Image Compression using K-Means clustering OpenMP

CSE 702 - Seminar

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
- OpenMP
- Parallel Algorithm
- Results
- Observations
- References

Problem statement

Using OpenMP for Parallel Implementation of Image Compression using K-Means clustering.

Image Compression

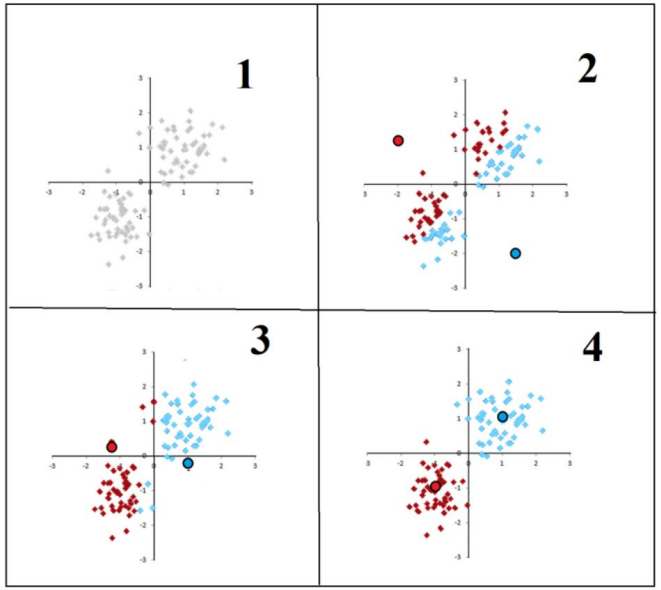
Image Compression is a type of data compression applied to digital images, to reduce their cost for storage or transmission.

Applications

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

K-Means Clustering Algorithm

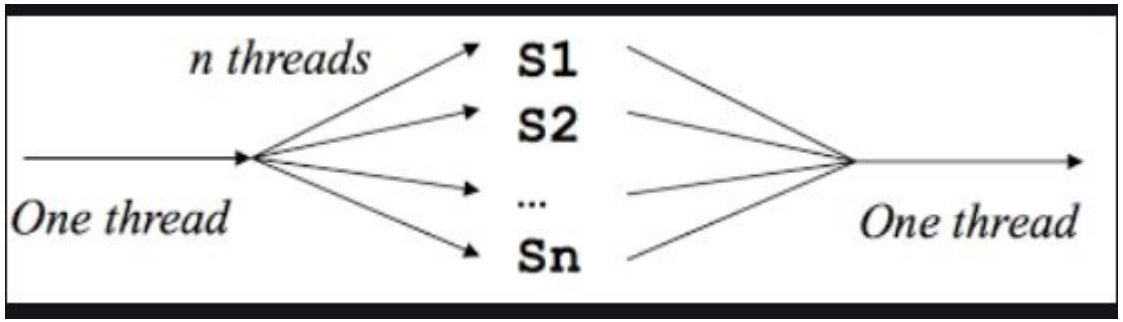
- K-means clustering is the optimization technique to find the ‘k’ clusters or groups in the given set of data points.
- Initially, select ‘k’ data points to be the cluster centers.
- Assignment step - Assign each data point to the closest cluster centers.
- Update step - Calculate the new cluster centers 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.

OpenMP



- Open Multi-Processing
- Parallel programming model using Shared memory
- One thread that runs from beginning to end - Master Thread
- Additional threads fork from the master thread and then join after the parallel implementation - Slave Threads

Parallel Algorithm

- Convert Image to pixels with RGB values in a text file.
- Consider P pixels distributed among N cores.
- Each core is assigned P/N pixel values from the text file.
- ‘k’ pixels are randomly selected as the cluster centers and assigned to shared memory space.
- Each core identifies the clusters all it’s pixels belongs to.
- The new global cluster centers are found by taking mean of all the local sums.
- Repeat the clustering for the specified number of iterations.
- Store information about each point’s final cluster center in a text file using the cluster centers of the final iteration.
- Convert pixels in text file back to the Image with reduced colors.

Results

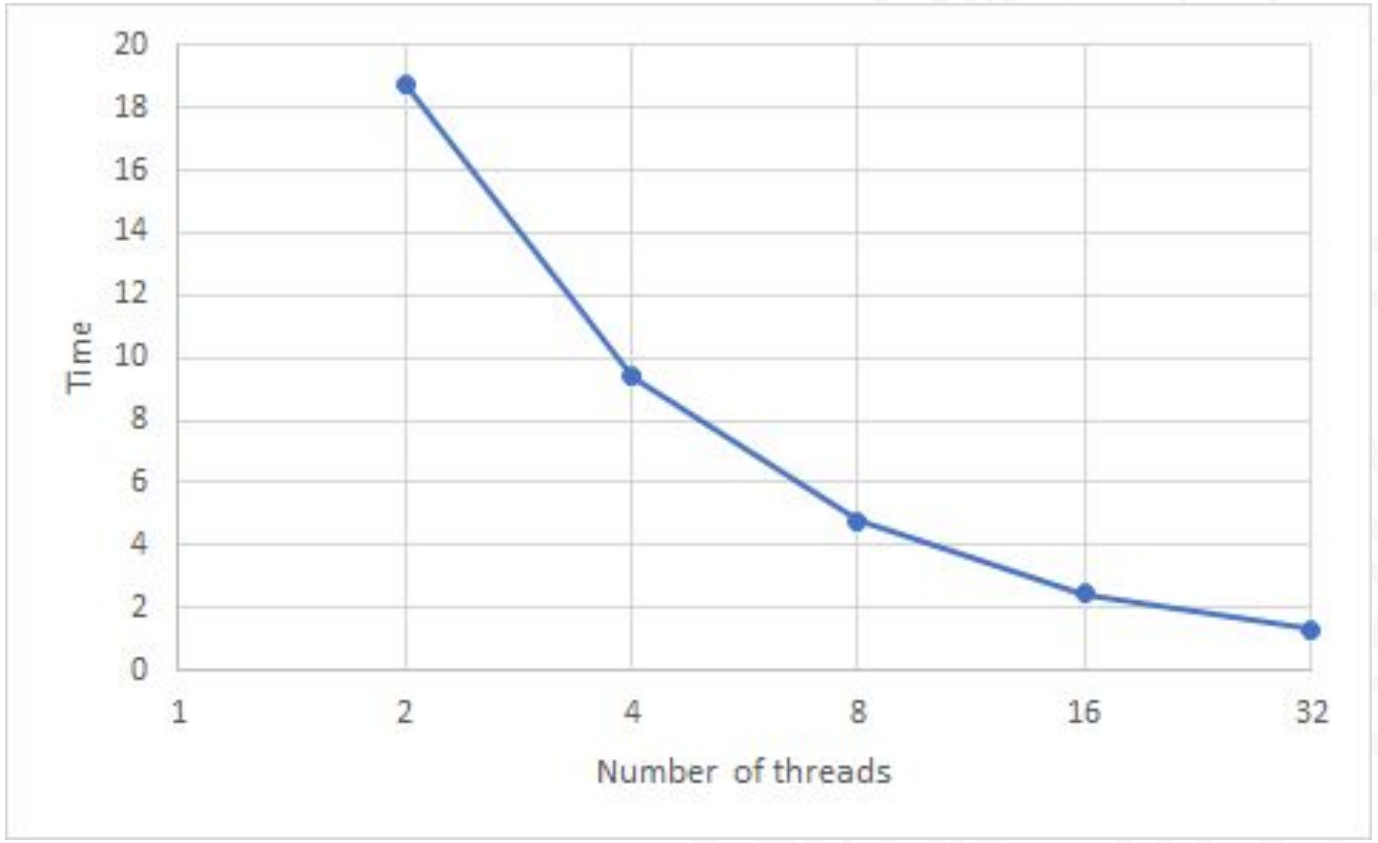
Original Image

Compressed Image - 5 Clusters

Compressed Image - 10 Clusters

Time Analysis

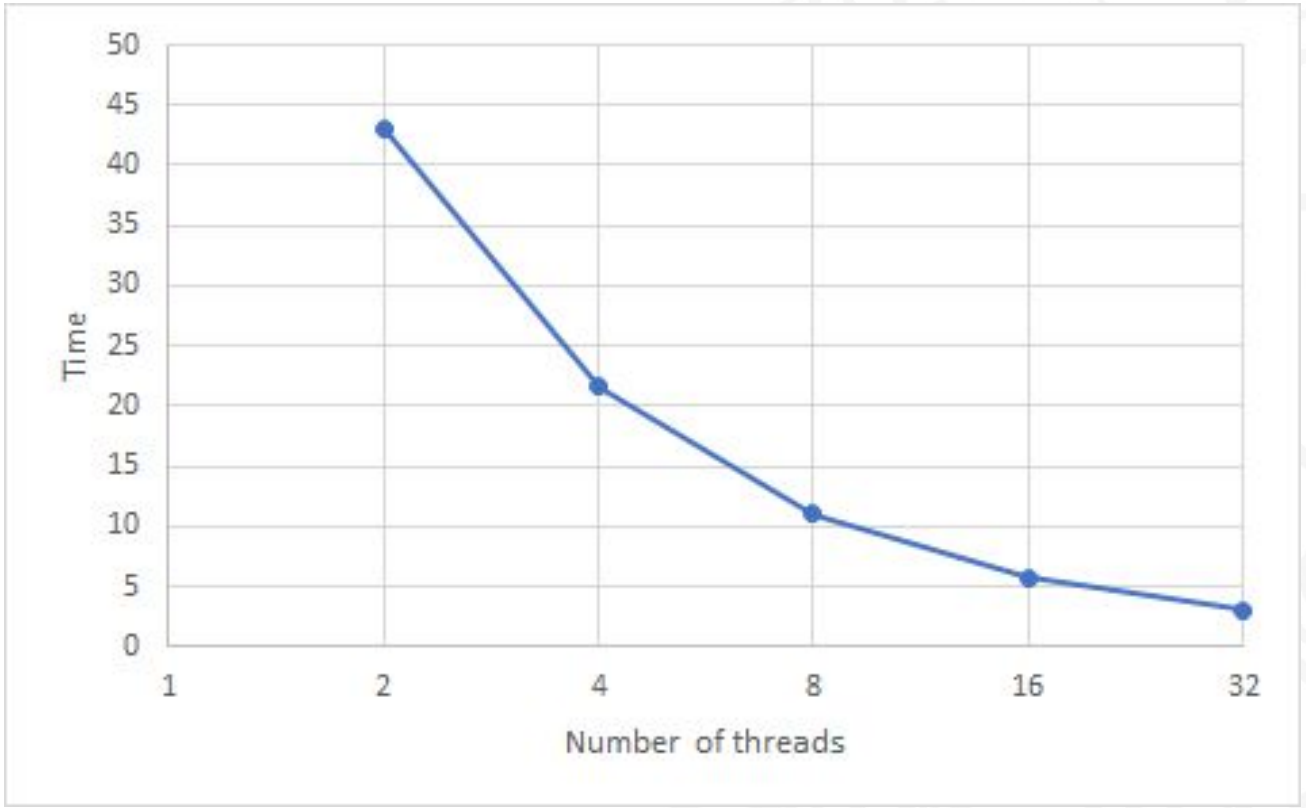
Number of Cores	Time (s)
2	18.77
4	9.38
8	4.78
16	2.47
32	1.29



Time Analysis

Number of Cores	Time (s)
2	43.07
4	21.58
8	11.01
16	5.71
32	2.98

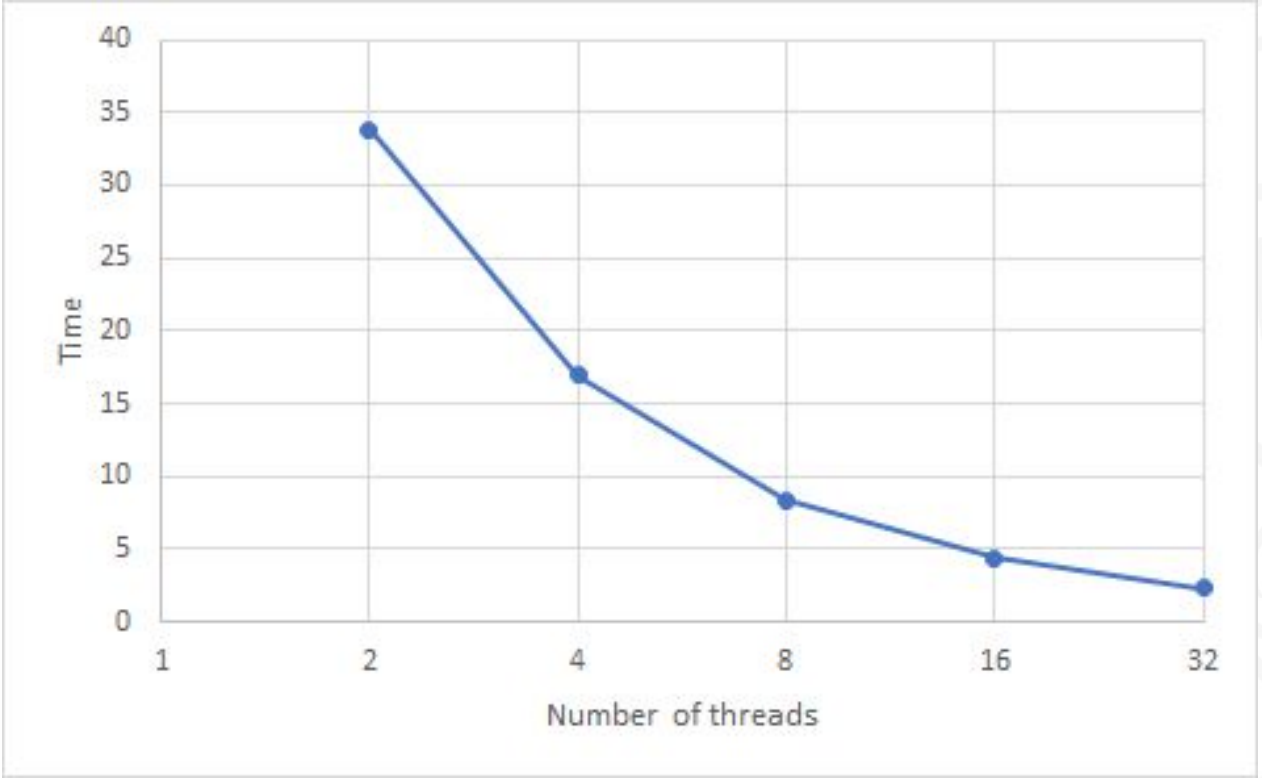
5 Clusters and 50 Iterations



Time Analysis

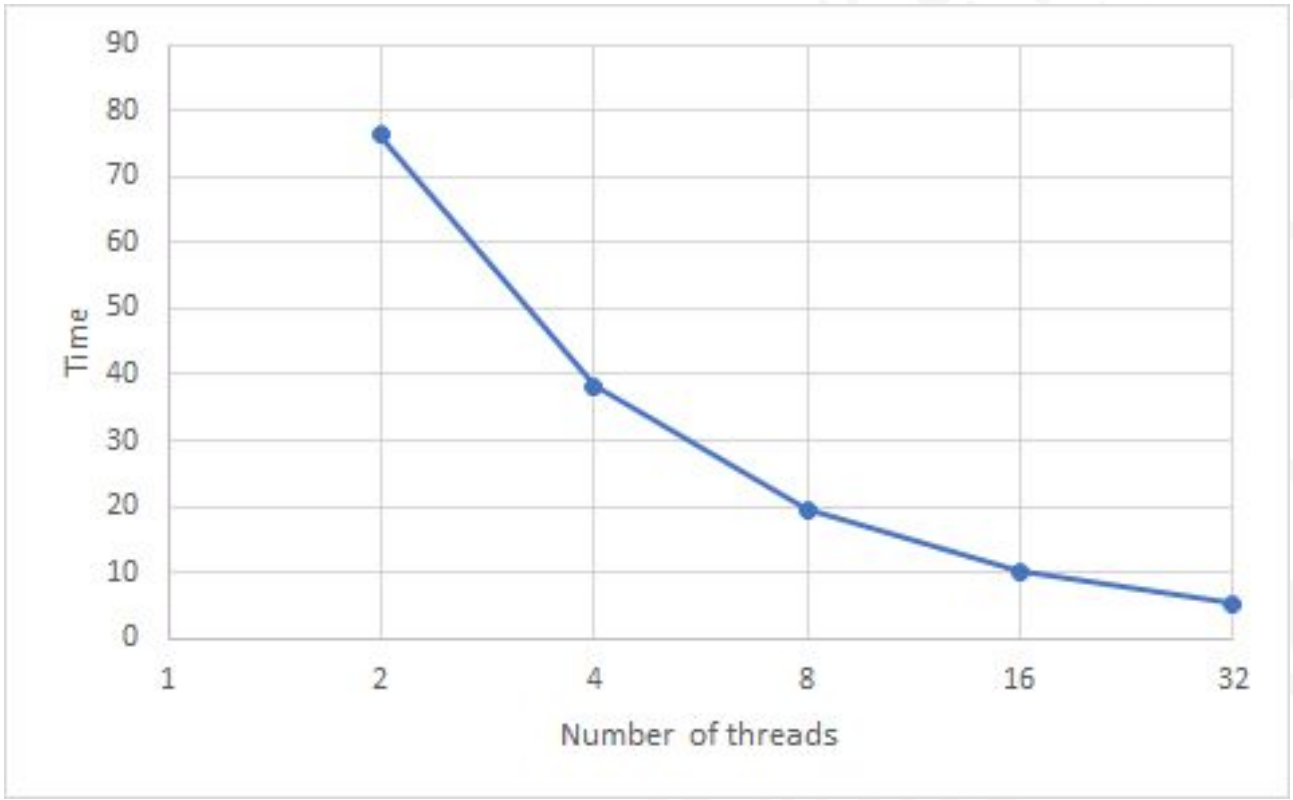
Number of Cores	Time (s)
2	33.87
4	16.95
8	8.364
16	4.4
32	2.29

10 Clusters and 20 Iterations



Time Analysis

Number of Cores	Time (s)
2	76.34
4	38.17
8	19.58
16	10.17
32	5.34



Observations

- Significant speedup observed as the cores were increased upto 32.
- The performance gets lower with 64 or more cores due to implicit context-switching and increased overheads.

References

- Algorithms Sequential & Parallel: A Unified Approach (Dr. Russ Miller, Dr.Laurence Boxer)
- https://en.wikipedia.org/wiki/OpenMP
- https://pages.tacc.utexas.edu/~eijkhout/pcse/html/omp-basics.html
- https://towardsdatascience.com/image-compression-using-k-means-clustering-aa0c91bb0eeb

Thank You