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

Dulce Ambrosio - 231143

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 : Indica que cada subproceso debe de tener su propia instancia en una variable.
   b) shared: Las variables se pueden compartir entre los subprocesos.
   c) firstprivate: Cada subproceso debe de tener su propia instancia y la variable con la que se inicializa debe de contener ya un valor existente en la construcción paralela.
   d) barrier: Los subprocesos se detienen hasta el punto de la barrera, hasta que todos se ejecuten.
   e) critical: El código se ejecuta un subproceso a la vez.
   f) atomic: Se utiliza para poder actualizar la ubicación de una memoria.

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.

   Como la variable1 es compartida esto provoca que se genera una tipo de carrera, es decir que diferentes hilos tratan de modificar el valor de está variable por lo que no se puede conocer el valor específico de está variable hasta que se termine el proceso. En cambio como la variable2 no es compartido, si no que es privada, no se genera este tipo de carrera, cada hilo realiza su proceso de manera independiente.

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.

   El key aparece 8 veces en el vector a.

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

Github: https://github.com/Dulce2004/Lab5---Micro.git

REFERENCIAS

TylerMSFT. (2023a, junio 16). Cláusulas de OpenMP. Microsoft Learn. https://learn.microsoft.com/es-es/cpp/parallel/openmp/reference/openmp-clauses?view=msvc-170

TylerMSFT. (2023b, junio 16). Directivas de OPenMP. Microsoft Learn. https://learn.microsoft.com/es-es/cpp/parallel/openmp/reference/openmp-directives?view=msvc-170#atomic