UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

Guatemala 21 de Agosto de 2024
Adrián López 231361

Programación de Microprocesadores

Laboratorio 5: Distribución de Carga de Trabajo por Hilos en C y OpenMP

---

1. (18 pts.) Explica con tus propias palabras los siguientes términos:

a) private

Directiva que declara que una variable será privada para cada hilo. Cada hilo obtiene su propia copia de la variable, como recurso no compartido. Al iniciar la región paralela, esta copia no tiene ningún valor inicial definido, y al salir de la región paralela, los cambios se descartan.

b) shared

Directiva que especifica que una variable es compartida entre todos los hilos. Todos los hilos acceden y modifican la misma variable, como recurso compartido.

c) firstprivate

Directiva que además de hacer una copia privada de la variable para cada hilo, inicializa esta copia con el valor que tenía la variable en el hilo principal antes de entrar en la región paralela.

d) barrier

Punto de sincronización donde todos los hilos deben detenerse y esperar hasta que todos los hilos hayan llegado a ese punto antes de continuar. Para asegurar que todos los hilos avancen juntos en ciertos momentos críticos del programa.

e) critical

Directiva que define una sección crítica del código, donde solo un hilo a la vez puede ejecutar ese bloque de código. Es útil para cuando se necesita asegurar que ciertas operaciones se realicen de forma atómica y, simultáneamente evitar conflictos cuando varios hilos intentan modificar la misma variable o recurso al mismo tiempo.

f) atomic

Es una versión más ligera y eficiente que “critical”, usada cuando solo una simple operación debe ser realizada de manera atómica.

(OpenMP Architecture Review Board, 2021)

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.

Programa en repositorio.

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.

Programa en repositorio.

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.

Programa en repositorio.

5. fcdx(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.

Las veces que aparece un valor key en el vector A es determinado por el programa mismo:

```
PS D:\28adr\OneDrive\Documents\Uni\Ciclo 4\Programación de Microprocesadores> gcc -fopenmp -o Ejercicio5A Ejercicio_5A.c
PS D:\28adr\OneDrive\Documents\Uni\Ciclo 4\Programación de Microprocesadores> ./Ejercicio5A
Numero de veces que 'key' aparece secuencialmente: 8
```

Versión paralela en repositorio.

Reflexión en un espacio diferente.

Referencias:

OpenMP Architecture Review Board. (2021, Noviembre 2). *OpenMP Application Programming Interface Specification Version 5.2 November 2021*. OpenMP.

https://www.openmp.org/wp-content/uploads/OpenMP-API-Specification-5-2.pdf