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
   Cuando una variable se declara como private, cada hilo (subproceso) en una región paralela tiene su propia copia privada de esa variable. Esto significa que cada hilo trabaja con su propia instancia de la variable, sin afectar a las demás.

   b) Shared
   Las variables declaradas como shared son compartidas entre todos los hilos en una región paralela. Todos los hilos pueden acceder y modificar la misma instancia de la variable. Si no se especifica nada, las variables se consideran compartidas por defecto.

   c) Firstprivate
   Similar a private, pero además de tener una copia privada, la variable firstprivate también se inicializa con el valor que tenía antes de entrar en la región paralela.

   d) Barrier
   La directiva barrier se utiliza para sincronizar todos los hilos en un equipo. Cuando se encuentra una barrera, todos los hilos se detienen hasta que todos los hilos han alcanzado la misma barrera.

   e) Critical
   Con critical, puedes especificar una sección de código que solo se ejecutará en un hilo a la vez. Es útil cuando necesitas proteger una sección crítica de código que no puede ser ejecutada simultáneamente por varios hilos.

   f) Atomic
   La directiva atomic se utiliza para actualizar una ubicación de memoria de manera atómica. Esto significa que la operación se realiza sin interferencias de otros hilos.

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.

    En esta versión paralela:
    - Utilizamos la directiva #pragma omp parallel for para crear un bucle paralelo que divide el trabajo entre los hilos disponibles.
    - La cláusula reduction(+:count) asegura que cada hilo acumule su propia suma parcial en la variable compartida count.
    - Cada hilo verifica si el valor en la posición i del arreglo es igual a 'key' y, si es así, incrementa su contador local.

    De esta manera, se generan tantas tareas como hilos disponibles, y cada hilo contribuye al conteo total de apariciones de 'key'.

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