Département d'informatique, Université de Biskra.

TP4: Programmation paralléle avec OpenMP

Exercice 1 :

La valeur de Pi ∏ peut être calculée par l'intégrale :

$$\int_0^1 \frac{4}{1+x^2}$$

Cette intégrale peut être approchée par:

$$\pi \approx \sum_{i=0}^{n} f(x_i)\Delta x \text{ où } \Delta x = \frac{1}{n} \text{ et } x_i = \frac{1}{2n} + i\frac{1}{n}$$

Plus **n** est grand, plus on se rapproche de **Pi**.

Q1: Proposez une implémentation séquentielle pour calculer de la valeur Pi ∏.

Q2: Proposez une implémentation parallèle avec OpenMP pour calculer l'approximation de la valeur Pi ∏.

Exercice 2 :

Dans ce programme on va calculer la suite syracuse que nous allons paralléliser. On va utiliser la fonction de bibliothèque omp_get_wtime pour mesurer le temps réel écoulé entre deux points du programme.

```
#include <stdio.h>
#include <omp.h>
#define TOTAL 2000000
int main(int, char *[])
{
  int pass=0, cur;
```

```
  double start = omp_get_wtime();
////////////////////// MODIFIER A PARTIR D'ICI UNIQUEMENT ////////////////////
  for(int i=0; i<TOTAL; i++) {
     cur=i;
     while (cur>1)
       cur=cur%2?3*cur+1:cur/2;
     pass++;
  }
////////////////////// MODIFIER JUSQU'ICI UNIQUEMENT ////////////////////
  double end = omp_get_wtime();
  printf("%d out of %d ! (delta= %d)\n", pass, TOTAL, TOTAL-pass);
  printf("time: %lf ms\n ", (end-start)*1000);
}
```

- Compilez et exécutez le programme. Vérifiez que la variable `pass` doit être égale à `TOTAL` en fin de programme.
- Parallélisez ce programme. Attention, le comportement du programme doit toujours être le même, il s'agit de répartir le travail entre plusieurs *threads* ! Si besoin, utilisez des sections critiques pour protéger l'accès aux variables partagées. Attention aussi à la variable `cur` qui mérite sans doute une version privée par `thread` !
- Essayez de varier le nombre de threads et observez le gain de temps ! L'accélération obtenue est-elle égale au nombre de threads ? Pourquoi ?
- *PS. On pourra s'aider d'une boucle shell comme ceci :*

  $ for np in 1 2 4 8 16; do OMP_NUM_THREADS=$np ./syracuse; done

Exercice 3 :

Dans ce programme on va calculer la suite syracuse que nous allons paralléliser. On va utiliser la fonction de bibliothèque omp_get_wtime pour mesurer le temps réel écoulé entre deux points du programme.