TP 1
Multithreading
Multiprocessing
Numba

Master 2 SID
Benoist GASTON
benoist.gaston@univ-rouen.fr

Introduction

- Le code est disponible sur le github du cours : https://github.com/gastoben/S3UE2_HPC/tree/main/TPs/TP01
- La fonction `factor_01(n)` de `TP01_01_factor.py` contient une construction naïve par compréhension de la liste des différents facteurs entiers de n strictement inférieur à `n`.
- La fonction `main(a, b)` de `TP01_01_factor.py` construit par compréhension la liste de la somme de tous les facteurs de chaque nombre entiers `n` compris entre `a` et `b` donnés en arguments du script.
- Le TP consiste à améliorer les performance du code à l'aide de Numba et à le paralléliser par multithreading et multiprocessing à l'aide du package `concurrent.futures`.

```
n=1  - factors :[] sum: 0
n=2  - factors :[1] sum: 1
n=3  - factors :[1] sum: 1
n=4  - factors :[1, 2] sum: 3
n=5  - factors :[1] sum: 1
n=6  - factors :[1, 2, 3] sum: 6
n=7  - factors :[1] sum: 1
n=8  - factors :[1, 2, 4] sum: 7
n=9  - factors :[1, 3] sum: 4
n=10 - factors :[1, 2, 5] sum: 8
```

factors from 1 to 10

Performance et Numba

Questions Préliminaires

- Nous allons prendre en main le code et estimer ses performances initiales en mononoeud
- **Questions**
    1. Exécuter le code avec différentes valeurs de `a` et `b`.
    2. Profiler le code à l'aide de `cProfile`, `%time`

Numba

- Nous allons tester la compilation du code avec Numba. Pour le moment nous ne désactiverons pas le GIL.
- **Questions**
    1. Ajouter le décorateur `@jit` à la déclaration de la fonction `factor01`. Quel impact sur les performances ?
    2. Ajouter le décorateur `@jit` à la déclaration de la fonction `main`. Quel impact sur les performances ?
    3. Ajouter l'option `nopython = True` au décorateur `@jit` (ou utiliser le décorateur `@njit`). Quel impact sur les performances ?

Multitasks

Multiprocessing

Pour cette partie, nous utiliserons la classe `ProcessPoolExecutor` de `concurrent.futures`

Questions

1. Retirer le décorateur `@jit` de la fonction `main`.

2. Réécrire la fonction main afin d'utiliser la fonction `map` native de Python.

3. Quel impact sur les performances ?

4. Adapter le code afin de dispatcher les tâches effectuées par le `map` entre les différents processus d'un pool de `n` processus, `n` donné en argument de la fonction `main`.

5. quel impact sur les performances ?

Multithreading

Pour cette partie, vous utiliserons la classe `ThreadPoolExecutor` de `concurrent.futures`

Questions

1. Adapter le code multiprocess afin de d'utiliser `n` threads plutôt que n processus

2. quel impact sur les performances ?

3. Utiliser l'option `nogil = True` du décorateur `@jit`.

4. Quel impact sur les performances ?