Práctica de programación paralela con OpenMP

J. Daniel García Sánchez (coordinador)

Arquitectura de Computadores
Grupo ARCOS
Departamento de Informática
Universidad Carlos III de Madrid

1. Objetivo

Esta práctica tiene como objetivo fundamental hacer el que los estudiantes se familiaricen con la **optimización de programas secuenciales** y con los **modelos de programación paralela** en arquitecturas de memoria compartida.

En concreto, la práctica se centrará en el desarrollo de software secuencial en el lenguaje de programación C++ (incluyendo las mejoras de C++17), así como en las técnicas de paralelismo mediante el uso de *OpenMP*.

2. Descripción del proyecto

En esta práctica se construirá una aplicación de filtrado de imágenes. Se desarrollarán dos versiones: una versión secuencial y otra versión paralela.

En las siguientes secciones se presentn los requisitos que deben cumplir con los programas desarrollados. Así mismo, se hará público un archivo binario con un programa ejectuable con la funcionalidad requerida. Esto hará más fácil la comparación de resultados.

2.1. Visión general

Se desarrollará una aplicación que aplica dos filtros sobre un conjunto de imágenes en formato BMP. El resultado será una nueva imagen en el mismo formato después de aplicar los mencionados filtros.

Para cada imagen se aplicarán los siguientes filtros:

- Difusión gaussiana (*Gaussian blur*).
- Operador de Sobel.

Se desarrollarán dos versiones distintas del programa:

- **image-seq**: Versión secuencial.
- **image-par**: Versión paralela.

La aplicación (**image-seq** o **image-par**) tomará 3 parámetros:

- Modo de ejecución.
- Directorio con imágenes de entrada.
- Directorio en el que dejar las imágenes de salida.

El modo de ejecución será uno de los siguientes:

- **copy**: No realiza ninguna transformación. Solamente copia los archivos.
- **gauss**: Aplica el filtro de suavizado gaussiano.
- **sobel**: Aplica el filtro de suavizado gaussiano y el operador de Sobel.

En cualquiera de los casos, la aplicación leerá todos los archivos del directorio de entrada, aplicará las transformaciones correspondientes y escribirá los archivos correspondientes con el mismo nombre en el directorio de salida.

El programa deberá comprobar que el número de parámetros es 3:

```
$image-seq
Wrong format:
  image-seq operation in_path out_path
    operation: copy, gauss, sobel
$
```

Si el primer argumento no toma un valor adecuado (**copy**, **gauss** o **sobel**), se presentará un mensaje de error y se terminará:

```
$image-seq gaus indir outdir
Unexpected operation:gaus
  image-seq operation in_path out_path
    operation: copy, gauss, sobel
$
```

Si el directorio de entrada no existe o no puede abrirse, se presentará un mensaje de error y se terminará:

```
$image-seq sobel indir outdir
Input path: inx
Output path: out
Cannot open directory [inx]
  image-seq operation in_path out_path
    operation: copy, gauss, sobel
$
```

Si el directorio de salida no existe, se presentará un mensaje de error y se terminará:

```
$image-seq sobel indir outx
Input path: indir
Output path: outx
Output directory [outx] does not exist
  image-seq operation in_path out_path
    operation: copy, gauss, sobel
$
```

En cualquier otro caso, se procesarán todos los archivos del directorio de entrada y se dejarán los resultados en el directorio de salida. Para cada archivo procesado se imprimirá la información del tiempo dedicado a cada tarea. Todos los valores de tiempo se expresarán en microsegundos:

```
$image-seq sobel indir outdir
Input path: indir
Output path: outdir
File: "indir/62096.bmp"(time: 73016)
  Load time: 3442
  Gauss time: 38909
  Sobel time: 29022
  Store time: 1642
File: "indir/169012.bmp"(time: 65724)
  Load time: 2602
  Gauss time: 32950
  Sobel time: 28515
  Store time: 1655
$
```

2.2. El formato BMP

Se utilizará el formato de imágenes BMP. Los ficheros de este tipo contienen una cabecera de al menos 54 bytes con la información especificada en el cuadro 1.

Comienzo	Longitud	Descripción
0	2	Caracteres **'B'** y **'M'**.
1	4	Tamaño del archivo.
6	4	Reservado.
10	4	Inicio de datos de imagen.
14	4	Tamaño de la cabecera de bitmap.
18	4	Anchura en pixeles.
22	4	Altura en pixeles.
26	2	Número de planos.
28	2	Tamaño de punto en bits.
30	4	Compresión.
34	4	Tamaño de la imagen.
38	4	Resolución horizontal.
42	4	Resolución vertical.
46	4	Tamaño de la tabla de color.
50	4	Contador de colores importantes.

Cuadro 1: Campos de cabecera para el formato BMP.

Se tendrán en cuenta las siguientes restricciones para que un fichero se pueda considerar válido.

- El número de planos debe ser **1**.
- El tamaño de cada punto debe ser de **24** bits.
- El valor de compresión debe ser **0**.

Obsérvese que no será necesario utilizar ciertos campos.

Los pixeles se almacenan a partir de la posición indicada por el campo *Inicio de datos de imagen*. Cada pixel se representa por 3 bytes consecutivos que representan el nivel de azul, verde y rojo para ese pixel. Dichos valores estarán siempre en el rango de **0** a **255**.

Al finalizar los píxeles de una fila se dejarán bytes de relleno, de forma que la siguiente fila de píxeles comience al principio de una palabra.

2.3. Difusión gaussiana

La difusión gaussiana tiene por objeto disminuir la nitidez de una imagen y se utiliza como paso previo a la detección de bordes.

En general, se utiliza una matriz cuadrada que actúa como máscara. Para una máscara de 5×5 la imagen resultado res, se obtiene a partir de la máscara m y de la imagen original im aplicando la siguiente expresión:

$$res(i,j) = \frac{1}{w} \sum_{s=-2}^{2} \sum_{t=-2}^{2} m(s+3, t+3) \cdot im(i+s, j+t)$$

donde la matriz de la máscara m y el peso w son:

$$m = \begin{pmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{pmatrix} \qquad w = 273$$

Importante: Tenga en cuenta que al aplicar la máscara en los bordes de la imagen, se considerará que la matriz de la imagen está rodeada de pixeles imaginarios de valor **0**.

2.4. Operador Sobel

El operador de Sobel es un filtro de realce, que ayuda a detectar bordes.
Se utilizarán dos máscaras de tamaño 3×3.
Las matrices de máscara m_x y m_y tendrán los valores:

$$m_x = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \qquad m_y = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

Con estas, se puede expresar los resultados parciales m_x y m_y como:

$$res_x(i,j) = \frac{1}{w} \sum_{s=-1}^{1} \sum_{t=-1}^{1} m_x(s+2, t+2) \cdot im(i+s, j+t)$$

y

$$res_y(i,j) = \frac{1}{w} \sum_{s=-1}^{1} \sum_{t=-1}^{1} m_y(s+2, t+2) \cdot im(i+s, j+t)$$

donde:

$$w = 8$$

Finalmente, la imagen resultado se obtiene como:

$$res(i,j) = |res_x(i,j)| + |res_y(i,j)|$$

3. Tareas

3.1. Desarrollo de versión secuencial

Esta tarea consiste en el desarrollo de la versión secuencial de la aplicación descrita en C++17.

Todos sus archivos fuente deben compilar sin problemas y no deben emitir ninguna advertencia del compilador. En particular, deberá activar los siguientes flags del compilador:

- `-std=c++17 -Wall -Wextra -Wno-deprecated -Werror -pedantic -pedantic-errors`

Tenga también en cuenta que deberá realizar todas las evaluaciones con las optimizaciones del compilador activadas (opciones del compilador `-O3` y `-DNDEBUG`), así como desactivadas (opción del compilador `-O0`).

Importante: No se podrá utilizar ninguna biblioteca externa a la biblioteca estándar de C++.

3.2. Evaluación del rendimiento secuencial

Esta tarea consiste en evaluar el rendimiento de la aplicación secuencial tanto con las optimizaciones del compilador activadas como desactivadas.

Para evaluar el rendimiento debe medir el tiempo de ejecución de la aplicación. Debe representar gráficamente los resultados. Tenga en cuenta las siguientes consideraciones:

- Debe incluir en la memoria todos los parámetros relevantes de la máquina en la que ha ejecutado (modelo de procesador, número de cores, tamaño de memoria principal, jerarquía de la memoria caché, ...) y del software de sistema (versión de sistema operativo, versión del compilador, ...).
  - Se deberá utilizar un computador que disponga de un procesador de al menos 4 núcleos físicos (excluyendo *hyperthreading*).
  - El sistema operativo deberá ser Linux.
  - Tenga en cuenta que la evaluación de la parte secuencial y la parte paralela deberá realizarse en el mismo computador y con las mismas condiciones de entorno.
- Realice cada experimento un número de veces y tome el valor promedio. Se recomienda un mínimo de 10 ejecuciones por experimento.

Represente en una gráfica todos los tiempos totales de ejecución obtenidos para imágenes de distinto tamaño.

Incluya en la memoria de esta práctica las conclusiones que pueda inferir de los resultados. No se limite simplemente a describir los datos. Debe buscar también una explicación convincente de los resultados.

3.3. Paralelización

Esta tarea consiste en desarrollar la correspondiente versión paralela de los programas usando OpenMP. En esta versión se deben paralelizar tanto la etapa de difusión gaussiana como la etapa del operador de Sobel. Deberá explicar en la memoria de forma detallada las modificaciones que ha realizado al código.

Tenga en cuenta lo siguiente:

- Cualquier práctica que emita un advertencia (*warning*) se considerará suspensa.

- Sus programas deben incluir entre los parámetros pasados al compilador, como mínimo, los siguientes: `-std=c++17 -Wall -Wextra -Wno-deprecated -Werror -pedantic -pedantic-errors`, o parámetros equivalentes.

En la memoria debe incluir las decisiones de diseño que ha tomado para alcanzar la paralelización. Para cada decisión debe incluir qué otras alternativa se podían considerar y justificar de forma razonada los motivos de la elección realizada.

3.4. Evaluación de versión paralela

Debe repetir las evaluaciones del primer apartado. Considere distinto número de hilos de ejecución, variando desde 1 hasta 16 hilos (1, 2, 4, 8 y 16).

Nota importante: Asegúrese de compilar todos los ejemplos con las optimizaciones activadas (opciones del compilador `-O3` y `-DNDEBUG`).

Además de las gráficas del primer apartado, represente de forma gráfica el *speedup*.

Incluya en la memoria de esta práctica las conclusiones que pueda inferir de los resultados. No se limite simplemente a describir los datos. Debe buscar también una explicación convincente de los resultados que incluya el impacto de la arquitectura del computador.

3.5. Impacto de la planificación

Realice un estudio para los casos de 4 y de 8 hilos del impacto que tienen los distintos modelos de planificación (*static*, *dynamic* y *guided*) incluidos en OpenMP. Incluya en la memoria las conclusiones que pueda inferir de los resultados obtenidos.

4. Calificación

La puntuación final obtenida en este práctica se obtiene teniendo en cuenta el siguiente reparto:

- Calificación de la implementación: 100 %
  - Implementación secuencial 40 %
    - Pruebas funcionales 30 %
    - Rendimiento alcanzado 30 %
    - Memoria 40 %
  - Implementación paralela 60 %
    - Pruebas funcionales 30 %
    - Rendimiento alcanzado 30 %
    - Memoria 40 %

Advertencias:

- Si el código entregado no compila, la nota final de la práctica será de 0.
- En caso de copia todos los grupos implicados obtendrán una nota de 0.

5. Procedimiento de entrega

La entrega del proyect se realizará a través de Aula Global.
Para ello se habilitarán 3 entregadores separados:

- **Entregador de memoria**. Contendrá la memoria del proyecto, que será un archivo en formato pdf con el nombre **memoria.pdf**.
- **Entregador de código secuencial**: Contendrá todo el código fuente necesario para compilar la versión secuencial.
  - Debe ser un archivo comprimido (formato zip) con el nombre **img-seq.zip**.
- **Entregador de código paralelo**: Contendrá todo el código fuente necesario para compilar la versión paralela.
  - Debe ser un archivo comprimido (formato zip) con el nombre **img-par.zip**.

La memoria no deberá exceder de 15 páginas y deberá contener, al menos, las siguientes secciones:

- **Página de título**: contendrá el nombre de la práctica, el nombre y NIA de los autores y el número de grupo pequeño.
- **Índice de contenidos**.
- **Introducción**.
- **Versión secuencial**: Sección en la que se explicará la implementación y optimización del código secuencial. No se debe incluir código en ella.
- **Versión paralela**: Sección en la que se explicará la implementación y optimización del código paralelo. No se debe incluir código en ella.
- **Evaluación de rendimiento**: Sección en la que se explicarán las diversas evaluaciones realizadas comparando la versión secuencial compilada sin optimizaciones, la versión secuencial compilada con optimizaciones, y la versión paralela optimizada.
- **Pruebas realizadas**: Descripción del plan de pruebas realizadas para asegurar la correcta ejecución de ambas versiones.
- **Conclusiones**.