TRABAJO FINAL
PROGRAMACION CONCURRENTE

Tema: Integración Numérica con OpenMP y MPI.

Profesor:

Eschoyez Maximiliano.

Integrantes:

Arias Fernando Agustín,

Brandan Agustín Darío.

Introducción

En este trabajo se nos dio la consigna de realizar un programa para realizar integración numérica con diferentes métodos de aproximación de la integral para luego aplicar técnicas de paralelización con el objetivo de acelerar el cálculo de los mismos.

Los métodos que se debían implementar en este trabajo eran:

● La Regla del Rectángulo

● La Regla del Punto Medio.

● La Regla del Trapecio.

● Método de Simpson 1/3

Para realizar la paralelización del cálculo se tenía que hacer uso de la biblioteca OpenMP, que facilita paralelizar un sistema en múltiples hilos, así como de MPI un protocolo de comunicación para la programación paralela en múltiples procesos.

Se deben realizar 2 versiones de cada aplicación del paralelismo:

OpenMP

● La primera versión ha de asignar un hilo a cada método de integración.

● La segunda versión debía dividir la carga de cálculo de cada uno de los procesos en N hilos lo cual nos daría un máximo de 4*N hilos en ejecución en un dado momento.

MPI

● La primera versión ha de asignar un proceso a cada método de

integración.

● La segunda versión debía dividir la carga de cálculo de cada uno de los procesos en N hilos lo cual nos daría un máximo de 4*N procesos en ejecución en un dado momento.

Una vez realizada la aplicación de los métodos de paralelismo se realizaron mediciones para comparar el rendimiento del cálculo secuencial de estos métodos de integración comparado a las versiones paralelizadas.

Desarrollo

Para poder desarrollar el software fuimos investigando sobre las dos librería a utilizar y aplicando los conocimientos adquiridos en clase pudimos llegar a la finalización.

1.El primer paso que se tomó fue investigar cómo se realiza el cálculo de cada uno de los métodos de integración a usar para de ahí poder derivar un algoritmo para el cálculo de los mismos.

2. Realización de la versión 1

2.a) MPI: En este caso cada uno de las funciones son ejecutadas por un proceso diferente a la cual le asignamos valores mediante MPI_BCAST. La división de tareas es realizada mediante el rango de cada proceso.

2.b) OPENMP: Se utilizó la sentencia #pragma para paralelizar el programa, talque como hay 4 métodos se ejecute un hilo por métodos eso haría que el programa se ejecuten 4 hilos. Se utilizó #pragma sections para resolver el problema.

3. Realización de la versión 2

3.a) MPI: A comparación con el anterior este programa comienza con 16 procesos los cuales se dividen en 4 por función. Dentro de cada función se hace una división de las tareas y se genera un rango local utilizado para el desarrollo de las integraciones. Una vez terminadas las tareas, cada proceso encargado realiza una reducción de tipo suma para obtener el resultado.

3.b OpenMP: Para resolver este problema se tuvo que paralelizar el proceso que realiza el bucle for. Ya que cada método se procesa con un bucle for, era necesario que ese proceso se divida a su vez en n hilos de acuerdo a la cantidad de hilos con él se inició el programa para dividir a los métodos.

Se utilizo la sentencia #pragma for shedule para dividir el proceso de un bucle for en n hilos.

4. Mediciones

Tiempo reales medidos en Gettime(offday).

OMP Version 1 4 HILOS	2481 us.
OMP Version 2 16 HILOS	1185 us.
MPI Version 1 4 PROCESOS	0.050171 seg.
MPI Version 2 16 PROCESOS	0.045553 seg.

Conclusión

El desarrollo de trabajo fue interesante y un desafío.

Se aprendió bastante cuando se buscó información sobre las librerías OPENMP y MPI, sabiendo que una librería trabaja con hilos (OPENMP) y la otra trabaja con procesos(MPI).

Estas librería nos permitieron paralelizar para mejorar la performance de nuestro software, en nuestro caso esa diferencia fue mínima, pero es seguro que en programas con grande cantidades de línea de códigos, con varios claster, con varios procesadores, multihilos; necesiten esta librería para aumentar la velocidad de procesamiento.