Image Segmentation using K-Means Clustering

CSE633 (Spring ‘20)

Instructor: Dr. Russ Miller

By Sneha Panicker

University at Buffalo
School of Engineering and Applied Sciences

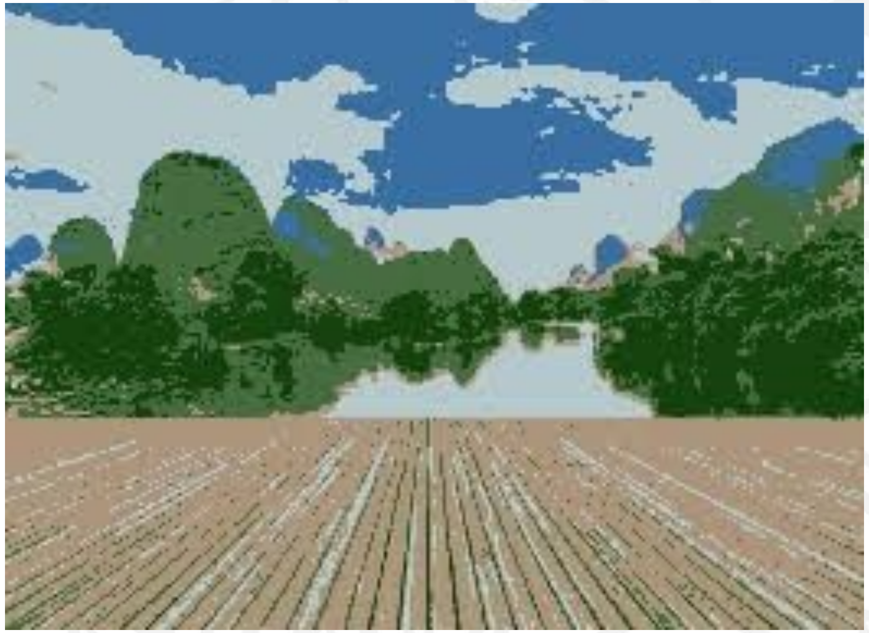
Outline

- Problem Definition
- k-Means Clustering Algorithm
- Parallel Implementation
- Results
- Observations
- Challenges
- References

Problem Definition

Image Segmentation using k-Means Clustering.

Original Image



Segmented Image with 5 clusters

k-Means Clustering Algorithm

1. Randomly select k pixels to be cluster centers.
2. For each pixel in the data set, associate it with the cluster which has its center closest to the pixel.
3. Calculate new cluster centers by averaging all pixels in a cluster.

Repeat 2 and 3 for a particular number of iterations.

Parallel Implementation

Creating Dataset from Image (Serial)

- Read the image using OpenCV for Python.
- Append the R, G, and B values of the pixels to a list for each pixel.
- Saving the list as a pickle file.

Parallel k-Means

1. Consider N pixels and P processors.
2. Assign N/P pixels to each processor using the pickle file.
3. Processor 0 randomly selects k pixels as cluster centers and broadcasts them.
4. Each processor for each of its pixels, finds the cluster to which the pixel belongs.
5. Each processor recalculates local sums for each cluster.
6. Each processor sends its local sums to processor 0 in order to find the global cluster centers.
7. Repeat the clustering for the specified number of iterations. (i.e. repeat steps 4-6)
8. Form a pickle file with information about each pixel's final cluster center.

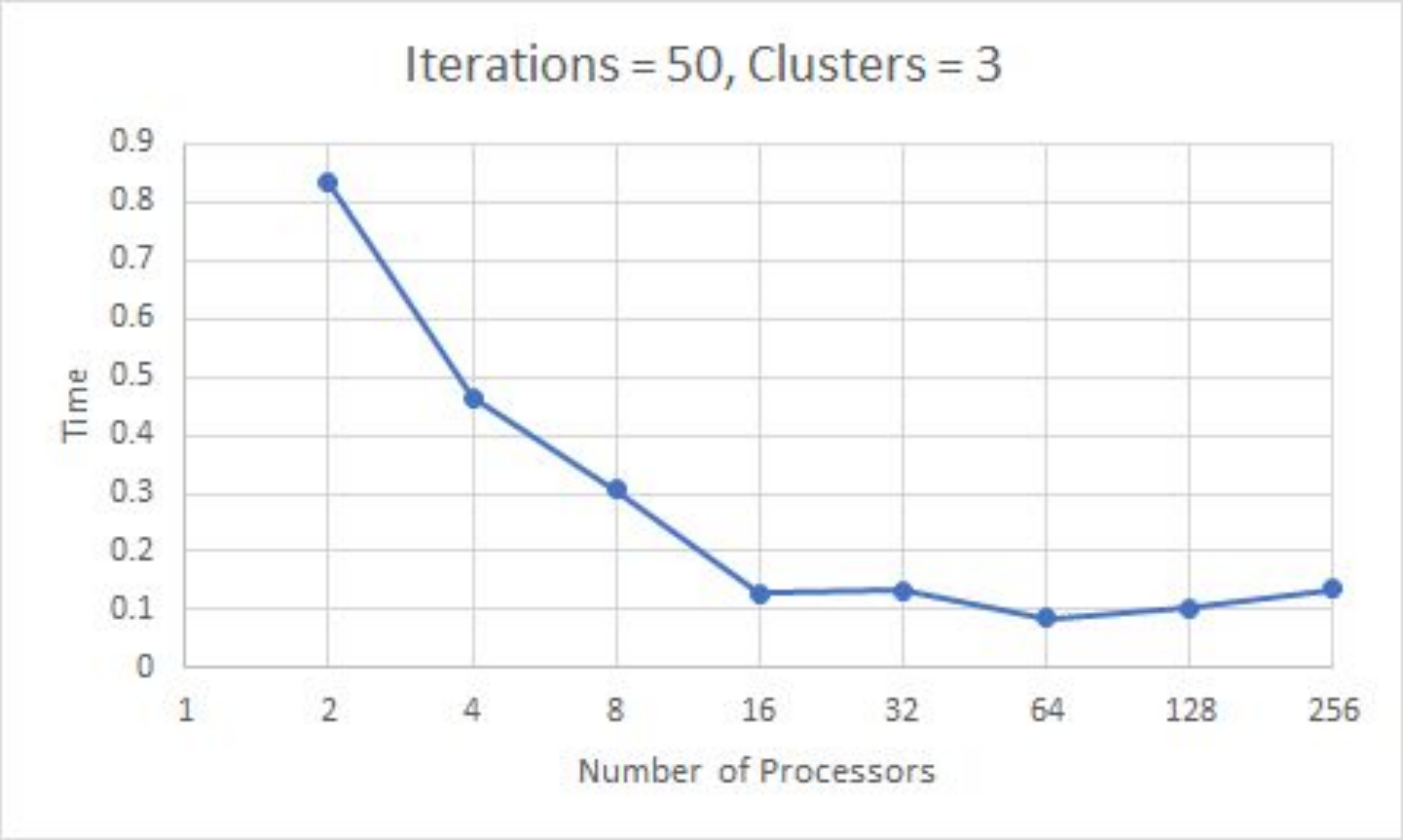
Segmented Image Formation (Serial)

- Read the file with information about each pixel's corresponding cluster centers.
- Read the image.
- For each pixel, overwrite the pixel value with the cluster center.
- Save the resulting image.

Results

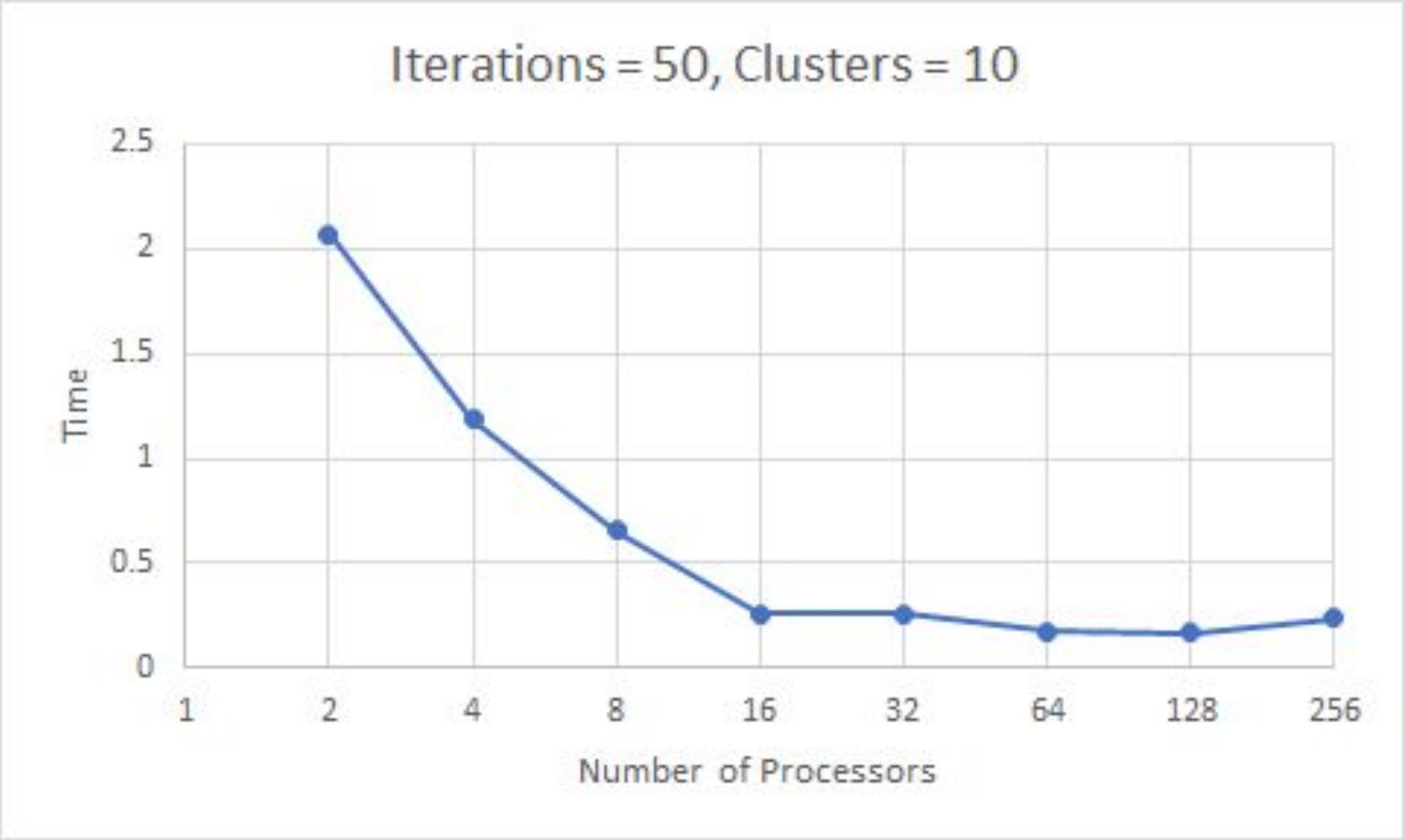
Analysis on Image size 128 x 128

PROCESSORS	TIME (s)
2	0.834
4	0.446
8	0.308
16	0.125
32	0.132
64	0.085
128	0.103
256	0.136



Analysis on Image size 128 x 128

PROCESSORS	TIME (s)
2	2.070
4	1.187
8	0.653
16	0.256
32	0.258
64	0.178
128	0.164
256	0.238

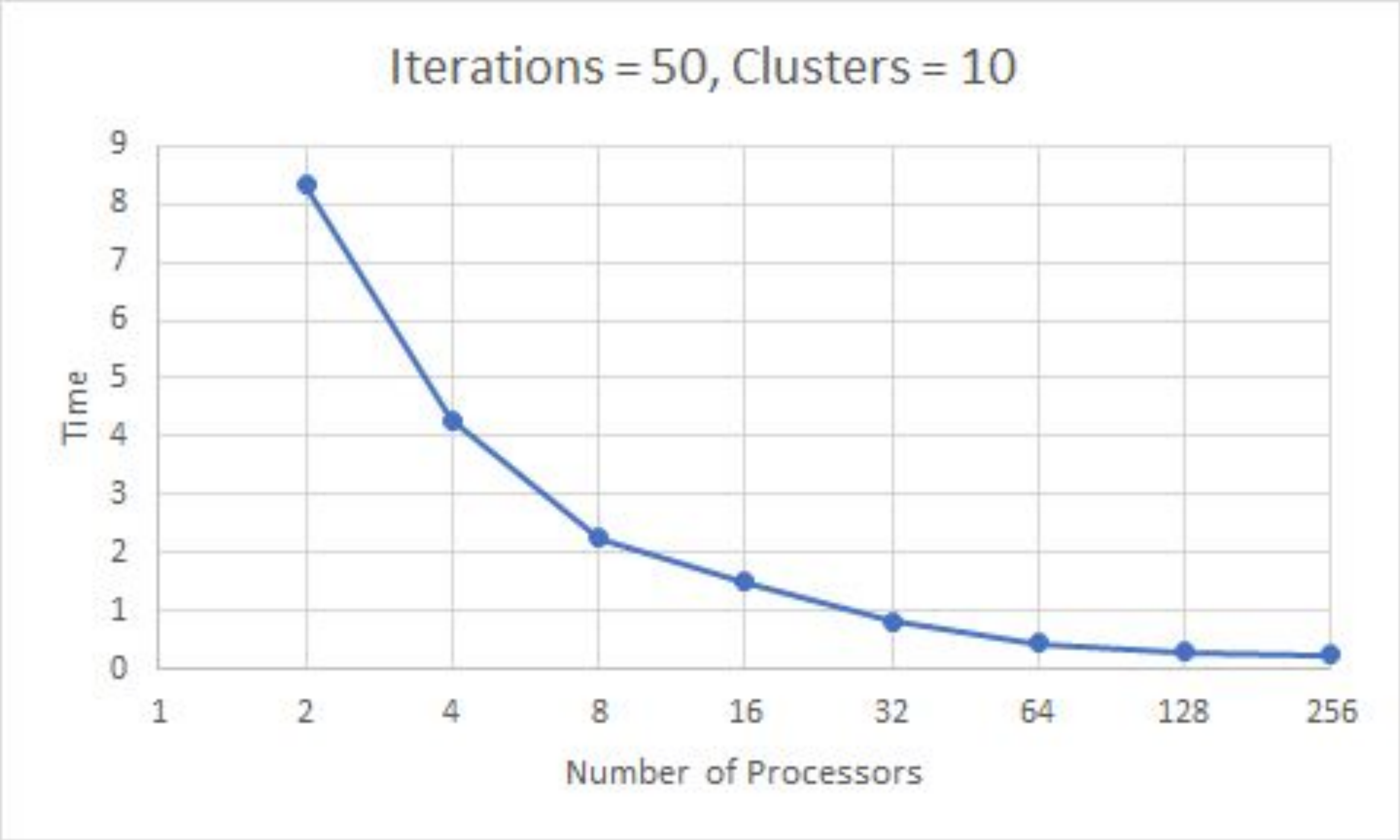


Analysis on Image size 256 x 256

PROCESSORS	TIME (s)
2	3.019
4	1.589
8	0.879
16	0.578
32	0.327
64	0.231
128	0.141
256	0.145

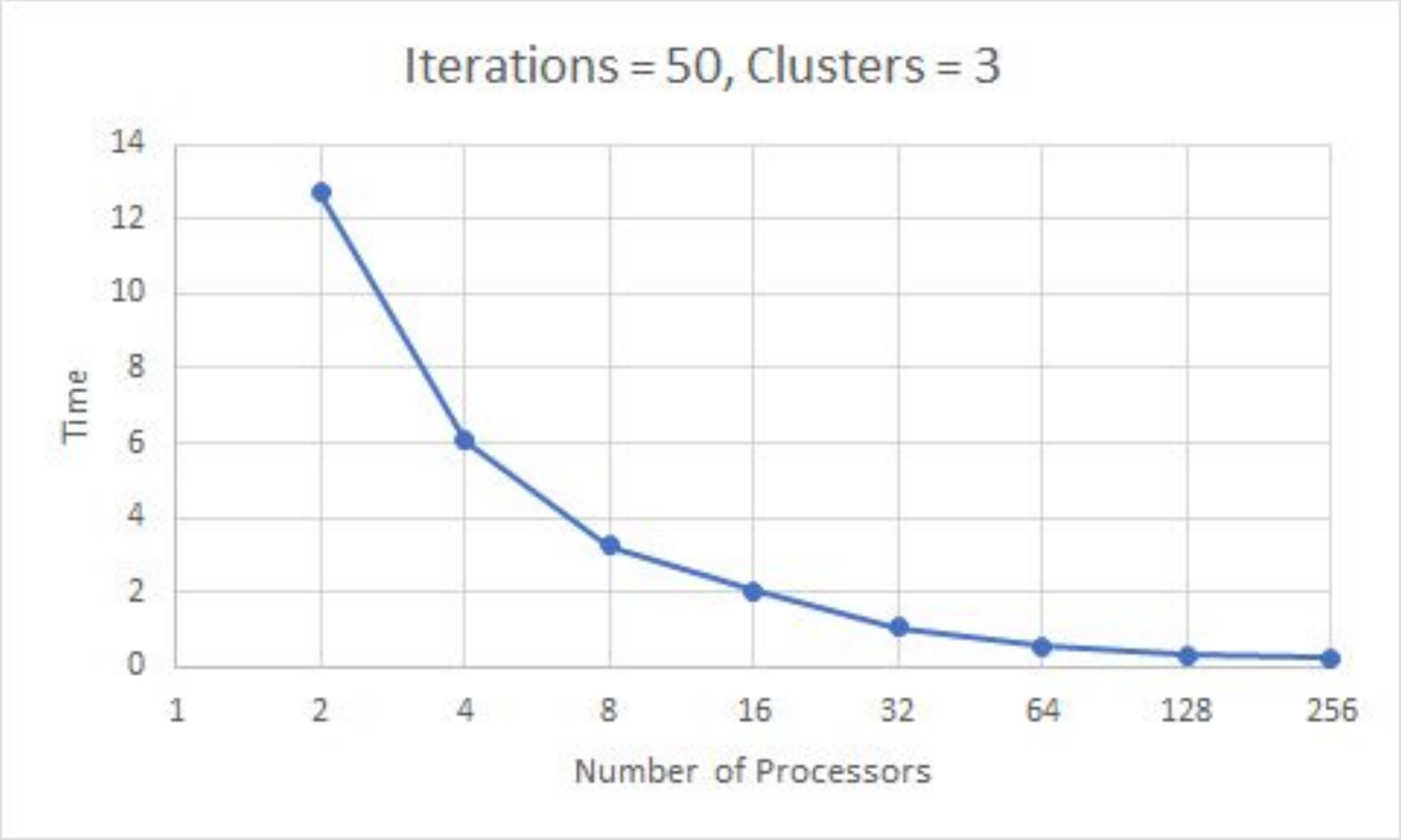
Analysis on Image size 256 x 256

PROCESSORS	TIME (s)
2	8.310
4	4.278
8	2.244
16	1.489
32	0.827
64	0.442
128	0.282
256	0.253



Analysis on Image size 512 x 512

PROCESSORS	TIME (s)
2	12.698
4	6.072
8	3.233
16	2.039
32	1.083
64	0.575
128	0.340
256	0.244

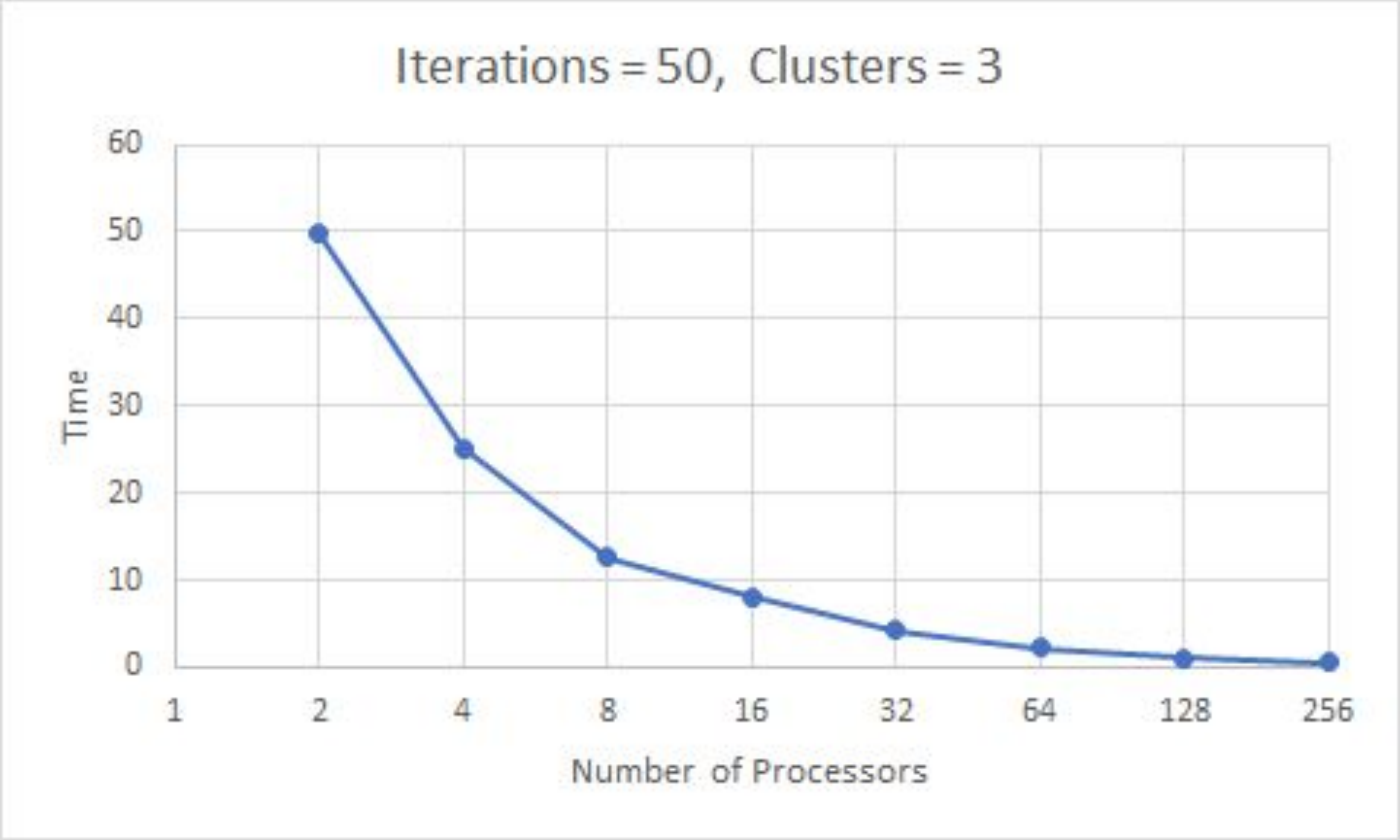


Analysis on Image size 512 x 512

PROCESSORS	TIME (s)
2	34.458
4	17.251
8	8.832
16	5.669
32	3.031
64	1.468
128	0.886
256	0.558

Analysis on Image size 1024 x 1024

PROCESSORS	TIME (s)
2	49.803
4	25.186
8	12.537
16	8.041
32	4.247
64	2.194
128	1.106
256	0.584



Analysis on Image size 1024 x 1024

PROCESSORS	TIME (s)
2	137.135
4	68.407
8	34.709
16	23.579
32	11.543
64	5.571
128	2.909
256	1.464

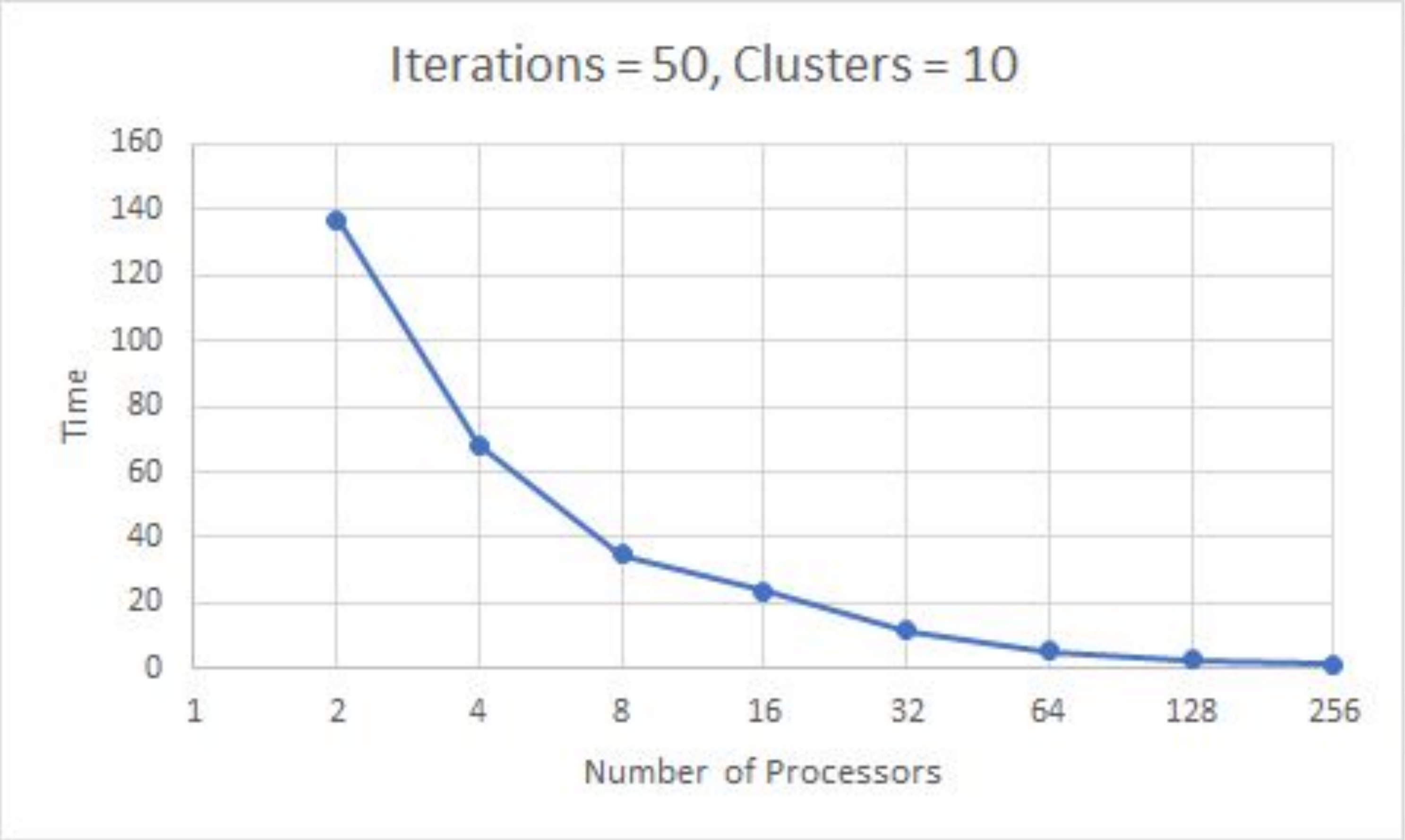


Image Results

Original Image

3 Clusters

10 Clusters

Observations

- For image size 128*128 pixels, the time taken starts to increase as we increase the number of processors beyond 64.
- For image size 256*256 pixels, the time taken for 128 processors and 256 processors remains comparable.
- For image sizes 512*512 pixels and 1024*1024 pixels, we see a consistent decrease in the time taken all the way till 256 processors.

Challenges

- The input pickle files and the output images had to be created serially because of lack of OpenCV support on UB clusters.
- The jobs scheduled to run with more than 64 processors were stuck in the queue for a long time because of receiving low priority and sometimes because of the resources being unavailable.

References

- Algorithms Sequential & Parallel: A Unified Approach (Dr. Russ Miller, Dr.Laurence Boxer)
- mpi4py package documentation
- https://ubccr.freshdesk.com/support/solutions/articles/5000686927
- http://www.buffalo.edu/ccr/support/research_facilities/general_compute.html
- Analytics Vidhya blog on Image Segmentation

Thank You.