

Parallelized Jacobi Method for Fast Poisson Image Blending

Amatur Rahman

Department of Computer Science and Engineering, Pennsylvania State University

E-mail: aur1111@psu.edu

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

Science and Problem Definition: 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 Runtime: 1.21 second

8X speedup

Parallel version is performing better

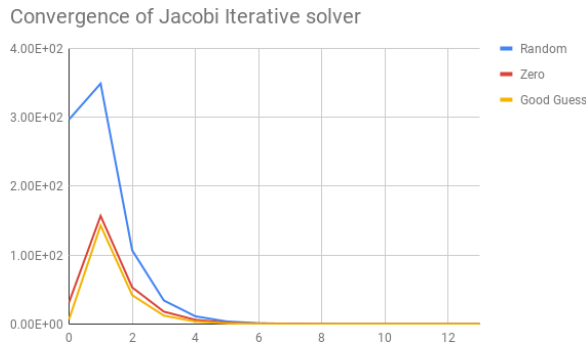
For smaller problem size, the problem size is much smaller to make effective use of parallelism.

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.

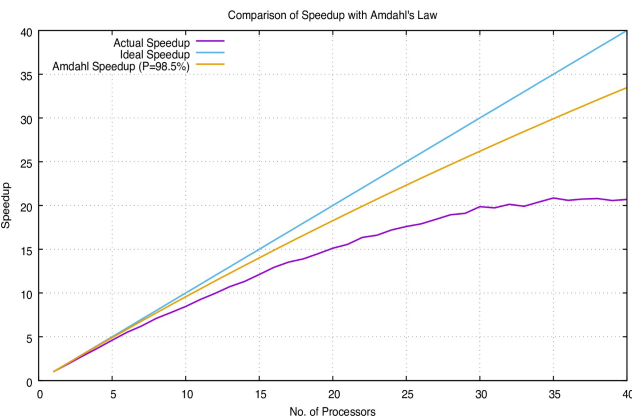
Direct vs. Indirect Solvers

- 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.



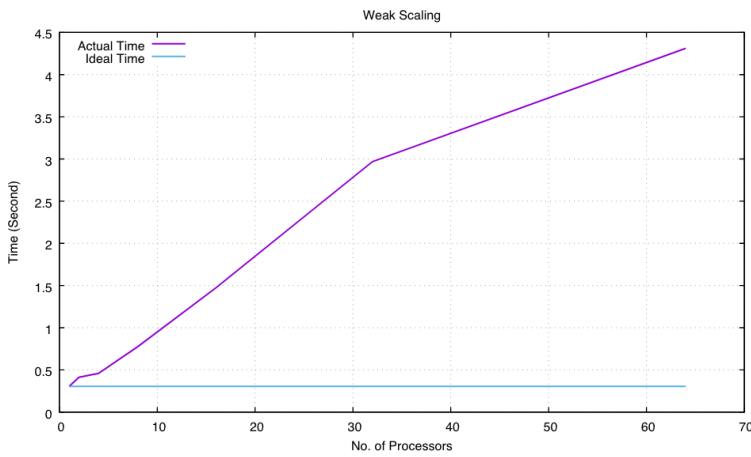
Strong Scaling

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 to External Library

TODO

Results Overview

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

Acknowledgements

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

All resources required for completing this project were provided by Dr. Adam Lavelly and Dr. Chris Blanton.

References

- [1] <http://eric-yuan.me/poisson-blending/>
- [2] <http://www.ctralie.com/Teaching/PoissonImageEditing/>
- [3] Barker, B. Message passing interface (mpi). In Workshop: High Performance Computing on Stampede (2015), vol. 262.

Code Availability

Serial and Parallel Code:

https://github.com/amatur/cse597_parallel_solver

Coupling to External Library and GPU Code:

<https://github.com/amatur/cuda597>