Enhancement of PNG Compression through Parallelization

Chad Greenspan ● Dev Gopal ● Joe Hoang ● Sebastian Reconco

The Portable Network Graphics (PNG) file format is widely used for images that require lossless compression. However, as image resolutions increase and datasets grow, the time needed to compress these images becomes a bottleneck. Traditional PNG compression algorithms, while effective in maintaining image quality, are computationally expensive, leading to slow processing times. The goal of this project is to enhance PNG compression performance by implementing parallelization techniques to reduce processing time while preserving image integrity.

PNG compression consists of multiple stages, including filtering, prediction, and entropy encoding. Each stage involves significant computation, particularly for high-resolution images. Standard sequential compression methods process images pixel by pixel, which is inherently slow. The challenge is to redesign the compression pipeline to leverage modern multi-core processors by parallelizing key computational steps.

The main issues we seek to address are:

- **High computation time**: Sequential image processing limits performance.
- **Inefficient resource utilization**: Modern CPUs and GPUs are underutilized in traditional implementations.
- **Preservation of image quality**: Compression of PNGs is already inherently lossless. If we were to increase efficiency through parallelization but lose visual fidelity, the usefulness of the implementation would be called into question.

We propose a multi-threaded approach to PNG compression by implementing parallelism in the filtering and compression stages. Our strategy includes:

1. **Image Segmentation**: Instead of processing the entire image sequentially, we divide it into smaller chunks (e.g., rows or blocks). Each segment is then processed independently.
2. **Parallel Filters**: Using C++ and the std::thread library, multiple threads will apply different PNG filters simultaneously. The applied filters include:
    - **None Filter**: Leaves pixel values unchanged.
    - **Sub Filter**: Modifies each pixel based on the value of the previous pixel in the same row.
    - **Up Filter**: Adjusts pixels based on the corresponding pixel in the row above.
    - **Average Filter**: Uses the average of the left and upper pixels for encoding.
    - **Paeth Filter**: Uses the best of left, above, and upper-left pixels.
3. **Concurrency in Entropy Encoding**: The final compression step involves applying the Deflate algorithm to encode the filtered image data. This stage can be further optimized using parallelized Huffman coding and run-length encoding.

Enhancement of PNG Compression through Parallelization

Chad Greenspan ● Dev Gopal ● Joe Hoang ● Sebastian Reconco

Implementation Details

- We utilize **LodePNG**, a lightweight PNG encoder/decoder, for handling image input and output.
- Filters are applied in parallel using **C++ threads** to distribute workload across multiple cores.
- The **filter.cpp** implementation defines multiple filtering strategies optimized for multi-threaded execution.

Expected Outcomes

By implementing a parallelized approach to PNG compression, we anticipate a significant reduction in processing time compared to traditional sequential methods. The introduction of multi-threading and concurrent execution in the filtering and compression stages should allow the program to efficiently utilize available CPU cores, leading to faster image processing. This improvement will be especially noticeable when working with high-resolution images, where traditional methods often struggle with long computation times. Additionally, by optimizing the compression pipeline, we aim to achieve these performance gains without compromising the quality of the final image.

Another key expected benefit is better resource utilization. Many modern computing systems come equipped with multi-core processors, yet traditional image compression techniques fail to leverage this capability. By implementing parallelization, the workload will be distributed across multiple cores, ensuring that the system's full potential is utilized. This will result in smoother performance and reduced bottlenecks, making it a more viable solution for real-time or large-scale image processing applications.

Furthermore, while efficiency is a primary focus, it is equally important that our approach maintains the integrity of the original image. Since PNG is a lossless compression format, any improvements to its efficiency must ensure that no data is lost in the process. Despite optimizing the computational aspects of compression, the fundamental characteristics of PNG encoding remain unchanged.

Notes for Further Completion

- Add benchmark results comparing traditional vs. parallelized PNG compression.
- Discuss potential trade-offs, such as increased memory consumption or thread synchronization overhead.
- Include a summary of test cases used for performance evaluation.
- Provide a conclusion summarizing findings and future improvements.