	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: TISTA GARCÍA EDGAR

Asignatura: ESTRUCTURA DE DATOS Y ALGORITMOS II

Grupo: 5

Número de Práctica(s): Guía práctica de estudio 12-13: Introducción a Algoritmos en Paralelo

Integrante(s): MURRIETA VILLEGAS ALFONSO | VALDESPINO MENDIETA JOAQUÍN

Núm. De Equipo de cómputo empleado: 38

Semestre : 2019 - 1

Fecha de entrega: 30 DE OCTUBRE DE 2018

Observaciones:

CALIFICACIÓN: _____

ALGORITMOS EN PARALELO PARTE 1 Y 2.

OBJETIVO

El estudiante conocerá y aprenderá a utilizar algunas de las directivas de OpenMp para implementar algún algoritmo paralelo.

INTRODUCCIÓN

En la práctica siguiente se revisarán nuevas clausulas y constructores de OpenMp, así como nuevos conceptos y ejemplos que ayudarán a la comprensión. A continuación, se muestran algunos constructores que se pueden encontrar en OpenMp.

PARTE TEÓRICA DE LA PRÁCTICA 12

CONSTRUCTOR CRITICAL

Porción de código que contienen la actualización de una o 2 variables compartidas, por lo que pueden ocurrir coincidencias. La exclusión mutua consiste en que un solo proceso excluye temporalmente a todos los demás por usar un recurso compartido dad del sistema.
Existe una directiva en OpenMP que permite que en un segmento de código no sea interrumpido por otros hilos, o sea la directiva delimita para un hilo. Sintaxis:

#pragma omp critical
{
}

CLAUSULA REDUCTION

La cláusula toma el valor de una variable aportada por cada hilo y aplica la operación indicada sobre de esos datos para obtener un resultado. Se puede plantear el producto punto. Cada hilo aporta el cálculo parcial de los datos y después se suman todos los resultados parciales y este quedará sobre una misma variable.

#pragma omp parallel freduction (operator:variable)
{
}

CONSTRUCTOR SECTION

Permite usar el paralelismo funcional debido a que permite asignar secciones de código independiente a hilos diferentes para que trabajen de forma concurrente/paralela. Cada sección paralela es ejecutada por un solo hilo, y cada hilo ejecuta ninguna o alguna sección. Este constructor tiene una barrera implícita que sincroniza el final de las secciones.
Sintaxis
#pragma opm parallel sections

```
{
        #pragma omp section
                …
}
```

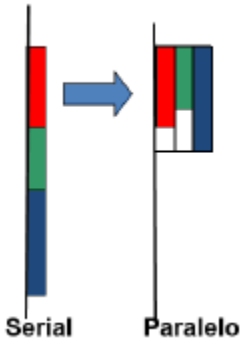


Imagen 1. Esquema de un algoritmo en su forma serial y posteriormente a su forma paralela.

CONSTRUCTOR BARRIER

El constructor barrier coloca una barrera explicita para que cada hilo espere hasta que todos los otros hilos lleguen a la barrera. Esto se utiliza para sincronizar los procesos. Aplica cuando se quiere obtener una secuencia apropiadas cuando hay dependencia de datos.
Sintaxis
#pragma omp barrier

CONSTRUCTOR SINGLE

Constructor que permite definir un bloque de código básico de código dentro de una región paralela, que debe ser ejecutada por un solo hilo. Nota: Todos los hilos esperan.
Sintaxis
#pragma omp single

CONSTRUCTOR MASTER

Este constructor es similar a single con la diferencia de que las actividades realizadas son hechas por el hilo maestro y no se tiene una barrera implícita, es decir, los hilos restantes no esperan a que el hilo maestro termine la actividad asignada.
Sintaxis

```
#pragma omp master
{
}
```

CLAUSULA NOWAIT
Esta cláusula permite quetar las barreras implícitas que tienen los constructores, como lo son el for, single, section, etc. Quitar las barreras que no son necesarias ayudan a mejorar el desempeño del programa.

PARTE TEÓRICA DE LA PRÁCTICA 13

En el proceso de análisis de problemas que se deseen paralelizar, surgen diversas opciones o formas de resolverlos. Ello dependerá, en gran medida, de las diferentes técnicas y herramientas que nos pueda proporcionar el software que empleemos en esta tarea. En esta guía revisaremos algunos problemas ejemplo que emplean estas técnicas con las directivas de OpenMP.

1] MULTIPLICACIÓN DE MATRICES

Los algoritmos basados en arreglos unidimensionales y bidimensionales a menudo son paralelizables de forma simple debido a que es posible tener acceso simultáneamente a todas las partes de la estructura, no se tienen que seguir ligas como en las listas ligadas o árboles. Un ejemplo común es el producto de dos matrices A y B de orden y. Recordando la fórmula y la forma de obtener un elemento de.

$$C_{i\,j} = \sum_{k=0}^{n-1} a_{ik} b_{k\,j}$$

para $0 \leq i, j < n$

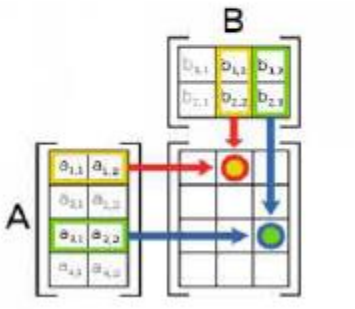


Imagen 2. Esquema de paralelización de matrices.

Para lograr dividir este problema se realiza un paralelismo de datos, donde cada hilo trabaje con datos distintos, pero con el mismo algoritmo. Un planteamiento de solución paralela es considerar que, en un ambiente con varios hilos, cada uno puede trabajar con diferentes datos, de acuerdo con las siguientes opciones:

- Calcular un elemento de la matriz resultante, se requiere un renglón de y una columna de B.
- Calcular todo un renglón de C, se requiere un renglón de A y toda la matriz B.
- Calcular renglones i de C, se requieren renglones de A y toda B.

Dependiendo de la granularidad escogida se necesitan un número de hilos en la región paralela y como lo ideal es tener un número de hilos igual al número de unidades de procesamiento, que no puede ser muy grande se toma la tercera opción.

2] HISTOGRAMA

El histograma es el número de pixeles de cada nivel de gris encontrado en la imagen. En el programa la imagen es representada por una matriz de cuyos elementos pueden ser mayores o iguales a 0 y menores a (tonos de gris) y se cuenta el número de veces que aparece un número(pixel) en la matriz y éste es almacenado en un arreglo unidimensional.

Por ejemplo, para la matriz (imagen) de 5 5 de la figura 13.2 que almacena valores de entre 0 y 5, en un arreglo unidimensional llamado histograma se almacena en el elemento con índice 0 el número de ceros encontrados, en el elemento con índice 1 el número de unos, en el elemento con índice 2 el número de dos etc…

Para paralelizarlo, se realiza una descomposición de datos, es decir, se dividirá la matriz de forma que cada hilo trabaje con un número de renglones diferentes y almacene la frecuencia de aparición de cada valor entre 0 y de la sub-matriz en un arreglo privado o propio de cada hilo llamado h []. Después se deben sumar todos los arreglos h[ ] de cada hilo para obtener el resultado final en el arreglo h [ ] . Figura 13.3.

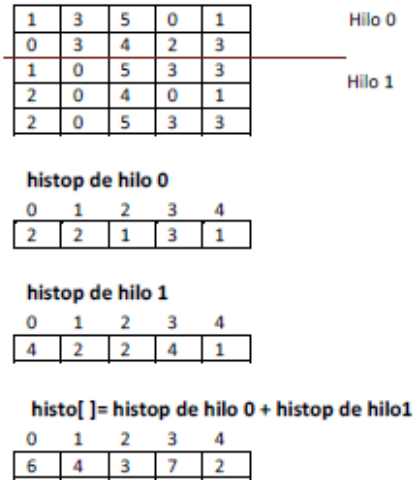
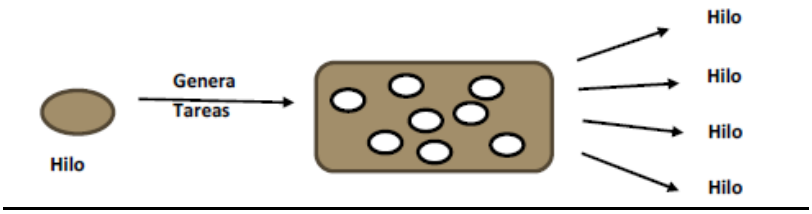


Imagen 3. Esquema de paralelización de como los datos del histograma de una imagen se distribuyen en distintos hilos.

3] SUCESIÓN DE FIBONACCI

Para este ejemplo se requiere conocer de otros dos constructores que son útiles para paralelizar otro tipo de problemas como los que requieren manejo de recursividad o los que contienen ciclos con un número indeterminado de iteraciones. Estos son task y taskwait.
El constructor task sirve para generar tareas de forma explícita y esas tareas generadas pueden ser asignadas a otros hilos que se encuentran en la región paralela. Figura 13.4.



Cuando un hilo encuentra el constructor task, se genera una nueva tarea, así que, si un hilo encuentra veces al constructor task, genera tareas. El constructor taskwait permite esperar a que todas las tareas hijas generadas desde la tarea actual terminen su actividad. Usar la directiva task en un programa recursivo para calcular el número de la sucesión Fibonacci.

EJERCICIOS DEL MANUAL PRÁCTICA 12

1] ACTIVIDAD

Completar la versión serie y paralela del ejemplo explicado de búsqueda del valor mayor de los elementos de un arreglo unidimensional de enteros.

2] ACTIVIDAD

Completar la versión serie y sus dos versiones paralelas del ejemplo explicado del producto punto de dos vectores de n elementos enteros.

3] ACTIVIDAD

Una forma de obtener la aproximación del número irracional PI es utilizar la regla del trapecio para dar solución aproximada a la integral definida

$$\pi = \int_0^1 \frac{4}{(1-x^2)}\, dx.$$

Se proporcionó el código en su versión serial y se requirió su versión paralela.

4] ACTIVIDAD

Para el siguiente programa obtener dos versiones paralelas, una utilizando el constructor section y otra con el constructor for. ¿Cuál de las dos versiones tarda más tiempo?

5] ACTIVIDAD

Para esta actividad se pidió probar el código dado en el manual y posteriormente contestar las siguientes preguntas:

- ¿Qué sucede si se quita la barrera?
- Si en lugar de utilizar el constructor master se utilizara single, ¿Qué otros cambios se tienen que hacer en el código?

EJERCICIOS DEL MANUAL PRÁCTICA 13

1] ACTIVIDAD

Completar la versión serie y paralela del ejemplo 1 explicado en la guía y medir el tiempo de ejecución de ambas versiones utilizando matrices de orden 500 x 500.

2] ACTIVIDAD

Completar la versión serie y paralela del ejemplo 2 explicado en la guía y medir el tiempo de ejecución de ambas versiones utilizando = 1000 y = 256. Tomar tres lecturas y sacar el tiempo promedio para cada caso.

- ¿Cuánto tiempo tardaron ambas versiones?
- ¿Por qué en la versión paralela el cálculo de h [] está delimitado con el constructor critical?

3] ACTIVIDAD

Probar las versiones serie y paralela del ejemplo 3 para verificar que se obtienen los mismos resultados. Además, agregar a la versión paralela la impresión del identificador del hilo en la generación de cada tarea para visualizar los hilos que participan en los cálculos.

- ¿Qué pasa si 1 y 2 no se colocan como compartidas? Probar
- ¿Por qué sucede lo observado?

4] ACTIVIDAD

Ejercicios sugeridos por el profesor Nota: Existen otros constructores y cláusulas y se espera que con lo visto en la guía 11,12 y 13 el estudiante pueda comprenderlas con mayor facilidad.

CONCLUSIONES

MURRIETA VILLEGAS ALFONSO

A lo largo de las prácticas 12 y 13 se conocieron distintas directivas y constructores de OpenMp donde a través de su análisis y aplicaciones es como se logró abordar mucho más el paradigma de programación en paralelo, además de que a lo largo de las distintas aplicaciones es como se logró también ver las ventajas y desventajas de la aplicación de algoritmos en versión paralela.

Por último, el paradigma de programación en paralelo como se vio en esta práctica realmente demostró tiempos de ejecución realmente distintos a los de la versión serial. Esta práctica es sin duda la demostración de cómo la versión paralela de los algoritmos puede ser totalmente buena en tiempo de ejecución debido a la forma en que se distribuye la carga de tareas en los procesadores.

VALDESPINO MENDIETA JOAQUÍN

REFERENCIAS

(2007). Introduction to parallel programming, Intel Software collage.

Galvin Baer Peter, Gagne Greg. *Fundamentos de Sistemas operativos*. MacGraw Hill.

Elba Karen Saenz García. (2017). *Manual de Prácticas del laboratorio de Estructuras de Datos y algoritmos 1I*. UNAM, Facultad de Ingeniería.