

Universidad Nacional del Altiplano
Facultad de Ingeniería Estadística e Informática
Docente: Fred Torres Cruz
Autor: Cliver Wimar Vilca Tinta

Trabajo Encargado - N° 001

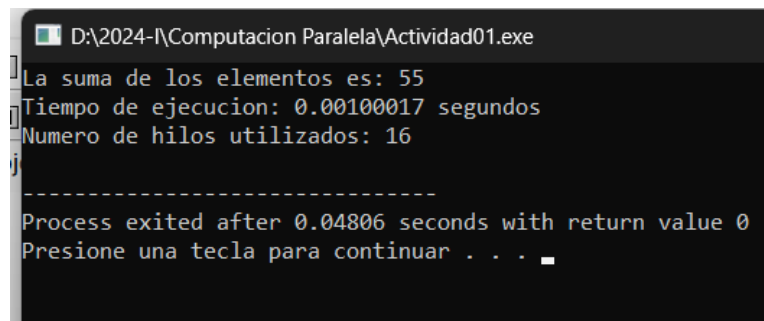
COMPUTACION PARALELA Y CONCURRENTES

Cuando intentamos acelerar nuestras tareas dividiendo el trabajo entre múltiples hilos, esperamos que el rendimiento mejore. Sin embargo, es importante tener en cuenta que trabajar con múltiples hilos implica tareas adicionales, como el manejo, la sincronización y la comunicación entre ellos. Estas tareas adicionales requieren tiempo y recursos del computador.

Código C++ Parallelism

```
#include <iostream>
#include <omp.h>
using namespace std;
int main() {
    int nums[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    int nums_size = sizeof(nums) / sizeof(nums[0]);
    int sum = 0;
    double start_time = omp_get_wtime();
    #pragma omp parallel for reduction(+:sum)
    for (int i = 0; i < nums_size; ++i) {
        sum += nums[i];
    }
    double end_time = omp_get_wtime();
    cout << "La suma de los elementos es: " << sum << endl;
    cout << "Tiempo de ejecucion: " << end_time - start_time << " segundos" << endl;
    int num_threads = omp_get_max_threads();
    cout << "Numero de hilos utilizados: " << num_threads << endl;
    return 0;
}
```

Salida:



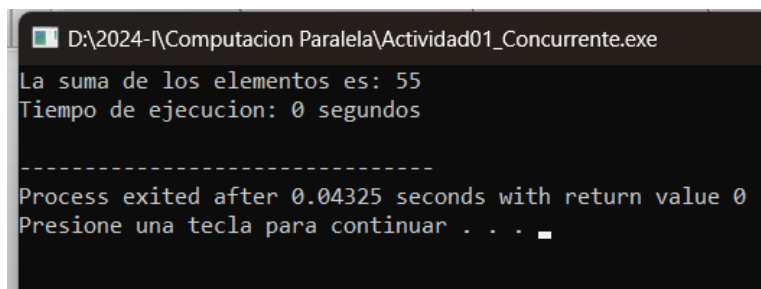
```
D:\2024-I\Computacion Paralela\Actividad01.exe
La suma de los elementos es: 55
Tiempo de ejecucion: 0.00100017 segundos
Numero de hilos utilizados: 16
-----
Process exited after 0.04806 seconds with return value 0
Presione una tecla para continuar . . .
```

Por otro lado, cuando hacemos cosas concurrentemente, significa que manejamos varias tareas a la vez, pero no necesariamente todas al mismo tiempo. Esto puede tener algunos retrasos porque tenemos que cambiar de tarea de vez en cuando y asegurarnos de que todas estén sincronizadas, pero no necesariamente es tan pesado como manejar múltiples hilos.

Código C++ Concurrency

```
#include <iostream>
#include <ctime> // Incluir la librería para medir el tiempo
using namespace std;
int main() {
    // Crear un arreglo de números
    int nums[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    int nums_size = sizeof(nums) / sizeof(nums[0]); // Tamaño del arreglo
    // Variable para almacenar la suma
    int sum = 0;
    // Iniciar temporizador
    clock_t tiempo_inicio = clock();
    // Calcular la suma de los elementos del arreglo de forma secuencial
    for (int i = 0; i < nums_size; ++i) {
        sum += nums[i];
    }
    // Detener temporizador
    clock_t tiempo_fin = clock();
    // Calcular la duración en segundos
    double duracion = (double)(tiempo_fin - tiempo_inicio) / CLOCKS_PER_SEC;
    // Imprimir la suma de los elementos
    cout << "La suma de los elementos es: " << sum << endl;
    // Imprimir el tiempo de ejecución en segundos
    cout << "Tiempo de ejecucion: " << duracion << " segundos" << endl;
    return 0;
}
```

Salida:

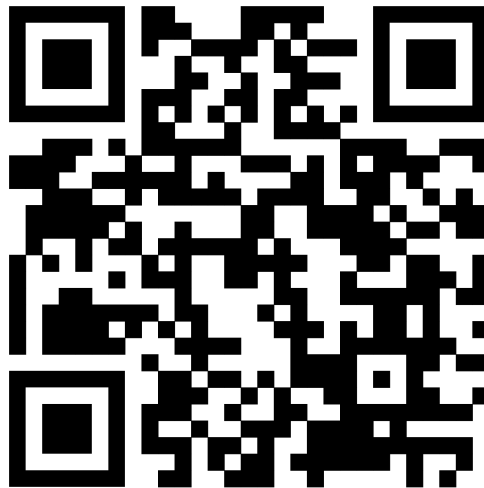


```
D:\2024-I\Computacion Paralela\Actividad01_Concurrente.exe
La suma de los elementos es: 55
Tiempo de ejecucion: 0 segundos

-----
Process exited after 0.04325 seconds with return value 0
Presione una tecla para continuar . . .
```

En conclusion, si bien esperamos que tanto el paralelismo como la concurrencia nos ayuden a hacer las cosas más rápido, a veces pueden causar más problemas de los que resuelven. Depende mucho del problema que estemos tratando de resolver, el hardware que estemos usando y cómo estemos escribiendo nuestro código. Lo importante es hacer pruebas para ver qué funciona mejor en cada caso específico. En nuestro caso, dado que el arreglo `nums` solo contiene 10 elementos y la tarea de suma es relativamente simple, es posible que la ejecución secuencial tenga un tiempo de ejecución más rápido en comparación con la ejecución paralela utilizando OpenMP. Esto se debe a que el overhead de paralelización y la sincronización entre los hilos pueden tener un impacto negativo en el rendimiento en este escenario específico.

Código QR Repositorio GutHub



Link Repositorio GutHub <https://github.com/CliverVilca/Parallel-Computing.git>