Arellano Granados Angel Mariano

218123444

Computación Tolerante a Fallas

D06 2023B

An Introduction to Scaling Distributed Python Applications

Introducción

La implementación de hilos en un programa de computo es una herramienta que nosotros como programadores podemos implementar para una infinidad de tareas, en este documento se explorara el uso de los hilos para crear un programa tolerante a fallas en Python.

Desarrollo

Para esta actividad se creo un programa en Python el cual recibe por consola los datos de una matriz de 4x4 la cual se busca multiplicar por dos, para esto la matriz de parte en dos matrices de 4x2 y cada una de estas se trata de resolver en un hilo.

La idea es que si la matriz ingresada contiene un dato erróneo no se pierda el 100% de los datos por un error de ejecución, sino que solo se pierda la mitad, sin perder rendimiento en el proceso. Este programa se podría escalar para perder cada vez menos datos si solo una pequeña parte de los datos es errónea.

Código:

```
import threading
import numpy as np

# Solicitar al usuario que ingrese los valores de la matriz
print("Ingrese los valores de la matriz 4x4:")
matriz = []
for i in range(4):
    fila = []
    for j in range(4):
        valor = input(f"Valor en la fila {i+1}, columna {j+1}: ")
        fila.append(valor)
    matriz.append(fila)

# Valor por el cual multiplicar la matriz (ejemplo: 2)
valor_multiplicacion = 2

# Matriz resultante (inicialmente llena de ceros)
matriz_resultante = np.zeros_like(matriz)

# Función para multiplicar una porción de la matriz
def multiplicar_por_valor(inicio_fila, fin_fila):
    try:
        for i in range(inicio_fila, fin_fila):
            for j in range(4):
                matriz_resultante[i][j] = int(matriz[i][j]) *
valor_multiplicacion
    except:
        print('Uno de los modulos a fallado')

# Crear dos hilos para multiplicar por valor en paralelo
thread1 = threading.Thread(target=multiplicar_por_valor, args=(0, 2))
thread2 = threading.Thread(target=multiplicar_por_valor, args=(2, 4))

# Iniciar los hilos
thread1.start()
thread2.start()

# Esperar a que todos los hilos terminen
thread1.join()
thread2.join()

# Imprimir la matriz original y la matriz resultante
print("\nMatriz original:")
```

```
for fila in matriz:
    print(fila)
print("\nMatriz resultante:")
for fila in matriz_resultante:
    print(fila)
```

Ejecución correcta:

```
Ingrese los valores de la matriz 4x4:
Valor en la fila 1, columna 1: 1
Valor en la fila 1, columna 2: 34
Valor en la fila 1, columna 3: 2
Valor en la fila 1, columna 4: 13
Valor en la fila 2, columna 1: 24
Valor en la fila 2, columna 2: 21
Valor en la fila 2, columna 3: 5
Valor en la fila 2, columna 4: 123
Valor en la fila 3, columna 1: 4
Valor en la fila 3, columna 2: 12
Valor en la fila 3, columna 3: 5
Valor en la fila 3, columna 4: 13
Valor en la fila 4, columna 1: 15
Valor en la fila 4, columna 2: 15
Valor en la fila 4, columna 3: 4
Valor en la fila 4, columna 4: 35

Matriz original:
['1', '34', '2', '13']
['24', '21', '5', '123']
['4', '12', '5', '13']
['15', '15', '4', '35']

Matriz resultante:
['2' '68' '4' '26']
['48' '42' '10' '246']
['8' '24' '10' '26']
['30' '30' '8' '70']
```

Ejecución errónea:

En este caso en vez de ingresar un numero ingresaremos una letra en la primera mitad de la matriz.

```
Ingrese los valores de la matriz 4x4:
Valor en la fila 1, columna 1: 12
Valor en la fila 1, columna 2: 56
Valor en la fila 1, columna 3: 32
Valor en la fila 1, columna 4: aa
Valor en la fila 2, columna 1: 78
Valor en la fila 2, columna 2: 12
Valor en la fila 2, columna 3: 356
Valor en la fila 2, columna 4: 14
Valor en la fila 3, columna 1: 54
Valor en la fila 3, columna 2: 63
Valor en la fila 3, columna 3: 26
Valor en la fila 3, columna 4: 45
Valor en la fila 4, columna 1: 81
Valor en la fila 4, columna 2: 19
Valor en la fila 4, columna 3: 26
Valor en la fila 4, columna 4: 35
Uno de los modulos a fallado

Matriz original:
['12', '56', '32', 'aa']
['78', '12', '356', '14']
['54', '63', '26', '45']
['81', '19', '26', '35']

Matriz resultante:
['24' '112' '64' '']
['' '' '' '']
['108' '126' '52' '90']
['162' '38' '52' '70']
```

El programa solo pierde 5 datos en vez de 16 en un programa normal.

Conclusión

Los hilos pueden simplificar o complicar el funcionamiento de un programa, no es obligatorio usarlos en todos nuestros programas, pero si son una herramienta que debemos de conocer para explotar a nuestro favor, pues usar paralelismos en un sistema nos asegura que usaremos el 100% de

nuestro procesador evitando los tiempos muertos mientras el programa espera una entrada.