GPGPU – Laboratorio 3

En este laboratorio usaremos la GPU para procesar una imagen.
Antes de comenzar descargar el proyecto de GitLab o el archivo comprimido desde la plataforma EVA.

```
git clone https://gitlab.fing.edu.uy/cursogpu/LabGPU02.git
```

Compilar con

```
make
```

El ejecutable generado se llama “blur” y recibe como parámetro un archivo de imagen en formato .ppm
Modifique el script que utilizó en el práctico anterior para encolar el trabajo en el cluster y ejecute el programa usando alguna de las imágenes de ejemplo.

El programa debería producir dos imagenes de salida `output_brillo.ppm`, y `output_blur.ppm`, la cual debe transferir a su estación de trabajo para poder visualizarla.

Ejercicio 1

La función `ajustar_brillo_cpu(...)` recorre la imagen sumando un coeficiente entre -255 y 255 a cada píxel, aumentando o reduciendo su brillo.

a) Construir la función `ajustar_brillo_gpu(...)` y los kernels correspondientes para realizar esta tarea en la GPU.

Configure adecuadamente la grilla (bidimensional) de threads para aceptar matrices de cualquier tamaño. En el kernel, utilice las variables blockIdx y threadIdx adecuadamente para acceder a la estructura bidimensional.

La recorrida de la imagen debe realizarse en dos versiones:

- En `ajustar_brillo_coalesced_kernel`, threads con índice consecutivo en la dirección x deben acceder a pixels de una misma fila de la imagen
- En `ajustar_brillo_no_coalesced_kernel`, threads con índice consecutivo en la dirección x deben acceder a pixels de una misma columna de la imagen

b) Registre los tiempos de cada etapa de la función `ajustar_brillo_gpu` (reserva de memoria, transferencia de datos, ejecución del kernel, etc.) para las dos variantes. ¿Nota algún cambio?

c) Compare los resultados con la salida de la herramienta `nvprof.` Registre dicho tiempo mediante “nvprof --profile-api-trace none --metrics gld_efficiency ./blur imagen.ppm” ¿Qué puede decir del resultado de la métrica gld_efficiency?

Ejercicio 2

Construiremos un filtro Gaussiano para reducir el ruido de una imagen en escala de grises. Consiste en sustituir el valor de intensidad de cada pixel por un promedio ponderado de los pixels vecinos. Los pesos por los cuales se pondera cada vecino en el promedio se almacenan en una matriz cuadrada (máscara).

a) Construya el kernel que aplica el filtro en la GPU y la función que invoca dicho kernel y transfiere los datos correspondientes hacia y desde la GPU.

b) Registre los tiempos de cada etapa de la función y compare las variantes de CPU y GPU. ¿Qué aceleración se logra? ¿Y considerando únicamente el tiempo del kernel?