

Universidad Nacional del Altiplano
Docente: Fred Torres Cruz
Autor : Henry Angel Pacco Zuvietta

Trabajo Encargado - N° 02

Introducción

En este documento se presenta un programa en Python que utiliza procesamiento paralelo para sumar dos arrays de manera eficiente. El código realiza una comparación entre la suma ordinaria y la suma utilizando múltiples procesos.

Código en Python

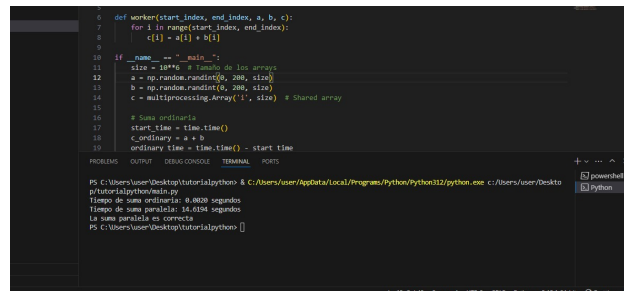
A continuación se muestra el código en Python:

```
1 import multiprocessing
2 import numpy as np
3 import time
4
5 def worker(start_index, end_index, a, b, c):
6     for i in range(start_index, end_index):
7         c[i] = a[i] + b[i]
8
9 if __name__ == "__main__":
10     size = 10**6 # Tamaño de los arrays
11     a = np.random.randint(0, 200, size)
12     b = np.random.randint(0, 200, size)
13     c = multiprocessing.Array('i', size) # Shared array
14
15     # Suma ordinaria
16     start_time = time.time()
17     c_ordinary = a + b
18     ordinary_time = time.time() - start_time
19
20     # Suma paralela
21     processes = []
22     chunk_size = size // multiprocessing.cpu_count()
23
24     start_time = time.time()
25     for i in range(multiprocessing.cpu_count()):
26         start_index = i * chunk_size
27         end_index = size if i == multiprocessing.cpu_count() - 1 else (i +
28             1) * chunk_size
29         process = multiprocessing.Process(target=worker, args=(start_index
30             , end_index, a, b, c))
31         processes.append(process)
32         process.start()
```

```
32     for process in processes:
33         process.join()
34     parallel_time = time.time() - start_time
35
36     # Imprimir tiempos
37     print(f"Tiempo de suma ordinaria: {ordinary_time:.4f} segundos")
38     print(f"Tiempo de suma paralela: {parallel_time:.4f} segundos")
39
40     # Verificación de resultados (opcional)
41     if np.array_equal(c_ordinary, c[:]):
42         print("La suma paralela es correcta")
43     else:
44         print("La suma paralela es incorrecta")
```

Listing 1: Código para suma paralela en Python

resultado de la ejecución del código :



```
0 def worker(start_index, end_index, a, b, c):
1     for i in range(start_index, end_index):
2         c[i] = a[i] + b[i]
3
4
5
6 if __name__ == "__main__":
7     size = 10**6 # tamaño de los arrays
8     a = np.random.randint(0, 200, size)
9     b = np.random.randint(0, 200, size)
10    c = multiprocessing.Array('i', size) # Shared array
11
12    # Suma ordinaria
13    start_time = time.time()
14    c_ordinary = a + b
15    ordinary_time = time.time() - start_time
16
17    # Suma paralela
18    start_time = time.time()
19    p = multiprocessing.Pool(4)
20    p.map(worker, [(start, end, a, b, c)] * 4)
21    parallel_time = time.time() - start_time
22
23    print(f"Tiempo de suma ordinaria: {ordinary_time:.4f} segundos")
24    print(f"Tiempo de suma paralela: {parallel_time:.4f} segundos")
25
26    # Verificación de resultados (opcional)
27    if np.array_equal(c_ordinary, c[:]):
28        print("La suma paralela es correcta")
29    else:
30        print("La suma paralela es incorrecta")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\user\Desktop\tutorialpython> python.exe c:\Users\user\Desktop\tutorialpython\main.py

Tiempo de suma ordinaria: 0.8808 segundos
Tiempo de suma paralela: 14.4104 segundos
La suma paralela es correcta

Figure 1:

Resultados

El código mide y compara el tiempo de ejecución de la suma ordinaria y la suma paralela, verificando además la corrección del resultado paralelo.

Conclusión

El uso de procesamiento paralelo puede reducir significativamente el tiempo de ejecución para operaciones computacionales intensivas como la suma de grandes arrays.