

Instituto Tecnológico y de Estudios Superiores de Monterrey

Maestría en Inteligencia Artificial Aplicada



Curso: Cómputo en la nube

Tarea 1. Programación de una solución paralela

Matthias Sibrian Illescas A01794249

Profesor: Dr. Eduardo Antonio Cendejas Castro

28 de enero 2024

Introducción

En el marco de la programación paralela, este trabajo se enfocó en diseñar e implementar algoritmos utilizando la librería OpenMP para abordar el desafío de la suma de arreglos. Con el objetivo de mejorar la eficiencia en la resolución de problemas numéricos, se realizó la ejecución simultánea de tareas independientes, aprovechando entornos de cómputo paralelo.

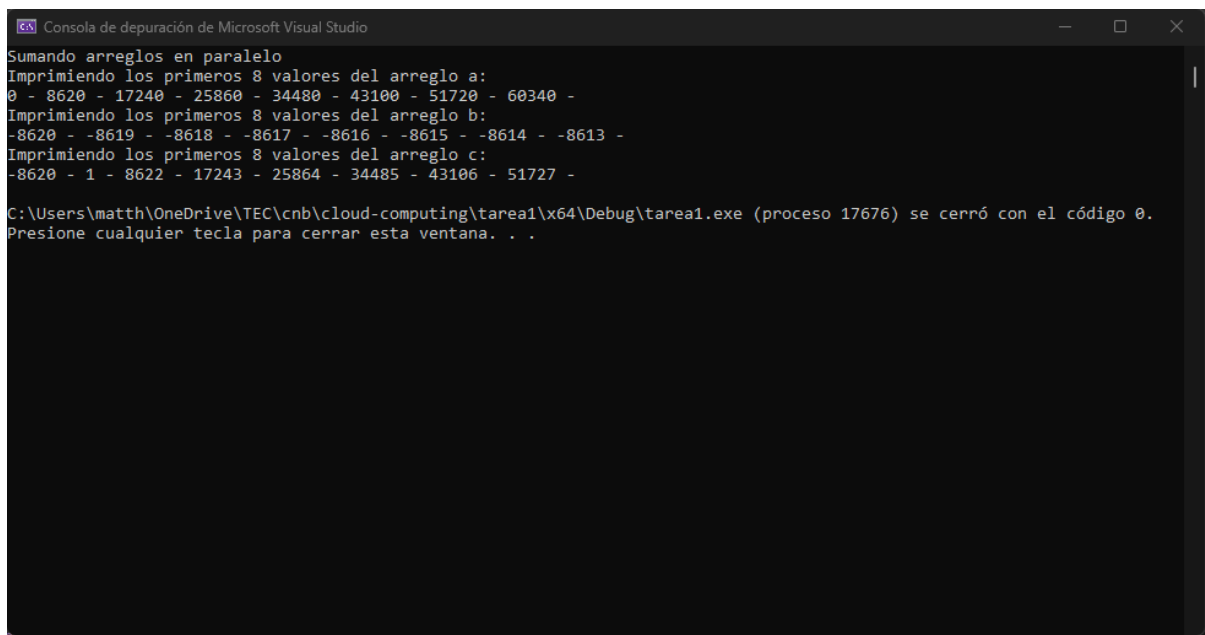
La tarea de suma de arreglos, aunque simple en ejecución secuencial, presenta ineficiencias con conjuntos de datos extensos. Para mitigar este problema, se hizo un enfoque paralelo mediante OpenMP. Para ello, se asignó valores aleatorios a dos arreglos y se usó el ciclo “for” en ejecuciones paralelas para realizar la suma de elementos correspondientes. Con esto se pudo ver la importancia que tiene un buen diseño de algoritmos paralelos para problemas numéricos.

Liga del repositorio de github donde se encuentra el proyecto desarrollado

<https://github.com/msibriani/cloud-computing>

Capturas de pantalla de al menos dos ejecuciones del proyecto

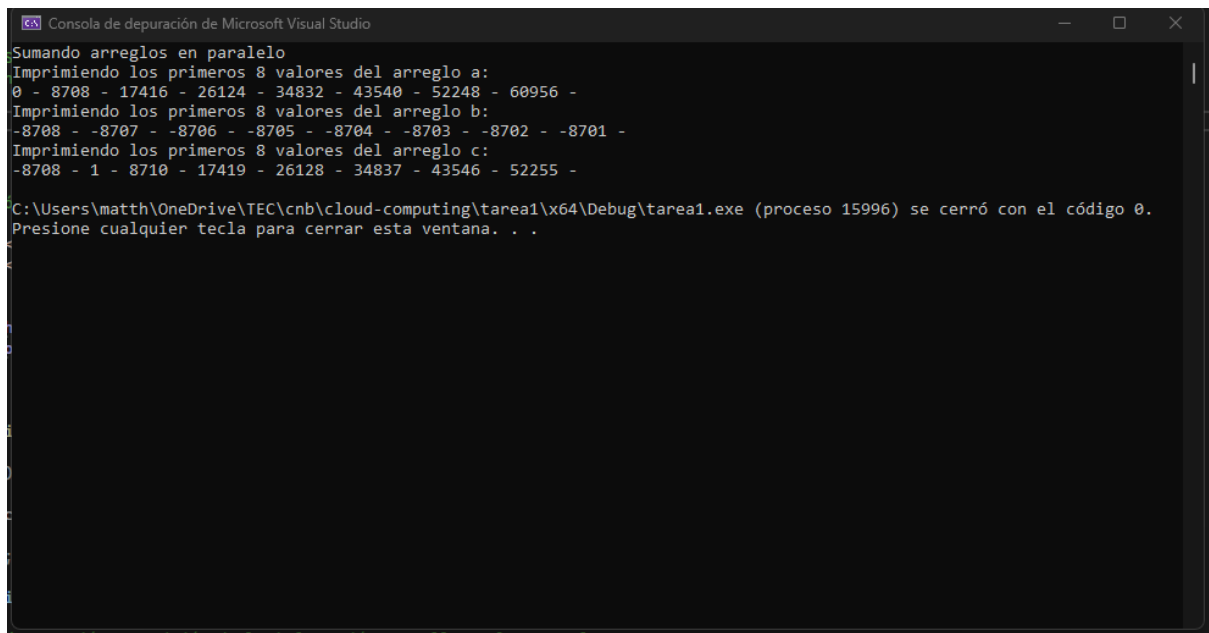
Ejecución 1



```
Consola de depuración de Microsoft Visual Studio
Sumando arreglos en paralelo
Imprimiendo los primeros 8 valores del arreglo a:
0 - 8620 - 17240 - 25860 - 34480 - 43100 - 51720 - 60340 -
Imprimiendo los primeros 8 valores del arreglo b:
-8620 - -8619 - -8618 - -8617 - -8616 - -8615 - -8614 - -8613 -
Imprimiendo los primeros 8 valores del arreglo c:
-8620 - 1 - 8622 - 17243 - 25864 - 34485 - 43106 - 51727 -

C:\Users\matth\OneDrive\TEC\cnb\cloud-computing\tarea1\x64\Debug\tarea1.exe (proceso 17676) se cerró con el código 0.
Presione cualquier tecla para cerrar esta ventana. . .
```

Ejecución 2



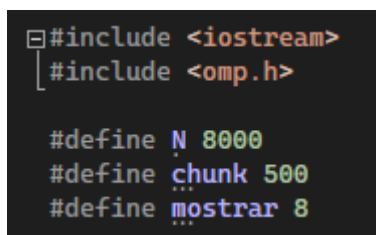
```
Consola de depuración de Microsoft Visual Studio

Sumando arreglos en paralelo
Imprimiendo los primeros 8 valores del arreglo a:
0 - 8708 - 17416 - 26124 - 34832 - 43540 - 52248 - 60956 -
Imprimiendo los primeros 8 valores del arreglo b:
-8708 - -8707 - -8706 - -8705 - -8704 - -8703 - -8702 - -8701 -
Imprimiendo los primeros 8 valores del arreglo c:
-8708 - 1 - 8710 - 17419 - 26128 - 34837 - 43546 - 52255 -

C:\Users\matth\OneDrive\TEC\cnb\cloud-computing\tarea1\x64\Debug\tarea1.exe (proceso 15996) se cerró con el código 0.
Presione cualquier tecla para cerrar esta ventana. . .
```

Explicación del código y los resultados

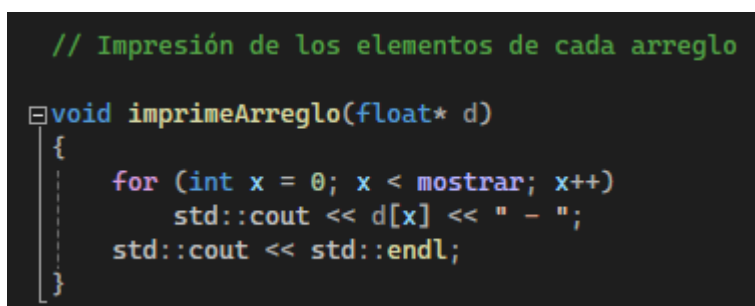
Acá se importa la librería estándar de in y out de C++, así como la librería OpenMP, la cual se definió sirve para el cómputo en paralelo.



```
#include <iostream>
#include <omp.h>

#define N 8000
#define chunk 500
#define mostrar 8
```

Esta función imprime los elementos tipo float de un array en la consola. Con la variable mostrar, se define cuántos se imprimen.



```
// Impresión de los elementos de cada arreglo

void imprimirArreglo(float* d)
{
    for (int x = 0; x < mostrar; x++)
        std::cout << d[x] << " - ";
    std::cout << std::endl;
}
```

Acá, en la función main, se inicializan tres arreglos (a, b, y c) de tamaño N y se llenan con valores influenciados por el índice y números aleatorios. Además, se establece la variable pedazos con el valor de chunk que se le pasa. El arreglo a es el índice del valor multiplicado por un número randomizado, mientras que el arreglo b es el índice del valor restado el valor randomizado.

```
void imprimeArreglo(float* d);

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

    for (i = 0; i < N; i++)
    {
        //Generación o petición de la información para llenar los arreglos

        /*
        * Para llenar los arreglos, se hizo uso de un cálculo a partir de un número random generado a través de una semilla.
        * Se están creando dos arreglos (a y b) donde los valores de cada elemento están influenciados tanto por el índice
        del elemento como por un número random. La semilla basada en el tiempo hace que sea random, y luego se logra crear
        el número con la función rand.
        */

        srand((unsigned)time(NULL));
        int random = rand();
        a[i] = i * random;
        b[i] = i - random;
    }

    int pedazos = chunk;
}
```

En esta parte se usa OpenMP para realizar la suma paralela de los elementos de dos arreglos (a y b), almacenando el resultado en un tercer arreglo (c). Se especifica la distribución estática de iteraciones entre hilos. Posteriormente, se imprimen porciones de los arreglos a, b y c usando la función que se definió.

```
//Uso correcto del for paralelo

#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];

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

En los resultados se ve que efectivamente, el valor resultante del arreglo c es la suma de los arreglos a y b element-wise. Es importante notar que porque el tipo de dato guardado es int, permite valores negativos, y las sumas resultan en el valor correcto.

Reflexión sobre la programación paralela

La programación paralela se presenta como una herramienta fundamental para mejorar el rendimiento de algoritmos frente a grandes conjuntos de datos. En esta actividad, se aborda específicamente la eficiencia en la suma de arreglos mediante la implementación de algoritmos paralelos con OpenMP.

Cuando se trata de operaciones con arreglos extensos, la ejecución secuencial se vuelve ineficiente debido a la cantidad de elementos y la velocidad del procesador. La programación paralela ofrece una solución al permitir la ejecución simultánea de tareas independientes.

A través de la configuración de un proyecto en Visual Studio con OpenMP y la implementación de un bucle paralelo, se maximiza el aprovechamiento de cómputo paralelo. La actividad no solo proporciona una solución eficiente para la suma de arreglos, sino que también fortalece habilidades prácticas en programación paralela, como la selección de modelos de paralelización y la creación de versiones paralelas correctas.