Fast Poisson Image Blending using Parallelized Jacobi Method

Amatur Rahman
Pennsylvania State University

Introduction

Goal:
Seamlessly cut and paste a source image on a target image, where the source and target are visually different.

- Gradient domain processing of image
- Computationally expensive, even more for gigabyte sized images.

Why parallel?

- Sequential algorithms do not scale well to solve these problems.
- Too large image size to fit into one node

Poisson Image Blending

Seamlessly blend two images into a single one.

Used in: Mainly Digital Image Processing, Data augmentation in deep neural networks, Medical Imaging.

Effect of Parallelizing the Code

For Image size = 40 x 40 and 10 processors

Optimized Serial Runtime: 10.6 second
Parallel Jacobi Runtim: 1.21 second

8X speedup
Parallel version is performing better

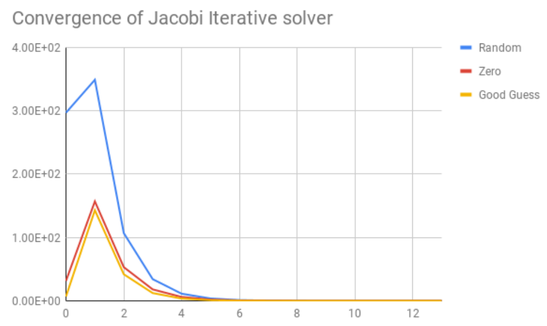
For smaller problem size, the problem size is much smaller to make effective use of parallelism.

For dimension 20*20=400, non-MPI serial version actually does better.

For trivial problems, setting up the overhead of costly parallel communication is not helpful, rather it increases runtime.

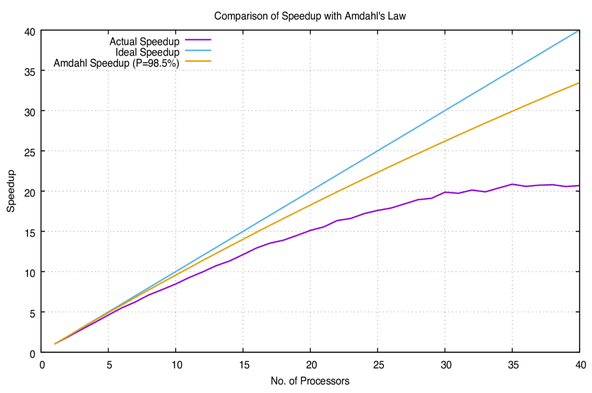
Direct vs. Indirect Solvers

- Our 'A' matrix is sparse, so indirect solver Jacobi does better.
- Sparse matrices do not generally have sparse LU decomposition, so it gets harder to fit the matrices in memory as the problem size goes larger.
- For Jacobi method, we store the A matrix in sparse representation.
- We tweaked the initial conditions to achieve fast convergence of direct solver Jacobi.



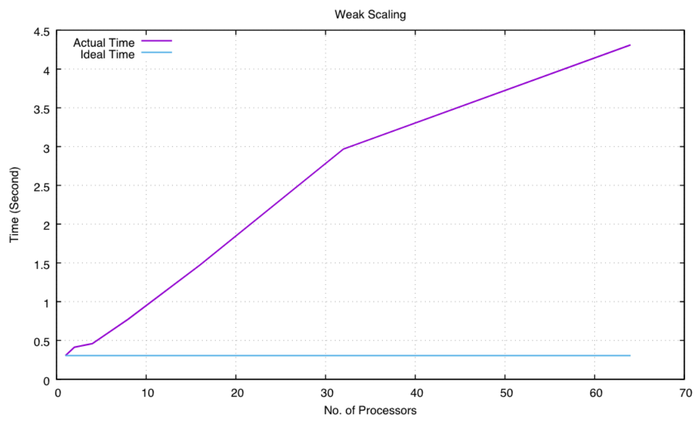
Strong Scaling

For an image with 6400x6400 pixel, the parallel jacobi iterative method can show upto **20x speedup**.



Weak Scaling

Keeping the workload per process constant, we aim to show the rate of increase in runtime.



Coupling

TODO

Results Overview

- Among direct and iterative solver, iterative solver runs **faster,** but gives inexact solution.
- Parallelization works well, but up to a point.
- Among serial and parallel method, parallel method proves it usefulness when the compute time and memory is exceeded in a single computer node.

Code Availability

The code is publicly available in github:
https://github.com/amatur/cse597_parallel_solver

Acknowledgements

ICS-ACI computing resources were used to perform the computations.

All resources were provided by Dr. Adam Lavely and Dr. Chris Blanton.

Bibliography

1] Poisson blending. Retrieved from http://eric-yuan.me/poisson-blending/ accessed 2018-09-23.
[2] Poisson image editing. Retrieved from http://www.ctralie.com/Teaching/PoissonImageEditing/, accessed 2018-09-23.
[3] Barker, B. Message passing interface (mpi). In Workshop: High Performance Computing on Stampede (2015), vol. 262.
[4] Gropp, W. D., Gropp, W., Lusk, E., Skjellum, A., and Lusk, A. D. F. E. E. Using MPI: portable parallel programming with the message-passing interface, vol. 1. MIT press, 1999.