilisant 2 threads
— la version multi-threads en utilisant 3 threads
— ...
— la version multi-threads en utilisant 6 threads

Interpréter les résultats.

3 Version multi-threads en utilisant les `std::threads`

Question 4.

Exécuter les codes `install/Release/power2` et `install/Release/power3`, comparer les temps de calcul.

Expliquer les différences éventuelles de temps calcul en examinant les fichiers sources C++ `src/valeur_propre/power2.cpp` (utilisé dans le code power2) et `src/valeur_propre/power3.cpp` (utilisé dans le code power3).

3.1 Tentative d'optimisation 3

Question 5.

Exécuter les codes `install/Release/power3` et `install/Release/power4`, comparer les temps de calcul.

Expliquer les différences éventuelles de temps calcul en examinant les fichiers sources C++ `src/valeur_propre/power3.cpp` (utilisé dans le code power3) et `src/valeur_propre/power4.cpp` (utilisé dans le code power4).

4 Transposition de matrice

4.1 Parcours par lignes ou par colonnes

On s'intéresse ici à l'opération de transposition des matrices :

$$A^T = (a^T_{i,j})_{i=1,\dots,n,j=1,\dots,n} = (a_{j,i})_{i=1,\dots,n,j=1,\dots,n}$$

où $a_{i,j}$ est le coefficient de la matrice d'origine à la ligne i et la colonne j

Question 6.

Exécuter les codes `install/Release/transpose1` et `install/Release/transpose2`.

Comparer les temps de calcul et expliquer les différences en examinant les fichiers source `src/transposee/transpose1.cpp` et `src/transposee/transpose2.cpp`.

4.2 Algorithme par bloc - version 1

On garde la structure des matrices comme dans `transpose1.cpp` et `transpose2.cpp`. Par contre le parcours de indices de matrice est différent.

Question 7.

Exécuter le code `install/Release/transpose3`.

Comparer les temps de calcul avec les 2 versions précédentes et expliquer les différences en examinant le fichier source `src/transposee/transpose3.cpp`.

4.3 Algorithme par bloc - version 2

Dans cette version, on utilise une structure des matrices par bloc. Chaque bloc est lui-même une matrice à coefficients scalaires. L'algorithme s'écrit formellement de la même façon.

Question 8.

Exécuter le code `install/Release/transpose4`.

Comparer les temps de calcul avec la version précédente et expliquer les différences en examinant le fichier source `src/transposee/transpose4.cpp`.