TP3 : programmation par tâches

Version du 9 janvier 2022

Exercice 1 – Fibonacci

La suite de Fibonacci est définie par

$$\mathcal{F}_0 = 0$$
$$\mathcal{F}_1 = 1$$
$$\forall n \geq 2, \qquad \mathcal{F}_n = \mathcal{F}_{n-1} + \mathcal{F}_{n-2}$$

1. Récupérer le code implémentant le calcul de la suite de Fibonacci en séquentiel, et le paralléliser avec OpenMP en vérifiant la justesse du calcul parallèle.
2. Comparer les performances obtenues avec la version séquentielle, par exemple pour $n = 40$.
   Comment améliorer les performances obtenues en parallèle ?

Exercice 2 – *QuickSort*

On rappelle le principe général de l'algorithme de tri *QuickSort*. On choisit tout d'abord un pivot (par exemple le premier élément du tableau) et on le place à sa place définitive. Pour cela on partitionne le tableau de sorte que les éléments inférieurs au pivot soient à sa « gauche » et que les éléments supérieurs au pivot soient à sa « droite » . Pour chacun des deux sous-tableaux à gauche et à droite du pivot, on procède récursivement, jusqu'à ce que l'ensemble des éléments soit trié.

1. Récupérer le code implémentant l'algorithme *QuickSort* en séquentiel, et le paralléliser avec OpenMP en vérifiant la justesse du calcul parallèle.
2. Comparer les performances obtenues avec la version séquentielle, par exemple pour 2^{27} éléments.
   Comment améliorer les performances obtenues en parallèle ?

