

Cómputo en la nube

Programación de una solución paralela

Emmanuel González Calitl
A01320739

Profesor tutor Cómputo en la nube
Alfredo Israel Ramírez Mejía

Profesor titular de Cómputo en la nube
Eduardo Antonio Cendejas Castro

Instituto Tecnológico y de Estudios Superiores de Monterrey

Fecha de entrega: *29 de enero del 2023*

Introducción:

Durante las últimas décadas los fabricantes de tecnología computacional hicieron pensar al público en general que el performance de una computadora se debe al Hardware que compone a una computadora, sin embargo, los últimos años se ha demostrado que el performance está definido por el Software y la calidad del mismo para poder remplazar la concurrencia por la manera de trabajar en paralelo.

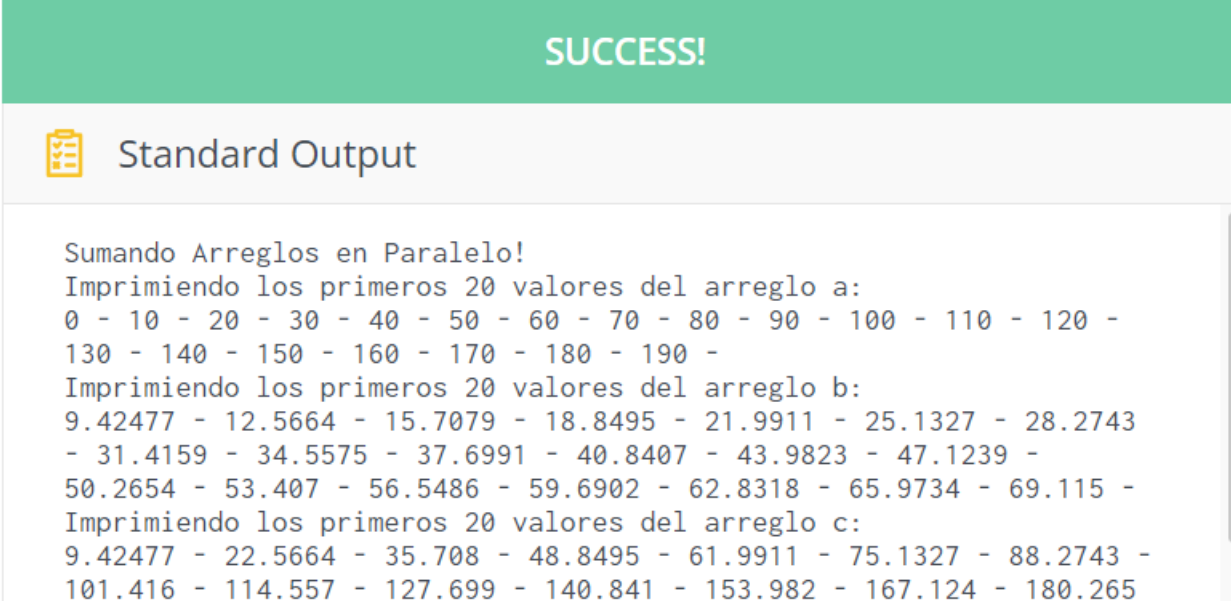
Dentro de esta práctica del curso se hace uso de esta técnica para poder realizar una suma en paralelo.

Liga del repositorio de github donde se encuentra el proyecto desarrollado:

<https://github.com/emm-gl/ComputoEnLaNube/tree/main/Tareas>

Capturas de pantalla de al menos dos ejecuciones del proyecto:


En las siguientes capturas de pantalla se muestran los resultados de dos ejecuciones con diferentes parámetros en el tamaño del arreglo y la generación del arreglo B. Podemos comprobar de manera manual que las sumas se realizaron de manera correcta:



```
Sumando Arreglos en Paralelo!  
Imprimiendo los primeros 20 valores del arreglo a:  
0 - 10 - 20 - 30 - 40 - 50 - 60 - 70 - 80 - 90 - 100 - 110 - 120 -  
130 - 140 - 150 - 160 - 170 - 180 - 190 -  
Imprimiendo los primeros 20 valores del arreglo b:  
9.42477 - 12.5664 - 15.7079 - 18.8495 - 21.9911 - 25.1327 - 28.2743  
- 31.4159 - 34.5575 - 37.6991 - 40.8407 - 43.9823 - 47.1239 -  
50.2654 - 53.407 - 56.5486 - 59.6902 - 62.8318 - 65.9734 - 69.115 -  
Imprimiendo los primeros 20 valores del arreglo c:  
9.42477 - 22.5664 - 35.708 - 48.8495 - 61.9911 - 75.1327 - 88.2743 -  
101.416 - 114.557 - 127.699 - 140.841 - 153.982 - 167.124 - 180.265
```

 test.cpp

SUCCESS!

 Standard Output

```
Sumando Arreglos en Paralelo!
Imprimiendo los primeros 20 valores del arreglo a:
0 - 10 - 20 - 30 - 40 - 50 - 60 - 70 - 80 - 90 - 100 - 110 - 120 - 130
- 140 - 150 - 160 - 170 - 180 - 190 -
Imprimiendo los primeros 20 valores del arreglo b:
24 - 32 - 40 - 48 - 56 - 64 - 72 - 80 - 88 - 96 - 104 - 112 - 120 -
128 - 136 - 144 - 152 - 160 - 168 - 176 -
Imprimiendo los primeros 20 valores del arreglo c:
24 - 42 - 60 - 78 - 96 - 114 - 132 - 150 - 168 - 186 - 204 - 222 - 240
- 258 - 276 - 294 - 312 - 330 - 348 - 366 -
```

Las impresiones de pantalla también se encuentran cargadas en el repositorio para su mejor lectura.

Explicación del código y los resultados:

Las siguientes líneas realizan la inclusión de las librerías necesarias para poder correr el siguiente código, se comenta la librería pch.h ya que correremos este código en un entorno web:

```
//#include "pch.h"
#include <iostream>
#include <omp.h>

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

A continuación, están los parámetros para la longitud de los arreglos, el tamaño de los segmentos a dividir las sumas y el número de índices que se mostrara del arreglo resultante de la suma:

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

Nuestra función principal, donde primero genera los arreglos y los rellena por medio de un ciclo for; posteriormente inicializa lo necesario para trabajar de manera paralela, para realizar la suma índice por índice se hace uso de un ciclo for que trabaja de manera paralela. Finalmente realiza la impresión de los arreglos que se suman, así como el tercer arreglo resultante de la suma:

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

    // Generación o petición de la información para llenar los arreglos:
    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)

        // Uso correcto del for paralelo:
        for (i = 0; i < N; i++)
            c[i] = a[i] + b[i];

    // Impresión de los elementos de cada arreglo:
    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);
}
```

Se genera una función adicional que imprime el número especificado de cada arreglo:

```
// Función para la impresión de los elementos de cada arreglo
void imprimeArreglo (float *d)
{
    for (int x = 0; x < mostrar; x++)
        std::cout << d[x] << " - ";
    std::cout << std::endl;
}
```

Reflexión sobre la programación paralela:

Se puede hablar profundamente acerca de la programación en paralelo, lo importante a recalcar es que se deben centrar los esfuerzos en tener una buena estrategia en como implementar la programación en paralelo, independiente del lenguaje de programación

Referencias:

Robey y Zamora. (2021). Parallel and High Performance Computing. Manning Publications*

B. Kirk y W. Hwu, 2016, Programming Massively Parallel Processors, 3er edition. Morgan Kaufmann.