



IIC2523 – Sistemas Distribuidos (II/2017)

Tarea 1

Entrega: Viernes 29 de Septiembre, 23:59

1. Objetivo

- Aprender a utilizar nodos *multicore* de un cluster para paralelizar un programa mediante la librería OpenMP.
- Analizar el rendimiento del sistema utilizado en el experimento considerando la arquitectura en que se ejecuta.

2. Descripción

Una imagen es una matriz de pixeles donde cada pixel tiene 3 canales (RGB) que son números del 0 al 255. La matriz de la imagen es tridimensional y de la forma `int imagen[H] [W] [3]` donde H es la altura y W el ancho de la imagen. Un filtro se representa como una matriz de coeficientes que son aplicado a cada celda de una imagen usando una operación de convolución.

En esa tarea deberá aplicar filtros sobre imágenes implementando una versión secuencial, una versión paralela, y analizan el rendimiento de ambas versiones.

La aplicación del filtro sobre la imagen se realiza de la siguiente forma: para cada pixel de la imagen de *input*, el valor del pixel correspondiente en la imagen de *output* corresponde a una ponderación entre el pixel y sus vecinos. Por ejemplo, para el pixel h en cualquier posición de la imagen, y su pixel correspondiente h' de la imagen resultante:

$$\begin{bmatrix} \dots & \vdots & \vdots & \vdots & \vdots & \ddots \\ \dots & x & y & z & \dots \\ \dots & w & h & k & \dots \\ \dots & l & m & n & \dots \\ \ddots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} * \begin{bmatrix} 0 & 2 & 0 \\ -1 & 8 & -1 \\ 0 & 1 & 0 \end{bmatrix} \implies h' = 2y - w + 8h - k + m$$

Entonces, centramos la matriz en el pixel, se multiplican sus vecinos y el pixel por el valor correspondiente en la matriz del filtro (también conocido como *kernel*) y se suman. Dado que cada pixel es un vector de 3 componentes, se pueden aplicar las operaciones normales de suma de vectores. Esto se conoce como *convolución* de matrices y tiene muchas otras aplicaciones en distintas áreas de la computación y el procesamiento de señales.

Hay distintas técnicas que se pueden utilizar para casos bordes como cuando los vecinos de un pixel estén fuera de la matriz. Para esta implementación dejamos a su criterio aplicar una acción razonable, por ejemplo, proyectar los valores de los pixeles.

3. Ejemplo

Tenemos la siguiente imagen en formato PNG.



Ella es Liliana, queremos aplicarle un filtro a Liliana para que se vea aún más poderosa.



Luego de 40 convoluciones del filtro *blur*, Liliana se ve distinta, tal vez no mucho más poderosa, pero el efecto del filtro es evidente. Afortunadamente sus habilidades de nigromante siguen intactas.

4. Ejecución

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 con que se aplicará la máscara

Deberá efectuar experimentos con distintos tamaños de filtros y cantidad de *threads*. Es importante que los experimentos sean ejecutados en un ambiente *multicore*.

Se le dará acceso a algunos nodos *multicore* del cluster GRIMA. Las instrucciones de acceso se enviarán a cada usuario. Si bien puede ejecutar sus experimentos en otra infraestructura a la que tenga acceso, su entrega debe ser reproducible en un nodo del cluster GRIMA.

5. Análisis de Rendimiento

Un aspecto importante a medir al momento de parallelizar un cómputo es calcular la ganancia que se obtiene. Para evaluar esto, debe ejecutar su implementación paralela con distintas cantidades de *threads*, y tomar nota de los tiempos de ejecución. Luego de eso, debe calcular las siguientes métricas:

- **Speedup**, o aceleración. Se define como la razón entre el tiempo que demora la versión secuencial, t_{seq} , y el tiempo que toma la versión paralela al ejecutarla con p *threads*, t_p . Es una magnitud sin unidades.

$$S_p = \frac{t_{seq}}{t_p}$$

- **Eficiencia** es una medida de cuánto se ha aprovechado el hecho de tener más unidades de procesamiento para mejorar el tiempo de ejecución de la aplicación. Se calcula como la razón entre el *speedup* alcanzado y la cantidad de *threads* utilizados, p . Debe ser un valor entre 0 y 1, aunque bajo ciertas condiciones podría salir de ese rango.

$$E_p = \frac{S_p}{p}$$

6. Informe

Como resultado de su experimento debe entregar un informe que contenga lo siguiente:

- Descripción breve de la solución implementada.
- Descripción de los experimentos realizados, incluyendo tamaños de imágenes, de filtros, número de iteraciones, cantidad de *threads*, y características física de la máquina utilizada (#cores, tamaño de memoria).
- Análisis de rendimiento detallando tiempo de ejecución, *speedup* y eficiencia para diferentes tamaños de filtros y cantidad de *threads*. Este análisis se debe apoyar utilizando gráficos pertinentes.
- Descripción de las observaciones efectuadas en relación a las características del nodo utilizado para esta tarea.

Es necesario que ejecuten en algún momento tests con menor, igual y mayor cantidad de *threads* que la cantidad de *cores* disponibles en la máquina.

7. Restricciones

La tarea debe ser programada en C utilizando únicamente las características de la biblioteca OpenMP para parallelizar. El algoritmo secuencial debe ser el mismo que el paralelizado ya que el trabajo consiste en distribuir un programa que previamente no lo estaba.

El filtro será cuadrado y dimensión impar. Si el *kernel* se sale de la imagen, puede realizar alguna técnica que desee para proyectar los bordes. Una técnica válida es que los espacios que queden fuera de la imagen se les asigne los valores de los pixeles más próximos que están dentro de la imagen.

Sus programas deben poder ser compilados y ejecutados en las máquinas del cluster a las que se les dio acceso.

8. Evaluación

- 10 %. Aspectos formales: entrega correcta, instrucciones de compilación apropiadas detalladas en un archivo **README**, en formato **txt** o **markdown**.
- 10 %. Descripción de experimentos.
- 10 %. Funcionamiento correcto de la versión secuencial.
- 35 %. Funcionamiento correcto de la versión paralela.
- 35 %. Análisis de rendimiento y descripción de los resultados observados.

9. Entrega

Independientemente de la plataforma en que haya ejecutado sus experimentos, la entrega debe realizarse en una carpeta de nombre **exactamente T1** en su directorio *home*. **NO** debe incluir archivos ejecutables (compilados).

Contacto

- Vicente Rafael Dragicevic (vrdragicevic@uc.cl)
- Pilar Ignacia Jadue (pijadue@uc.cl)
- Francesca Lucchini (flucchini@uc.cl)

10. Integridad académica

Los alumnos de la Escuela de Ingeniería de la Pontificia Universidad Católica de Chile deben mantener un comportamiento acorde a la Declaración de Principios de la Universidad. En particular, se espera que mantengan altos estándares de honestidad académica. Cualquier acto deshonesto o fraude académico está prohibido; los alumnos que incurran en este tipo de acciones se exponen a un Procedimiento Sumario. Es responsabilidad de cada alumno conocer y respetar el documento sobre Integridad Académica publicado por la Dirección de Docencia de la Escuela de Ingeniería.

Específicamente, para los cursos del Departamento de Ciencia de la Computación, rige obligatoriamente la siguiente política de integridad académica. Todo trabajo presentado por un alumno para los efectos de la evaluación de un curso debe ser hecho individualmente por el alumno, sin apoyo en material de terceros. Por “trabajo” se entiende en general las interrogaciones escritas, las tareas de programación u otras, los trabajos de laboratorio, los proyectos, el examen, entre otros. Si un alumno copia un trabajo, obtendrá nota final 1,1 en el curso y se solicitará a la Dirección de Docencia de la Escuela de Ingeniería que no le permita retirar el

curso de la carga académica semestral. Por “copia” se entiende incluir en el trabajo presentado como propio partes hechas por otra persona.

Obviamente, está permitido usar material disponible públicamente, por ejemplo, libros o contenidos tomados de Internet, siempre y cuando se incluya la referencia correspondiente.

Lo anterior se entiende como complemento al Reglamento del Alumno de la Pontificia Universidad Católica de Chile. Por ello, es posible pedir a la Universidad la aplicación de sanciones adicionales especificadas en dicho reglamento.