Parallel Seam Carving

Augusto Blomer
Victor Konen

Introduction

Seam carving is an effective way of resizing images while maintaining the clarity of the most significant content. This method can reduce the dimensions of a digital image without any discernable warping of the content portrayed in the original image. This allows the aspect ratio of an image to be altered in a way that preserves the subject(s) by eliminating the least important sections.

When a large image needs to be greatly reduced in size, performing seam carving can require significant computational resources. This can take a long time with a serial program. The purpose of this project is to determine what kind of speedup can be achieved through parallelization of distinct components of a seam carving algorithm. In the algorithm outlined below, the computation of the energy function and the generation of the seams will be parallelized. This will be tested and compared against a serial implementation. Although attempts have been made to improve the efficiency of the seam carving process in the past, this project will determine whether this explicit serial implementation can benefit from parallelization.

Additional Background

Extensive research has already been conducted for optimal energy functions and dynamic removal of seams in serial. Plenty of background information can be found in the cited paper: Seam Carving for Content-Aware Image Resizing, by Shai Avidan and Ariel Shamir. The cited wikipedia page for the concept also provides an adequate summary.

Algorithm

1. Store each pixel of the original image and obtain the desired dimensions
2. Remove seams from the image until it is the desired dimensions
3. For each seam to be removed from the image:
    - Compute the energy of each pixel based on an energy function
    - Compute the seam with the least total energy relative to all other possible seams
    - Remove the seam with the least total energy
4. Save the carved version of the image in the desired format

Expected Advantages of Parallelization

The energy values of each pixel in the image must be computed every time a single seam is to be removed from the image. Dividing this work evenly among multiple processors should result in a significant speedup. Since this information must be communicated among all of the processors when it comes time to remove the next seam, there will likely be a communication performance tradeoff as the number of processors increases beyond a certain point.

Computation of the seam with the least total energy is an exercise in dynamic programming. Depending on the direction of the seam, vertical or horizontal, the program must iterate over the rows or columns of pixels in the image, and determine the lowest energy step to that pixel from the three adjacent pixels in the previous row or column. This computation can be divided among multiple processors, however there exists a bottleneck that prevents processors from advancing to the next row or column before communication occurs due to pixel adjacency dependencies. This parallelization should result in a moderate speedup.

A combination of the parallelization of these two separate components of the algorithm should demonstrate a reasonable speedup over the serial implementation, given the right balance of processors that provide the optimal computation versus communication tradeoff.

Methodology

The functionality of the two components to be parallelized was implemented as follows in the subsections below.

Energy Function Parallelization

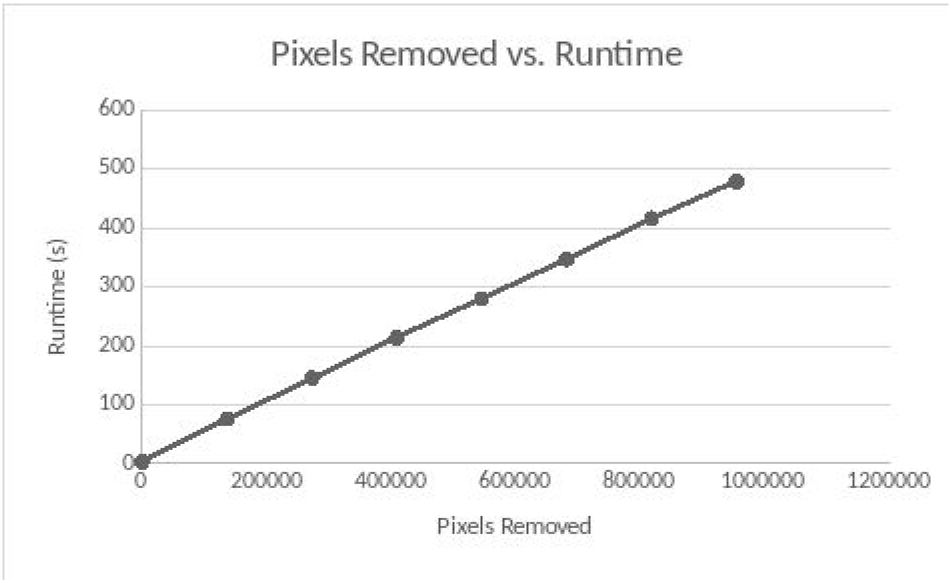
This functionality was parallelized by dividing the pixels in the image evenly among the processors, assigning an extra pixel to some of the processors as needed. Each time the energy must be computed for the entire image, each processor computes the energy for each pixel it is assigned to, using the same gradient energy function from the serial implementation of the program. Each processor then sends the energy data for the pixels it computed the energy for to all of the other processors so that each processor has the energy data for all of the pixels in the image.

Lowest Energy Seam Parallelization

Need to explain this

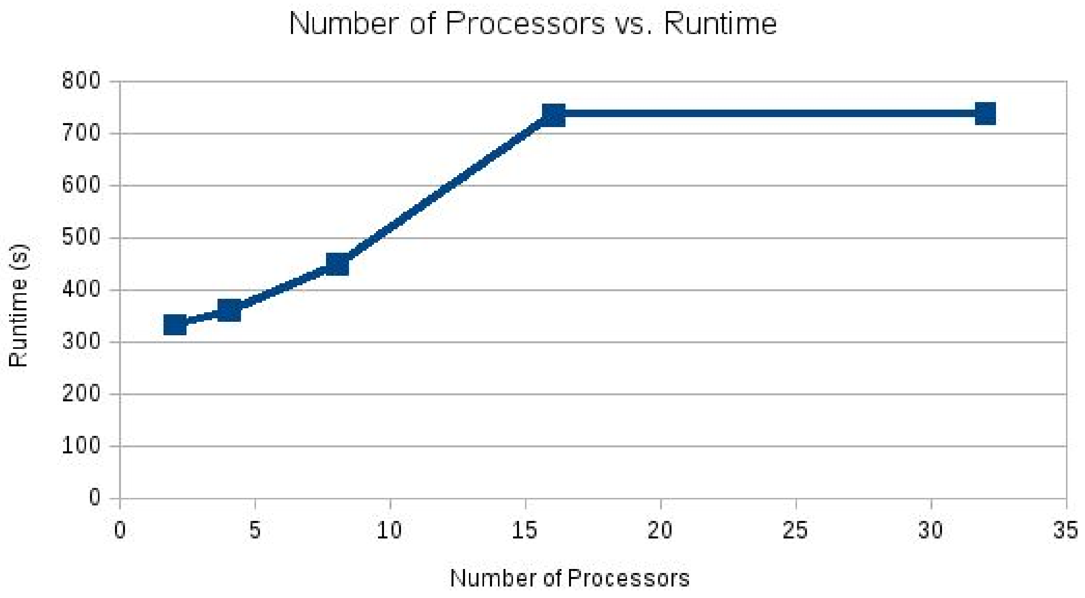
Results

The parallel implementations of the energy function component and the lowest energy seam component were tested separately in order to determine the speedup factors over the serial implementation independent of each other. The two components were then combined and tested to yield the final complete parallel implementation results. Each of the tests were conducted on the same image (pups.jpg), with a removal of 700 seams equivalent to approximately one million pixels. This test was also the most resource intensive test performed for the serial implementation of the program. The serial implementation was tested on this image removing an increasing number of pixels and the results are outlined in the graph below.



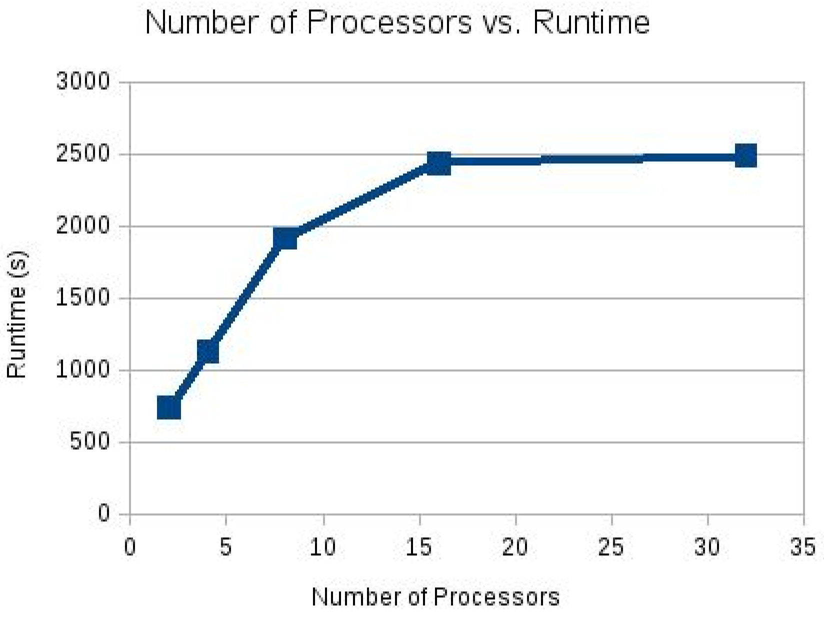
Energy Function Test Results

This component was tested at processor counts 2, 4, 8, 16, and 32, yielding the following results outlined in the graphs below.



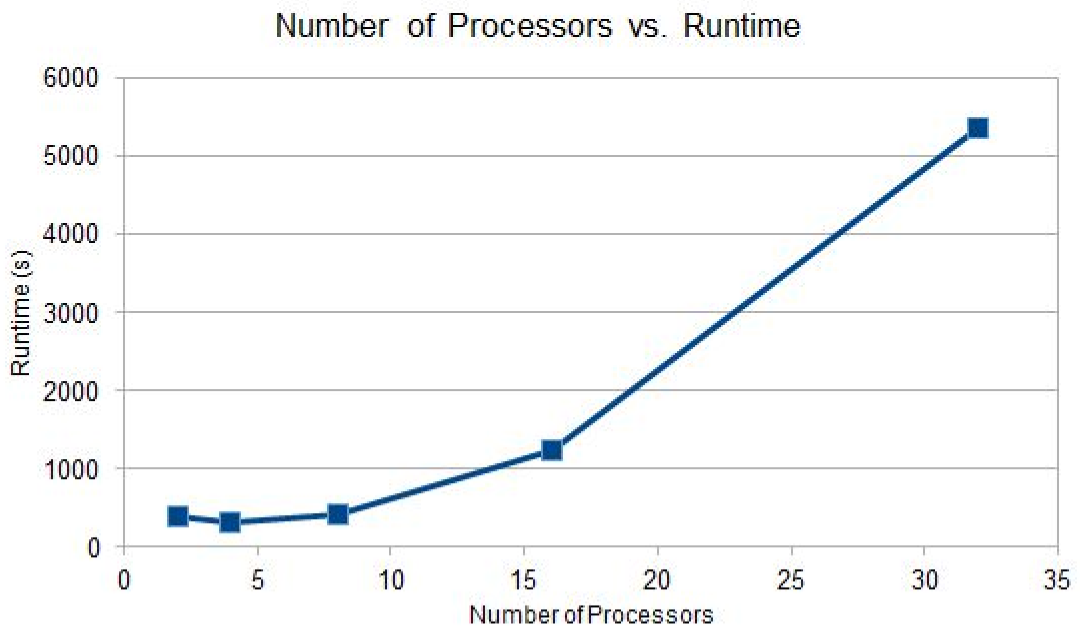
Lowest Energy Seam Test Results

This component was tested at processor counts 2, 4, 8, 16, and 32, yielding the following results outlined in the graph below.



Combined Components Test Results

Both parallel implementations of the components were combined and tested at processor counts 2, 4, 8, 16, and 32, yielding the following results outlined in the graph below.

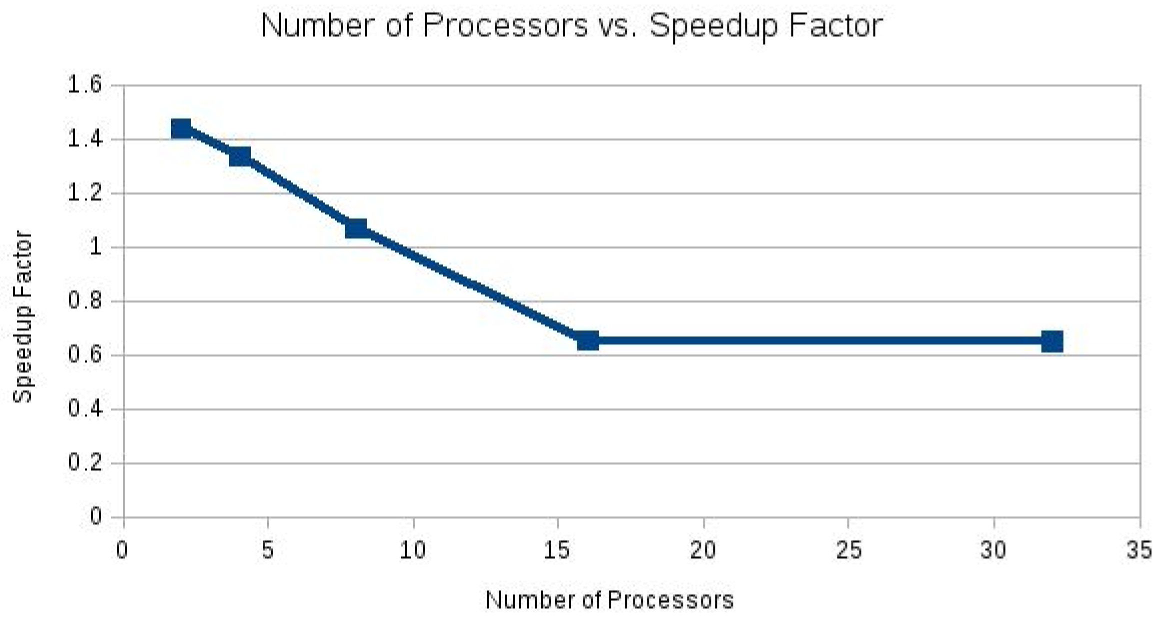


Discussion

Each of the components, and the combination of the two will be analyzed separately based on the test results outlined above.

Energy Function Analysis

The speedup factor of the parallel implementation of the energy function component is illustrated in the graph below as a function of the number of processors.

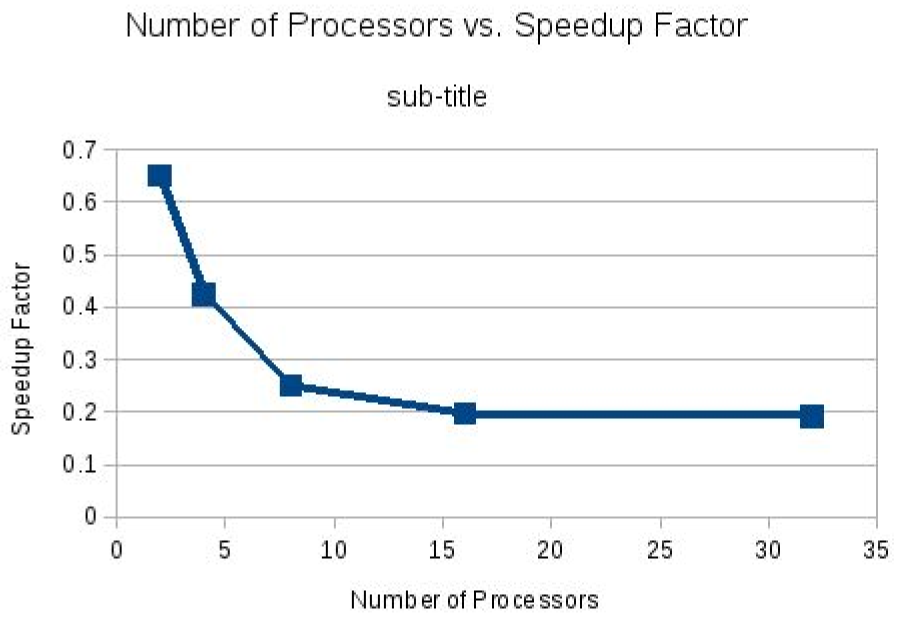


With a serial runtime of 479.214 seconds

For this component, the speedup factor indicates a significant performance increase of the parallel implementation over the serial implementation for 2 and 4 processors. As the number of processors increases beyond 8, the parallel implementation experiences a performance decrease as compared to the serial implementation. This is due to the communication tradeoff for an increasing processor count. Each time the energy of every pixel in the image is computed, every processor must communicate with all others and therefore must wait until this communication is complete before proceeding to the next iteration of the algorithm.

Lowest Energy Seam Analysis

The speedup factor of the parallel implementation of the lowest energy seam component is illustrated in the graph below as a function of the number of processors.



With a serial runtime of 479.214 seconds

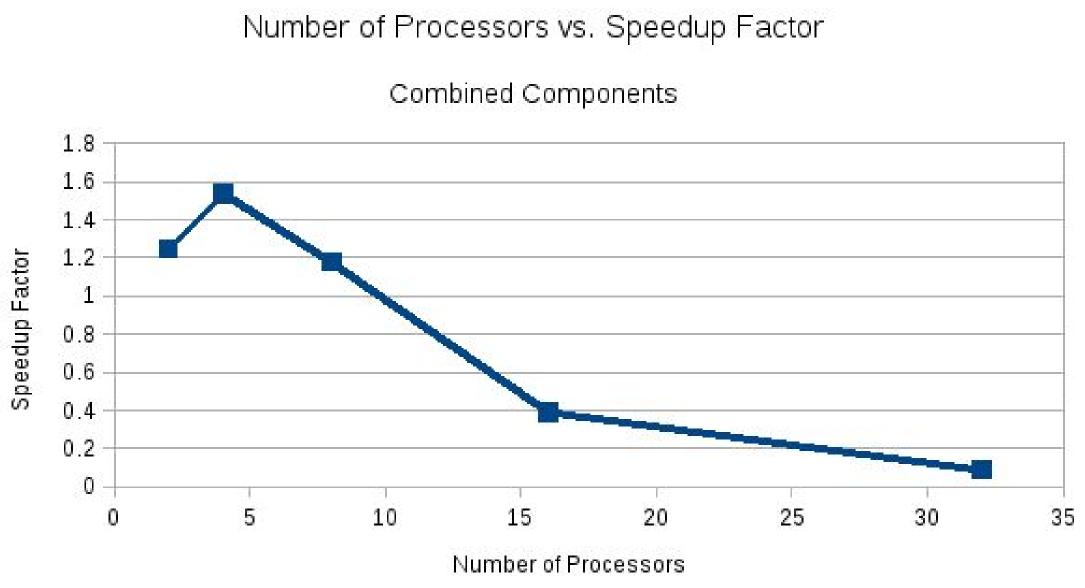
For this component, the speedup factor indicates an exponential performance decrease for the parallel implementation as compared to the serial implementation as the number of processors increases. This result was unanticipated for this component, however it can be explained by several factors.

Each processor is computing the energy for each pixel in the entire image for each seam to be removed. For each iteration of seam computation, neighboring processes but communicate bordering pixel path cost information in order to advance through the algorithm. All processes must also communicate after seam costs are computed in order to retrieve all necessary information for seam removal in serial. This communication overhead far exceeds any potential gains in computation time that the parallel implementation may have over the serial implementation.

This trend holds for images of varying sizes and variable seam removal tests for this particular parallelized component. Contrary to the team's thoughts that the parallelization of this component would offer substantial speedup for large enough seam carving requests, this method incurs additional costs that lead to significant slow down.

Combined Components Analysis

The speedup factor of the parallel implementation of the combined components is illustrated in the graph below as a function of the number of processors.



Running the algorithm with all the parallel optimizations revealed a speedup of 1.54 with 4 processors when compared to the serial implementation. The tests that ran with 2, 4, and 8 processors all provided speedup, but running with 4 processors provided the best results. When more processors are added, the running time becomes greater than that of the serial implementation, increasing exponentially.

As the number of parallel processors increase, the amount of work done per process decreases and the amount of communication between processes increases. This is especially true for the lowest energy seam parallelization because each process depends on information from every other process. However, parallelizing both parts of the algorithm does provide a benefit to run time for a small number of processors.

Expected Schedule Success

As expected, implementing and testing our parallel version of the algorithm took four weeks. Once this was completed, completing performance tests and analysis took two days. This was a few days ahead of the expected requirement, but overall, the expected schedule turned out to be accurate.

Expected Difficulties Resolution

Unfortunately we were not able to establish a line of communication with the server administrator for the Bluewave cluster at UMBC to fulfill the software dependencies for the CImg library used by the project for image processing. In order to overcome this, imgproc was implemented to convert images to raw text data that could be processed by the seam carving program. This program was executed on personal computers that could fulfill the software dependencies for the image processing with the CImg library. The raw text data was then transferred to the Bluewave cluster to be run with the seam carver. Output from the seam carver was then transferred back to personal computers and converted back into an image format using imgproc to ensure correctness of functionality.

Acknowledgements

The project was tested on the Bluewave cluster at The University of Maryland, Baltimore County under the user accounts ablomer1 and vkonen1.

The CImg library and the SL Vector library were used to process images by imgproc and to store pixel data in 3D vectors respectively.

Bibliography

Avidan / Mitsubishi Electric Research Labs, Shamir / The Interdisciplinary Center & Mitsubishi Electric Research Labs. Seam Carving for Content-Aware Image Resizing. Retrieved March 5, 2016, from
http://www.win.tue.nl/~wstahw/edu/2IV05/seamcarving.pdf
Provides information on the method of seam carving, energy functions to be used for the process, as well as seam insertion which is not relevant to this project.

Seam carving. (2016, January 16). Retrieved March 05, 2016, from
https://en.wikipedia.org/wiki/Seam_carving
Provides a summary of seam carving, the typical algorithm, its downsides, and possible improvements.

CImg Library Project. The C++ Template Image Processing Toolkit. Project Manager David Tschumperle. Included library file for serial implementation.
http://datahole.ddns.net/uploads/seam_carver/cimg.h
The CImg library used for image processing to manipulate pixel data.

Berkeley Fluid Animation & Simulation Toolkit. Primary Author James F. O'Brien. Regents of the University of California. Included library files for serial implementation.
http://datahole.ddns.net/uploads/seam_carver/sl_io.h
http://datahole.ddns.net/uploads/seam_carver/sl_vector.h
Vector 3 implementation to store pixel data for energy function calculations and comparisons.

Project Documents

Both the serial and parallel implementations, as well as relevant project documentation, test results, and resources can be found on the project web page.

http://datahole.ddns.net/cmsc483/main/index.html