Tecnológico de Monterrey

Tarea 1. Programación de una solución paralela

Cómputo en la nube

Profesor: Dr. Eduardo Antonio Cendejas Castro

A00819828 - Yocelin Juarez Arroyo

Introducción:

La paralelización es la ejecución paralela de tareas en un sistema con múltiples núcleos de procesador. Por ejemplo, en una solución tradicional, si tenemos un programa que debe realizar diferentes cálculos en una gran cantidad de datos. Estos cálculos se realizan secuencialmente, uno después del otro, utilizando un solo núcleo de procesador.

Con la paralelización, se divide el conjunto de datos en partes más pequeñas y asignamos cada parte a un núcleo de procesador diferente. Cada núcleo realiza sus cálculos de manera independiente y simultánea, lo que acelera significativamente el tiempo total de procesamiento. Esto aprovecha la capacidad de procesamiento en paralelo de los múltiples núcleos del procesador.

Las ventajas son que mejora la eficiencia y acelera el procesamiento, por lo que es fundamental para enfrentar desafíos computacionales complejos.

Liga del repositorio del proyecto en Github:

[a00819828/Tarea1 (github.com)]

```
// Tarea 1. Programación de una solución paralela_A00819828.cpp : This file contains the 'main' function.
//

#include <iostream>

#include <omp.h>
```

- #include <iostream>: Importa la librería estándar de entrada/salida.
- #include <omp.h>: Importala librería OpenMP para programación paralela.

```
#define N 1000
#define chunk 100
#define mostrar 10
```

Declarar 3 constantes:

- #define N 1000: Define la cantidad de arreglos 1000.
- #define chunk 100: Cantidad de “pedazos” de datos que cada hilo de ejecución procesa en paralelo.

- #define mostrar 10: Define el número de elementos que se mostrarán al imprimir los arreglos.

```
void imprimeArreglo(float* d);
```

Imprimimos los primeros elementos del arreglo.
Iniciamos el código principal.

```
int main()
{
    std::cout << "Sumando arreglos en paralelo!\n";
    float a[N], b[N], c[N];
    int i;

    for (i = 0; i < N; i++)
    {
        a[i] = i * 10;
        b[i] = (i + 3) * 3.7;
    }
    int pedazos = chunk;

    #pragma omp parallel for \
    shared(a, b, c, pedazos) private(i) \
    schedule(static, pedazos)

    for (i = 0; i < N; i++)
        c[i] = a[i] + b[i];
```

Se inicializan los arreglos a y b.
Usamos **#pragma omp parallel for** para indicar que se ejecutarán en paralelo.
shared(a, b, c, pedazos): Indica que las variables **a, b, c y pedazos** son compartidas entre los hilos.
private(i): Indica que cada hilo tendrá su propia copia privada en **i**.
schedule(static, pedazos): Divide la tarea en partes iguales y asigna, el tamaño de cada parte es se determina por “pedazos”, posteriormente **for** realiza la suma de manera paralela, y el resultado se almacena en el arreglo c.

```
std::cout << "Imprimiendo los primero" << mostrar << " valores del arreglo a: " << std::endl;
imprimeArreglo(a);
std::cout << "Imprimiendo los primero" << mostrar << " valores del arreglo b: " << std::endl;
imprimeArreglo(b);
std::cout << "Imprimiendo los primero" << mostrar << " valores del arreglo c: " << std::endl;
imprimeArreglo(c);
```

Finalmente, se imprime una porción de los primeros elementos de los arreglos a, b, y c mediante la función **imprimeArreglo**.

Resultados:

El código realiza la suma de dos arreglos de manera paralela utilizando OpenMP y muestra los primeros elementos de los arreglos involucrados en la operación. La paralelización se logra distribuyendo la carga de trabajo entre varios hilos de ejecución para mejorar el rendimiento en sistemas con múltiples núcleos de CPU.

```
Sumando arreglos en paralelo!
Imprimiendo los primero10 valores del arreglo a:
0-10-20-30-40-50-60-70-80-90-
Imprimiendo los primero10 valores del arreglo b:
11.1-14.8-18.5-22.2-25.9-29.6-33.3-37-40.7-44.4-
Imprimiendo los primero10 valores del arreglo c:
11.1-24.8-38.5-52.2-65.9-79.6-93.3-107-120.7-134.4-
```

Reflexión sobre la programación paralela:

Como se menciona en la Guia, en este caso tenemos pocas tareas, sin embargo para problemas más complejos que requieren múltiples cálculos este tipo de soluciones agiliza los procesos, como ingeniera industrial, lo veo como la manufactura de un producto, en donde si tenemos una sola estación de trabajo la producción no es tan eficiente que en caso de tener múltiples estaciones que realicen tarea en simultáneo.