

An Introduction to Scaling Distributed Python Applications



Centro Universitario de Ciencias Exactas e Ingenierías

Asignatura: **Computación tolerante a fallas**

Sección: D06

An Introduction to Scaling Distributed Python Applications

Alumno: Luis Jaime Portillo Correa

Código: 217546155

Profesor: **Michel Emanuel López Franco**

Fecha: **08/09/2023**

Objetivo

(Application checkpointing) Generar un programa utilizando hilos, procesos, demonios y concurrencia.

Desarrollo

Para el Desarrollo de esta práctica decidí realizar un programa que utilizará multihilos para ejecutar diferentes funciones de manera concurrente, las funciones que ejecutaré tratarán de descargar archivos de youtube; tanto video como música, para de esta forma generar una carpeta con los archivos e irlos descargando a la vez.

```
# Función 1: Imprimir números del 1 al 5
def descargarVideo():
    url = 'https://www.youtube.com/watch?v=kinQsc6y9Kw'
    # Crear un objeto YouTube
    yt = YouTube(url)

    # Seleccionar la mejor resolución disponible
    video = yt.streams.get_highest_resolution()
    #video.download(output_path='/ruta/de/destino/') DIRECTORIO DEFINIDO

    # Descargar el video
    video.download(filename="video.mp4")

    print("Descarga completada.")

def descargarCancion():
    url = 'https://www.youtube.com/watch?v=L0ZJugltLRs'
    # Crear una instancia de la clase YouTube
    yt = YouTube(url)

    # Seleccionar la mejor resolución disponible (en este caso, el audio)
    stream = yt.streams.filter(only_audio=True).first()

    # Descargar el video
    stream.download(filename="cancion.mp3")

    # Renombra el archivo a una extensión .mp3

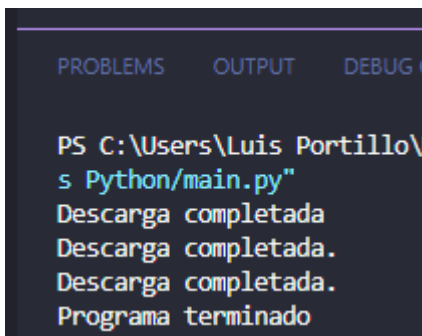
    print("Descarga completada")
```

En primera instancia, creamos las funciones encargadas de descargar archivos multimedia, en este caso la función 1 descargará el video de youtube albergado en el link que le indiquemos con ayuda de la librería Pytube de Python.

La segunda función realizará lo mismo pero sólo descargará el sonido del video en un archivo mp3, estas dos funciones y otra de vídeo se ejecutarán de manera simultanea por medio de hilos, de la siguiente manera:

```
55
56 # Crear hilos para cada función
57 hilo1 = threading.Thread(target=descargarVideo)
58 hilo2 = threading.Thread(target=descargarCancion)
59 hilo3 = threading.Thread(target=descargarVideo2)
60
61 # Iniciar los hilos
62 hilo1.start()
63 hilo2.start()
64 hilo3.start()
65
66 # Esperar a que todos los hilos terminen
67 hilo1.join()
68 hilo2.join()
69 hilo3.join()
70
71 print('Programa terminado')
```

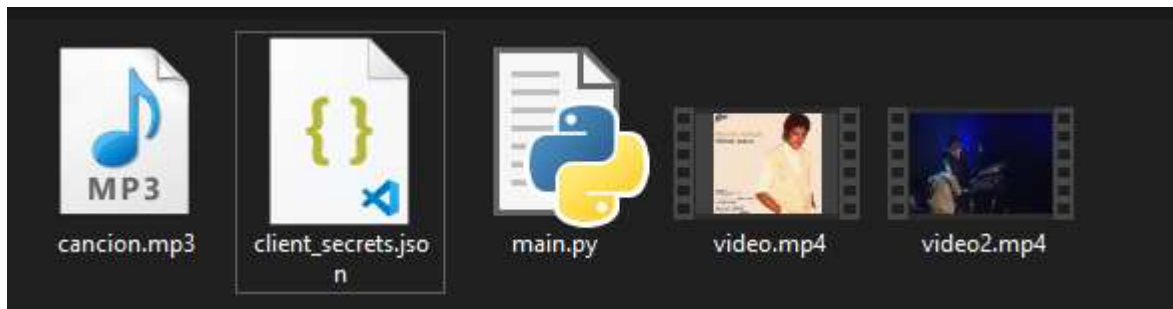
Creamos los hilos para cada función con ayuda de la librería threading, posteriormente los ejecutamos y estos correrán de manera concurrente y finalmente esperaremos a que los tres terminen para dar el programa por terminado.



```
PROBLEMS OUTPUT DEBUG CONSOLE
PS C:\Users\Luis Portillo\> python Python/main.py
Descarga completada
Descarga completada.
Descarga completada.
Programa terminado
```

El programa dedica tiempo a cada hilo de manera equitativa, de manera que todos se ejecutan de manera alternada y al terminar, el programa lo indica.

Así es como se ve la carpeta de descargas generada por la app.



Conclusión

A lo largo de estas semanas del curso he podido explorar aspectos básicos de la tolerancia a fallas y cada tema me parece más interesante para complementar un sistema robusto; este tema no fue la excepción.

En esta ocasión pude adentrarme en conceptos de programación multiprocesos, es decir, no ejecutar las funciones y elementos de manera estructurada, si no de manera simultanea o concurrente en el procesador. Considero que este tipo de herramientas fortalecen la tolerancia a fallas pues se evita sobrecarga en el sistema al trabajar con los distintos núcleos del CPU y podemos garantizar un mejor rendimiento.

Bibliografía

- <https://github.com/LujaMX/Tolerancia-a-fallas/blob/main/Multihilos%20Python/main.py>