Mariana Osorio Rojas - 464679
Fabio Andrés Guzman Figueroa
Universidad Pontifica Bolivariana - Facultad de Ingeniería
Ingeniería de Sistemas e Informática - Sistemas Operativos

PRÁCTICA N9 - MULTI PROCESSING WITH PYTHON

En el mundo de la programación, la eficiencia y el rendimiento son aspectos clave para lograr aplicaciones rápidas y efectivas. Una forma de mejorar estos aspectos es mediante el uso del multiprocesamiento, una técnica que permite ejecutar múltiples procesos de forma simultánea, aprovechando así los múltiples núcleos de las CPU modernas.

En Python, la biblioteca multiprocessing facilita la implementación de multiprocesamiento, lo que puede resultar útil en diversas situaciones, como el procesamiento de grandes conjuntos de datos, la paralelización de tareas intensivas en CPU o la creación de aplicaciones más responsivas.

|Descripción del Problema|

En esta práctica, se busca explorar y comprender el uso del multiprocesamiento en Python. Para ello, se plantea un ejercicio práctico en el que deberán:

- Familiarizarse con la biblioteca multiprocessing.
- Comparar la ejecución serial y paralela de una función simple para comprender las diferencias de rendimiento.
- Utilizar colas (Queue) en multiprocesamiento para coordinar la ejecución de funciones en paralelo.
- Experimentar con la creación y ejecución de sus propias funciones en paralelo, como elevar al cubo un número o sumar uno a un número.

|Código Fuente|

```python
from multiprocessing import Queue, Process

def cube(numbers, queue):
    for i in numbers:
        queue.put(i*i*i)

def add_one(numbers, queue):
    for i in numbers:
        queue.put(i + 1)

q = Queue()
numbers = range(1, 8)

p1 = Process(target=cube, args=(numbers, q))
p2 = Process(target=add_one, args=(numbers, q))

p1.start()
p2.start()

p1.join()
p2.join()

while not q.empty():
    print(q.get())
```

```python
from multiprocessing import Queue, Process

def square(numbers, queue):
    for i in numbers:
        queue.put(i*i)

def make_negative(numbers, queue):
    for i in numbers:
        queue.put(-1*i)

q = Queue()
numbers = range(1,5)
```

```python
p1 = Process(target=square, args=(numbers, q))
p2 = Process(target=make_negative, args=(numbers, q))

p1.start()
p2.start()

p1.join()
p2.join()

while not q.empty():
    print(q.get())
```

|Captura de Ejecución|