Proyecto Paralelizacion de programas con OpenMP

Autor: Jon Echevarria

Informe Técnico

Índice:

1. Introducción:

En este informe técnico se presenta el trabajo realizado en el proyecto de Arquitectura de Computadores, el cual ha consistido en paralelizar y optimizar dos programas utilizando la biblioteca OpenMP. El proyecto se ha llevado a cabo en la segunda convocatoria de la asignatura, habiendo realizado previamente el proyecto de evaluación continua en la primera convocatoria.

El objetivo principal de este proyecto ha sido aprender a utilizar OpenMP para aprovechar el paralelismo en programas computacionalmente intensivos y mejorar su rendimiento en un entorno multiprocesador. Se nos proporcionaron dos programas completos, "mcol.c" y "histo.c", y se nos pidio que los paralelizaramos para obtener un mayor rendimiento.

En este informe se detallarán las soluciones propuestas, los resultados obtenidos y se extraerán algunas conclusiones. También se incluirá información sobre las herramientas utilizadas, tanto en términos de hardware (arquitectura, procesadores, memoria) como de software (sistema operativo, compiladores, versiones de software).

2. Fundamentos Teóricos:

En esta sección, se presentarán los fundamentos teóricos necesarios para comprender el trabajo realizado en el proyecto de paralelización de programas de C con OpenMP. Se abordarán los conceptos clave de paralelismo, la biblioteca OpenMP y las técnicas utilizadas para aprovechar el paralelismo en los programas.

2.1 Paralelismo: El paralelismo se refiere a la capacidad de ejecutar tareas simultáneamente en un sistema computacional con el objetivo de mejorar el rendimiento y la eficiencia. Al dividir una tarea en subprocesos o hilos independientes, se pueden ejecutar de forma concurrente, aprovechando los recursos de hardware disponibles y reduciendo el tiempo de ejecución global.

2.2 Biblioteca OpenMP: OpenMP (Open Multi-Processing) es una interfaz de programación de aplicaciones (API) que facilita la programación paralela en sistemas de memoria compartida. Proporciona directivas de compilación, bibliotecas y variables de entorno que permiten a los desarrolladores aprovechar el paralelismo de forma sencilla. OpenMP se utiliza comúnmente en programas escritos en lenguaje C y C++.

2.3 Técnicas para aprovechar el paralelismo: Existen diversas técnicas para aprovechar el paralelismo en los programas utilizando OpenMP. Algunas de las técnicas más utilizadas son:

2.3.1 Directivas de compilación: OpenMP ofrece un conjunto de directivas de compilación que se insertan en el código fuente para indicar al compilador dónde y cómo se deben aplicar las operaciones paralelas. Estas directivas especifican regiones paralelas, bucles paralelos, tareas, entre otros, permitiendo al programador controlar la ejecución paralela del programa.

2.3.2 Creación de hilos: OpenMP proporciona funciones para la creación y gestión de hilos, permitiendo a los programadores especificar cómo se deben asignar los hilos a las tareas paralelas. Los hilos permiten la ejecución concurrente de secciones de código y facilitan la distribución de la carga de trabajo entre los procesadores disponibles.

2.3.3 Sincronización: La sincronización es fundamental para garantizar la consistencia y coherencia de los datos compartidos entre los hilos. OpenMP ofrece mecanismos de sincronización, como barreras y cláusulas de reducción, que permiten a los hilos coordinar su ejecución y compartir resultados de manera segura.

3. Aplicación:

En esta sección se presentará la solución desarrollada, las decisiones adoptadas y los resultados obtenidos para cada uno de los programas.

3.1 Ejercicio 2: histo.c

El programa "histo.c" realiza varias operaciones en una matriz, como calcular el histograma de la imagen, el valor mínimo del histograma y su posición, y la suma de cada fila y columna de la matriz. Para paralelizar el código utilizando OpenMP, se aplicaron las siguientes modificaciones.

Se utilizó una región paralela para encapsular las operaciones que se pueden realizar de forma concurrente: el cálculo del histograma, la suma de filas y la suma de columnas.

En el bucle para calcular el histograma, se utilizó la directiva "#pragma omp atomic" para garantizar que los incrementos en los elementos del histograma se realicen de forma segura en un entorno paralelo.

Para la suma de filas y columnas, se aplicó la directiva "#pragma omp for nowait schedule(dynamic, 500)" para paralelizar los bucles de manera eficiente. La cláusula "nowait" se utilizó para evitar esperas innecesarias entre las iteraciones del bucle.

Además, se utilizó una sección crítica "#pragma omp critical" para asegurar que la actualización final del histograma global se realice de forma segura.

En cuanto al cálculo del valor mínimo del histograma y su posición, se decidió no utilizar la cláusula de reducción "reduction(min:hmin)" para evitar conflictos de escritura concurrente en la variable "hmin". En su lugar, se utilizó una comparación simple dentro de un bucle paralelo para encontrar el mínimo.

3.2 Ejercicio 1: mcol.c

En este caso, el código es más simple ya que el bucle principal es paralelizable sin requerir estructuras adicionales.

Los cambios realizados son los siguientes:

- Se ha aplicado la directiva `#pragma omp parallel for` para paralelizar el bucle principal. Esta directiva distribuye las iteraciones del bucle entre los hilos disponibles y asegura la ejecución en paralelo de cada iteración.
- Se han declarado las variables `i`, `columna`, `y` y `suma` como privadas utilizando la cláusula `private`. Esto garantiza que cada hilo tenga su propia copia de estas variables y evita conflictos de escritura concurrente.
- El cálculo de la suma y la asignación del resultado se realizan dentro del bucle paralelo de manera segura, ya que cada hilo trabaja con su propia variable `suma` y asigna el resultado al elemento correspondiente en el array `resultados`.

4. Conclusiones Generales:

La paralelización de programas con OpenMP brinda mejoras significativas en rendimiento y eficiencia. Al dividir tareas en regiones paralelas y utilizar directivas específicas, se aprovechan los recursos en sistemas multiprocesador. Las operaciones concurrentes se ejecutan en paralelo, acelerando el procesamiento. Directivas como `#pragma omp parallel for` y `#pragma omp atomic` facilitan la implementación y gestionan conflictos de escritura. Se deben considerar variables compartidas, secciones críticas y evitar condiciones de carrera. En resumen, OpenMP optimiza programas al aprovechar el paralelismo y mejorar la velocidad y escalabilidad.

5. Bibliografía:

ejemplos de openMP
pdf de Tim Mattson