Image Compression using K-Means clustering

CSE 633 - Parallel Algorithms

Instructor: Dr. Russ Miller

By: Aashna Mahajan - 50317416

University at Buffalo The State University of New York

Outline

- Problem statement
- Image Compression
- K-Means clustering algorithm
- Sequential Algorithm
- Parallel Algorithm
- Results
- Observations
- References

Problem statement

Compressing the Image using K-Means clustering.

Image Compression

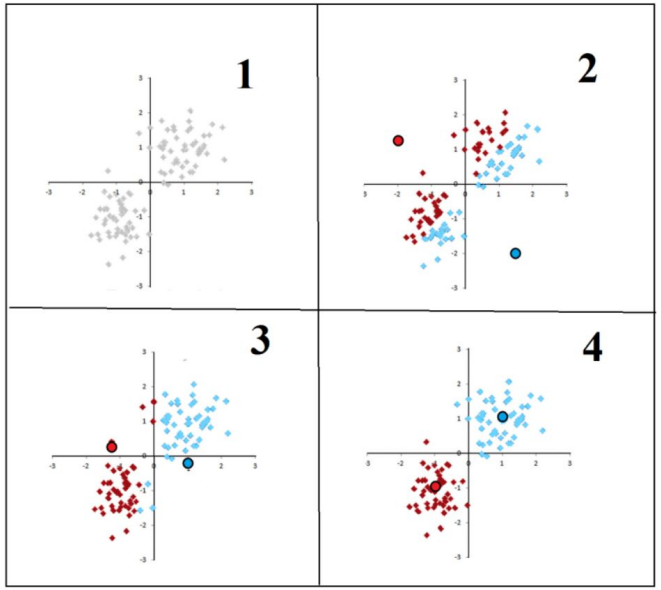
Image Compression is a type of data compression applied to digital images, to reduce their cost for storage or transmission.

Applications

- Medical Imaging
- Face Recognition and Detection
- Satellite Remote Sensing
- Software and Security Industry
- Retail Stores
- Federal Government Agencies, etc.

K-Means Clustering Algorithm

- K-means clustering is the optimization technique to find the ‘k’ clusters or groups in the given set of data points.
- Initially, select ‘k’ data points to be the cluster centers.
- Assignment step - Assign each data point to the closest cluster centers.
- Update step - Calculate the new cluster centers by taking average of all the data points in each cluster.
- Repeat the assignment and updation steps for a particular number of iterations.



Sequential Algorithm

- Read the image using Python OpenCV.
- Select ‘k’ number of clusters.
- Randomly, select ‘k’ pixels from the image to be the cluster centers.
- Iterate through each pixel in the image and assign it to the closest cluster center.
- Take average of all the pixels in each cluster, which will give us the new cluster centers.
- Repeat the assignment and updation steps for a particular number of iterations.
- Update the image with the new pixels.

Original Image

Compressed Image - 5 Clusters

Compressed Image - 10 Clusters

Compressed Image - 20 Clusters

Parallel Algorithm

- Convert Image to pixels with RGB values in a text file.
- Consider P processors and N pixels of the image.
- Assign N/P pixels to each processor.
- Processor 0 selects ‘k’ pixels randomly as the cluster centers and broadcasts them.
- Each processor assigns each of it’s pixels to the closest cluster.
- Calculate local sums for each clusters in each processor.
- Each processor sends it’s local sums to processor 0 to find the global cluster centers.
- Repeat the clustering for the specified number of iterations.
- Save the final cluster centers of the final iteration in another text file.
- Iterate each pixel of the original image and cluster them according to the final cluster points.
- Save the New Image.

Results

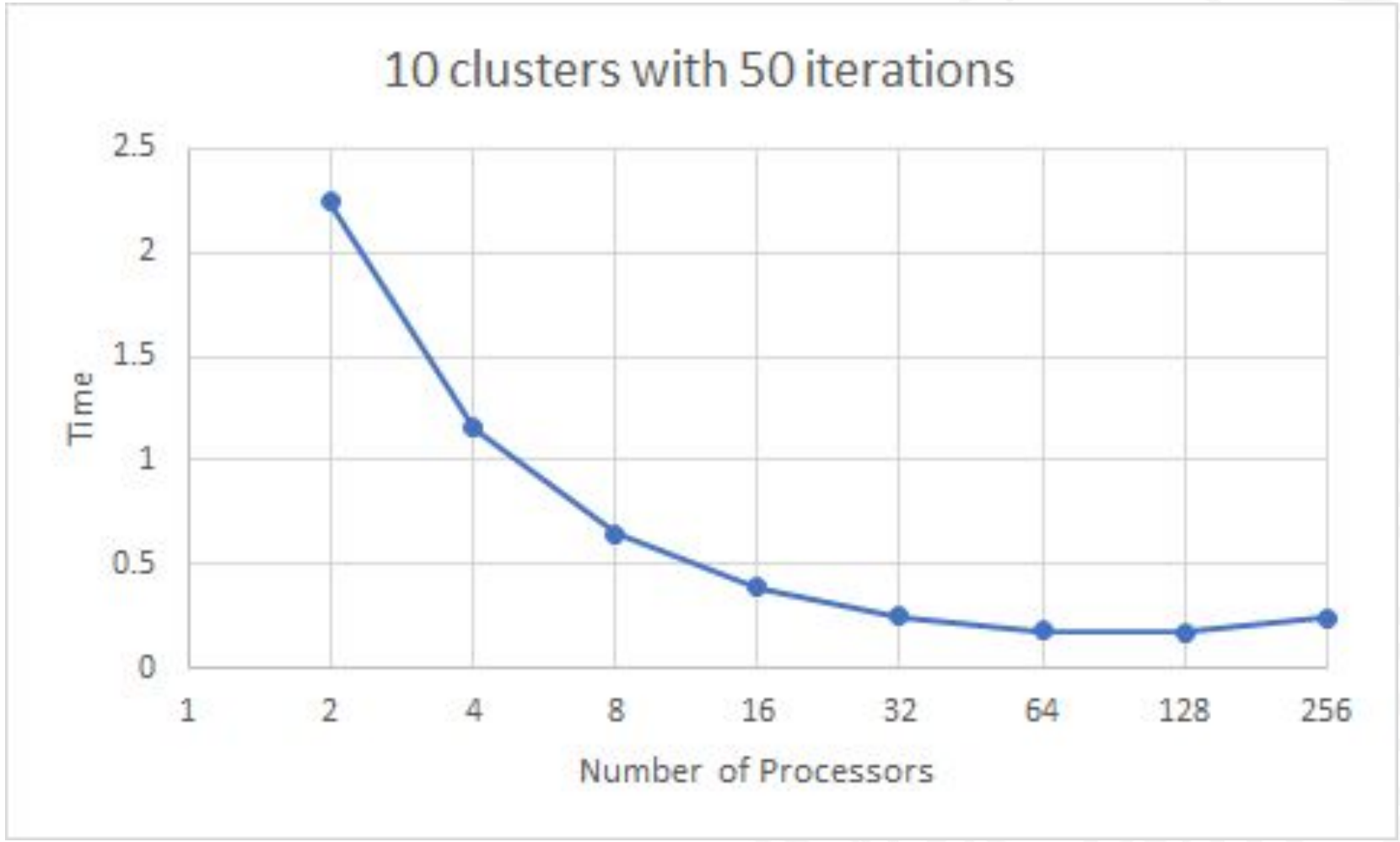
Original Image

Compressed Image - 5 Clusters

Compressed Image - 10 Clusters

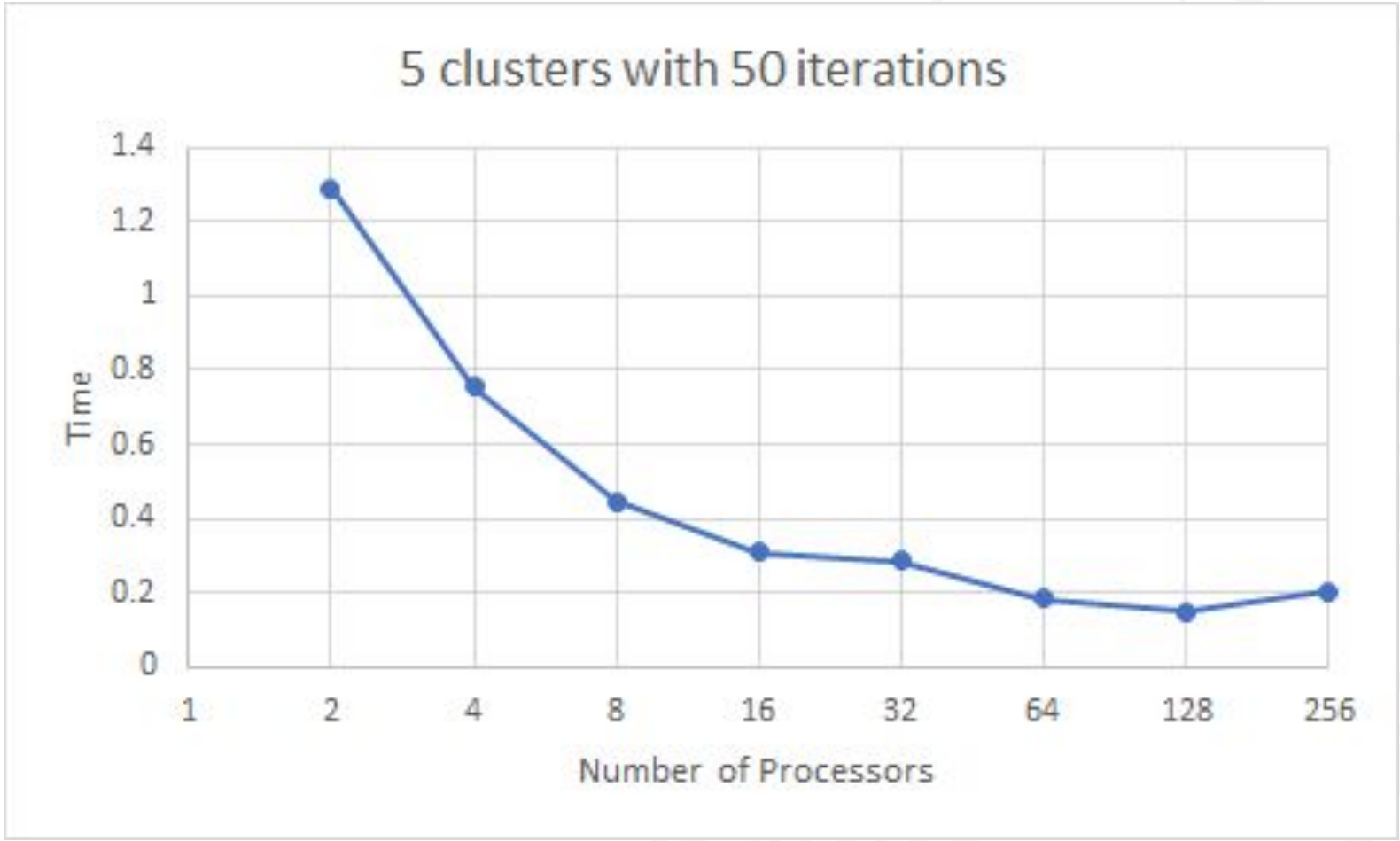
128 * 128 Sized Image

PROCESSORS	TIME
2	2.244983
4	1.156922
8	0.647604
16	0.389776
32	0.247434
64	0.180877
128	0.175467
256	0.243613



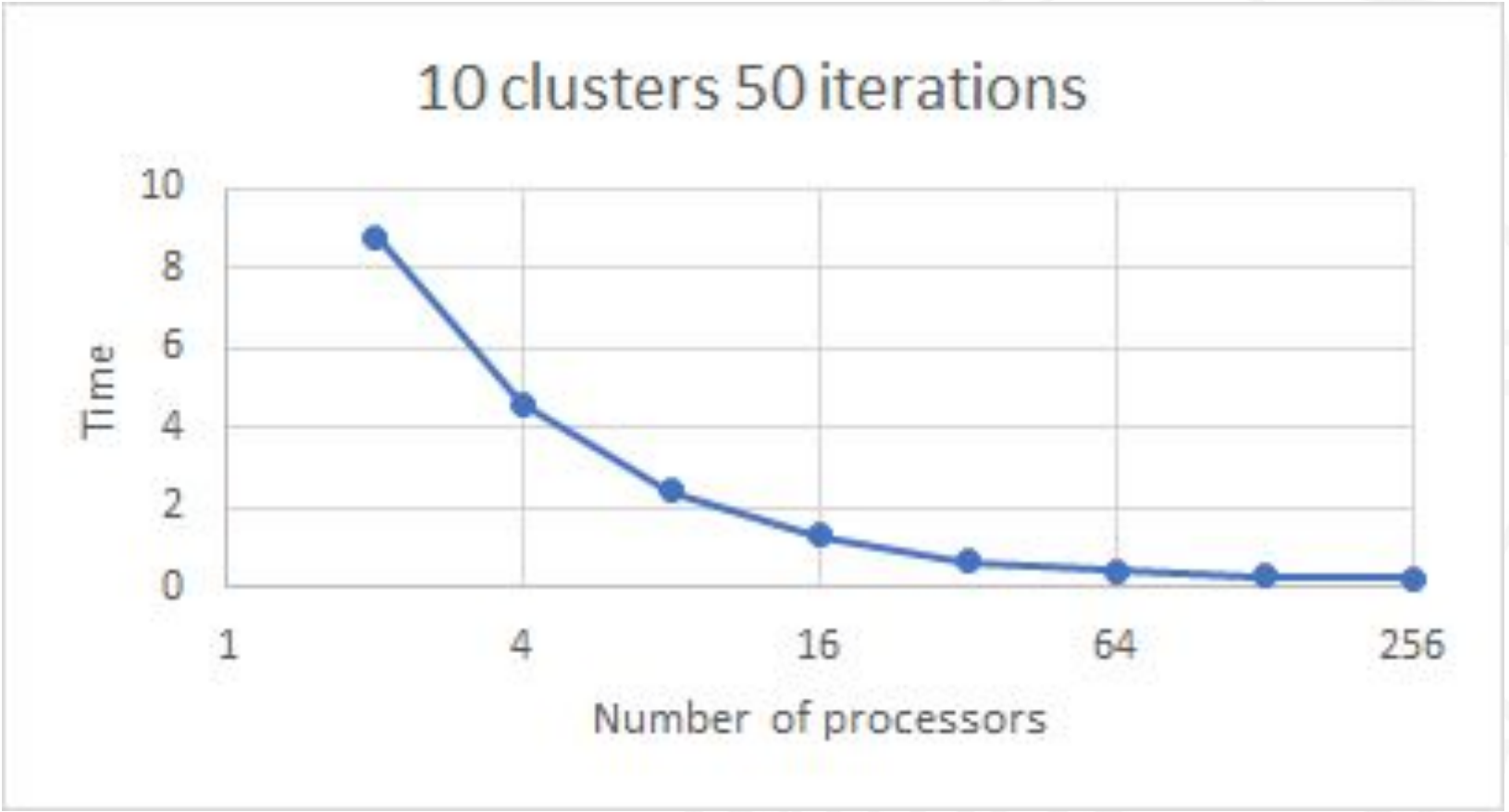
128 * 128 Sized Image

PROCESSORS	TIME
2	1.28728
4	0.75517
8	0.446025
16	0.309565
32	0.285484
64	0.183389
128	0.149291
256	0.204485



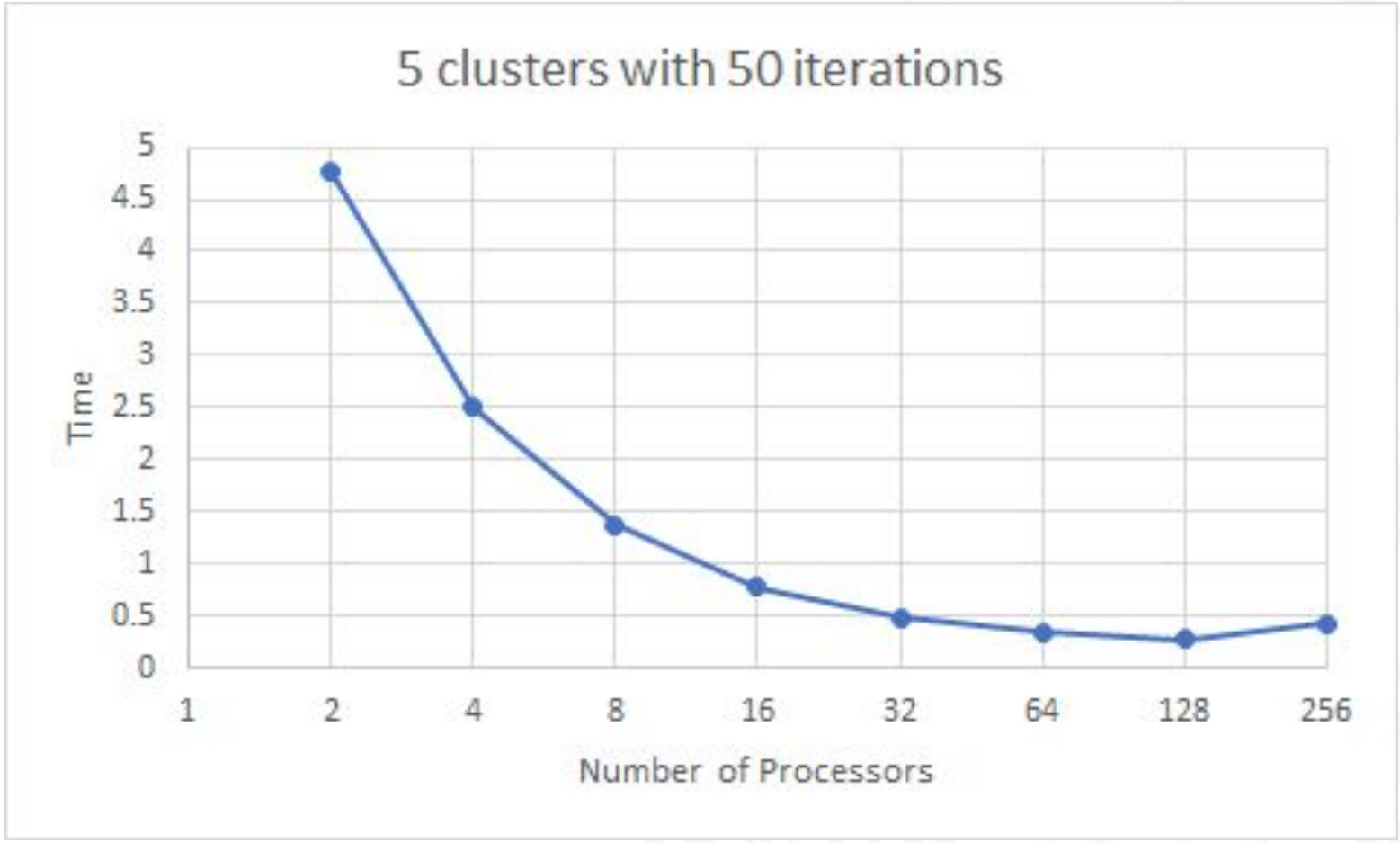
256 * 256 Sized Image

PROCESSORS	TIME
2	8.77687
4	4.61229
8	2.444103
16	1.307961
32	0.685142
64	0.416719
128	0.301695
256	0.261068



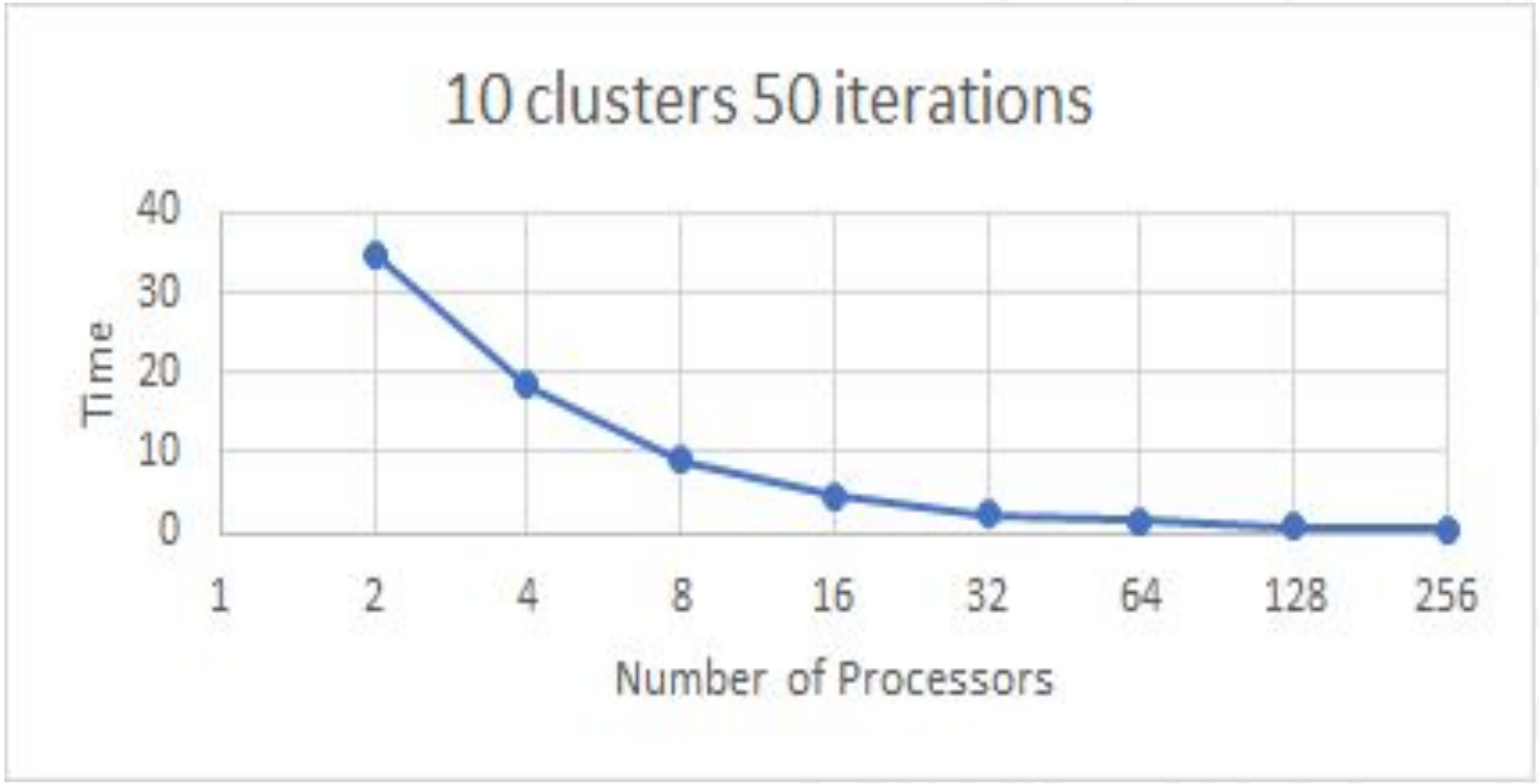
256 * 256 Sized Image

PROCESSORS	TIME
2	4.760425
4	2.495988
8	1.376339
16	0.774094
32	0.485675
64	0.342563
128	0.272038
256	0.419926



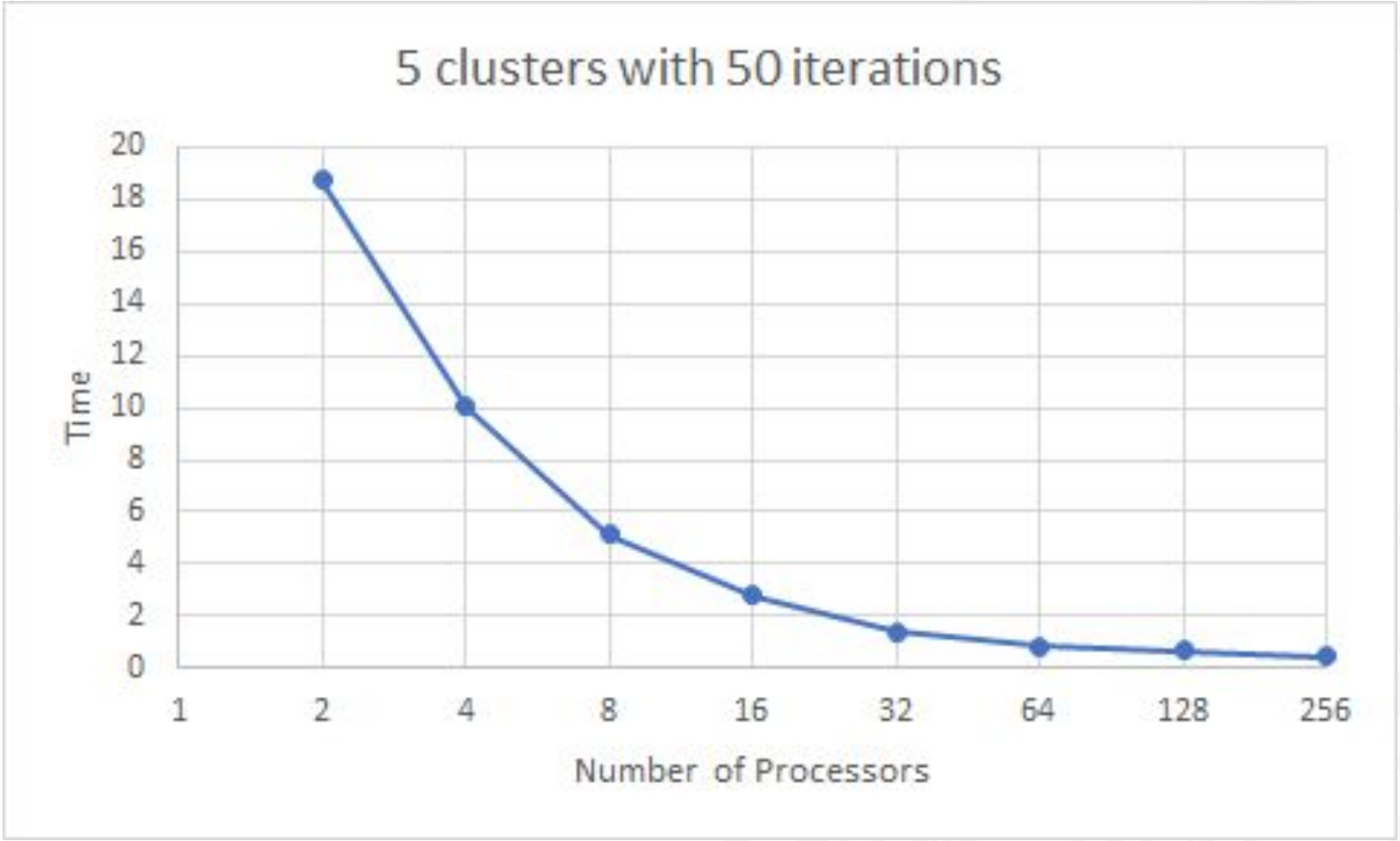
512 * 512 Sized Image

PROCESSORS	TIME
2	34.77792
4	18.31805
8	8.942027
16	4.609761
32	2.399916
64	1.411803
128	0.781993
256	0.523263



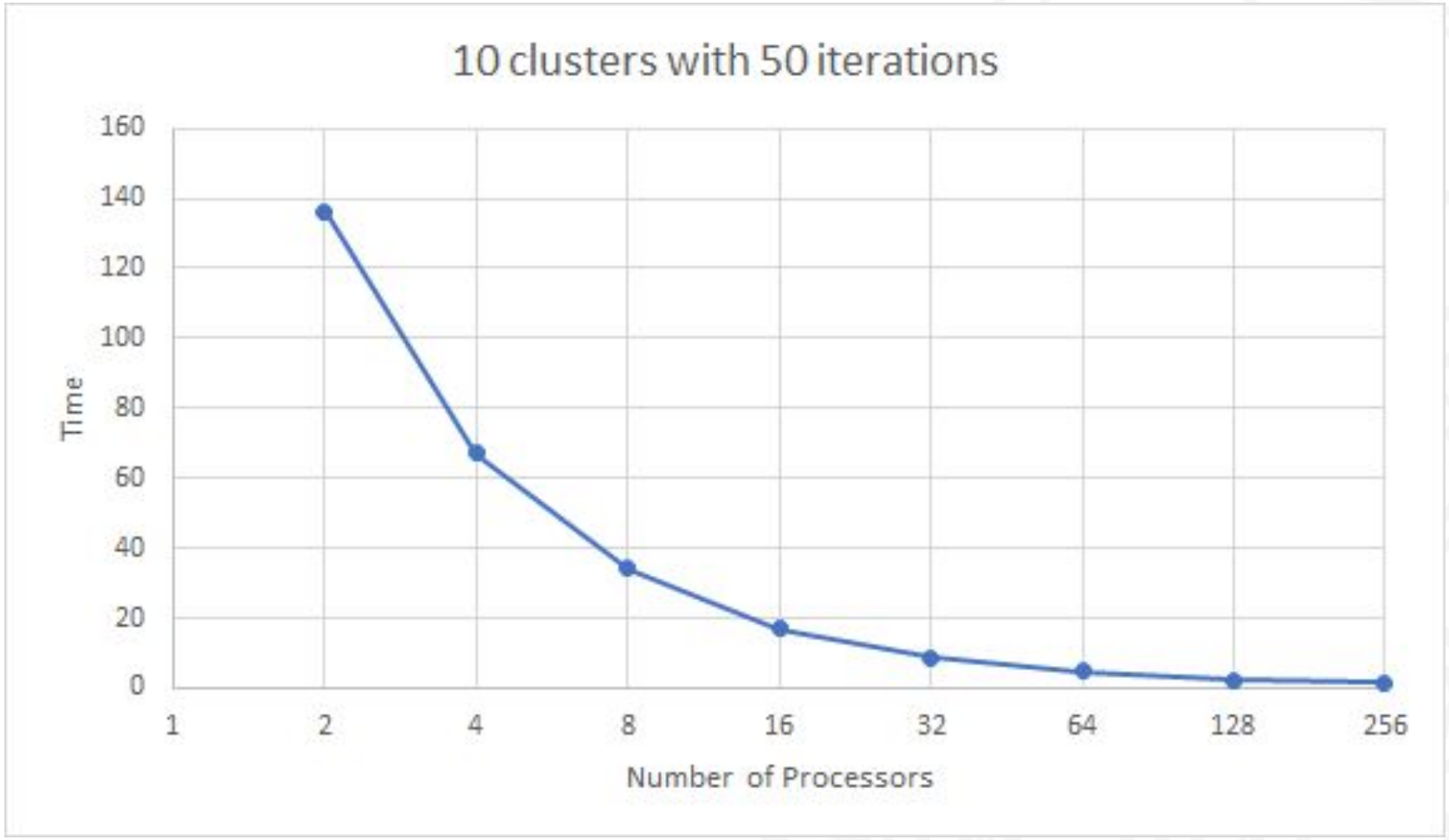
512 * 512 Sized Image

PROCESSORS	TIME
2	18.73687
4	10.01309
8	5.117878
16	2.779889
32	1.421484
64	0.860769
128	0.645885
256	0.438779



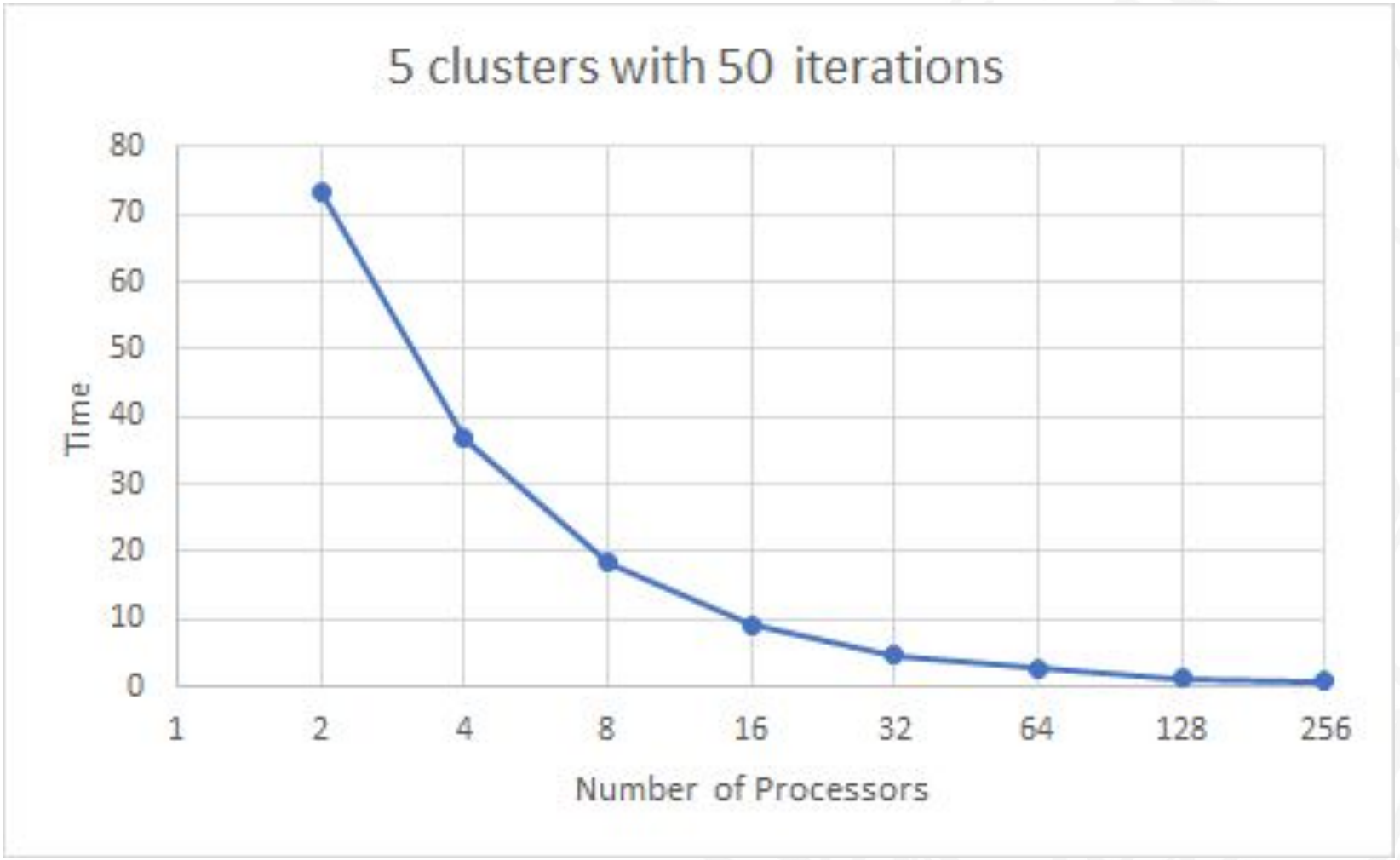
1024 * 1024 Sized Image

PROCESSORS	TIME
2	136.2366
4	67.17114
8	34.04873
16	16.96487
32	8.771979
64	4.697546
128	2.43751
256	1.401541



1024 * 1024 Sized Image

PROCESSORS	TIME
2	73.33249
4	36.74979
8	18.44813
16	9.190316
32	4.789679
64	2.758551
128	1.327619
256	0.868226



Observations

- For the images with the size of 128*128 pixels and 256*256 pixels, the time decreases consistently but then increases as we increase the number of processors beyond 128.

- For the images with the size of 512*512 pixels and 1024*1024 pixels, the time taken consistently decreases till 256 processors.

References

- Algorithms Sequential & Parallel: A Unified Approach (Dr. Russ Miller, Dr.Laurence Boxer).
- https://benalexkeen.com/k-means-clustering-in-python/
- https://mpi4py.readthedocs.io/en/stable/tutorial.html
- https://ubccr.freshdesk.com/support/home
- http://pubs.sciepub.com/jcsa/6/1/4/index.html

Thank You