Procesamiento paralelo de imágenes con Python y Numba
Proyecto Final – Programación Concurrente (GPU)

“Aplicación práctica de programación paralela con GPU”

ALUMNOS: JESICA LENCINA

Universidad Nacional Guillermo Brown

En este proyecto se implementó un sistema de procesamiento de imágenes utilizando Python y Numba, con el objetivo de aplicar los conceptos de programación concurrente y paralelismo en GPU.

VOLVER A INICIO

INTRODUCCIÓN

EL PROPÓSITO PRINCIPAL FUE DEMOSTRAR CÓMO LA PARALELIZACIÓN PUEDE MEJORAR EL RENDIMIENTO EN TAREAS DE ALTO COSTO COMPUTACIONAL, COMO EL PROCESAMIENTO DE IMÁGENES.

A TRAVÉS DEL USO DE LIBRERÍAS COMO OPENCV Y MATPLOTLIB, SE PROCESARON IMÁGENES APLICANDO FILTROS Y COMPARANDO LOS RESULTADOS OBTENIDOS EN CPU Y GPU.

Metodología y Desarrollo

El desarrollo del proyecto se realizó en Visual Studio Code, utilizando un entorno de programación en Python.

Se creó una estructura de carpetas organizada con los siguientes elementos:

📁 *src/ → Archivos de código (cpu_version.py y gpu_version.py)*
📁 *imagenes/ → Imagen de entrada (mariposa color)*
📁 *result/ → Resultados generados*

El código se divide en dos partes:
Versión CPU: procesa la imagen de forma secuencial, aplicando un filtro de escala de grises.

Versión GPU: aplica el mismo procesamiento utilizando Numba para paralelizar el cálculo en la tarjeta gráfica.

Durante la ejecución, se midió el tiempo de procesamiento de cada versión para analizar el rendimiento.

Resultados

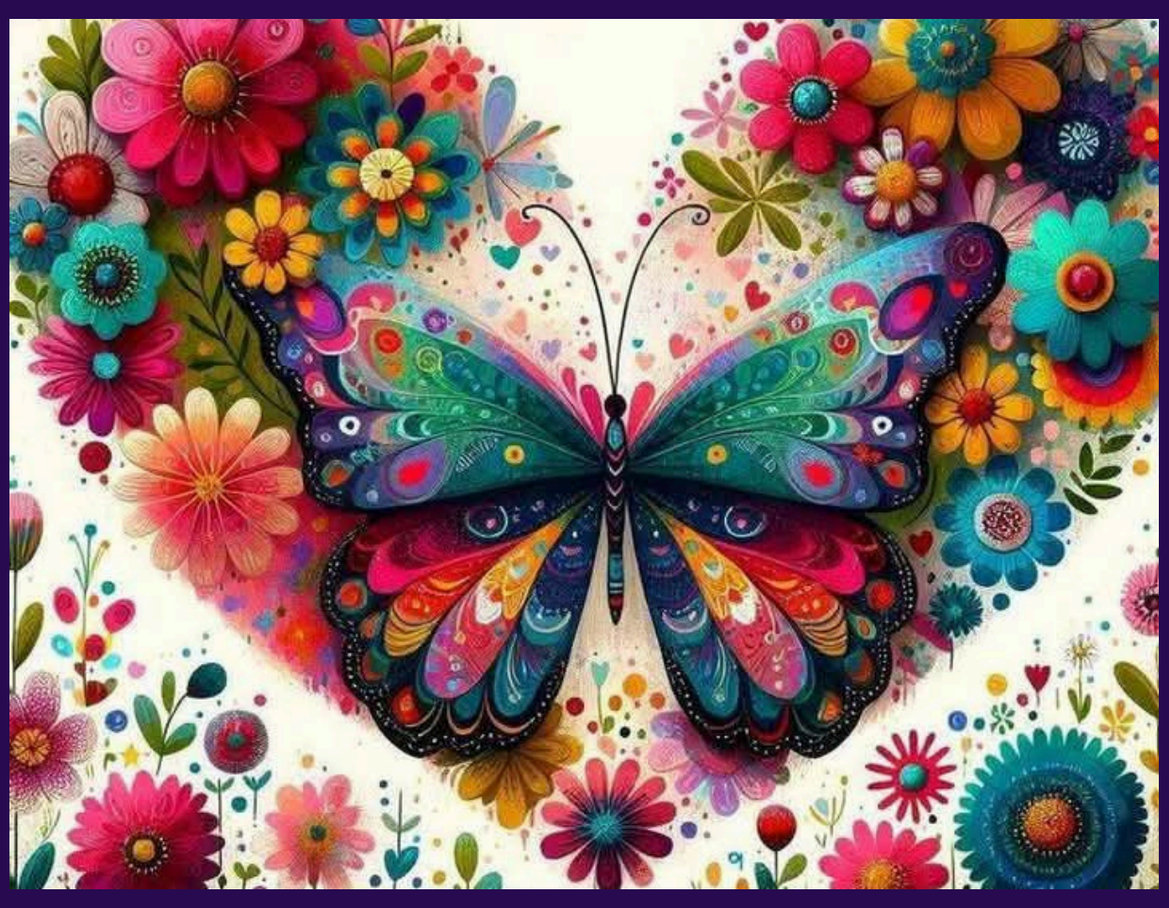
A continuación se observan los resultados obtenidos tras ejecutar el código en CPU y GPU.

🖼️ *Figura 1. Imagen original (arriba) y procesada en escala de grises (abajo).*

Tiempos de ejecución:

CPU secuencial: 0.0006 segundos
CPU paralela (Numba): 3.8401 segundos

El sistema logró procesar correctamente la imagen, demostrando el funcionamiento de la conversión a escala de grises tanto en modo secuencial como en paralelo.



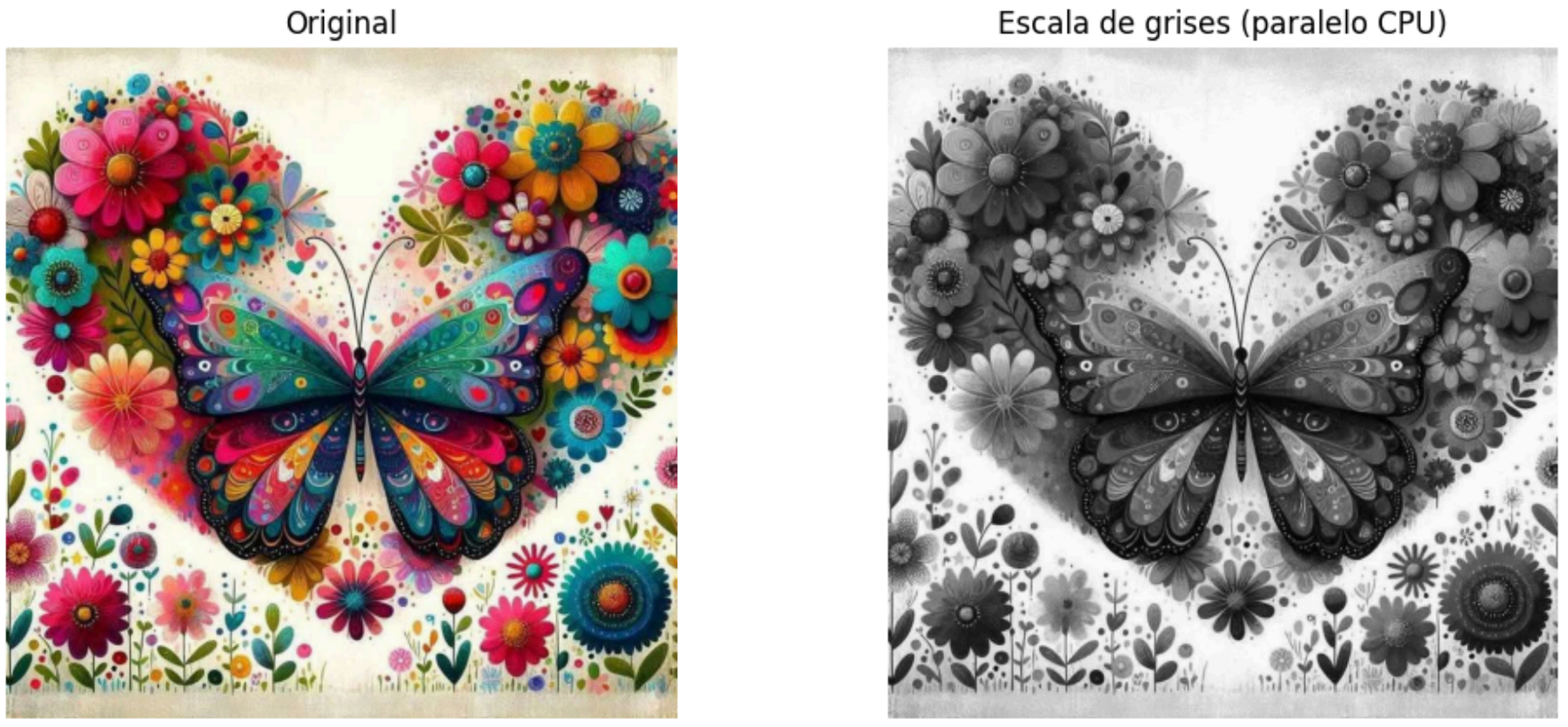


Imagen original (izquierda) y procesada (derecha).

Análisis y Conclusión

El proyecto permitió aplicar los fundamentos de programación concurrente, sincronización y paralelismo, observando las diferencias en la ejecución de un mismo proceso en CPU y GPU.

Aunque la versión GPU presentó un tiempo mayor debido a limitaciones de hardware, se comprendió la importancia de la distribución de carga y la optimización de recursos.

Esta experiencia ayudó a fortalecer la comprensión del modelo de ejecución data-parallel, las barreras de sincronización y los conceptos de speedup y eficiencia.

En futuras versiones, podría optimizarse la transferencia de datos entre CPU y GPU, incorporar nuevos filtros de imagen y comparar los resultados con otros enfoques paralelos.

“El proyecto permitió evidenciar el impacto del procesamiento paralelo en tareas visuales y reforzar la importancia del diseño eficiente en la programación concurrente.”

Referencias y Agradecimientos
Bibliografía:

Apuntes de clase – Programación Concurrente, UNaB (2025).
Documentación oficial de Numba, OpenCV y Matplotlib.

Agradecimientos:

A mis profesores y compañeros por el acompañamiento durante el proceso de aprendizaje. 💜

Y a mí misma, por no rendirme y seguir aprendiendo paso a paso. 🦋