Universidad Nacional Autónoma de México

Instituto de Investigaciones en Matemáticas Aplicadas y Sistemas

Resumen procesos e hilos en Python

Autor:
Vázquez Rojas José David
Hugo Carlos Moran P.
Luis Fernando Flores T.
Jose Marcos Yañez E.

Octubre 2020

Por definición un proceso es una ejecución de un programa, con un fin definido y que necesitan recursos para ejecutarse correctamente (CPU,memoria…). Por el contario, un hilo (thread) es simplemente una tarea que puede ser ejecutada al mismo tiempo que otra tarea. También llamados: Actividades concurrentes, Procesos ligeros ó Contextos de ejecución.

En Python, nos ofrecen herramientas preparadas para lograr manipular operaciones concurrentes con procesos e hilos en ejecución. Para ello, existen módulos en Python para aprovechar técnicas ya implementadas .

Módulo **threading**- Incluye una interfaz de alto nivel orientada a objetos para trabajar con concurrencia desde Python. Los objetos Thread se ejecutan al mismo tiempo dentro del mismo proceso y comparten memoria. Usando hilos es una forma sencilla de escalar para tareas que son más enlazadas a la E/S que al CPU.

Módulo **multiprocessing -** Refleja threading, excepto que en lugar de una clase Thread proporciona un Process. Cada Process es un verdadero proceso del sistema sin memoria compartida, pero multiprocessing proporciona funciones para compartir datos y pasar mensajes entre ellos de manera que en muchos casos de conversión de hilos a procesos es tan simple como cambiar unas pocas declaraciones.

La librería **multiprocessing** no sólo se encarga de hacer “forks”, su habilidad más interesante es la de facilitar la gestión de estos procesos, para ello provee de los mecanismos necesarios para que todos los procesos puedan comunicarse entre si, puedan ponerse en sincronía e incluso gestionar cola comunes a todos ellos de una forma fiable y segura.Igualmente y si se busca resolver una tarea que no requiere trabajadores demasiado inteligentes o autónomos, pone a disposición del programador una piscina o Pool de procesos. A continuación, un ejemplo:

```
from multiprocessing import Pool // Se importa el modulo Pool
de multiprocessing

def f(x): // Se crea la función que devolvera el cuadrado del
paraemtro
    return x*x

if __name__ == '__main__':
    with Pool(5) as p:// se crea la piscina de procesos como p
        print(p.map(f, [1, 2, 3]))//para cada elemento,
realiza la funcion f respectivamente
```

```
[1, 4, 9]
```

En este ejemplo se busca resolver un problema mediante la técnica del “divide y vencerás“, separando el trabajo a hacer entre varios procesos para que entre todos consigan terminar lo antes posible porque cada parte de la solución se resolvería en paralelo con las demás.

Otro ejemplo es creando un objeto proceso usando Process como a continuación:

```
from multiprocessing import Process //Se importa el modulo
process

def f(name)://function que imprimira el nombre
    print('hello', name)

if __name__ == '__main__':
    p = Process(target=f, args=('bob',))// creamos el proceso
que realizará la función f y como argumento el nombre
    p.start()//iniciamos el proceso
    p.join()//espera hasta que el proceso hijo termine
```