OpenMP

Trabajo práctico Nº 2

Sistemas Operativos II
DC - FCEFyN - UNC

Abril de 2020

1 Introducción

Los niveles de integración electrónica han permitido la generación de procesadores de arquitecturas multiprocesos, multicore y ahora many integrated core (MIC). Este avance hace necesario que los programadores cuenten con un profundo conocimiento del hardware sobre el que se ejecutan sus programas, y que dichos programas ya no pueden ser monoproceso.

Entre las técnicas y estándares más utilizados para sistemas de memoria compartida y memoria distribuida, se encuentra OpenMP y MPI respectivamente.

2 Objetivo

El objetivo del presente trabajo práctico es que el estudiante sea capaz diseñar una solución que utilice el paradigma de memoria compartida, utilizando OpenMP.

3 Desarrollo

3.1 Requerimientos

Para realizar el presente trabajo práctico, es necesario descargar la librería *SimpleBmp*[01] en el sistema sobre el cual se diseñe, desarrolle y pruebe la solución al problema. Esta librería provee funciones para crear, leer y escribir imágenes BMP de 24 bits sin paleta de colores.

3.2 Problema a desarrollar

Deberá implementar un programa en lenguaje C que realice el procesamiento de una imagen BMP utilizando la librería anteriormente mencionada. El procesamiento consiste en aplicar dos filtros, en dos zonas diferentes de la imagen. La primer zona, llamada zona A, esta definida en el interior de un circulo de radio

R centrado en la imagen, este radio sera recibido como parámetro y su unidad de medida será en píxeles. El exterior de este circulo se llamara zona B.

El procesamiento a realizar en la zona A consiste en la aplicación de un *filtro lineal* el cual se corresponde a la siguiente ecuación

$$p(x, y) = p(x, y) * K + L \tag{1}$$

Donde K representa el contraste y L el brillo.Esto deberá estar parametrizado en el código, de manera tal, que pueda ser modificado en tiempo de compilación o ejecución (a elección del programador).

El procesamiento a realizar en la zona B consiste en la aplicación de un filtro *Blur* el cual se corresponde a la siguiente ecuación (Una convolucion de 2 dimenciones)

$$g(x, y) = \omega * p(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} \omega(s, t) p(x - s, y - t) \tag{2}$$

Donde $g(x, y)$ es la imagen filtrada, $p(x, y)$ es la imagen original y ω es la matriz que se define a continuación, con límites $-a <= s <= a$ y $-b <= t <= b$.

$$\omega = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Debe tenerse en cuenta que es necesario normalizar el resultado para evitar la saturación de los colores

Como metodología para resolver este problema, se solicita que, primero, se realice un diseño que sea solución al problema sin explotar el paralelismo en lenguaje C. Luego, a partir de este, realizar una nueva implementación que realice el proceso mediante el uso de la librería OpenMP, explotando el paralelismo del problema. Para ello, se requiere reconocer qué tipo de paralelismo exhibe el problema en cuestión y luego, diseñar la solución del mismo determinando cuáles son los datos/operaciones paralelizables. Se tendrá en cuenta, el nivel de paralelismo alcanzado.

Además, el informe debe contener gráficos/tablas de los datos recopilados de varias ejecuciones del programa (30, estadístico suficiente), tanto en el clúster de la UNC como localmente en una sola PC, indicando qué ganancia en pérformance existe al distribuir este proceso en multiples hilos en comparación a la ejecución monohilo, obtener el speedup y la escalabilidad. Comparar la ejecución procedural y de memoria compartida, explicando la diferencias observadas. También se deberá investigar acerca de qué utilidades de profiling gratuitas existen y que brinda cada una (un capítulo del informe), optando por una para realizar las mediciones de tiempo de ejecución de la aplicación diseñada

Al igual que en el trabajo anterior, se exige el uso de Cppcheck y la compilación con el uso de las flags de warning -Werror, -Wall y -pedantic -Wconversion. Se pide utilizar el estilo de escritura de código de GNU o el estilo de escritura del kernel de Linux.

Figure 1: Imagen base

Figure 2: Imagen resultante

4 Entrega

Se deberá proveer los archivos fuente, así como cualquier otro archivo asociado a la compilación, archivos de proyecto "Makefile" y el código correctamente documentado, todo en un repositorio a elección por el Estudiante y compartido a los profesores del práctico o con acceso público. También se debe entregar un informe, con el formato adjunto. Gráficos y análisis de comparación entre la ejecución procedural y la distribuida. El informe además debe contener el diseño de la solución y la comparativa de profilers. Se debe asumir que las pruebas de compilación se realizarán en un equipo que cuenta con las herramientas típicas de consola para el desarrollo de programas (Ejemplo: gcc, make), y NO se cuenta con herramientas "GUI" para la compilación de los mismos (Ej: eclipse).

NOTA: todavía no se encuentra habilitada la cuenta de acceso al clúster de la UNC.

Cuando esté disponible la misma, se informará por LEV los datos necesarios.

5 Evaluación

El presente trabajo práctico es individual deberá entregarse antes de las 23:50 ART del día 14 de Mayo de 2020 mediante el LEV. Será corregido y luego deberá coordinar una fecha para la defensa oral del mismo.

6 Referencias

1. Simple Bmp SOII (SimpleBmp), https://github.com/Sistemas-Operativos-II-2020/simple_bmp
2. BMP File Structure , https://itnext.io/bits-to-bitmaps-a-simple-walkthrough-of-bmp-image-format-765dc6857393
3. https://en.wikipedia.org/wiki/Kernel_(image_processing)
4. https://hub.packtpub.com/image-filtering-techniques-opencv