Taller 3:
Procesamiento de imágenes

El objetivo de este ejercicio es escribir un programa paralelo con MPI que agilice un tipo de procesamiento de imágenes. Una imágen en escala de grises puede representarse como un arreglo bidimensional de enteros que indican la intensidad del pixel. Esto puede darnos la idea de partir la imagen en regiones o dominios, de manera que cada proceso pueda realizar las operaciones necesarias sobre el dominio de la imagen que le corresponde. Por simplicidad, descompondremos las imágenes en una sola de las dimensiones.

1. Detección de contornos

El contorno de una figura en una imagen demarca una diferencia entre los pixeles que pertenecen a la figura y los que pertenecen al fondo de la imagen. Una manera simple de detectar contornos es buscando pixeles que sean suficientemente diferentes de sus vecinos y resaltandolos en blanco. Para esto, partiendo de la imagen original en escala de grises representada por la matriz de enteros $A_{i,j}$, se puede redefinir la intensidad (un valor entero) de cada pixel ($I_{i,j}$) de la siguiente manera:

$$I_{i,j} = A_{i-1,j} + A_{i+1,j} + A_{i,j-1} + A_{i,j+1} - 4A_{i,j}, \tag{1}$$

de manera que si un pixel es igual a sus vecinos su intensidad será 0 y se verá negro. Para detectar los bordes exteriores de la imagen de dimensión $M \times N$ (M filas, N columnas) tomaremos los pixeles en el rango $1 \leq i \leq M$ y $1 \leq j \leq N$ como pertenecientes a la imagen y tendremos un marco de pixeles blancos (valor 255 en 8 bits) para los pixeles que tengan coordenadas $i = 0$, $i = M + 1$, $j = 0$ o $j = N + 1$. A éste marco lo llamaremos el halo. Un ejemplo del resultado de este proceso se muestra en la figura 1.

2. Invirtiendo la operación

El proceso inverso de aproximar la imagen original a partir de los contornos es posible en este caso. Sin embargo es un proceso mas dispendioso que el original, que requiere hacer múltiples iteraciones. Esto lo hace más apropiado para

Figura 1: .

observar los efectos de la paralelización.

Los archivos `edgeMxN.pgm` suministrados contienen la información de los pixeles del contorno de una imagen. Para poder visualizarlos puede normalizarlos. Esto se logra con las funciones en la librería `pgmio.hpp` que se ejecuta desde el archivo `normalize.cpp`. El archivo `out.pgm` producido puede visualizarse como una imagen en escala de grises.

La tarea será producir una aproximación a la imagen original a partir de la imagen de contorno `edgeMxN.pgm`.

La imagen puede obtenerse iterando N veces el proceso inverso para cada pixel

$$New_{i,j} = \frac{1}{4}(Old_{i-1,j} + Old_{i+1,j} + Old_{i,j-1} + Old_{i,j+1} - I_{i,j}), \qquad (2)$$

en donde Old y New son las matrices al inicio y al final de cada iteración, y se parte de una imagen totalmente blanca. Recuede que los bordes de la imagen deben permanecer siempre blancos (255).

El código serial se presenta en el archivo `reconstruct_ser.cpp`. Note que el arreglo 2D se guarda como un arerglo de 1D en donde las filas vienen una después de la otra, y el elemento i, j de la matríz aumentada (para incluir el halo) de dimensión $(M+2)\times(N+2)$ se accede como $i*(N+2)+j$, $0 \leq i \leq M+1$, $0 \leq j \leq N+1$.

3. Paralelización

La estrategia consite en dividir el némero total de filas de la imagen entre el número de procesos a ejecutar. Cada proceso se encargará de sus filas, pero hay que tener en cuenta que en las fronteras entre región y región un proceso debe tener en cuenta los valores actualizados de la frontera de sus vecinos.

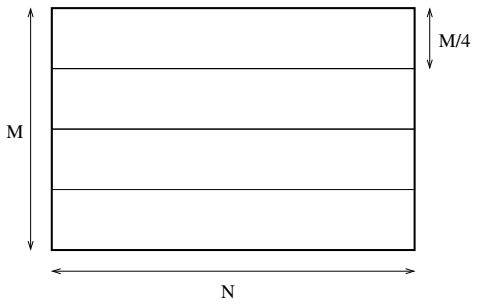

Figura 2: .