Parcial No. 1
Programación Paralela

Pontificia Universidad JAVERIANA Cali

Descripción General

La siguiente lista de desafíos fue diseñada para poner en práctica los conocimientos adquiridos en el curso de programación paralela. Con este examen se logra:

1. Identificar varios aspectos relacionados con la programación paralela
2. Analizar un problema y generar una solución usando programación paralela
3. Utilizar librerías/bibliotecas para paralelización de memoria compartida y distribuida
4. Determinar cuáles ventajas y desventajas tiene una solución paralela

Pregunta de Interés

¿Cómo influye la programación paralela, utilizando OpenMP ó Pthreads y MPI (*En un entorno simulado con docker*), en el tiempo de ejecución del filtrado de imágenes PPM y PGM en comparación con una implementación secuencial?

Los archivos de imágenes en formato **PPM** (Portable Pixmap) y **PGM** (Portable Graymap) son simples, sin comprimir e ideales para entender los conceptos básicos del procesamiento de imágenes. Además, el contenido suele ser texto plano o binario sin procesar, esto permite la fácil transferencia, lectura e incluso entendimiento del contenido y su presentación visual.

Aplicación base

Desarrollar un programa en C++ que pueda leer y procesar imágenes en formato **PPM** (Portable Pixmap) y **PGM** (Portable Graymap). Utilice el paradigma orientado a objetos de tal manera que pueda extender la capacidad a medida que se consideren otros requerimientos.

1. El formato de los archivos PPM y PGM es el siguiente:
    - 1.1. **Línea 1:** El "número mágico", que es P3 para PPM y P2 para PGM.
    - 1.2. **Línea 2:** Los comentarios, que son opcionales y comienzan con #.
    - 1.3. **Línea 3:** El ancho y el alto de la imagen, separados por un espacio, en píxeles.
    - 1.4. **Línea 4:** El valor máximo de color (generalmente 255).
    - 1.5. **Líneas siguientes:** Los valores de los píxeles.
        - 1.5.1. Si es P2, cada píxel se representa con un número (intensidad).
        - 1.5.2. Si es P3, cada píxel se representa con tres números
2. Debe cargar archivos de color (PPM) o escala de grises (PGM) de forma automatica no interactiva.
3. La lectura se realizará desde la entrada estándar (**stdin**).

```
./processor lena.ppm lena2.ppm ##lectura desde la entrada estándar
```

4. Utilice como base este repositorio para comprender la lectura/escritura de estos archivos https://github.com/japeto/netpbm_filters/tree/main

Versión secuencial

Los filtros de procesamiento de imágenes son operaciones que modifican los valores de los píxeles y logran un efecto visual. Modifique la aplicación base de tal forma que permita uno o varios filtros.

1. Implemente los filtros: suavizado (*blur*), laplace y realce (*sharpening*)
2. Ponga a prueba su implementación con las entradas lena, fruit y puj

```
./filterer fruit.ppm fruit_blur.ppm --f blur ## filtro blur a fruit
```

3. Mida y registre el tiempo en CPU y de ejecución total que le toma al computador realizar el filtrado
4. Utilice como base este repositorio para comprender los filtros y su aplicación. https://github.com/japeto/netpbm_filters/tree/main

Memoria compartida

Utilice pthreads para implementar la paralelización de los filtros. Modifique la versión secuencial para que:

1. Se divida un archivo de entrada en cuatro regiones, arriba-izquierda, arriba-derecha, abajo-izquierda y abajo-derecha
2. Se entregue a los hilos una de las regiones de la imagen y el filtro a aplicar
3. Ponga a prueba su implementación con las entradas damma y sulfur

```
./pth_filterer fruit.pgm fruit_blur2.ppm --f blur ## filtro blur a fruit
```

4. Mida y registre el tiempo en CPU y de ejecución total que le toma al computador realizar el filtrado

Utilice OpenMP para implementar la paralelización de los filtros. Modifique la versión secuencial para que:

1. Se apliquen los filtros paralelo a una imagen de entrada
2. Ponga a prueba su implementación con las entradas damma y sulfur

```
./omp_filterer sulfur.pgm sulfur_N.pgm ## filtros a sulfur
```

3. Mida y registre el tiempo en CPU y de ejecución total que le toma al computador realizar el filtrado

Memoria distribuida

Implemente un esquema de paso de mensajes entre contenedores Docker con MPI.

1. Modifique la versiones secuencial y usando MPI permita que los nodos (hosts) intercambien imágenes y apliquen los tres filtros.
2. Ponga a prueba su implementación con al menos dos de las entradas

3. Mida y registre el tiempo en CPU y de ejecución total que le toma a cada nodo realizar el filtrado

Recomendaciones

1. Asegúrese de que su código es modular y sigue el paradigma orientado a objetos
2. Utilice arreglos en lugar de vectores y punteros a Char en lugar de strings
3. Revisé los enlaces y busqué información adicional sobre el formato de los archivos de entrada y sus características.

Entrega

1. Los archivos de código deben ser publicados a través de un repositorio de acceso público. Puede ser un fork de https://github.com/japeto/netpbm_filters/tree/main
2. Cree un documento a manera de informe donde presente sus hallazgos y responda a la pregunta de interés de acuerdo a lo que encontró en las mediciones de tiempo.
3. Cree un video presentando la implementación y el desarrollo de cada uno de los puntos del parcial (Puede utilizar su informe y/o código para guiar la presentación)