TP nº 3 : synchronisation et tâches

1 Nombre de threads

Étudiez et corrigez le code suivant de deux manières différentes, en ajoutant seulement des directives OpenMP et sans utiliser la clause **`reduction`** [Lien] :

```
void nb_thread_kernel() {
  size_t nb_threads = 0;
  #pragma omp parallel
  {
    nb_threads++;
  }
  printf("nb_threads = %zu\n", nb_threads);
}
```

2 Calcul des premiers nombres premiers

Étudiez et parallélisez le code suivant [Lien] :

```
void prime_kernel(size_t primes[], size_t* ptr_nb_primes) {
  size_t nb_primes = 0;
  size_t divisor;
  bool is_prime;

  for (size_t i = PRIME_MIN; i < PRIME_MAX; i+=2) {
    is_prime = true;
    divisor = PRIME_MIN;

    while ((divisor < i) && is_prime) {
      if ((i % divisor) == 0)
        is_prime = false;
      divisor += 2;
    }

    if (is_prime) {
      primes[nb_primes] = i;
      nb_primes++;
    }
  }

  *ptr_nb_primes = nb_primes;
}
```

3 Graphe de tâches

Étudiez et parallélisez le code suivant en supposant que votre *runtime* OpenMP ne supporte pas le parallélisme imbriqué (on ne peut pas créer de région parallèle dans une région parallèle). Vous effectuerez une première parallélisation sans tâches OpenMP, puis vous exploiterez les tâches qui devraient vous permettre d'améliorer encore le temps d'exécution [Lien] :

```
void dag_kernel(double r1, double r2, double r3, double* r) {
  double d1, d2, d3, d4, d5, d6;

  d1 = f(r1, r2, 1);
  d2 = f(r2, r3, 1);
  d3 = f(d1, d2, 1);
  d4 = f(r1, r3, 2);
  d5 = f(r2, d2, 1);
  d6 = f(d5, d4+d3, 1);

  *r = d6;
}
```

4 Boucles non itératives : nombre de Fibonacci

Étudiez et parallélisez le code suivant [Lien] :

```
int fibok(int n) {
  if (n < 2)
    return n;

  return fibok(n-1) + fibok(n-2);
}

void fibonacci_kernel(int n, int* fibo) {
  *fibo = fibok(n);
}
```

5 Parallélisation d'un code à l'état sauvage

Dans la nature, les codes parallélisables sans modifications sont rares !

- Récupérez les codes suivants (crédit Dominique Béréziat) : [Lien] et [Lien].
  - Ce code calcule les points de l'ensemble de Mandelbrot.
  - Enregistre le résultat sous forme d'image au format ras (ouvrable avec LibreOffice).
- Lisez ce programme et étudiez son parallélisme.
- Rendez possible la parallélisation par OpenMP.
  - Cherchez à avoir une forme itérative acceptable [doc].
  - Modifiez certains calculs pour rendre le code parallèle.
- Implémentez une version parallèle avec OpenMP.