td2a_cenonce_session_2D_parallelisation_local

April 14, 2021

1 2A.i - Parallélisation locale (énoncé)

Parallélisation avec joblib.

[1]:
```
from jyquickhelper import add_notebook_menu
add_notebook_menu()
```

[1]:
```
<IPython.core.display.HTML object>
```

La parallélisation des calculs permet d'accélérer l'exécution d'un programme au prix de deux contraintes :

- La synchronisation : écrire un programme avec des calculs en parallèle est plus complexe. Il faut souvent attendre que des processus s'exécutant en parallèle se terminent avant de passer à l'étape suivante avec l'ensemble des résultats.
- La communication : il faut communiquer aux différents processus d'exécutant en parallèle les tâches qu'ils doivent exécuter. Selon la parallélisation choisie, le coût n'est pas négligeable (communication réseau par exemple).

La parallélisation est assez récente. Elle est née avec les premiers processeurs multitâche et les premiers systèmes d'exploitation avec des interfaces graphiques.

- thread : async, await, queue, threading
- processus : concurrent.futures, multiprocessing, joblib
- cluster

1.1 joblib

[2]:
```
from math import sqrt
from joblib import Parallel, delayed
%timeit Parallel(n_jobs=2)(delayed(sqrt)(i ** 2) for i in range(100000))
```

```
1 loop, best of 3: 2.23 s per loop
```

[3]:
```
%timeit [sqrt(i ** 2) for i in range(100000)]
```

```
10 loops, best of 3: 70.7 ms per loop
```

[4]:
```

```