Parallel ODETLAP for Terrain Compression and Reconstruction

Jared Stookey
Dan Tracy
Zhongyi Xie
Barbara Cutler
W. Randolph Franklin
Marcus V. A. Andrade

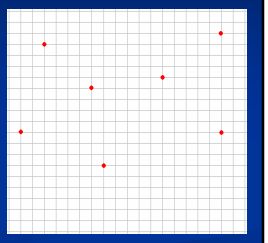
Outline

- Quick Overview of our research
- ODETLAP (Non-Patch)
- Motivation for Parallelization
- Our Approach
- MPI Implementation
- Results
- Current and Future Work

Quick Overview

- Our research
  - Terrain compression
  - Compress terrain by selecting subset of points
  - Reconstruct the terrain by solving a system of equations to fill in missing points
  - The method we use to reconstruct the terrain is slow for large datasets
  - We came up with a method for reconstructing very large datasets quickly using MPI

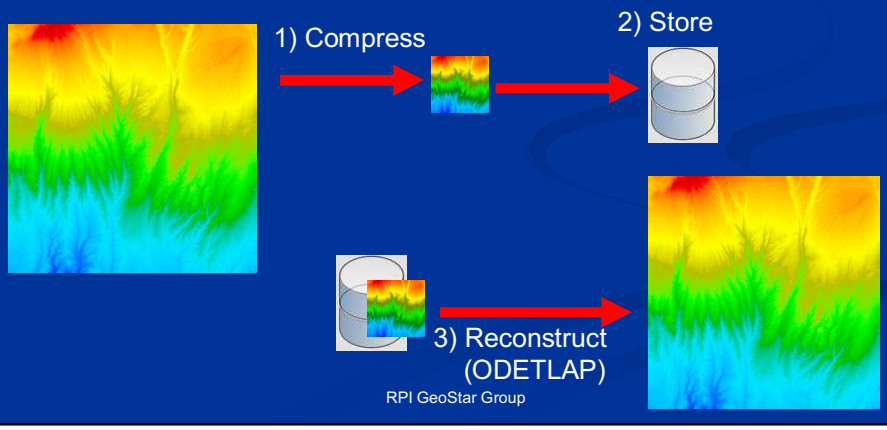
ODETLAP



- Over-Determined Laplacian
- Two Equations:
  - $4z_{ij} = z_{i-1,j} + z_{i+1,j} + z_{i,j-1} + z_{i,j+1}$
  - $z_{ij} = h_{ij}$
  - Multiple values for some points
  - Require a smooth parameter R to interpolate when multiple values exist
- Reconstruct an approximated surface from $\{h_{ij}\}$ (Red points)

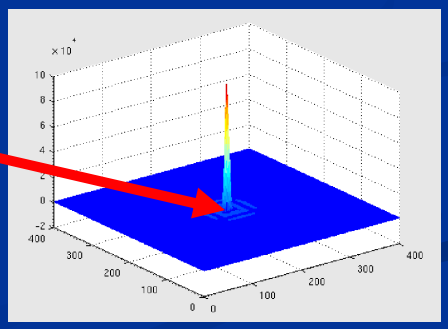
ODETLAP Compression

- Lossily compress image by selecting subset of points
- ODETLAP reconstruction solves for the whole terrain



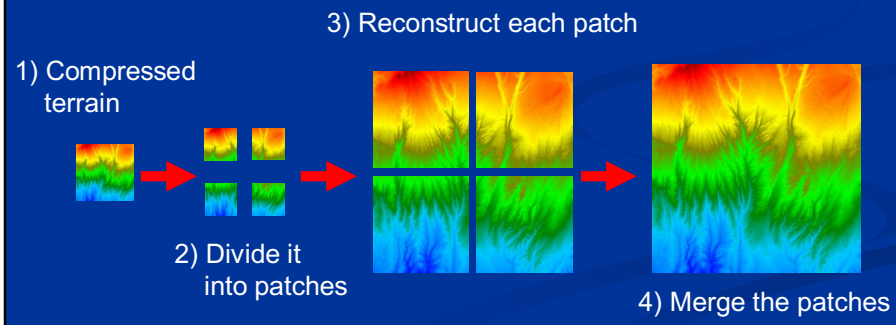
Motivation for Parallelization

- ODETLAP prohibitively slow for large datasets
- We need a scalable implementation
- Only a small neighborhood of points will affect a particular elevation.
- 1 pixel only affected an area of 62x62



Our Approach

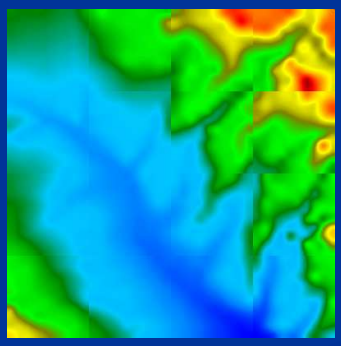
- Divide the terrain into individual patches
- Run ODETLAP on each patch separately



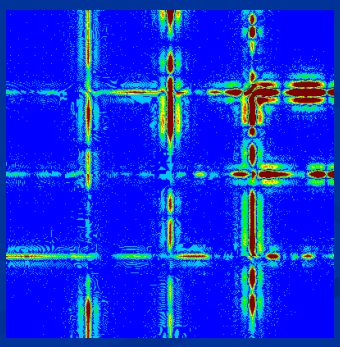
There is a problem! (continued)

- We get discontinuity if we naively merge the patches

Naively reconstructed terrain:



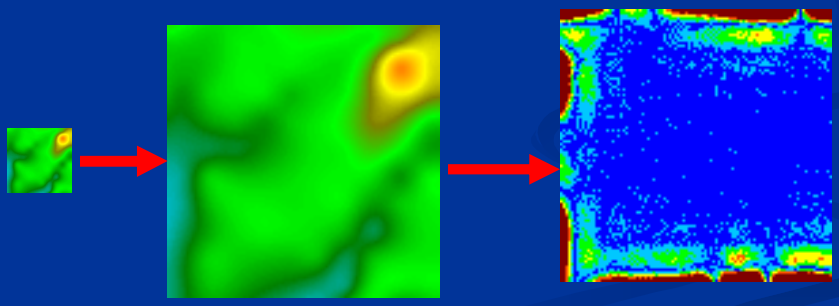
Errors:



There is a problem!

- Points near the edges of patches have incomplete data which causes errors

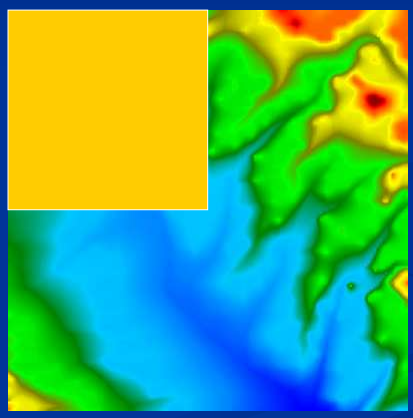
Pixels in red show erroneous results



Pixels in blue show correct results

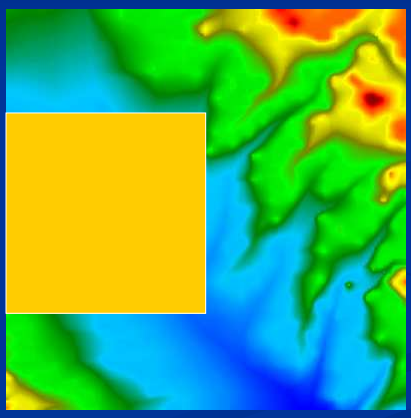
Solution

- Use overlapping layers of patches



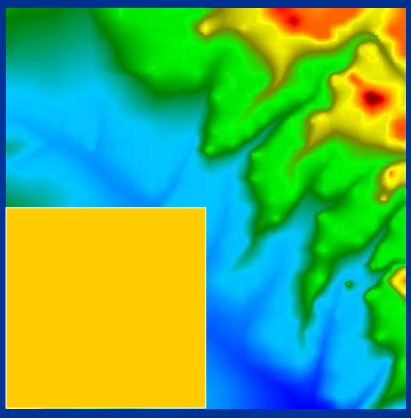
Solution

- Use overlapping layers of patches



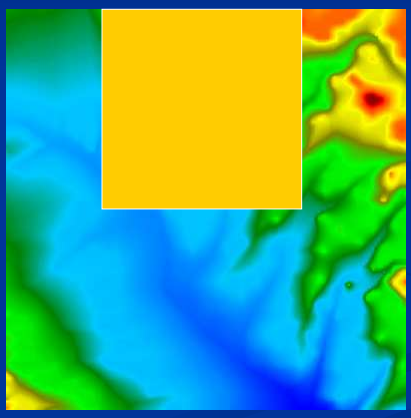
Solution

- Use overlapping layers of patches



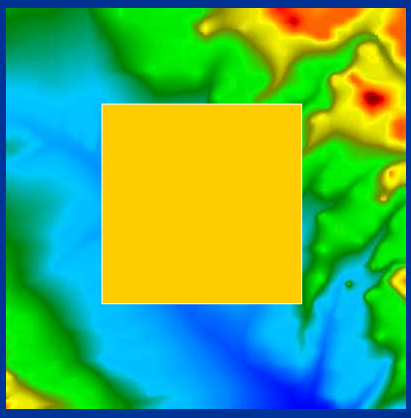
Solution

- Use overlapping layers of patches



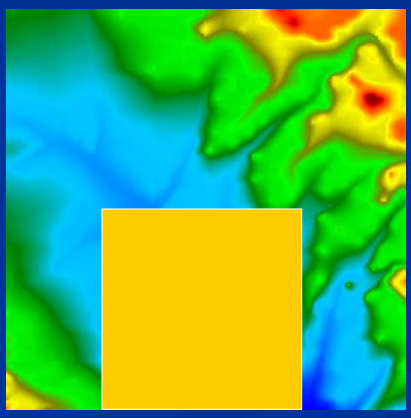
Solution

- Use overlapping layers of patches



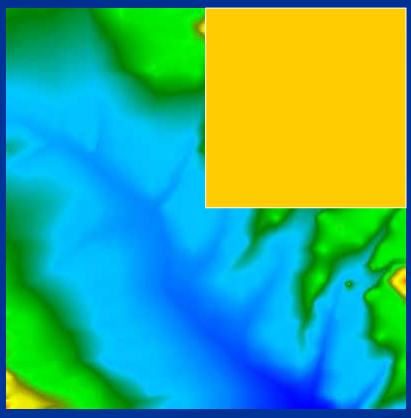
Solution

- Use overlapping layers of patches



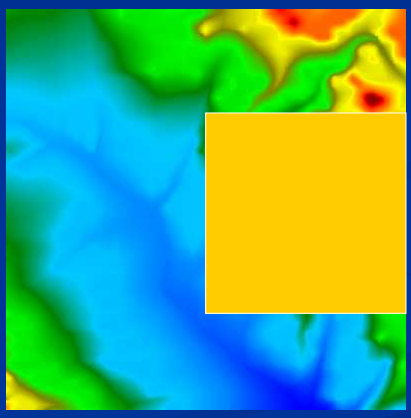
Solution

- Use overlapping layers of patches



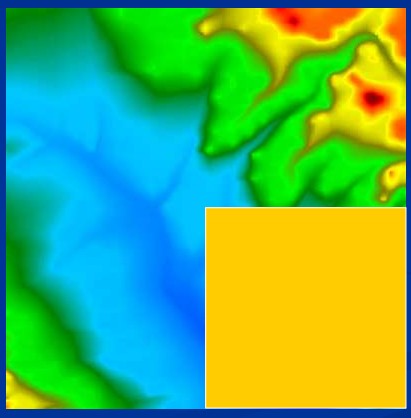
Solution

- Use overlapping layers of patches



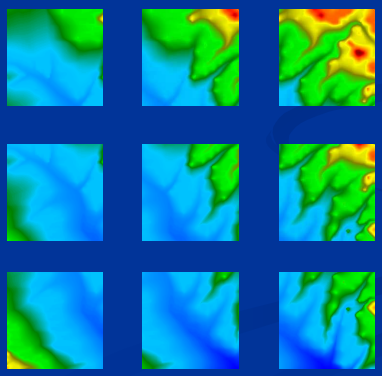
Solution

- Use overlapping layers of patches



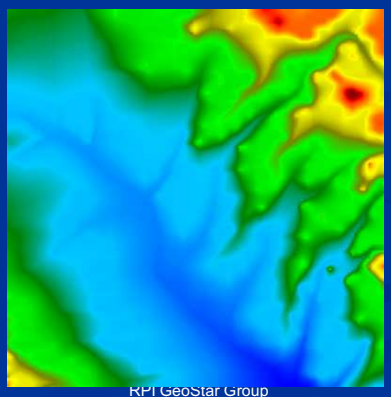
Solution

- Use overlapping layers of patches
- Then merge the results



Solution

- Use overlapping layers of patches
- Then merge the results

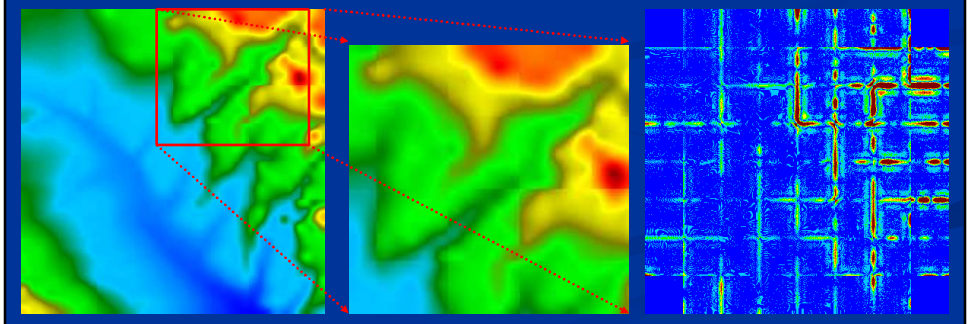


Problem: Averaging the patches

- A simple averaging of the patches incorporates the border error into the reconstructed terrain:

Terrain reconstructed using averaged patches

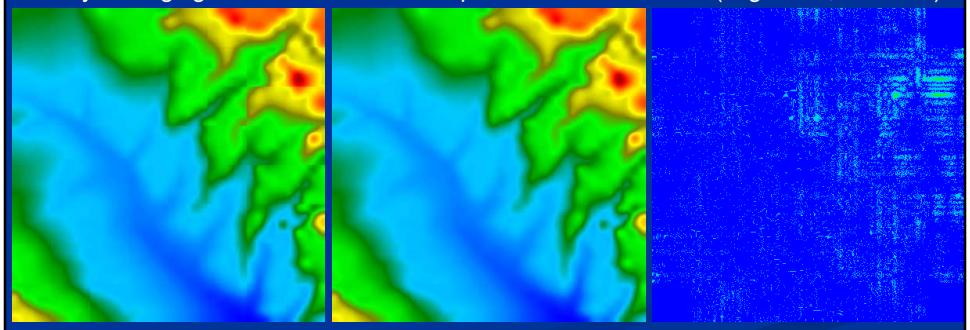
Errors:



Solution: Bilinear Interpolation

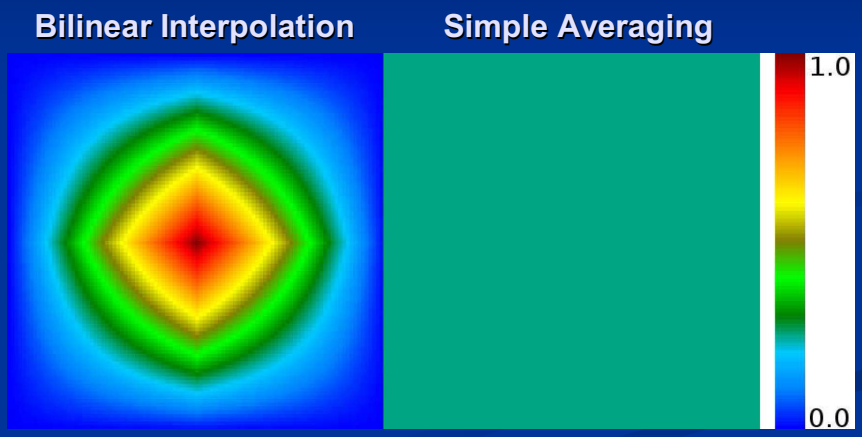
- Use bilinear interpolation to do a weighted average such that border values fall off to zero:

Naively averaging results | Bilinear interpolation results | Error (avg: 0.1m, max: 2m):



Elevation Range of the Original: 1105m..1610m
Using DTED Level 2 (30m spacing)

Weighting Pattern for Bilinear Interpolation vs. Simple Averaging

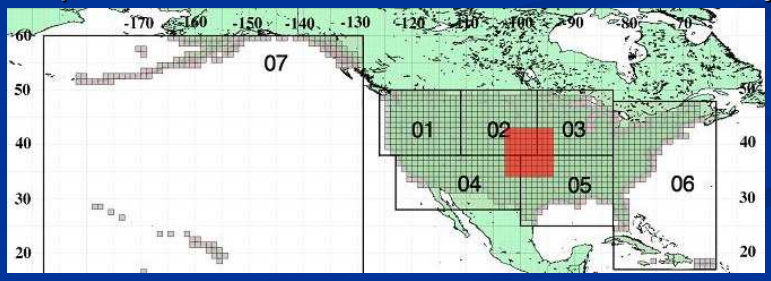


MPI Implementation

1) Each processor (except central process) is pre-assigned one or more patches
2) Every MPI process does the following for each patch assigned to it:
   - Load patch
   - Run ODETLAP on the patch
   - MPI_send the patch to the central process
3) When all of the patches have been received by the central process, merge them using bilinear interpolation.

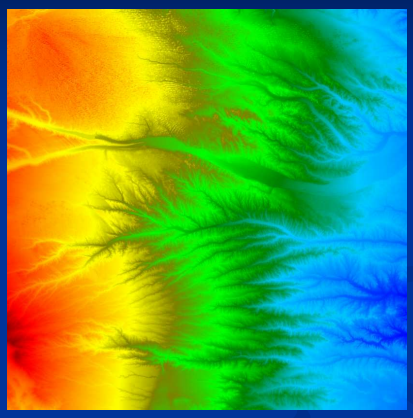
Results

- 16,000*16,000 Central USA terrain data
- Use 128 2.6 GHz processors on RPI CCNI cluster
- Divide into 101,761 patches of 100x100 size
- Completed in 28 minutes and 32 seconds
- Non-patch ODETLAP would have taken 179 days



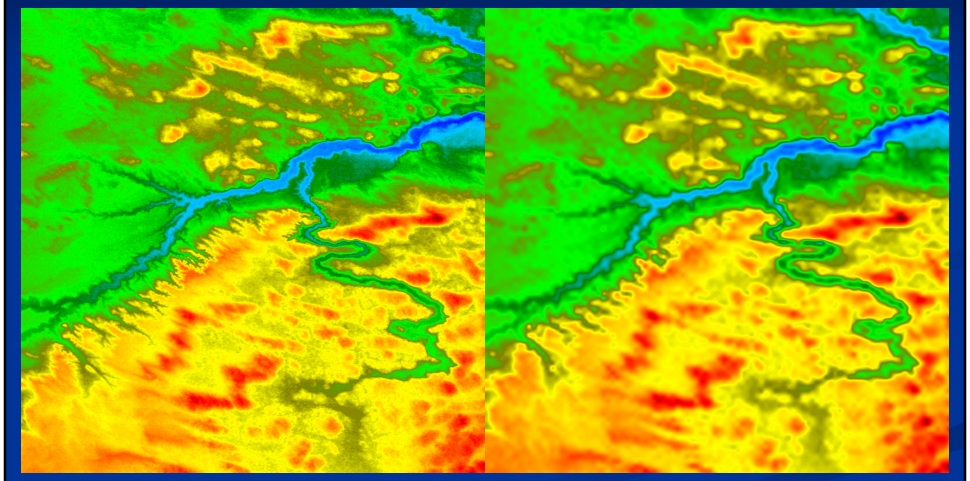
Results(cont.)

- Size: 16K*16K
- STD: 217
- Range: 1013
- Mean Error: 1.96
- Max Error: 50
- RMS Error: 2.76



The terrain was compressed by a factor of 100, with a mean error within 0.2% of the range.

Original and reconstructed Terrain



Original Terrain (1000 * 1000) Reconstruction Result (1000 * 1000)

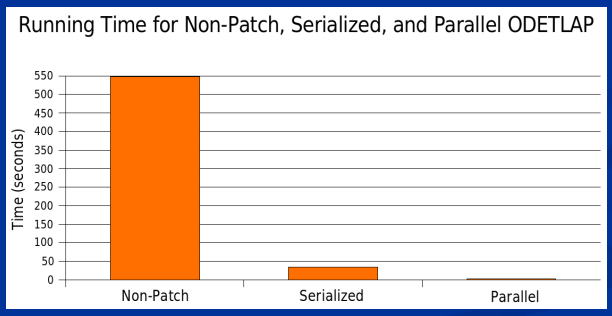
Patch Size vs. Time & Error

Total size	#points used	Patch size	Running time	Mean absolute Error	Max Error	RMS Error
2000*2000	39894	50*50	0m 38s	0.6640	13	0.9617
2000*2000	39894	100*100	0m 55s	0.6598	13	0.9530
2000*2000	39894	200*200	5m 25s	0.6598	13	0.9527
2000*2000	39894	400*400	18m 49s	0.6598	13	0.9527

These results come from an 8-processor machine

Serialized vs. Parallel

- Serialized: A single worker processor runs each patch sequentially (speedup of 9.5 in the test)
- Parallel: Several processors run on many patches in parallel (additional speedup of 5.6 in the test)



Test data: 800 x 800 size with mean elevation of 107

Running Time Comparison

Method	Running time	Mean Error	Max Error	RMS Error
Original ODETLAP	549s	0.6150	7	0.8835
Serial ODETLAP	34s	0.6156	7	0.8846
Parallel ODETLAP	9s	0.6156	7	0.8846

Test data: 800 x 800 size with mean elevation of 107, run on 8 processors. Parallel ODETLAP is 50 times faster, while introducing only 0.1% additional error.

Current and Future Work

- Improvements to our implementation
  - Reduce data size – regular grid can be more compact
  - Each process should grab the next available patch
  - Optimize for the Blue Gene/L system (see next slide)
- Reduce errors from the patch method
  - Improve the method for merging patches

Blue Gene/L System

- Computational Center for Nanotechnology Innovations (CCNI) at RPI
- 32,768 CPU's @ 700 Mhz
- 512-1024MB memory/CPU (non-shared)
- Opportunity to run very large data sets quickly
- New method
  - Source, Sink, Workers, and Coordinator
  - DEM size is not limited by process memory size
  - Use processors as cache instead of the disk
    - On the BG, disk is slow, network and memory is very fast
    - We must reduce the overhead to take advantage of all CPU's