Universidad del Valle de Guatemala
Facultad de Ingeniería
Departamento de Ciencias de la Computación
CC3086 Programación de microprocesadores

Ciclo 2 de 2,024
Temario A

Laboratorio 05

Competencias para desarrollar
Distribuir la carga de trabajo entre hilos utilizando programación en C y OpenMP.

Instrucciones
Esta actividad se realizará individualmente. Al finalizar los períodos de laboratorio o clase, deberá entregar este archivo en formato PDF y los archivos .c en la actividad correspondiente en Canvas.

1. **(18 pts.)** Explica con tus propias palabras los siguientes términos:
   a) private:
   Esto significa que las variables declaradas como `private` dentro de una región paralela no comparten valores entre hilos. Cada hilo trabajará con su versión de la variable, evitando interferencias con otros hilos. Esto es útil cuando deseas que los hilos realicen operaciones independientes sin afectar a los demás.

   b) shared:
   El término `shared` en OpenMP se refiere a las variables que son compartidas entre todos los hilos. Todos los hilos tienen acceso a la misma dirección de memoria para estas variables, lo que significa que cualquier cambio realizado por un hilo será visible para los otros hilos. Este comportamiento es útil cuando varios hilos necesitan colaborar y trabajar sobre los mismos datos, pero también requiere cuidado para evitar condiciones de carrera.

   c) firstprivate:
   Especifica que cada hilo debe tener su propia instancia de una variable, y que la variable debe ser inicializada con el valor de la variable, porque existe antes de la construcción paralela.

   d) barrier:
   Es un punto en el programa donde todos los hilos deben detenerse y esperar hasta que todos hayan llegado antes de continuar. Es como una línea de meta que todos deben cruzar antes de seguir a la siguiente etapa

   e) critical
   `critical` es una sección de código que solo un hilo puede ejecutar a la vez. Cuando un hilo entra en una región `critical`, otros hilos deben esperar hasta que el primero haya terminado. Esto es útil para proteger operaciones que no pueden ser ejecutadas en paralelo.

   f) atomic
   `atomic` es similar a `critical`, pero es más eficiente para operaciones simples, como incrementar o decrementar una variable. `atomic` asegura que una operación específica se realice sin interrupciones por otros hilos, permitiendo una actualización segura de variables compartidas con un impacto mínimo en el rendimiento.

2. **(12 pts.)** Escribe un programa en C que calcule la suma de los primeros N números naturales utilizando un ciclo ***for*** **paralelo**. Utiliza la cláusula **reduction con +** para acumular la suma en una variable compartida.
   a) Define N como una constante grande, por ejemplo, N = 1000000.

b) Usa omp_get_wtime() para medir los tiempos de ejecución.

3. **(15 pts.)** Escribe un programa en C que ejecute tres funciones diferentes en paralelo usando la **directiva #pragma omp sections**. Cada sección debe ejecutar una función distinta, por ejemplo, una que calcule el factorial de un número, otra que genere la serie de Fibonacci, y otra que encuentre el máximo en un arreglo, operaciones matemáticas no simples. Asegúrate de que cada función sea independiente y no tenga dependencias con las otras.

4. **(15 pts.)** Escribe un programa en C que tenga un ciclo for donde se modifiquen dos variables de manera paralela usando #pragma omp parallel for.
   a. Usa la cláusula shared para gestionar el acceso a la variable1 dentro del ciclo.
   b. Usa la cláusula private para gestionar el acceso a la variable2 dentro del ciclo.
   c. Prueba con ambas cláusulas y explica las diferencias observadas en los resultados.

5. **(30 pts.)** Analiza el código en el programa Ejercicio_5A.c, que contiene un programa secuencial. Indica cuántas veces aparece un valor key en el vector a. Escribe una versión paralela en OpenMP utilizando una descomposición de tareas **recursiva**, en la cual se generen tantas tareas como hilos.

6. **REFLEXIÓN DE LABORATORIO: se habilitará en una actividad independiente.**

Bibliografía

Using OpenMP with C — Research Computing University of Colorado Boulder documentation (curc.readthedocs.io)

OpenMP Clauses | Microsoft Learn