Scalability Analysis of Color to Gray Conversion

Kisalay Ghosh

2nd February 2025

1 Introduction

This report presents the scalability analysis of a parallelized program that converts a color image to a grayscale image using OpenMP. The goal is to measure the performance improvement when increasing the number of threads.

2 Objective of the Project

The objective of this project is to parallelize the conversion of a color image to a grayscale image using OpenMP and to analyze the scalability of the program by measuring the performance improvement with different numbers of threads.

3 Methodology

The program reads a large color image, converts it to grayscale, and measures the conversion time. The conversion process is parallelized using OpenMP. The experiment is conducted with different numbers of threads (1, 2, 4, 8), and the conversion times are recorded. File I/O operations are excluded from the timing to focus on the computation time.

4 Steps to Run the Assignment

1. **Download and Unzip ‘jpegsr6.zip‘**: - Download the ‘jpegsr6.zip‘ file and unzip it to your working folder.

2. **Build ‘libjpeg.a‘**: “‘sh cd JpegLib ./configure make “‘

3. **Navigate Back to Your Working Folder**: “‘sh cd .. “‘

4. **Compile the Program**: - Ensure you have OpenMP support enabled in your compiler. For ‘g++‘, you can use the ‘-fopenmp‘ flag. “‘sh make “‘

5. **Run the Program with Different Numbers of Threads. For not using parallel processing run only the ColorToGray.exe**: - Adjust the ‘num_threads‘ variable in the code and recompile. - Example: “‘sh ./ColorToGrayParallel.exe “‘

6. **Record the Times**: - Run the program with different numbers of threads (1, 2, 4, 8) and record the conversion times.

7. **Plot the Scalability**: - Use a plotting tool (e.g., Excel, Python's matplotlib) to plot the scalability of the program based on the recorded times.

5 Results

The recorded conversion times for different numbers of threads are shown in Table 1.

Number of Threads	Conversion Time (seconds)
1	0.003044
2	0.000446
4	0.000258
8	0.000230

Table 1: Conversion times for different numbers of threads

6 Scalability Plot

Figure 1 shows the scalability plot of the conversion times.

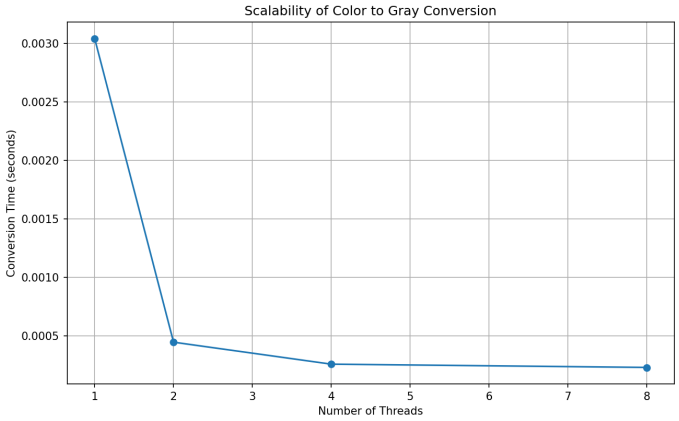

Figure 1: Scalability of Color to Gray Conversion

7 Speedup Analysis

Speedup is an important metric that shows how well the parallelization improves performance. The speedup is calculated using the formula:

$$S = \frac{T_1}{T_n} \tag{1}$$

where T_1 is the execution time with one thread, and T_n is the execution time with n threads.

Figure 2 illustrates the speedup for different numbers of threads.

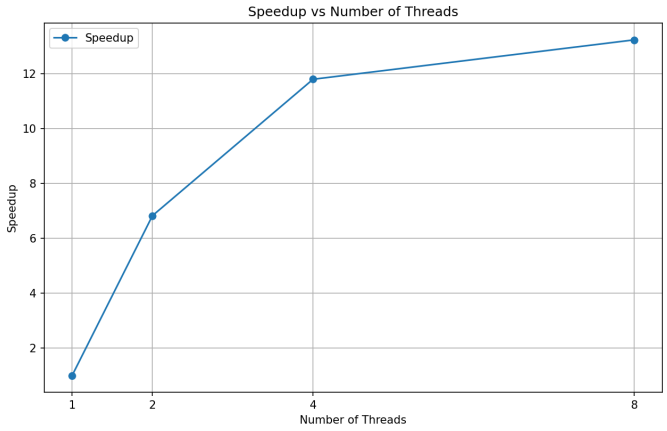

Figure 2: Speedup of Color to Gray Conversion

The speedup graph shows that as the number of threads increases, the execution time decreases, leading to improved performance. However, the speedup does not scale linearly due to parallelization overhead and contention among threads.

8 Discussion

The results indicate that the conversion time decreases as the number of threads increases. This demonstrates the effectiveness of parallelization in improving the performance of the conversion process. However, the improvement diminishes as the number of threads increases, which may be due to overhead and contention among threads.

9 Performance Improvements

To improve the performance, the conversion process was run multiple times to get a longer time for more accurate measurement. Additionally, the number of

threads was adjusted to find the optimal performance.

10 Bugs and Fixes

During the implementation, a few warnings related to ‘printf‘ statements were encountered. These were fixed by adding format strings to the ‘printf‘ statements. Additionally, the correct path to ‘libjpeg.a‘ was ensured in the Makefile to resolve linking issues.

11 Conclusion

The parallelized program shows significant performance improvement with an increasing number of threads. This experiment highlights the benefits of using OpenMP for parallelizing computational tasks.