PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC 3524 — Tópicos Avanzados en Sistemas Distribuidos – 2016/2

Tarea 1 – OpenMP

Fecha de Entrega: Martes 16-Mayo, 23:59
Composición de trabajo: individual
En esta tarea deben utilizar la librería OpenMP para aplicar un filtro a imágenes usando máscaras de convolución.

El Problema

Considere una imágen en formato PNG (Portable Network Graphics), con H pixeles de alto y W pixeles de ancho. Si cada pixel está compuesto por 3 canales para los valores Rojo (R), Verde (G) y Azul (B) con valores entre 0 y 255. Es posible representar la imagen en C usando un arreglo tridimensional: `char image[H][W][3]`.

La aplicación de una máscara lineal de desenfoque (*blur*) consiste en aplicar a cada pixel la siguiente máscara de $M \times N$.

$$K = \frac{1}{M \times N} \times \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

En particular si consideramos una máscara de 3×3 lo que estamos haciendo es reemplazar cada pixel por el promedio los pixeles cubiertos por la máscara. Este proceso se puede aplicar iterativamente incrementando el efecto. Por ejemplo, aplicando 10 veces la máscara de 3×3 a la siguiente imagen, se obtiene:

$$\Rightarrow \frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \Rightarrow$$

Este proceso debe aplicarse a cada pixel y a cada canal RGB.

Versión secuencial

Una versión secuencial de este algoritmo funciona de la siguiente manera:

- Leer una imagen I de dimensiones $H \times W$ en el arreglo `I[H][W][3]`
- En cada iteración r
  - Para cada pixel (i, j) de la imagen I, en la posición `I[i][j]`
    - Para cada canal $k \in \{R, G, B\}$:
      - Calcular el nuevo valor $I'_k(i,j) = \frac{1}{9} \times (I_k(i-1,j-1) + I_k(i-1,j) + I_k(i-1,j+1) + I_k(i,j-1) + I_k(i,j) + I_k(i,j+1) + I_k(i+1,j-1) + I_k(i+1,j) + I_k(i+1,j+1))$
  - Copiar la imagen resultante I' a I.
- La imagen resultante queda en I.

La misión

Debe implementar una versión **secuencial** y una versión **paralela** de este algoritmo **usando OpenMP**, que permita aplicar una máscara cuadrada de desenfoque de tamaño arbitrario a una imagen (arbitraria). Para ello deberá determinar una manera apropiada de paralelizar la ejecución entre múltiples *cores*.

Su programa deberá recibir como argumentos de entrada:

- La imagen a analizar
- La dimensión N de la máscara cuadrada a aplicar (un número impar). El ejemplo usa $N = 3$.
- El número de iteraciones en que se aplicará la máscara

Funcionalidad adicional (opcional): permitir aplicar máscaras personalizadas. Por ejemplo, leyendo la máscara desde otro archivo.

Análisis de rendimiento

Una vez implementada la versión paralela, debe realizar un análisis de rendimiento comparando el tiempo de ejecución, *speedup* y eficiencia para diferentes tamaños de máscara y *threads*. Debe reportar gráficamente estos valores. Debe haber al menos un caso en que la cantidad de *threads* sea mayor a la cantidad *cores* en la máquina a ejecutar.

Para que la comparación sea válida, asegúrese que el algoritmo sea el mismo tanto en la versión secuencial o paralela. Ejecutar algoritmos distintos puede distorsionar los resultados.

Finalmente debe incluir un párrafo donde discuta sus resultados mencionando, por ejemplo, los efectos de crear más *threads* que la cantidad de *cores* disponibles y otros aspectos que considere relevantes.

Restricciones

Puede programar su aplicación en C, C++, ó Python 3.x. Para la lectura de imágenes puede usar una biblioteca *open source* como `opencv`[1] (u otra que sea *open source*). En cualquier caso **NO** puede usar funciones de biblioteca que incorporen automáticamente el paralelismo requerido (tienen que implementarlo).

El programa debe funcionar en un sistema Linux. Se les proveerá una cuenta en una máquina del cluster GRIMA para que puedan probar su programa. Su programa **DEBE** funcionar en los nodos del cluster GRIMA (no vale: "en mi computador funcionaba").

El reporte debe entregarse como un archivo `.pdf`. Debe adjuntar un archivo `readme` con instrucciones de compilación y ejecución. Si es en C, C++, debe incluir un `Makefile`. **NO** entregar código compilado.

Evaluación

- 10 %. Restricciones de entrega (si no se cumplen suficiente condiciones, no se revisa el resto).
- 10 %. Funcionamiento de versión secuencial.
- 50 %. Funcionamiento de versión paralela.
- 30 %. Análisis de rendimiento.
- 10 %. Funcionalidad adicional.

Entrega

La entrega consiste en un archivo `.zip` a través de SIDING que incluya:

- Código fuente
- Instrucciones (readme, Makefile, ...)
- Reporte (.pdf)

[1] http://opencv.org/