**Curso Computo en la nube**

Tarea 1: Programación de una solución paralela

Jesús Esteiner Alonso

*Estudiante Máster en Inteligencia Artificial Aplicada*

*Universidad Tecnológico de Monterrey, Ciudad de México*

[*A1793554@tec.mx*](mailto:A1793554@tec.mx)

Enero 2023

Dr. Eduardo Antonio Cendejas Castro

*Profesor titular MNA CN*

**Introducción**

Debido a la reciente proliferación del uso compartido de memoria en las computadoras *(DSM distributed shared-memory)* por parte de la comunidad científica, hay demasiado interés en cómo utilizar de la mejor manera tanto la partición de memoria distribuida como la memoria compartida de estos sistemas computacionales. *MPI (message passing interface)* proporciona un medio eficiente de comunicación paralela entre una colección distribuida de máquinas de cómputo, sin embargo, no todas las implementaciones de MPI aprovechan la memoria compartida cuando está disponible entre los procesadores (la premisa básica es que dos procesadores, que comparten memoria común, pueden comunicarse entre si más rápido mediante el uso del medio compartido que a través de otros medios de comunicación) . *OpenMP (open multi processing)* se introduce para proporcionar un medio para implementar el paralelismo de memoria compartida en programas como C/C++. Específicamente, OpenMP crea un conjunto de variables de entorno, directivas de compilación y rutinas que se utilizarán para la paralelización de memoria compartida. OpenMP se diseñó específicamente para explotar ciertas características de las arquitecturas de memoria compartida, como la capacidad de acceder directamente a la memoria en todo el sistema con baja latencia y bloqueos de memoria compartida de una manera muy rápida.

Está surgiendo un nuevo paradigma de programación paralela en el que tanto MPI como OpenMP se pueden utilizar para estos procesos de paralelización. En una arquitectura de memoria compartida distribuida DSM, OpenMP usa la comunicación entre nodos (es decir, entre una colección de procesadores que comparten el mismo subsistema de memoria) y MPI se usaría para la comunicación entre los nodos (es decir, entre las distintas colecciones distribuidas de procesadores). La combinación de estas dos metodologías de paralelización puede proporcionar los medios más efectivos para explotar completamente los sistemas DSM modernos.

En esta oportunidad, estaremos implementando con la utilización de arreglos en C++, la sumatoria en paralelo de un conjunto de arreglos en donde podamos apreciar y aplicar la programación paralela sumando algunos elementos de los arreglos A y B, y generando un tercer arreglo C con el resultado obtenido.

Para mucha más información acerca de OpenMP favor referirse a [www.openmp.org](http://www.openmp.org).

**Liga del repositorio del proyecto en Github:**

[**https://github.com/PosgradoMNA/actividades-de-aprendizaje-JesusAlonsoTecM/blob/main/PruebaOMP.cpp**](https://github.com/PosgradoMNA/actividades-de-aprendizaje-JesusAlonsoTecM/blob/main/PruebaOMP.cpp)

**Explicación del código a través de capturas de pantalla**

Pantalla 1: Llamado de librerías y definición de constantes

Text

Description automatically generated

Pantalla 2: Definición y creación de 3 arreglos float, con N número de posiciones, en nuestro caso inicial, colocaremos 1500. Adicionalmente, se utiliza un FOR para recorrer los arreglos a y b, y llenarlos con valores aleatorios usando la función rand() de c++.

Text

Description automatically generated

Pantalla 3: Explicación de la instrucción más importante dentro del código #pragma

Text

Description automatically generated

Pantalla 4: Asignación del resultado de la suma de los arreglos a y b al arreglo c, y adicionalmente, se imprima el resultado de la sumatoria mostrando los primeros “mostrar” lugares, en nuestro caso 20 por el momento.

Text

Description automatically generated

Pantalla 5: Definición de la función imprimeArreglo()

Text

Description automatically generated

**CAPTURAS DE PANTALLA CON LOS RESULTADOS de ejecución del algoritmo:**

Ejecución 1, con parámetros:

Text

Description automatically generated

Resultados:

A screenshot of a computer

Description automatically generated

Ejecución 2, con parámetros:

Text

Description automatically generated

Resultados:

A screenshot of a computer

Description automatically generated

**REFLEXION SOBRE LA PROGRAMACION PARALELA**

Ya hemos logrado comprobar que la programación paralela tiene muchas ventajas, por supuesto también un par de desventajas, pero es un recurso que podemos aprovechar en gran variedad de procesos permitiéndonos ahorrar tiempo y recursos cuando este es posible. Sabemos ahora, que la utilización por ejemplo de la interfaz OpenMP como un modelo para la programación paralela para memoria compartida y memoria distribuida en multiprocesadores, nos permite tener a disposición varios hilos de ejecución concurrentes que dan acceso a variables alojadas en zonas de memoria compartida, y esto facilita enormemente la obtención de resultados mucho más rápido, y de una forma flexible acelerando la ejecución de un programa como resultado de la paralelización. La programación en paralelo ya es hoy día un paradigma de programación dominante, por supuesto principalmente en arquitecturas de computadores que mantienen procesadores multinúcleo. Entendemos sin embargo, que hay una curva de aprendizaje mucho más alta en la programación paralela que lleva a incrementar su complejidad de implementación, así como incrementos en energía y algunos otros aspectos, pero ello no detendrá que cada día más, este paradigma sea y se convierta aún más, en el paradigma dominante en la programación de computadoras.