Image Compression using k-Means Clustering (OpenMP)

CSE702 Fall ‘19

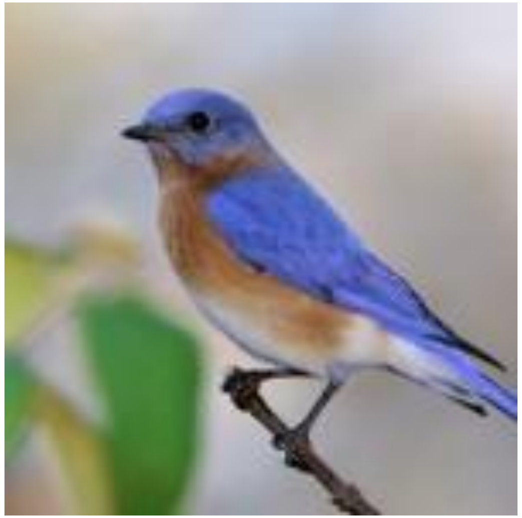
Instructor- Dr. Russ Miller

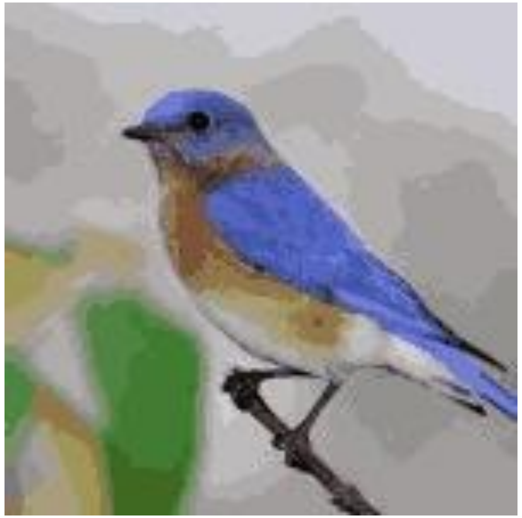
By Utkarsh Bansal

Outline

Problem Definition

Compressing an Image using k-Means Clustering.


11783 unique colors


40 unique colors

k-Means for Clustering

1. Randomly select k points to be cluster centers.
2. For each point in the data set, put it in the cluster which has its center closest to the point.
3. Calculate new cluster centers by taking means of all points in a cluster.

   Repeat 2 and 3 until convergence or exit condition reached.

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 string one by one.
- Saving the string to a .txt file.

Parallel k-Means

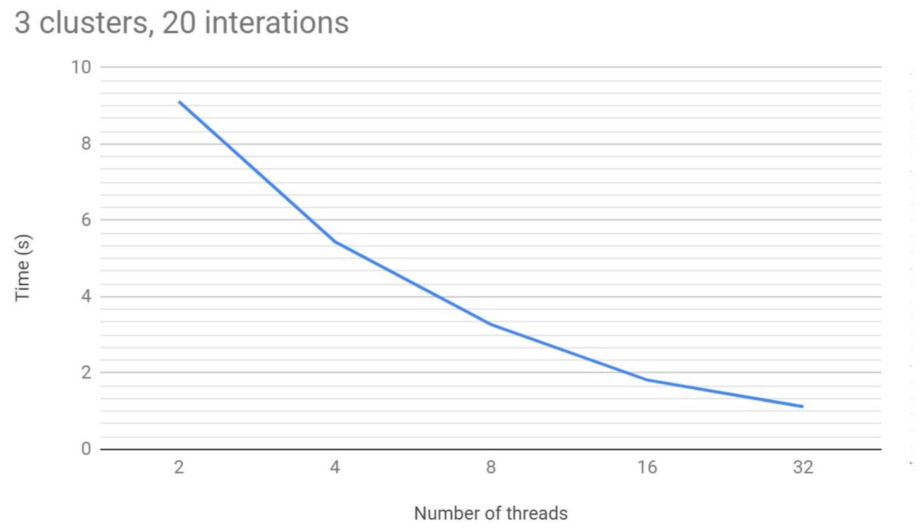
1. Consider N data points and P cores.
2. Assign N/P data points to each core using the .txt file.
3. Core 0 randomly elects k points as cluster centers.
4. Each core for each of its points, finds the cluster to which the point belongs.
5. Recalculate local sums for each cluster in each core.
6. Add all local sums for each core to find the global means.
7. Repeat the clustering for number of iterations. (loop back to step 4)
8. Save the cluster centers of the final iteration.
9. Form a file with information about each point’s final cluster center.

Compressed Image Formation (Serial)

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

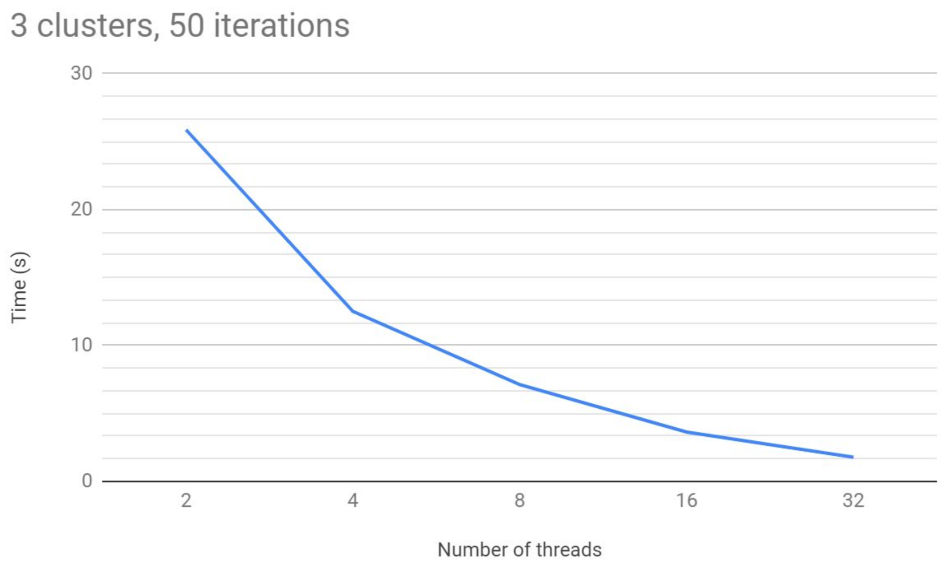
Results

Time Analysis for 3 Clusters



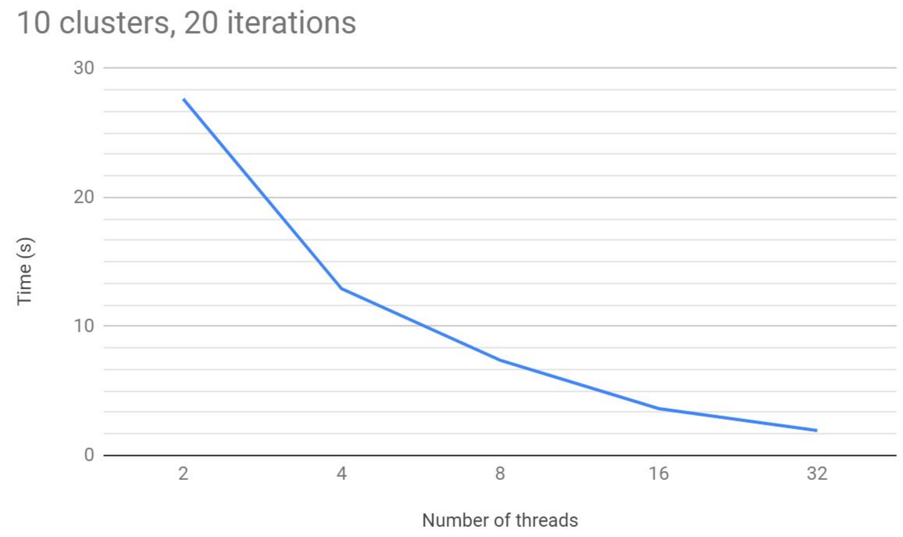
Number of threads	Time in seconds
2	9.12
4	5.44
8	3.27
16	1.82
32	1.12

Time Analysis for 3 Clusters



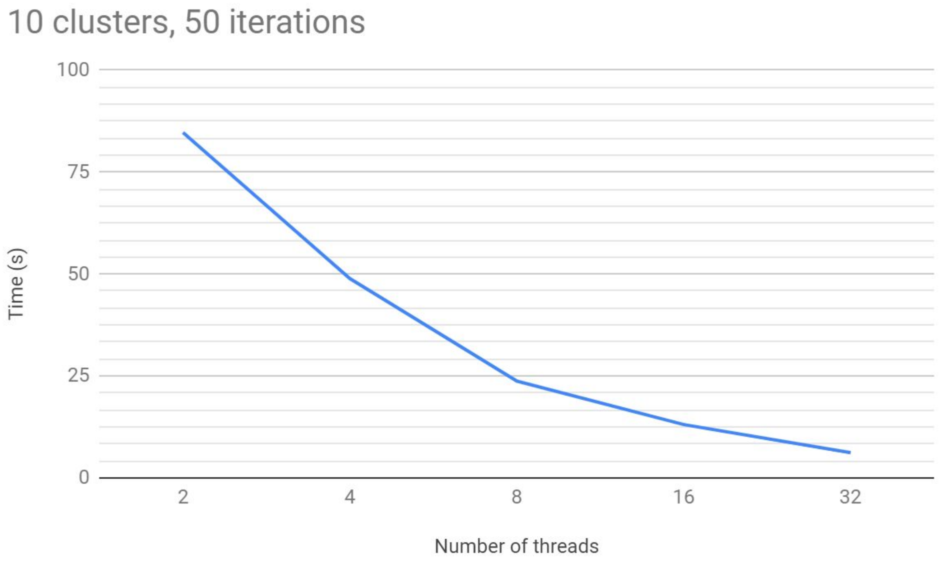
Number of threads	Time in seconds
2	25.88
4	12.52
8	7.13
16	3.63
32	1.78

Time Analysis for 10 Clusters



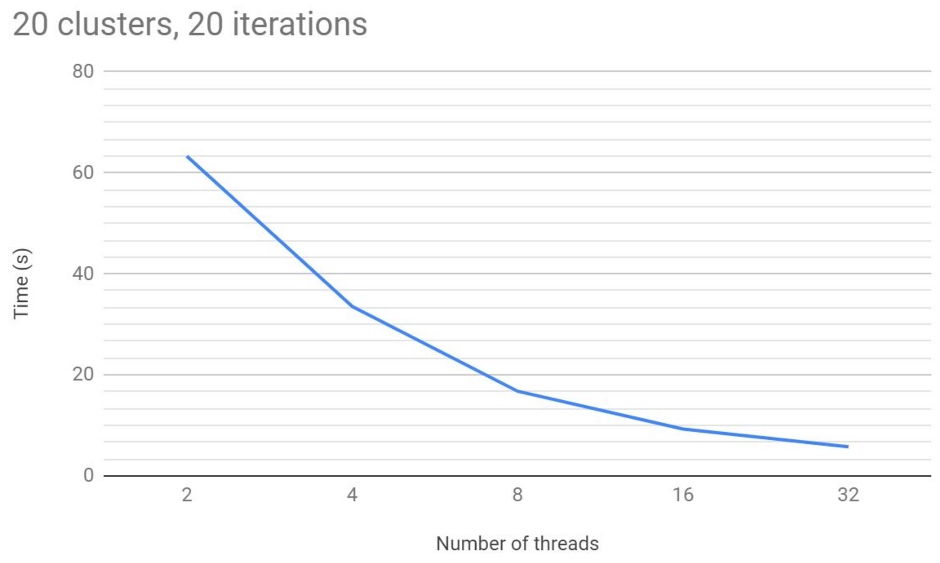
Number of threads	Time in seconds
2	27.64
4	12.92
8	7.38
16	3.63
32	1.93

Time Analysis for 10 Clusters



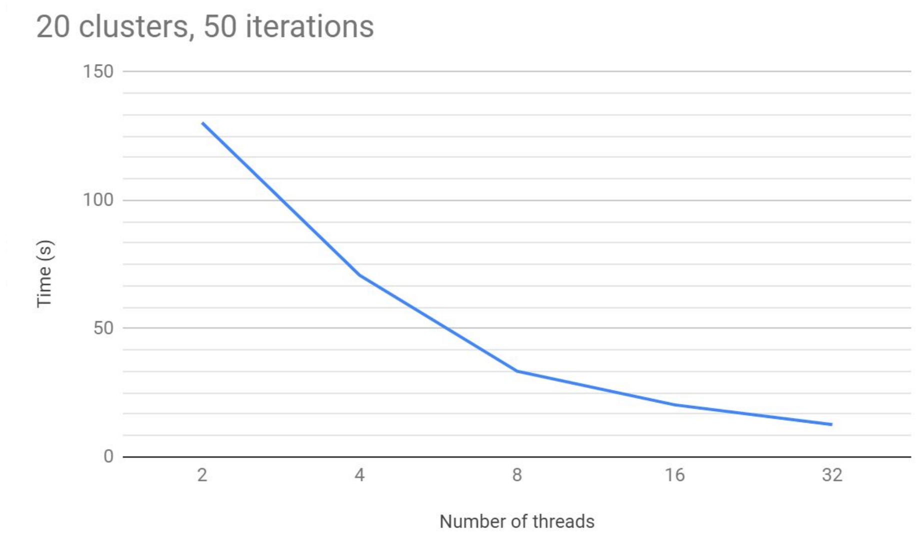
Number of threads	Time in seconds
2	84.74
4	48.9
8	23.75
16	13.08
32	6.21

Time Analysis for 20 Clusters



Number of threads	Time in seconds
2	63.34
4	33.56
8	16.76
16	9.29
32	5.76

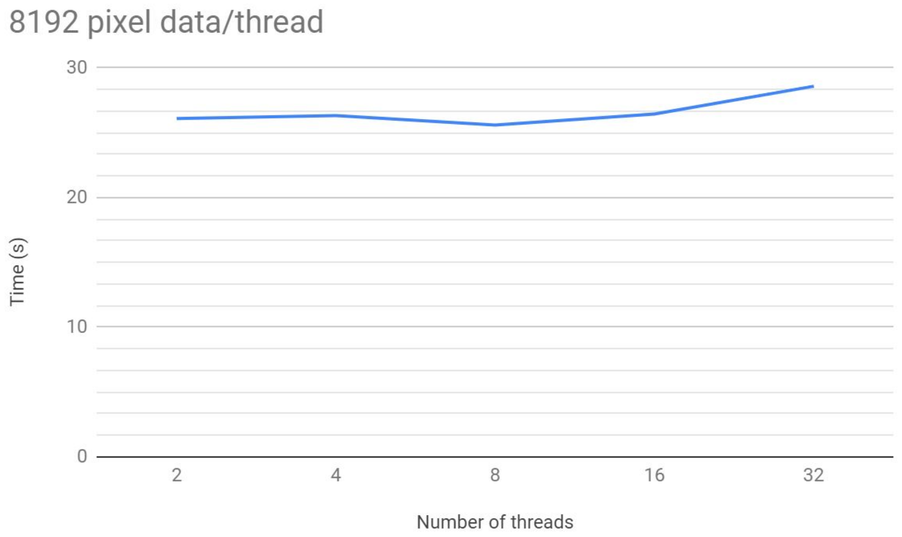
Time Analysis for 20 Clusters



Number of threads	Time in seconds
2	130.24
4	70.84
8	33.41
16	20.32
32	12.63

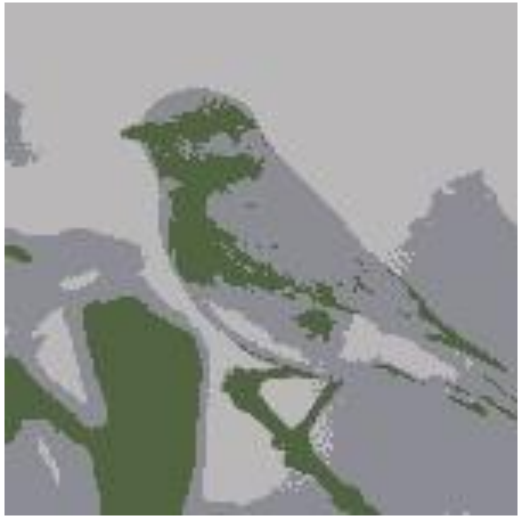
We had varying data/thread till now. Now keeping the data/thread constant at data 256*32 = 8192 pixel data.

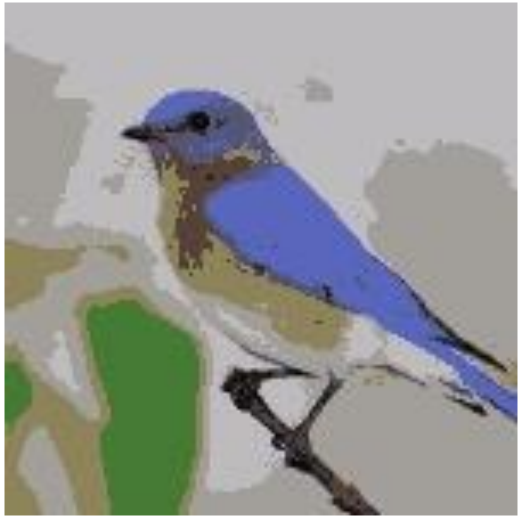
Time Analysis for Constant Data/Processor

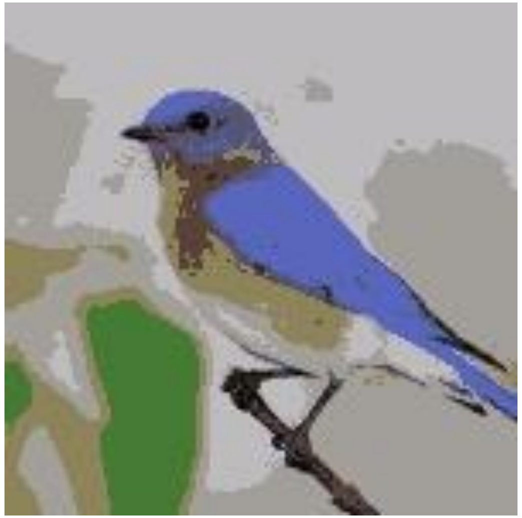


Number of threads	Time in seconds
2	26.09
4	26.32
8	25.6
16	26.45
32	28.59

Output Images


3 colors


10 colors


20 colors

Observations

- Significant speedup is observed upto 32 cores.
- For the input size used, using more than 32 cores may not be practical.

Challenges

- The input .txt files and the output images had to be created serially.
- Ran into insufficient memory errors when the images were too big. (>4000*4000 pixels)

References

- Algorithms Sequential & Parallel: A Unified Approach (Dr. Russ Miller, Dr.Laurence Boxer)
- A “Hands-on” Introduction to OpenMP

Thank You!