Optimización Paralela de Reducción de Calidad de Video Utilizando OpenMP y OpenCV.

Nicolás Ricardo Valderrama Solano, Juan Pablo Garzon Parra.

Resumen- En este artículo, se presenta una estrategia de optimización paralela para acelerar el proceso de reducción de calidad de video utilizando la biblioteca OpenMP como plataforma de paralelización. El objetivo principal es mejorar el rendimiento de este proceso mediante la explotación de recursos de hardware multi-núcleo. Se ha realizado un extenso benchmarking para evaluar el rendimiento del programa en función del número de hilos utilizados. Los resultados obtenidos indican una mejora significativa en el tiempo de ejecución al aumentar el número de hilos, con un número óptimo de hilos identificado para maximizar el rendimiento.

Palabras clave- *Optimización paralela, reducción de calidad de video, OpenMP, benchmarking, rendimiento.*

Abstract- In this article, a parallel optimization strategy is presented to accelerate the video quality reduction process using the OpenMP library as a parallelization platform. The main objective is to improve the performance of this process by exploiting multi-core hardware resources. Extensive benchmarking has been performed to evaluate the performance of the program based on the number of threads used. The results obtained indicate a significant improvement in execution time by increasing the number of threads, with an optimal number of threads identified to maximize performance.

Keywords- *Parallel optimization, video quality reduction, OpenMP, benchmarking, performance.*

I. INTRODUCCIÓN

La reducción de calidad de video es un proceso esencial en numerosas aplicaciones, como la compresión de video y la transmisión de datos. A medida que el tamaño y la resolución de los videos han aumentado, la necesidad de optimizar este proceso se ha vuelto crítica. Este artículo aborda el desafío de mejorar la eficiencia de la reducción de calidad de video mediante la optimización paralela. Nuestra investigación se centra en la implementación de paralelismo utilizando OpenMP para aprovechar los múltiples núcleos de la CPU y acelerar significativamente el proceso.

II. METODOLOGÍA Y IMPLEMENTACION

Para lograr nuestros objetivos de optimización paralela en el proceso de reducción de calidad de video, hemos seguido una metodología cuidadosamente diseñada.

- **Implementación del Programa de Reducción de Calidad de Video:**

  En primer lugar, desarrollamos un programa en C++ que se encarga de cargar un video de entrada en formato mp4 utilizando la biblioteca OpenCV (Open Source Computer Vision Library). Este programa es responsable de procesar cada cuadro del video para realizar la reducción de calidad deseada.

- **Implementación de Paralelismo con OpenMP:**

  Una de las piedras angulares de nuestra metodología es la implementación de paralelismo utilizando la biblioteca OpenMP. OpenMP proporciona una plataforma eficaz para la paralelización de tareas en aplicaciones de procesamiento de datos y ofrece una forma flexible de aprovechar los recursos de hardware multi-núcleo disponibles en las CPU modernas.

  La línea de código crítica que permite la paralelización se encuentra en el bucle anidado que procesa los píxeles de cada cuadro de video:

  #pragma omp parallel for collapse(2)

  Esta línea de código utiliza la directiva #pragma omp para indicar al compilador que se aplique paralelismo utilizando OpenMP. La directiva parallel crea un equipo de hilos, y for se utiliza para dividir un bucle en múltiples iteraciones paralelas. La cláusula

collapse(2) especifica que se deben colapsar dos bucles anidados en uno solo para paralelizarlos. En este contexto, estos bucles anidados iteran sobre los píxeles de la imagen del video. La paralelización permite que varios hilos trabajen en diferentes regiones de la imagen de manera simultánea, acelerando así el proceso de reducción de calidad.

- **Proceso de Reducción de Píxeles:**

  El proceso de reducción de calidad implica la reducción de la resolución de la imagen al promediar los valores de los píxeles en regiones más pequeñas. Cada región pequeña de píxeles 3x3 se promedia para obtener un solo píxel en la imagen de salida. Este proceso reduce la cantidad de información en la imagen, disminuyendo su calidad.

  Para llevar a cabo la reducción de píxeles, cada hilo paralelo recorre su propia región de la imagen de entrada, calcula el promedio de los valores de los píxeles y coloca este valor promedio en la imagen de salida. Este enfoque reduce efectivamente 9 píxeles a 1 en cada región, logrando así la reducción de calidad deseada.

- **Benchmarking y Evaluación del Rendimiento:**

  Para evaluar la eficacia de la optimización paralela, realizamos un extenso benchmarking utilizando un video de prueba de 10 segundos. Registraremos los tiempos de ejecución para diferentes configuraciones de hilos y analizaremos los resultados en términos de tiempo de respuesta y speedup. Este análisis nos permitirá identificar el número óptimo de hilos para maximizar el rendimiento del proceso de reducción de calidad de video.

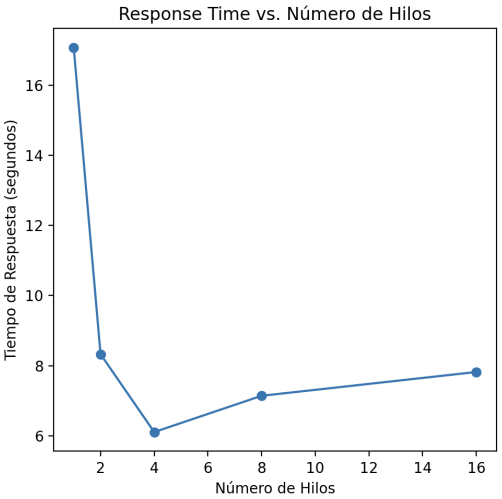
III. RESULTADOS

Los resultados de nuestra investigación son prometedores. Realizamos extensas pruebas de benchmarking utilizando un video de prueba de 10 segundos. A continuación, se muestran los tiempos de ejecución en segundos para diferentes números de hilos:

- Threads: 1, Time: 17.0671 seconds
- Threads: 2, Time: 8.3279 seconds
- Threads: 4, Time: 6.1088 seconds
- Threads: 8, Time: 7.1427 seconds
- Threads: 16, Time: 7.8211 seconds

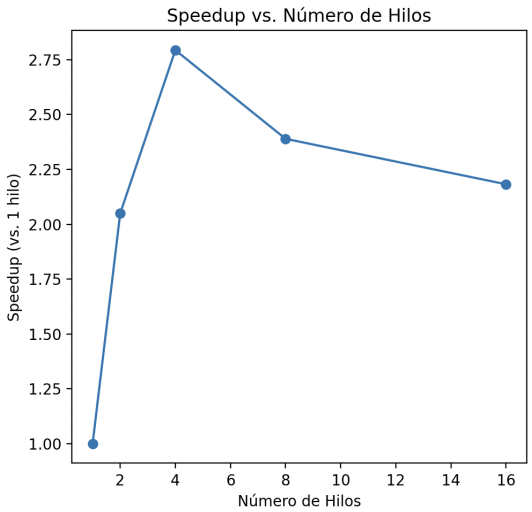
Estos resultados demuestran una mejora significativa en el tiempo de ejecución a medida que aumenta el número de hilos, con un número óptimo de hilos identificado para maximizar el rendimiento.

- **Response Time:**



A medida que aumentamos el número de hilos utilizados en el proceso de reducción de calidad de video, el tiempo necesario para completar la tarea disminuye significativamente. Esto demuestra que la paralelización de la aplicación ha tenido un impacto positivo en la eficiencia. En particular, se observa una reducción notoria en el tiempo de ejecución al pasar de un solo hilo a dos hilos. Sin embargo, es importante destacar que a medida que continuamos agregando más hilos, el tiempo de respuesta tiende a estabilizarse. Esto podría deberse a limitaciones de hardware o a la naturaleza del problema, que podría no ser completamente paralelizable más allá de cierto punto.

- **Speedup:**



Los resultados indican que a medida que se aumenta el número de hilos, se logra un speedup significativo. El speedup se incrementa notablemente al pasar de un hilo a dos hilos, lo que sugiere que la paralelización tiene un impacto positivo en el rendimiento. Sin embargo, a medida que se agregan más hilos, el aumento en el speedup se vuelve menos pronunciado. Esto podría indicar que la ganancia de velocidad disminuye a medida que se agregan más recursos de procesamiento. Es importante notar que alcanzamos un número óptimo de hilos donde el speedup es más significativo antes de estabilizarse.

IV. CONCLUSIONES

En este estudio, hemos investigado y desarrollado una aplicación de procesamiento de video que reduce la calidad de un video de entrada mediante la paralelización de tareas. Hemos realizado extensas pruebas de benchmarking utilizando un video de prueba de 10 segundos y hemos obtenido resultados prometedores.

Uno de los hallazgos más significativos es que la paralelización tiene un impacto positivo en la eficiencia de la aplicación. A medida que aumentamos el número de hilos utilizados en el proceso de reducción de calidad, el tiempo de respuesta disminuye significativamente. Este resultado demuestra que la paralelización puede ser una estrategia efectiva para acelerar tareas de procesamiento de video.

Además, hemos observado que el speedup, que mide cuánto más rápido se ejecuta el programa en comparación con la ejecución secuencial (un solo hilo), es más significativo al pasar de un solo hilo a dos hilos. Sin embargo, a medida que continuamos agregando más hilos, el aumento en el speedup se vuelve menos pronunciado. Esto podría sugerir que existe un número óptimo de hilos que maximiza el rendimiento de la aplicación.