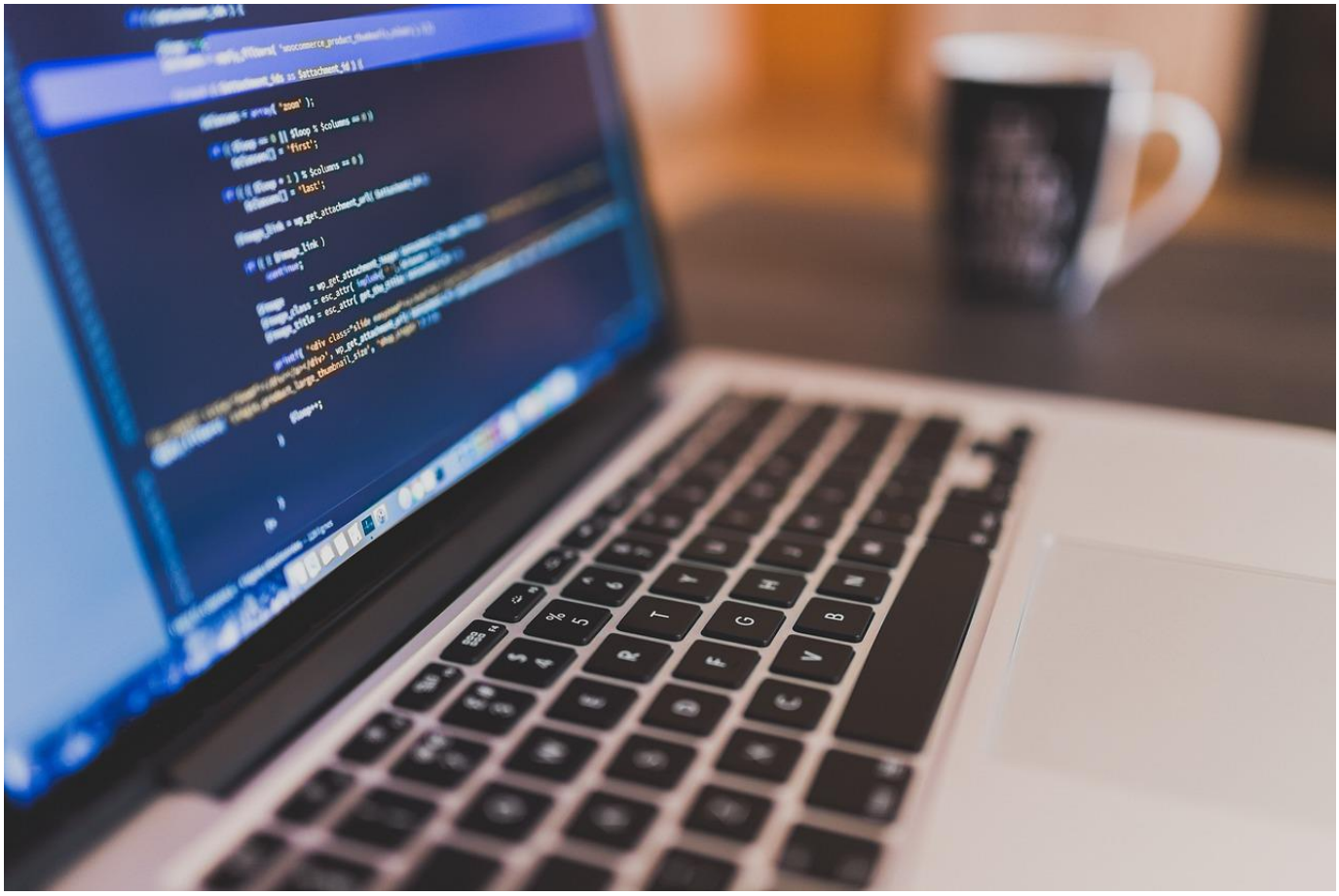
An Introduction to Scaling Distributed Python Applications

COMPUTACIÓN TOLERANTE A FALLAS – D06 – 2024A

MICHEL EMANUEL LOPEZ FRANCO

Navarro Velazquez Andres – 218127792

Fecha: 19/02/2024



Este programa es sencillo pero muestra claramente como crea múltiples unidades de ejecución (hilos, procesos y tareas de concurrencia) para simular diferentes formas de programación concurrente y muestra cómo iniciar, ejecutar y manejar estas unidades de manera simultánea en Python.

La diferencia entre cada uno de esos puede ser confusa pero a grandes rasgos es así:

Hilos (threads):

- Un hilo es la unidad más pequeña de procesamiento que puede ser ejecutada por un sistema operativo.
- Los hilos comparten el mismo espacio de memoria y recursos de su proceso padre.
- Los hilos permiten la ejecución concurrente dentro de un mismo proceso y pueden ser utilizados para realizar múltiples tareas simultáneamente.

Procesos:

- Un proceso es un programa en ejecución. Cada proceso tiene su propio espacio de memoria, recursos y estado de ejecución.
- Los procesos son independientes entre sí y no comparten recursos de memoria, lo que proporciona un mayor nivel de aislamiento.
- Los procesos son útiles para ejecutar tareas independientes que pueden requerir recursos diferentes o pueden no compartir datos entre sí.

Demonios:

- Un demonio es un tipo especial de proceso que se ejecuta en segundo plano sin interacción directa con el usuario.
- Los demonios a menudo realizan tareas de mantenimiento o servicios que no requieren interacción directa con los usuarios.
- Los demonios suelen iniciar su ejecución durante el inicio del sistema y se ejecutan continuamente hasta que el sistema se apaga o el demonio es detenido explícitamente.

Concurrencia:

- La concurrencia se refiere a la capacidad de un sistema para manejar múltiples tareas simultáneamente.
- La concurrencia no implica necesariamente que las tareas se ejecuten al mismo tiempo, sino que pueden avanzar de manera simultánea.
- Los hilos, los procesos y otras técnicas de programación como la programación asincrónica son utilizadas para lograr la concurrencia en los programas.

```python
import threading
import multiprocessing
import concurrent.futures
import time

# Función para ejecutar en un hilo
def thread_function(name):
    print(f'Hilo: {name} iniciado.')
    time.sleep(2)
    print(f'Hilo: {name} finalizado.')

# Función para ejecutar en un proceso
def process_function(name):
    print(f'Proceso: {name} iniciado.')
    time.sleep(2)
    print(f'Proceso: {name} finalizado.')

if __name__ == "__main__":
    # Crear un hilo
    thread = threading.Thread(target=thread_function, args=('Hilo 1',))
    thread.start()

    # Crear un proceso
    process = multiprocessing.Process(target=process_function,
args=('Proceso 1',))
    process.start()

    # Crear un demonio
    daemon_thread = threading.Thread(target=thread_function, args=('Demonio
1',))
    daemon_thread.daemon = True
    daemon_thread.start()

    # Utilizar concurrent.futures para concurrencia
    with concurrent.futures.ThreadPoolExecutor() as executor:
        futures = []
        for i in range(3):
            futures.append(executor.submit(thread_function, f'Concurrencia
{i}'))

        for future in concurrent.futures.as_completed(futures):
            print(f'Concurrencia: {future.result()} finalizada.')
```

programas > Scaling Distributed.py > ...

```python
import threading
import multiprocessing
import concurrent.futures
import time

# Función para ejecutar en un hilo
def thread_function(name):
    print(f'Hilo: {name} iniciado.')
    time.sleep(2)
    print(f'Hilo: {name} finalizado.')

# Función para ejecutar en un proceso
def process_function(name):
    print(f'Proceso: {name} iniciado.')
    time.sleep(2)
    print(f'Proceso: {name} finalizado.')

if __name__ == "__main__":
    # Crear un hilo
    thread = threading.Thread(target=thread_function, args=('Hilo 1',))
    thread.start()

    # Crear un proceso
    process = multiprocessing.Process(target=process_function, args=('Proces
    process.start()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python: Scaling Distributed

```
PS C:\Users\Andre\OneDrive\Escritorio\Escuela\Seminario de Sistemas operativos\programas> & C:/Users/Andre/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/Andre/OneDrive/Escritorio/Escuela/Seminario de Sistemas operativos/programas/Scaling Distributed.py"
Hilo: Hilo 1 iniciado.
Hilo: Demonio 1 iniciado.
Hilo: Concurrencia 0 iniciado.
Hilo: Concurrencia 1 iniciado.
Hilo: Concurrencia 2 iniciado.
Proceso: Proceso 1 iniciado.
Hilo: Hilo 1 finalizado.
Hilo: Demonio 1 finalizado.
Hilo: Concurrencia 2 finalizado.
Hilo: Concurrencia 1 finalizado.
Hilo: Concurrencia 0 finalizado.
Concurrencia: None finalizada.
Concurrencia: None finalizada.
Concurrencia: None finalizada.
Proceso: Proceso 1 finalizado.
PS C:\Users\Andre\OneDrive\Escritorio\Escuela\Seminario de Sistemas operativos\programas>
```

Conclusión:

En conclusión pude darme cuenta entre la diferencia entre los hilos, procesos, demonios y concurrencia, mientras que los hilos y los procesos son unidades de ejecución concurrente, los demonios son procesos específicos que se ejecutan en segundo plano y la concurrencia es el concepto general que abarca la ejecución simultánea o concurrente de múltiples tareas. Todos conforman las actividades que realiza un sistema operativo para poder funcionar correctamente, esto gracias a los avances de poder realizar varias tareas simultáneamente, en cambio eso no siempre fue así antes solo se podía realizar un proceso a la vez con pocos hilos o ninguno, y funcionaban secuencialmente, quiere decir que empieza un proceso, lo termina y empieza con otro, esto me muestra la gran diferencia de avances que tenemos hoy en día

Bibliografía:

Dixit, P., & Dixit, P. (2023, 3 mayo). Concurrency in Python: Threading, Processes, and Asyncio - StatusNeo. *StatusNeo - Cloud Native Technology Services & Consulting*. https://statusneo.com/concurrency-in-python-threading-processes-and-asyncio/

Educative. (s. f.). *An Introduction to Scaling Distributed Python Applications*. https://www.educative.io/blog/scaling-in-python