Parallel Image and Video Processing in Java

Mohammad Atwi 6328

Mohammad Karaki 6393

Presented to: Dr Mohammad Aoude

Table of Contents

Introduction to Image and Video Processing in Java

In today's digital era, images and videos play a central role in how we communicate, consume information, and express creativity. As the quality of media continues to increase—especially with the rise of 4K and higher resolutions—the demand for efficient processing becomes critical. This project addresses that challenge by implementing image and video filtering using Java through the Fork/Join framework and ExecutorService. Since videos are essentially sequences of images, and images can be represented as large matrices of pixel values, we leverage a tile-based parallel processing approach to significantly improve performance and scalability.

Project Overview and Objectives

Goal of the Project

The primary goal of this project is to apply image and video filtering using Java's Fork/Join and ExecutorService.

This approach aims to improve processing speed through parallelism.

Understanding the Data

Videos are sequences of images, and images are large matrices of pixel values.

This understanding forms the basis for the project's processing methodology.

Tile-Based Parallel Processing

The project adopts a tile-based parallel processing approach to divide images into manageable blocks.

This method allows concurrent processing, significantly boosting performance.

Parallel vs Sequential Processing

Parallel Processing

- Parallel processing divides the image into smaller blocks processed concurrently across multiple cores, drastically reducing processing time.
- ExecutorService assigns each block to a thread, while ForkJoinPool uses recursive task division and work-stealing to optimize thread use.

Sequential Processing

- Sequential processing handles the entire image in a single thread.
- This approach leads to longer processing times.
- Sequential processing often results in bottlenecks.
- It is less suitable for tasks requiring real-time or high-volume processing.

Tile-Based Parallel Processing Approach

1

Dividing Images into Tiles

Our approach divides images into smaller blocks, or tiles, to facilitate parallel processing.

This segmentation allows for independent handling of each tile, optimizing the workflow.

2

Independent Tile Processing

Each tile is processed independently by a thread that loops through its pixels both vertically and horizontally.

This ensures that the desired filter is applied systematically across all pixels within the tile.

3

Efficient Multi-Core Utilization

The tile-based method efficiently utilizes multi-core processors by breaking down large images into smaller units.

This approach maximizes the processing power of modern hardware for concurrent operations.

4

Scalable and Faster Filtering

By enabling concurrent processing of tiles, the method ensures scalable and faster image filtering.

This makes it suitable for handling large-scale image processing tasks effectively.

Video Processing Approach

1

Frame Extraction

For video processing, we utilized JavaCV to extract individual frames from videos.

This step ensures that each frame can be processed independently.

2

Parallel Image Filtering

Each frame was then processed through our parallel image filtering pipeline concurrently.

This approach leverages parallelism to handle multiple frames at the same time.

3

Recombination of Frames

After all frames were filtered, they were recombined to reconstruct the processed video.

This step ensures the integrity and continuity of the video after processing.

4

Efficient Dataset Handling

This pipeline effectively extends the tile-based parallel image processing approach to video.

It is designed to handle large video datasets efficiently.

Speedup Results and Analysis

4.5x

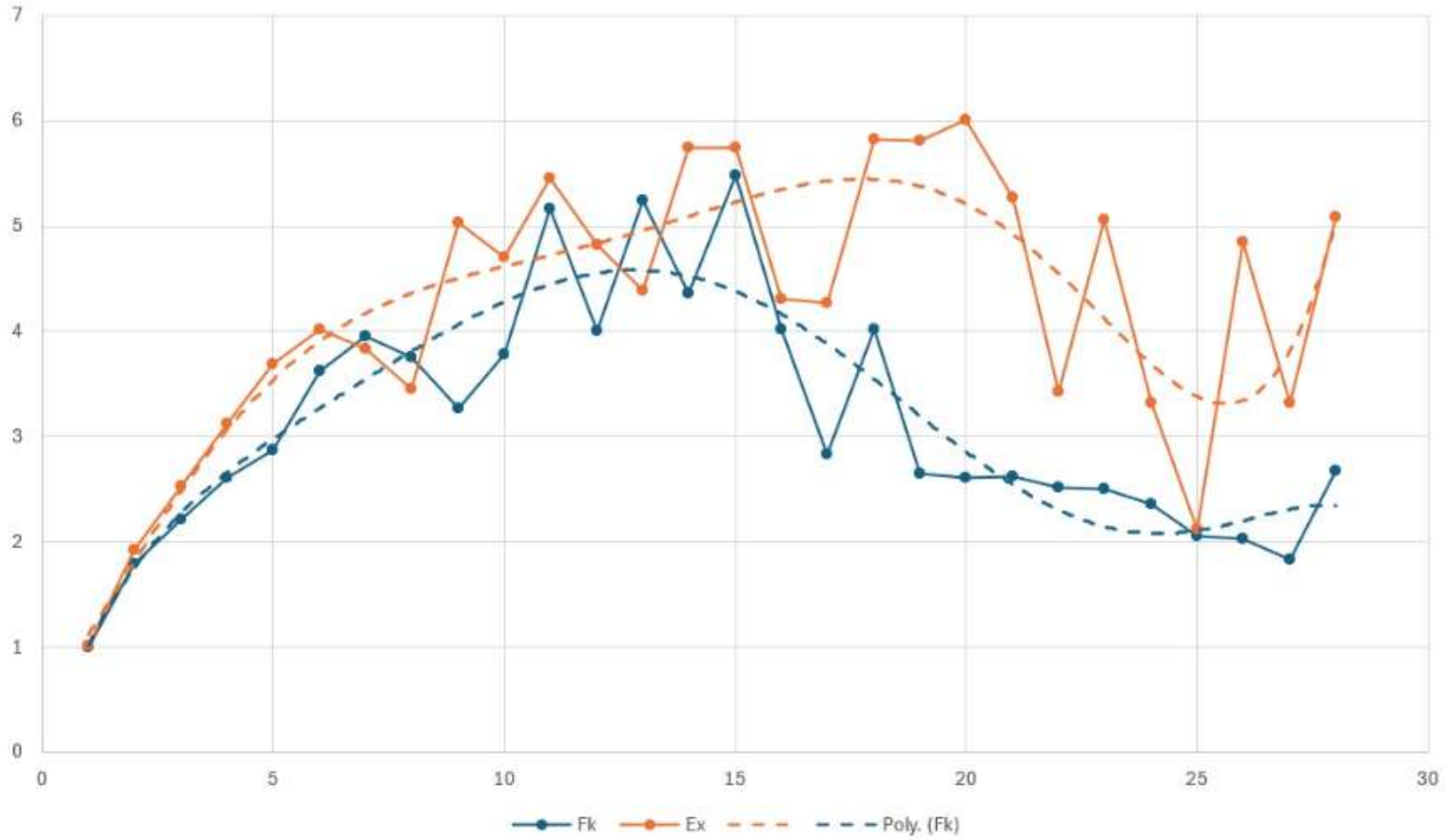
Average Speedup Achieved

Our optimized design achieved an average speedup of 4.5x.

7x

Peak Speedup Observed

Some configurations reached peak speedups of 5x, 6x and even 7x.



Conclusion and Future Directions

Project Success

This project successfully demonstrates the power of parallel processing for image and video filtering.
By leveraging Java's ExecutorService and ForkJoinPool, significant performance improvements were achieved.

Real-Time Applications

The approach is suitable for real-time and large-scale media applications.
Multi-core processors were effectively utilized to enhance processing efficiency.

Future Exploration

Future work includes exploring GPU acceleration with CUDA.
This will enable further performance enhancements for computational tasks.

Advanced Techniques

Utilizing SIMD instructions and Java's Vector API are key future directions.
These techniques aim to explicitly vectorize computations for better efficiency.