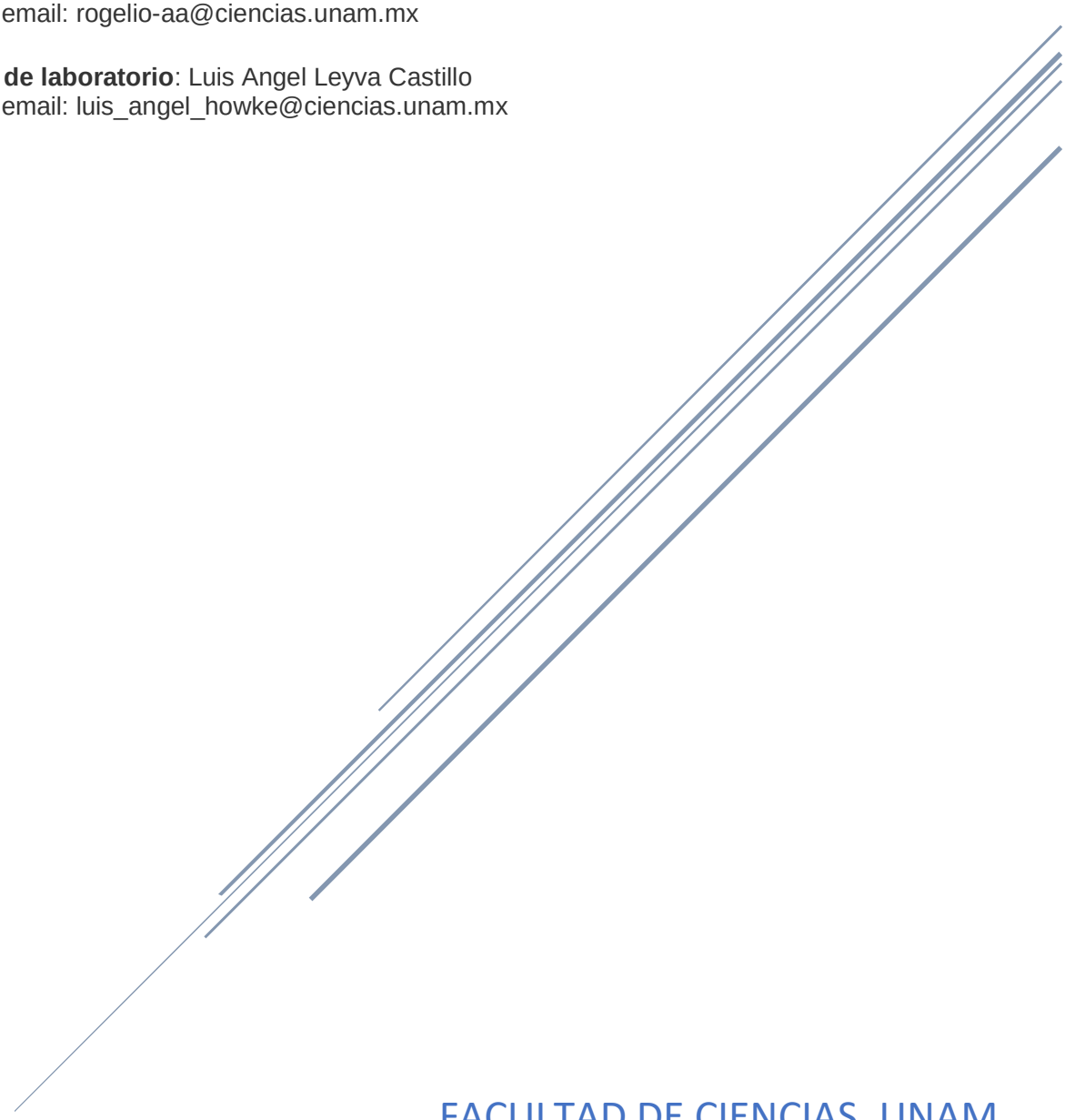
PRACTICA 1

Introducción a Hilos

Profesor: Salvador Gonzalez Arellano.
email: salvador_gonzalez_a@ciencias.unam.mx

Ayudante de teoría: Rogelio Alcantar Arenas.
email: rogelio-aa@ciencias.unam.mx

Ayudante de laboratorio: Luis Angel Leyva Castillo
email: luis_angel_howke@ciencias.unam.mx



FACULTAD DE CIENCIAS, UNAM
Computación Concurrente

INTRODUCCIÓN

Los sistemas operativos modernos se basan en el principio de la multiprogramación, es decir, en la posibilidad de manejar distintos hilos de ejecución de manera simultánea con el objetivo de paralelizar el código e incrementar el rendimiento de la aplicación.

Esta idea también se plasma a nivel de lenguaje de programación. Algunos ejemplos representativos son los APIs de las bibliotecas de hilos Pthread, Win32, Java, C, C++, C#, Go, Ruby, etc. Incluso existen bibliotecas de gestión de hilos que se enmarcan en capas software situadas sobre la capa del sistema operativo, con el objetivo de independizar el modelo de programación del propio sistema operativo subyacente.

La idea de hilo surge de la posibilidad de compartir recursos y permitir el acceso concurrente a esos recursos. La unidad mínima de ejecución pasa a ser el hilo, asignable a un procesador. Los hilos tienen el mismo código de programa, pero cada uno recorre su propio camino con su PC, es decir, tiene su propia situación aunque dentro de un contexto de compartición de recursos.

Informalmente, un hilo se puede definir como un proceso ligero que tiene la misma funcionalidad que un proceso pesado, es decir, los mismos estados. Por ejemplo, si un hilo abre un fichero, éste estará disponible para el resto de hilos de una tarea.

Las ventajas de la programación multihilo se pueden resumir en las tres siguientes:

- Capacidad de respuesta, ya que el uso de múltiples hilos proporciona un enfoque muy flexible. Así, es posible que un hilo se encuentra atendiendo una petición de E/S mientras otro continúa con la ejecución de otra funcionalidad distinta. Además, es posible plantear un esquema basado en el paralelismo no bloqueante en llamadas al sistema, es decir, un esquema basado en el bloqueo de un hilo a nivel individual.
- Compartición de recursos, posibilitando que varios hilos manejen el mismo espacio de direcciones.
- Eficacia, ya que tanto la creación, el cambio de contexto, la destrucción y la liberación de hilos es un orden de magnitud más rápida que en el caso de los procesos pesados. Recuerde que las operaciones más costosas implican el manejo de operaciones de E/S. Por otra parte, el uso de este tipo de programación en arquitecturas de varios procesadores (o núcleos) incrementa enormemente el rendimiento de la aplicación.

EJERCICIOS

1.- Multiplicación de Matrices Secuencial

Realiza un programa que realice la multiplicación de matrices.

Considera que seran matrices de tamaño n x n

De entrada seran 2 matrices de tamaño n x n y el retorno sera la matriz resultante.

2.- Multiplicación de Matrices Concurrente

Es posible realizar el producto de dos matrices usando varios hilos de forma independiente. Primero recordemos cómo se obtiene el producto de dos matrices:

Sean A = (a_{ij}) una matriz de n x r y B = (b_{ij}) una matriz de r x m. La matriz producto C = AB es una matriz de tamaño n x m que está definida por las entradas

$$c_i j = \sum_{k=1}^{r} a_{ik} * b_{kj}$$

Es decir, el elemento c_{ij} es igual al producto punto entre la i-ésima fila de A y la j-ésima columna de B. Directamente de esta definición obtenemos un algoritmo secuencial para calcular AB, donde el cálculo principal es el producto punto, que ser vería simila al siguiente código:

```
For (k = 0; k< r; ++k){
        C[i][j] += A[i][k] * B[k][j]
}
```

Para calcula el producto de forma concurrente usando varios hilos de ejecución, podemos asignar parejas de la forma (i,j) a cada hilo y este se encargará de obtener la entrada c_{ij}.

Para el caso de una matriz de n x n tenemos la opción de partir en bloques (submatrices) y luego hacer que cada hilo se encargargue de calcular las entradas de un bloque.

En pocas palabras una manera de resolverlo es asignarle un hilo a cada renglon y que este se ejecute.

Realiza un programa que realice la multiplicación de matrices con hilos.

Considera que seran matrices de tamaño n x n

De entrada seran 2 matrices de tamaño n x n y el retorno sera la matriz resultante.

3.- Filtros de Manera Secuencial

La edición de imágenes se ha vuelto muy popular ultimamente, por lo que se han creado diversos algoritmos para crear nuevos efecto, ya sea para agregar más lumonisidad a la imagen, detallar alguna sombra, o que se marque un contorno. En este instante se les presentaran algunos filtros de los cuales deberan implementar 5 de los 10, sera 2 faciles, 2 intermedios y 1 dificil.

GRISES (FACILES)

- Promedio, tomamos cada pixel, sumamos sus componente RGB (es decir sumamos Rojo, Verde y Azul) y lo divididimos entre 3 (Gray = (Red+Green+Blue)/3 y esto se lo asignamos a cada pixel, el resultado es bueno.
- Correctud, obtenemos cada componente RGB y aplicamos la siguiente formula Gray = (Red*.03+Green0.59+Blue*0.11), el resultado es un tanto mejor que en el anterior, mas cuando las imágenes tienen mucha luz, a este filtro tambien se le conoce como LUMA
- Correctud2, siempre se puede mejorar, es basicamente lo mismo que el anterior, la formula a usar ahora es Gray = (Red*.2126+Green0.7152+Blue*0.0722)
- Single Color, en este caso solo obtenemos una componente y ese sera nuestro valor de gris, es decir, si eligen usar Single Color Red, el color sera Gray = pixel.getRed() y ese se asigna a las componentes de dichi pixel

Blurs (Dificiles)

- Blur, este filtro lo que hace es dar un poco la imagen en movimiento o como si se desfazara la imagen, lo que se hace es aplicar la siguiente matriz a cada pixel de la imagen, la manera de aplicarla es tomando los pixeles dadas las coordenadas y obtener sus componentes, posteriormetne se multiplica por la casilla de la matriz para obtener el valor nuevo, al final se asigna al nuevo pixel (ver ejemplo al final de estos filtros)

```
double blur[][] = {
        {0.0,0.2,0.0},
        {0.2,0.2,0.2},
        {0.0,0.2,0.0}
};
```

- Blur2, siempre se puede mejorar mucho más el efecto, basicamente es lo mismo que el anterior, lo que cambia es la matriz, al final dividimos entre 13 cada pixel, pues la suma de las componentes de la matriz debe dar 1

```
double [][] blur = {
    {0,0,1,0,0},
    {0,1,1,1,0},
    {1,1,1,1,1},
    {0,1,1,1,0},
    {0,0,1,0,0}
};
```

- Motion Blur, ahora usaremos el conocido Motion Blur, da un efecto mucho mas fuerte y se nota mucho mejor, en princpio es lo mismo que el anterior, , al final dividimos entre 13 cada pixel, pues la suma de las componentes de la matriz debe dar 1.

```
double blur[][] ={
    {1, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 1, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 1, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 1, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 1, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 1, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 1, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 1, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 1}
};
```

Varios (Medios)

- Sharpen, de un efecto como si estuviera dibujado con pluma, usamos la siguiente matriz

```
int sharpen[][] = {
    {-1,-1,-1},
    {-1,9,-1},
    {-1,-1,-1}
};
```

- Componentes RGB, lo que se hace es aplicar una capa del color que indiquemos, los parametros son int rojo, int verde, int azul, lo que se hace es sumar directamte los valores que damos a cada componente, es decir, r = pixel.getRed() + rojo, para el componente rojo, el resultado es el siguiente

Se le asigno 50 de rojo extra, 0 de verde y 25 de azul

- Altro contraste, es un filtro relativamten sencillo, el primer paso es pasar a grises (usando cualquier filtro), de ahí si el valor de rojo verde y azul es mayor a 127 entonces asignamos 255 en otro caso 0, al final el pixel tendra o 0 o 255

Filtros De Matrices

Como tal para hacer estos filtros es seguir los siguientes pasos:

1. Tendremos que recorrer toda la imagen
2. Creamos 3 variables rojo, verde y azul
3. Dependiendo la matriz a usar, sera el numero de pixeles a usar (en este caso usaremos la de 3x3), el pixel central sera el pixel en el que nos encontremos (usando coordenadas x,y este corresponde a estas), si el pixel no es el central lo que tenemos que hacer es localizar su ubicación de este, si nos encontramos en una esquina deberemos aplicar el modulo para poder adquirir dicho pixel)
4. Si la suma de los valores de la matriz es igual a 1, el ultimo paso es sumar los valores
5. La suma de cada valor sera asignado al nuevo pixel para generar la imagen nueva
6. Se repite para cada pixel de toda la imagen (reiniciando los valores de rojo, verde y azul en 0)

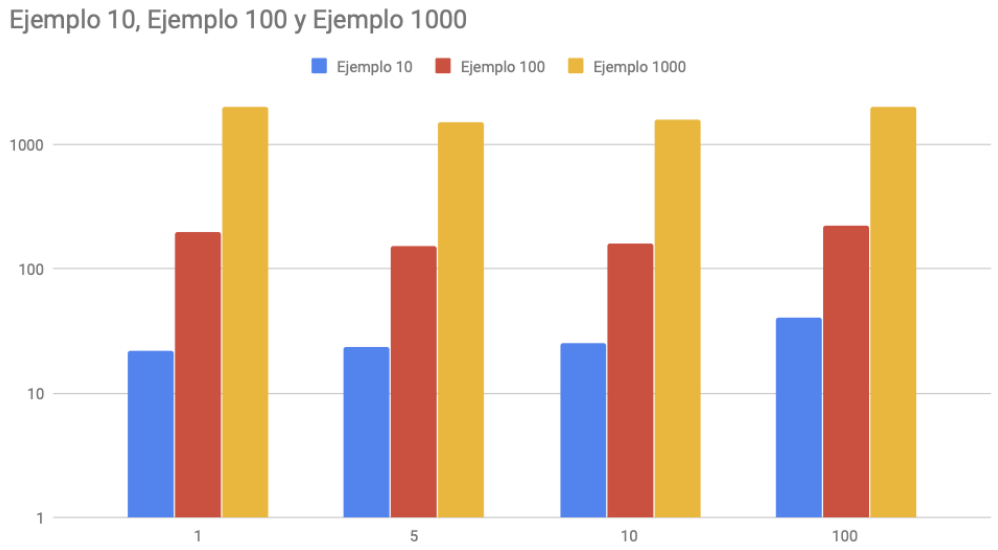
4.- Filtros de Manera Concurrente

La manera de hacer concurrente los filtros es similar a la multiplicación de matrices, solamente le asignamos una submatrices (o renglon) a cada hilo para que este ejecute el proceso.

TEORÍA

Para esta parte es necesario entregar un reporte con los siguientes resultados:

Usa los archivos de prueba incluidos en la carpeta matrices para probar tu programa (o puedes llenarlas de manera aleatoria). Se incluyen matrices de tamaños 10×10, 100×100 y 1000×1000. Para cada prueba repite el cálculo 20 veces y obtén el promedio del tiempo tardado, tanto en la opción secuencial como la opción concurrente. Con estos datos elabora una gráfica que compare tamaño de matriz frente a tiempo promedio. La gráfica debe incluir tanto los datos de la opción secuencial como los de la opción concurrente. En la figura 1 se muestra un ejemplo.



Donde 1, 5, 10 y 100 son el número de hilos usados.

Repite el mismo procedimiento para la parte de los filtros.

En cada ejercicio se hará una comparación entre los tiempos de ejecución de la forma secuencial y la forma concurrente. Además, la aceleración obtenida en la solución concurrente será comparada con lo que nos dice la ley de Amdahl.

Para hacer las comparaciones, realiza una tabla de la siguiente forma:

# Hilos	Aceleración Teórica	Aceleración Obtenida	% Código en paralelo

Posteriormente contesta las siguientes preguntas:

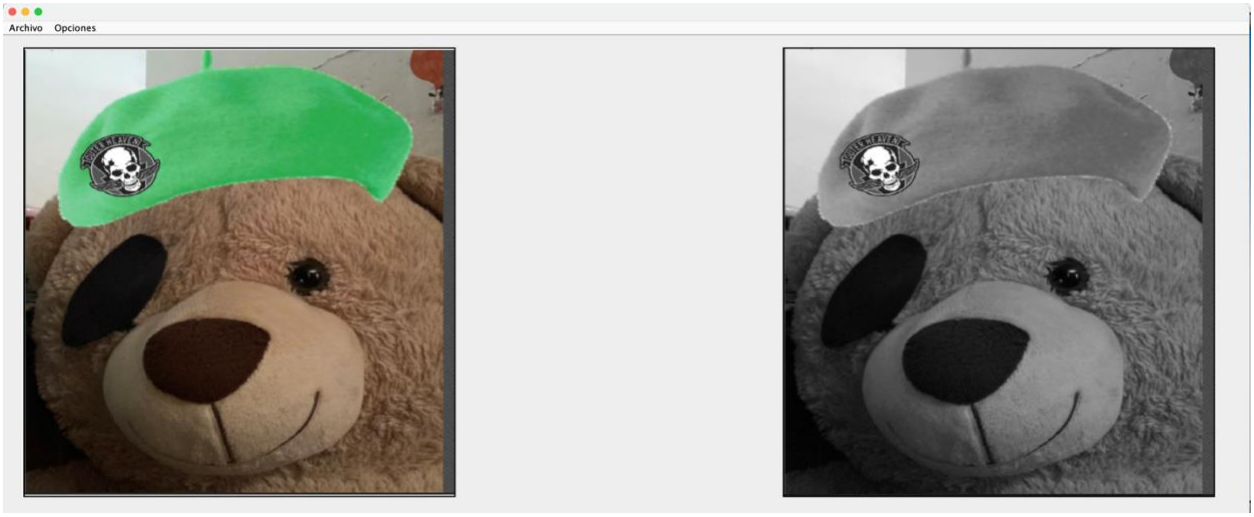
1. ¿Hubo una mejora significativa?
2. ¿Crees que si agregamos mas nucleos a nuestro CPU mejore el rendimiento o sera mejor aumentar la frecuencia de reloj? Justifica.
3. ¿Qué pasaria si tuvieramos una cantidad infinita de hilos, mejoraria la ejecución o no? ¿Hasta que nivel de mejora obtendremos?

PUNTOS EXTRA

Para obtener los puntos extra realiza lo siguiente:

1. Compara tus resultados y de tus compañeros de equipo al ejecutar, realiza una tabla donde muestre el tiempo de ejecución y las especificaciones de cada computadora utilizada (CPU, SO, RAM, MB, etc)
2. Realiza todos los filtros tanto de maner secuencial como de manera paralela
3. Realiza una interfaz gráfica para visualizar los filtros

Ejemplo:



ENTREGABLE

Para la parte práctica, que es la programación, los métodos deben de ir en un mismo archivo, también deben de poner un método main con algunas pruebas de sus métodos, todo debe ir debidamente documentado.

Para la parte teórica, es necesario que se haga a computadora, en el editor de su elección. Deben de poner las referencias bibliográficas en donde consultaron la información, esta debe de ir en formato APA.

Se les dará 0.5 extra si la realizan en LaTeX.

Todo debe de ir comprimido en un archivo ZIP, Taro cualquiera de tu preferencia, este debe de llevar el siguiente formato:

[NOMBRE DEL EQUIPO.[EXTENSION]

EJEMPLO:

ENCHILADASDEYAKULT.zip