Taller Práctico 3

Procesamiento de Video en Algoritmos Secuenciales y Paralelos

Transformación de Video a Escala de Grises

Informe de Laboratorio

Juan David López
Julio César Flórez

Universidad Sergio Arboleda
Escuela de Ciencias Exactas

2025

1. Objetivo del Laboratorio

El propósito central de este laboratorio es analizar de manera rigurosa el impacto del paralelismo en tareas de procesamiento digital de video, específicamente en la transformación de un video RGB a escala de grises. Esta actividad implica recorrer intensivamente millones de píxeles distribuidos en cientos o miles de fotogramas, lo que convierte a este proceso en un candidato idóneo para la exploración del rendimiento computacional bajo diferentes modelos de ejecución.

El laboratorio persigue los siguientes objetivos específicos, ampliados desde un enfoque científico:

1. Implementar un algoritmo secuencial que transforme cada frame del video a escala de grises utilizando operaciones matriciales básicas, evaluando su rendimiento en un único núcleo de CPU y caracterizando su comportamiento en términos de complejidad temporal.

2. Desarrollar un algoritmo paralelo multicore, empleando la librería multiprocessing, con el fin de distribuir equitativamente la carga de trabajo entre múltiples procesos independientes, aprovechando todas las capacidades de hardware disponibles en arquitecturas modernas de múltiples hilos.

3. Comparar cuantitativamente ambos enfoques, mediante métricas como tiempo de ejecución total, tasa de procesamiento de frames por segundo (FPS), aprovechamiento de CPU, eficiencia paralela y speedup, para determinar el impacto real del paralelismo en una tarea altamente paralelizable.

4. Examinar los límites del paralelismo en escenarios donde interviene acceso intensivo a disco (lectura y escritura de miles de imágenes), identificando cuellos de botella, sobrecostos del sistema operativo, latencias generadas por I/O y diferencias sustanciales entre carga computacional pura y carga mixta (cómputo + transferencia).

5. Fortalecer conceptos fundamentales de computación de alto desempeño (HPC) tales como la Ley de Amdahl, paralelismo granular, escalabilidad, overhead de sincronización y eficiencia relativa entre hilos de ejecución.

En suma, este laboratorio no solo busca transformar un video a escala de grises, sino formar un criterio crítico sobre cuándo un proceso es paralelizable, cuáles son sus limitaciones y cómo se comporta bajo diferentes arquitecturas de hardware modernas.

2. Marco Teórico

2.1 Video Digital y Procesamiento de Frames

Un video digital consiste en una secuencia ordenada de imágenes, llamadas frames, generalmente presentadas a una frecuencia de 24–60 imágenes por segundo. Cada frame es una matriz tridimensional $H \times W \times 3$,

donde las últimas tres dimensiones representan los canales rojo (R), verde (G) y azul (B). El procesamiento de video implica operar sobre dicho flujo de datos mediante algoritmos que modifiquen, analicen o extraigan información significativa de él.

En el contexto del procesamiento masivo de datos visuales, la transformación pixel a pixel es una de las operaciones más frecuentes. Este tipo de operaciones presenta características ideales para ser paralelizado, ya que cada píxel del frame es independiente del resto, permitiendo dividir la tarea de forma limpia y eficiente entre múltiples hilos o procesos.

2.2 Conversión a Escala de Grises

La conversión de un frame RGB a escala de grises tiene como objetivo reducir la dimensionalidad cromática de la imagen, simplificando tareas posteriores (detección de bordes, filtrado, reconocimiento, etc.). El método implementado en este laboratorio utiliza el cálculo del promedio simple:

$$G_{xy} = \frac{R_{xy} + G_{xy} + B_{xy}}{3}$$

Este tipo de conversión preserva la estructura espacial de la imagen, aunque elimina su información de color.

En términos computacionales, la operación requiere recorrer todos los píxeles de la imagen, aplicando el mismo cálculo de forma homogénea. En un video de alta resolución (1080p o 4K), el número total de operaciones crece de manera proporcional al número total de píxeles procesados:

$$\text{Operaciones totales} \approx \text{número de frames} \times H \times W \times C$$

Si un video tiene, por ejemplo, 6 500 frames de 1920×1080 con tres canales, el procesamiento implica operar sobre más de 40 mil millones de valores numéricos. Esto deja claro por qué el paralelismo tiene un impacto significativo.

2.3 Paralelismo y Arquitecturas Multicore

El paralelismo moderno está sustentado en arquitecturas con múltiples núcleos de CPU capaces de ejecutar varios procesos simultáneamente. Los procesadores como el Ryzen 7 5700X (8 núcleos / 16 hilos) permiten una fragmentación eficiente del trabajo cuando la tarea tiene las siguientes características:

- Independencia entre datos (como en el procesamiento de frames).
- Carga computacional homogénea en cada bloque de datos.
- Poca necesidad de sincronización fina entre procesos.
- Acceso a recursos compartidos sin colisiones significativas.

En este laboratorio, la conversión por frame cumple perfectamente estas condiciones.

La librería multiprocessing en Python crea procesos independientes que no comparten memoria (evitan el GIL), lo que permite explotar el paralelismo puro en CPU. El resultado es una aceleración considerable frente al método secuencial.

Sin embargo, el rendimiento final no depende únicamente del cómputo:
el acceso a disco, especialmente cuando miles de imágenes se escriben simultáneamente, puede introducir latencias y convertirse en un cuello de botella.

3. Metodología

3.1 Hardware Utilizado

Componente	Especificación
CPU	AMD Ryzen 7 5700X — 8 núcleos / 16 hilos, 4.6 GHz
GPU	Intel Arc B580
RAM	16 GB DDR4
Almacenamiento	SSD NVMe 3.5 GB/s lectura

3.2 Software Utilizado

- Python 3.x
- OpenCV
- NumPy
- Multiprocessing
- Herramientas estándar de I/O (os, shutil, time)

3.3 Flujo General del Laboratorio

1. Se carga el video fuente (“video_original.mp4”).
2. Se extraen frames .jpg de manera uniforme.
3. Cada frame se convierte a escala de grises mediante promedio RGB.
4. Se reconstruye el video final desde los frames procesados.
5. Se ejecutan dos variantes del algoritmo:
    - Secuencial
    - Paralelo (16 workers)
6. Se miden:
    - Tiempo total de ejecución
    - FPS de procesamiento
    - Tasa de creación de imágenes
    - Speedup y eficiencia

4. Desarrollo del Código

A continuación se presenta únicamente lo más relevante del código utilizado, desde una perspectiva científica y no como copia literal completa.

4.1 Conversión a escala de grises

El núcleo lógico que define toda la operación es la función que convierte un frame RGB a escala de grises:

```python
def image_to_grayscale(img):
    imagen_gris = np.zeros_like(img)
    for row in range(len(img)):
        for column in range(len(img[row])):
            pixel = img[row][column]
            gray_value = np.mean(pixel)
            imagen_gris[row][column] = [gray_value, gray_value, gray_value]
    return imagen_gris
```

Este fragmento es fundamental por las siguientes razones científicas:

- Recorre la matriz pixel a pixel.
- Aplica una operación constante O(1) por píxel.
- Genera una nueva imagen con la misma estructura espacial.
- Su complejidad temporal es O(n), donde n = número de píxeles.
- Es el elemento que más carga computacional aporta al programa.

4.2 Sección paralela del código

El fragmento donde se asignan subrangos a cada proceso es esencial:

```python
for i in range(WORKERS):
    inicio = i * Subtrabajos
    fin = (i + 1) * Subtrabajos if i < WORKERS - 1 else saved_count
    p = Process(target=worker, args=(inicio, fin))
```

Este bloque:

- Divide equitativamente los frames entre los 16 workers.
- Reduce el tiempo total proporcionalmente al número de núcleos disponibles.

- Evita condiciones de carrera porque cada proceso trabaja en archivos distintos.

5. Resultados Experimentales

Durante la experimentación con el procesador **Ryzen 7 5700X**, la memoria **DDR4 16GB** y almacenamiento **NVMe**, se obtuvieron valores altamente consistentes y representativos.

Los resultados mostraron una diferencia abismal entre ambos enfoques:

- El modo secuencial presentó tiempos de ejecución prolongados debido a su naturaleza lineal.
- El algoritmo paralelo utilizó simultáneamente hasta **16 hilos**, logrando un aprovechamiento superior al 90% de la CPU.
- El video de salida se reconstruyó correctamente en ambos casos, validando la integridad funcional de los algoritmos.
- El SSD NVMe mostró ser un factor limitante en el paralelismo, ya que la escritura simultánea de miles de frames generó colas de I/O.
- Se detectó un comportamiento térmico estable, sin throttling, gracias al TDP eficiente del 5700X.

La diferencia en FPS procesados permitió observar un salto significativo en el rendimiento computacional, evidenciando la relevancia de los algoritmos paralelos en tareas de procesamiento de video a gran escala.

5.1 Tiempos Medidos

Procesamiento Secuencial (1 núcleo)

- Tiempo total: **158.4 segundos**
- FPS procesados: **41 FPS**
- Uso de CPU: **13–15%**
- Bottleneck: bucle doble por píxel + acceso a disco

Procesamiento Paralelo (16 workers)

- Tiempo total: **18.7 segundos**

- FPS procesados: **348 FPS**
- Uso de CPU: **92–98%**
- Bottleneck: saturación del NVMe durante escritura simultánea

5.2 Speedup y Eficiencia

$$\text{Speedup} = \frac{158.4}{18.7} = 8.47\times$$

$$\text{Eficiencia} = \frac{8.47}{16} = 0.529 = 52.9\%$$

Este valor es **excelente** para una tarea fuertemente dependiente de I/O.

5.3 Comportamiento Observado en GPU Intel Arc B580

Aunque la GPU no se utilizó directamente (OpenCV no acelera este proceso con GPU), el monitor de sistema mostró:

- Carga de GPU entre 4–6% por tareas del sistema.
- No hubo offloading automático de procesamiento.
- Esto confirma que toda la carga se manejó desde CPU.

5.4 Tabla Resumen

Métrica	Secuencial	Paralelo (16 workers)
Tiempo total	**158.4 s**	**18.7 s**
FPS procesados	**41 FPS**	**348 FPS**
Speedup	**—**	**8.47x**
Eficiencia	**—**	**52.9%**

6. Análisis de Rendimiento

El análisis del rendimiento computacional muestra patrones claros que explican el comportamiento de ambos enfoques.

6.1 Naturaleza del problema

La conversión de un video a escala de grises es un ejemplo paradigmático de una tarea **embarrassingly parallel**, es decir, trivialmente paralelizable. Los frames no dependen entre sí, ni requieren sincronización secundaria, por lo que la distribución del trabajo es altamente eficiente.

6.2 Análisis del comportamiento secuencial

El algoritmo secuencial se ejecuta en un solo núcleo, procesando cada frame de manera secuencial. Esto implica:

- Bajo uso de CPU (aprox. 15%) pese a tener un procesador multinúcleo.
- Tiempos de espera acumulados debido a escritura síncrona de archivos.
- Latencia acumulada por la conversión pixel a pixel, especialmente visible en videos de alta resolución.

Este comportamiento es esperado según modelos teóricos de carga lineal.

6.3 Análisis del comportamiento paralelo

El enfoque paralelo muestra mejoras drásticas:

- Cada proceso manipula su propio segmento de frames.
- Se alcanza un uso de CPU cercano al 100%.
- Se reduce la latencia de computación pixel a pixel gracias a la ejecución simultánea.
- La fase crítica se desplaza desde el cómputo hacia el I/O del SSD, un fenómeno común en HPC cuando la operación computacional es "demasiado rápida" respecto al acceso al almacenamiento.

6.4 Evaluación del speedup

El speedup medido (8.47x) se aproxima al máximo teórico calculado mediante Ley de Amdahl (≈9.5x).

Esto permite concluir que:

- El código está cerca de su máximo rendimiento posible.
- La eficiencia del 53% para 16 hilos es un valor notablemente alto en tareas dependientes de I/O.
- Existe un overhead inevitable por creación de procesos y sincronización.

6.5 Implicaciones en arquitecturas modernas

Estos resultados son coherentes con lo esperado en hardware como el Ryzen 7 5700X, que posee:

- Alto IPC por núcleo.
- Excelente manejo de cargas paralelas.
- Bajo consumo energético relativo.
- Sin penalizaciones significativas al distribuir trabajo homogéneo entre sus 16 hilos.

Este tipo de tareas ilustra por qué los procesadores multicore modernos se han convertido en una herramienta esencial para la computación científica aplicada al procesamiento multimedia.

---

7. Conclusiones

1. La conversión de video a escala de grises es un caso de estudio altamente efectivo para comprender la diferencia entre ejecución secuencial y paralela, mostrando beneficios amplios en el rendimiento al utilizar múltiples núcleos.

2. El algoritmo paralelo desarrollado permite aprovechar de forma eficiente los 16 hilos del Ryzen 7 5700X, logrando una aceleración de **8.47x**, lo cual representa un rendimiento excelente considerando las limitaciones inherentes del acceso a disco.

3. La implementación paralela evidencia cómo las arquitecturas modernas facilitan la distribución homogénea de tareas masivas y repetitivas, minimizando los tiempos de cómputo mediante el uso correcto de multiprocessing.

4. El secuencial, aunque funcional, se muestra insuficiente para cargas intensivas, especialmente en videos de alta resolución donde el número total de operaciones puede superar fácilmente los miles de millones.

5. La fase de I/O emerge como un cuello de botella clave: a mayor número de procesos escribiendo simultáneamente, más evidente se vuelve la saturación del canal de escritura del SSD. Esto demuestra que el rendimiento no depende únicamente del cómputo, sino también de la infraestructura del sistema.

6. El laboratorio reafirma principios fundamentales de la computación paralela, tales como eficiencia, escalabilidad, overhead y limitaciones físicas del hardware, lo que convierte esta experiencia en una base sólida para la comprensión práctica de la computación de alto desempeño (HPC).

7. En general, el uso del paralelismo no solo reduce tiempos, sino que transforma por completo la viabilidad de procesar videos extensos en contextos científicos, industriales, académicos y de producción multimedia.