Image Segmentation using K-Means Clustering with OpenMP

Team 20

member : 310551154 林子恒、310554047 張方華、310552059曾宇廷

Outline

- Introduction/motivation
- Problem statement
- Proposed solution
- Evaluation
- Conclusion
- Contributions of each member
- Q&A

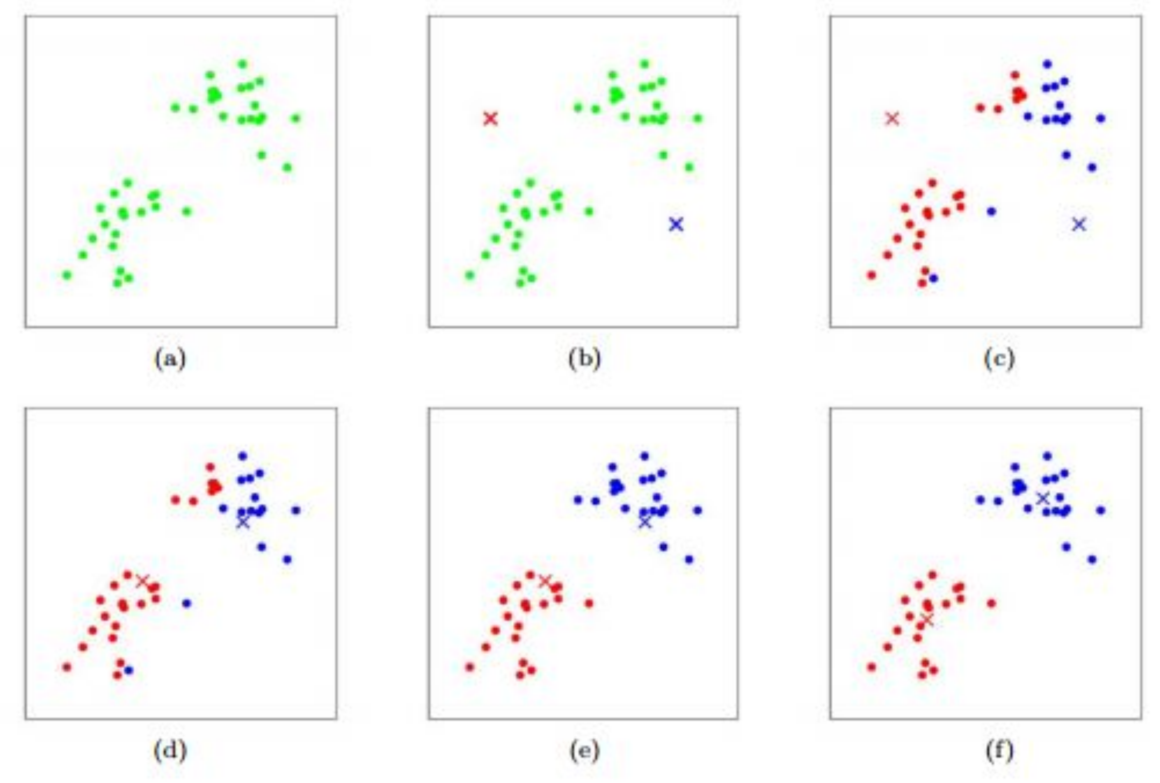
Platform

AMD Ryzen™ Threadripper™ 3960X

- # of CPU Cores: 24
- # of Threads: 48
- Max. Boost Clock: Up to 4.5GHz
- Base Clock: 3.8GHz
- L1 Cache: 1.5MB
- L2 Cache: 12MB
- L3 Cache: 128MB
- Technology: TSMC 7nm FinFET

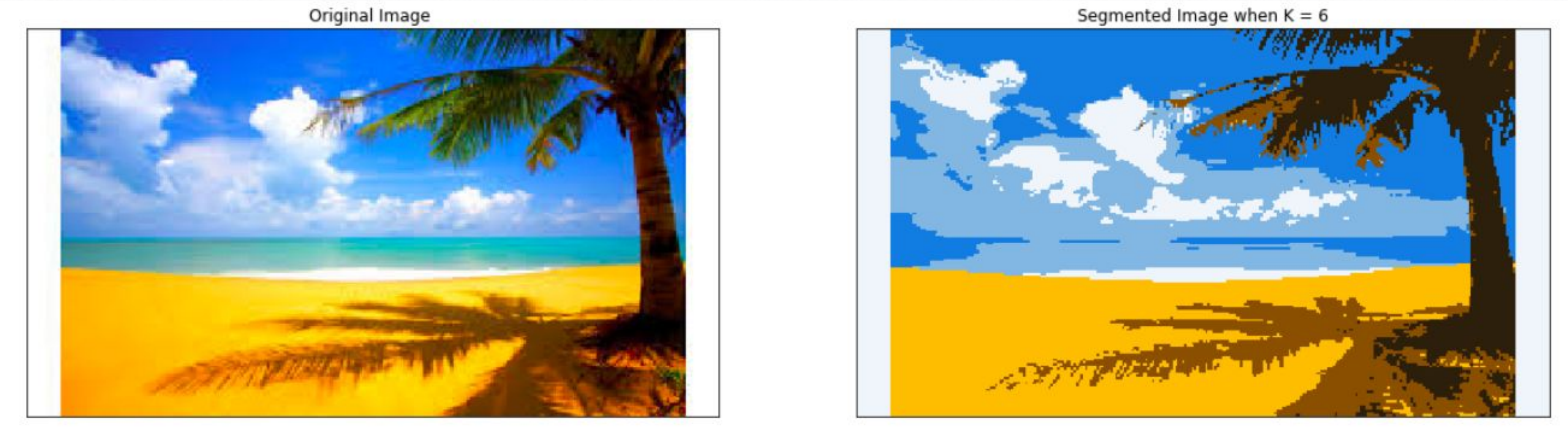
Introduction / motivation

k-means



Introduction / motivation

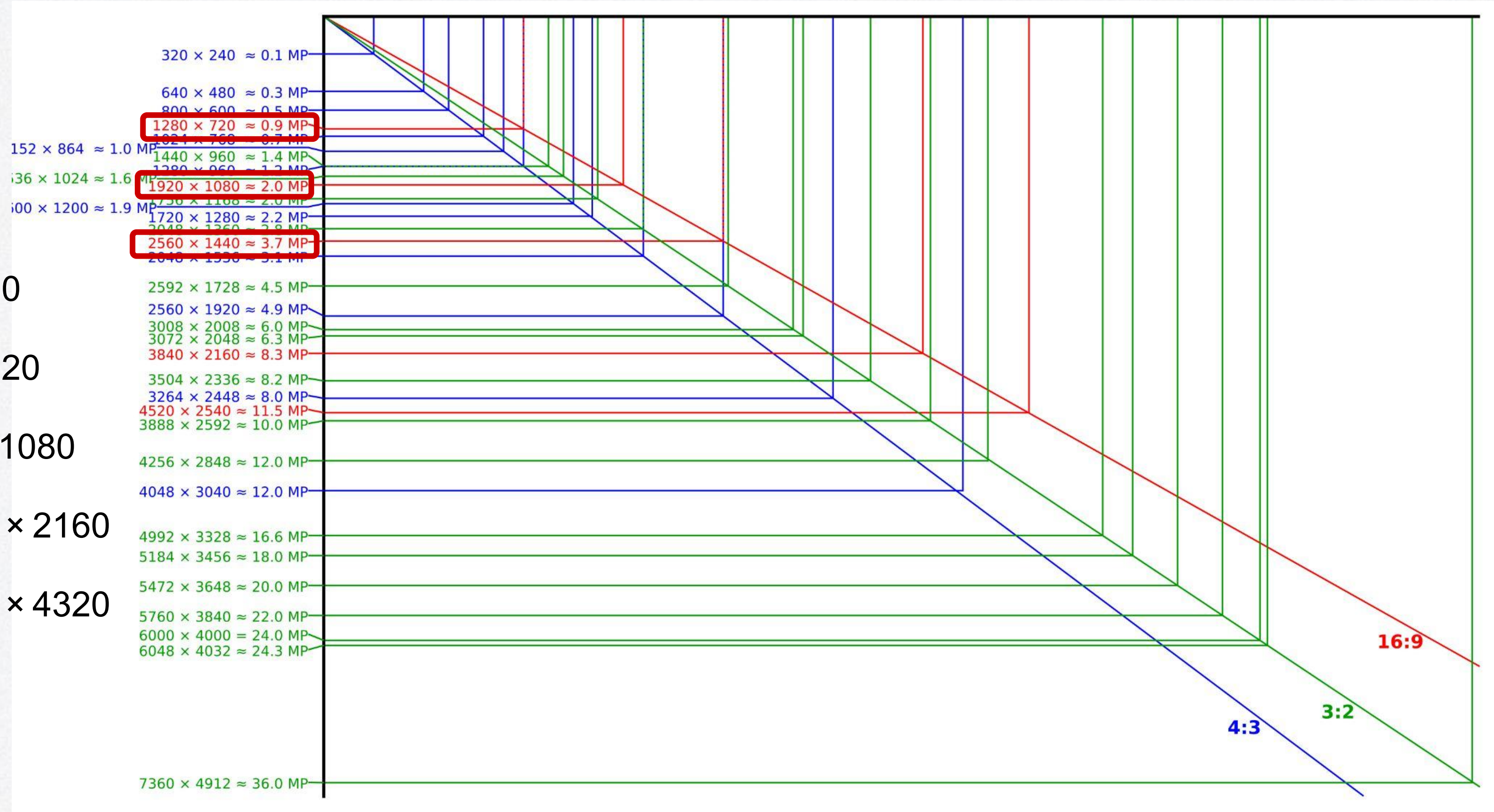
K-means image segmentation / compression



1. findAssociatedCluster()
2. adjustClusterCenters()
3. applyFinalCluster()

Problem statement

A **megapixel** (**MP**) is a million pixels



480p = 720 × 480

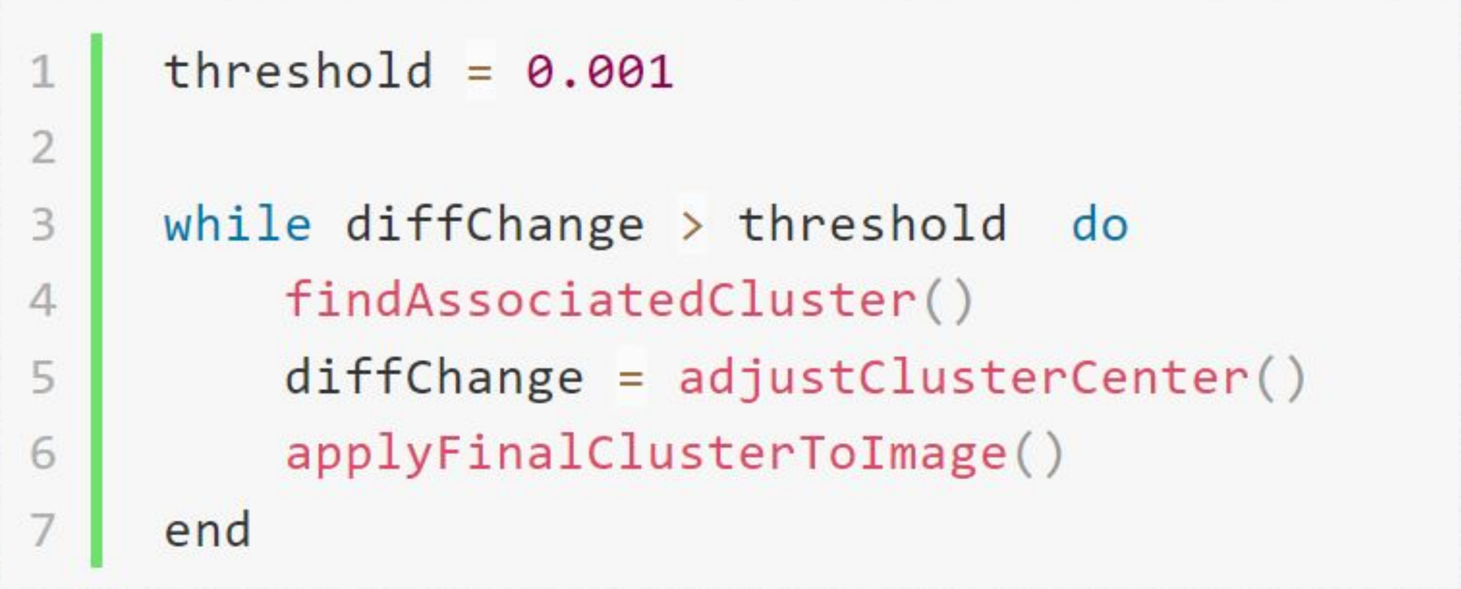
720p = 1280 × 720

1080p = 1920 × 1080

4K UHD = 3840 × 2160

8K UHD = 7680 × 4320

Proposed Solution



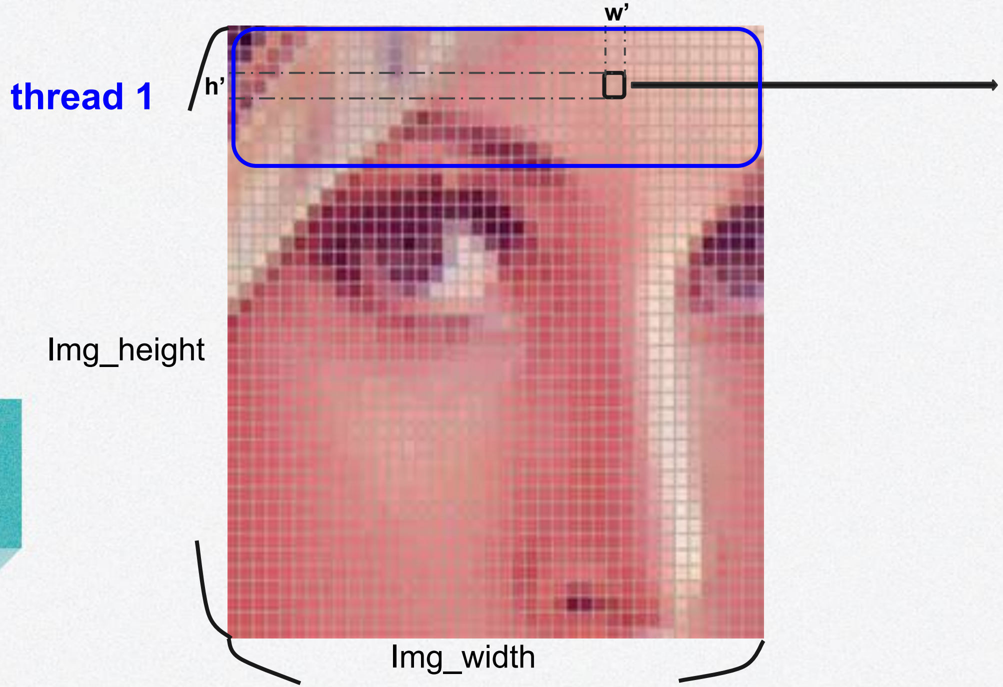
```
threshold = 0.001

while diffChange > threshold  do
    findAssociatedCluster()
    diffChange = adjustClusterCenter()
    applyFinalClusterToImage()
end
```

Proposed Solution

findAssociatedCluster()

1. Assign **Img_height/P** to each processor to find associated cluster for each pixel
2. Store the reallocated cluster number **k’** in a dynamic 2d array of Img_height * Img_width size

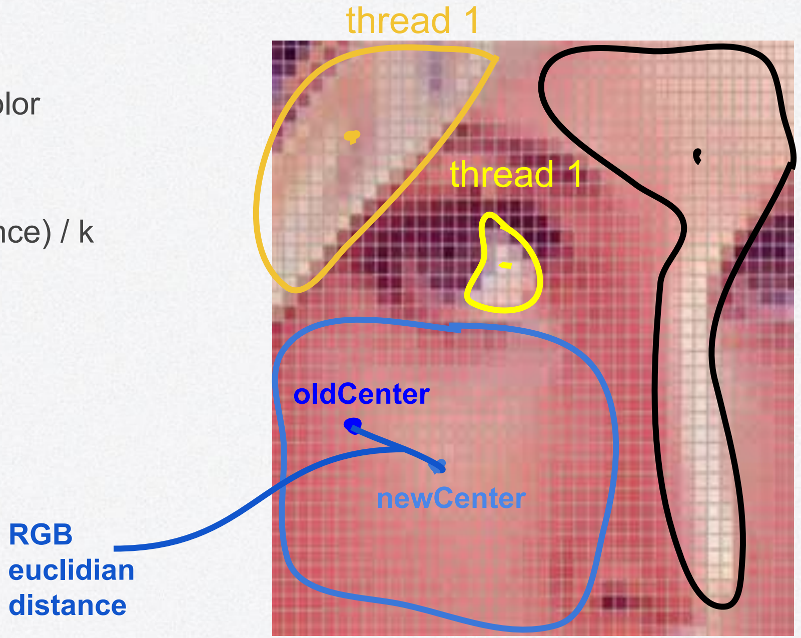


arr[h’][w’] = 2
assign pixel h’ w’ value to cluster 2

Proposed Solution

adjustClusterCenters()

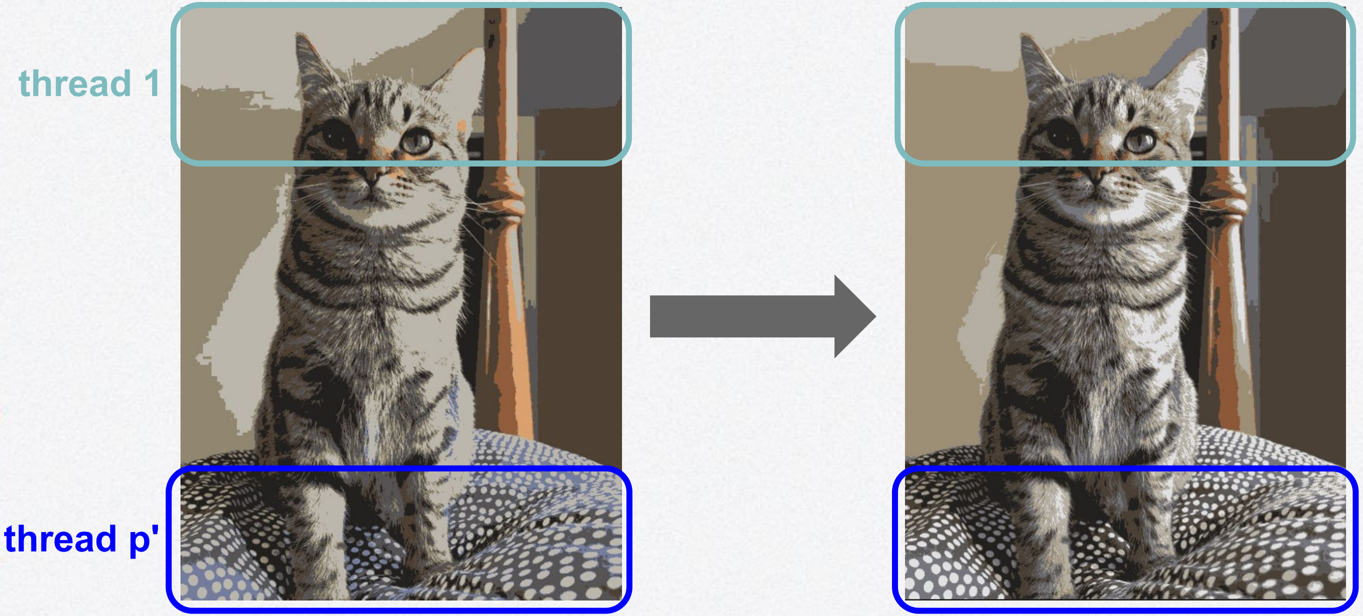
1. Assign **K/P** to each processor to compute the center color euclidian distance
2. Update new center to each cluster
3. meanNewCenter = (sum up newCenter euclidian distance) / k
4. Calculate **diffChange**
   = abs(meanOldCenter - meanNewCenter)



Proposed Solution

applyFinalClusterToImage()

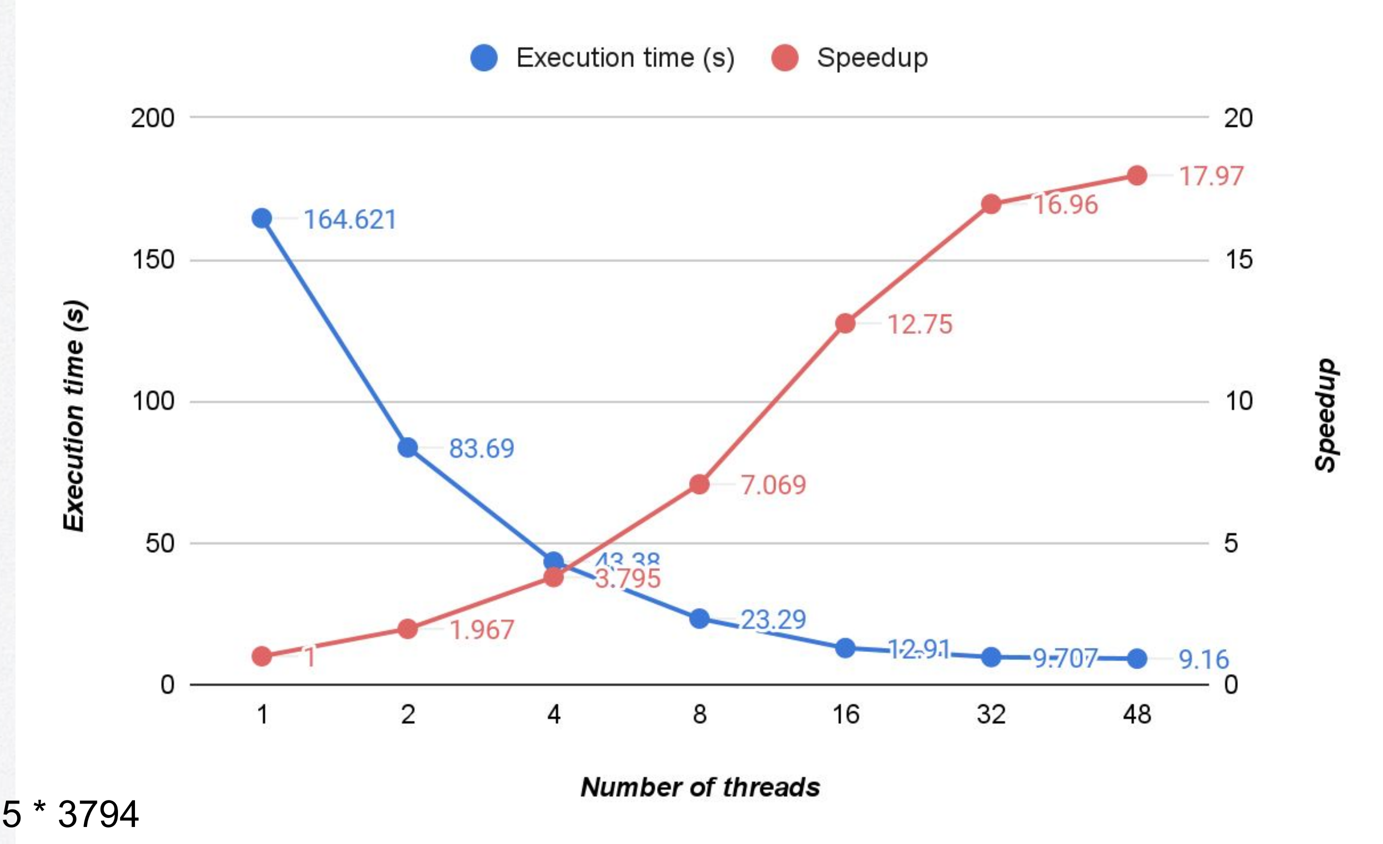
1. Assign **Img_height/P** to each processor to write Image



Evaluation

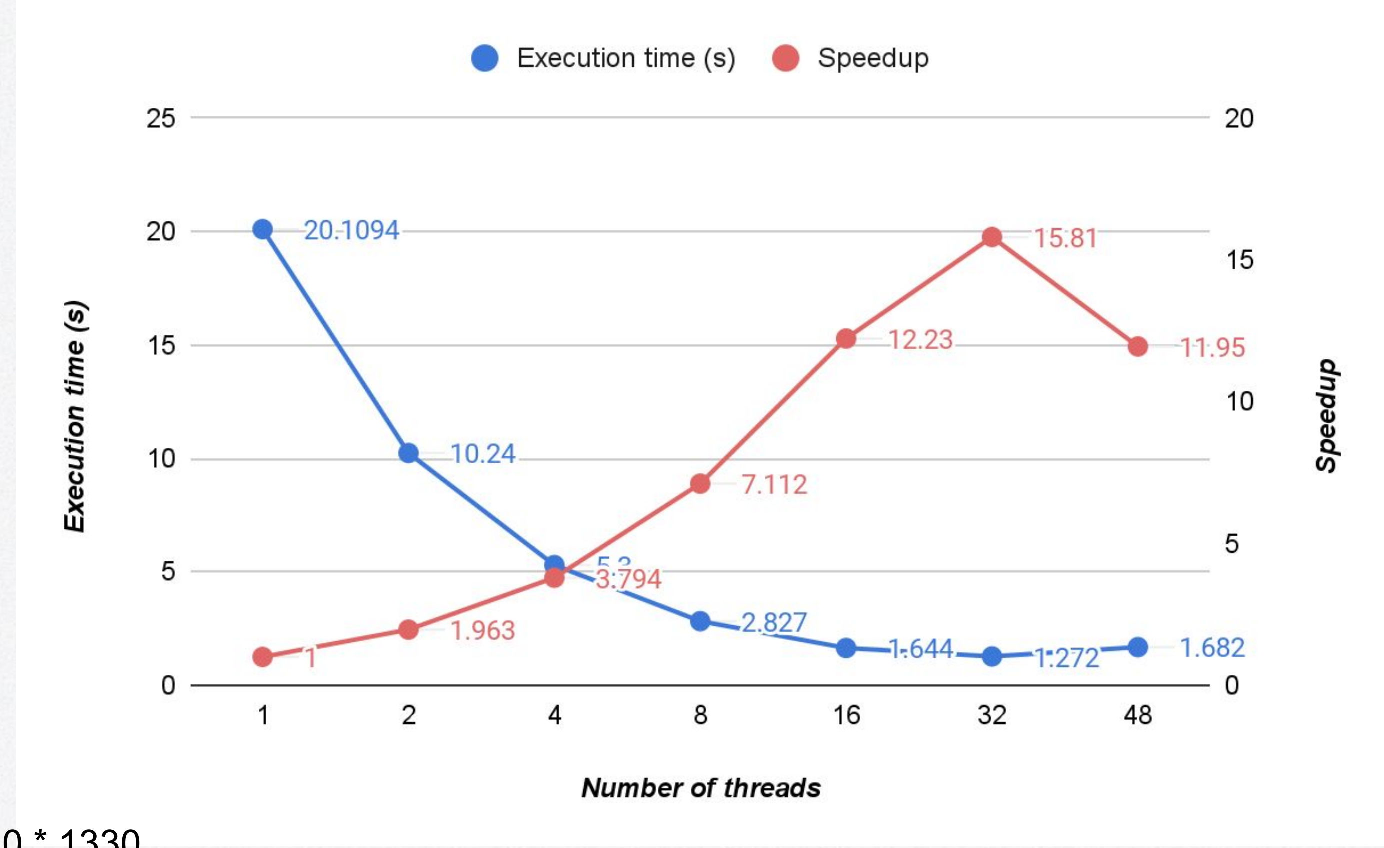
- Computation time decreases as image size decreases.
- Speedup decreases when the number of threads increases to 32.
- Efficiency drops significantly when the number of threads increases to 32.

Evaluation - Original, K = 128



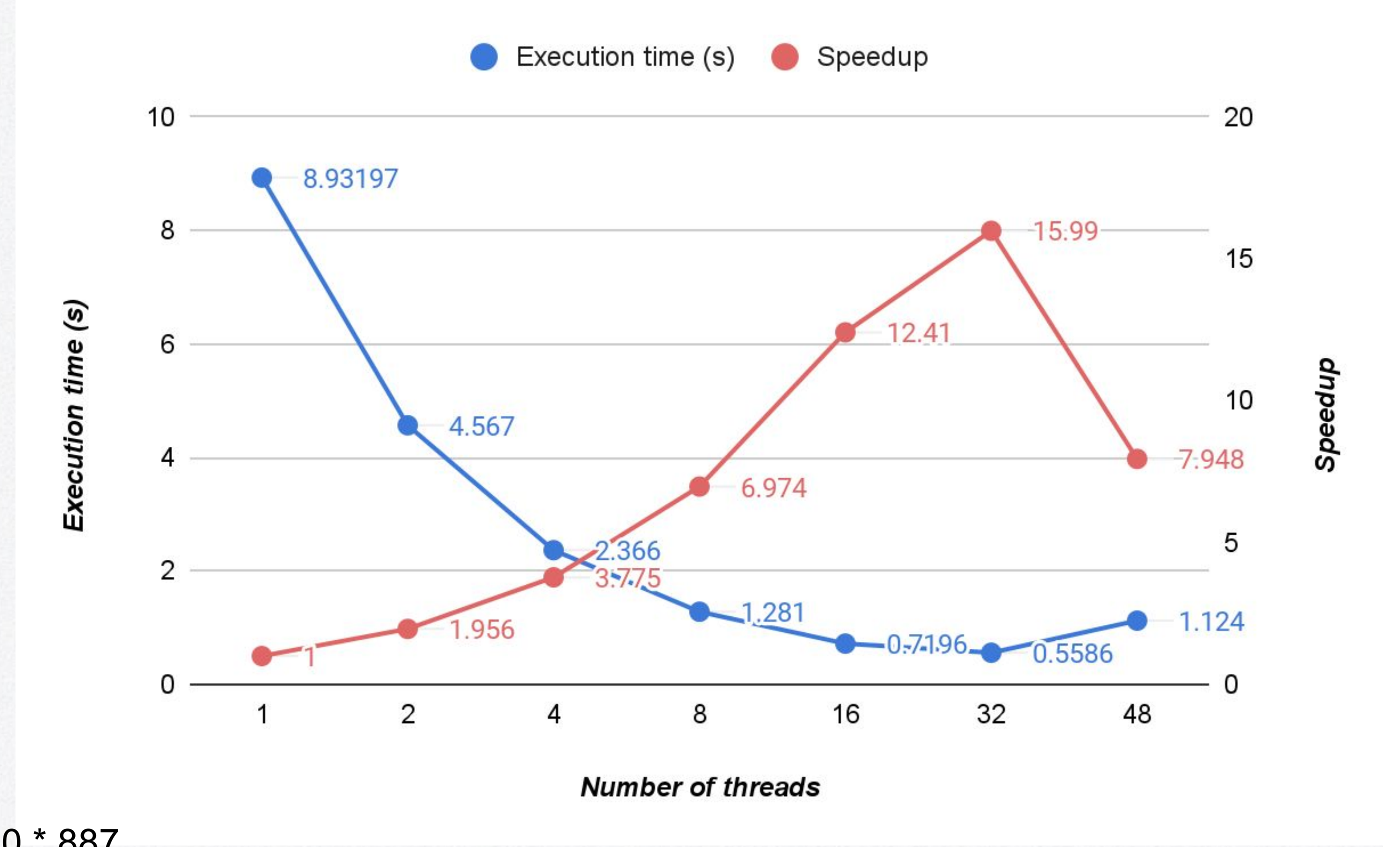
size = 5475 * 3794

Evaluation - Large, K = 128



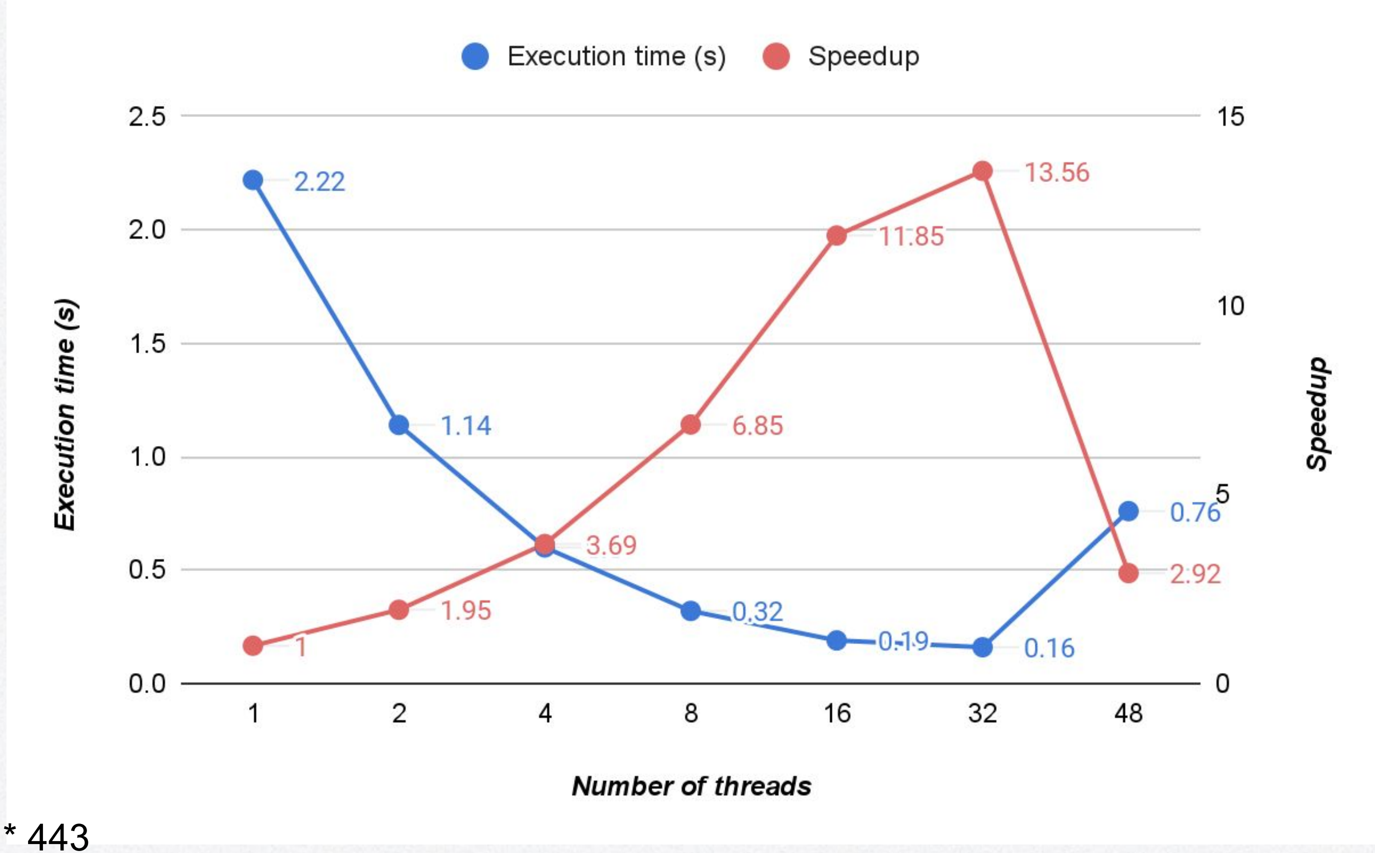
size = 1920 * 1330

Evaluation - Medium, K = 128



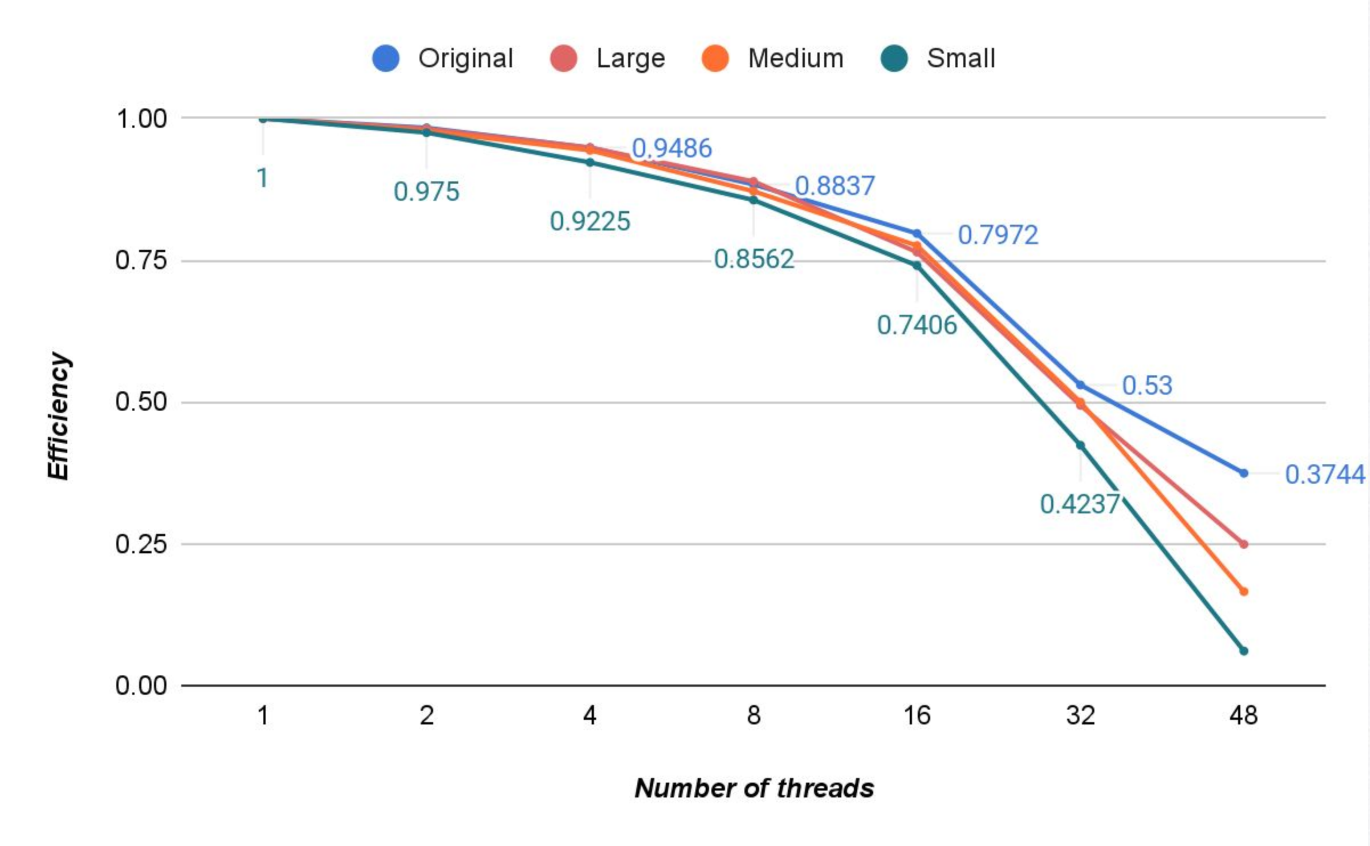
size = 1280 * 887

Evaluation - Small, K = 128



size = 640 * 443

Evaluation - Efficiency



Conclusion

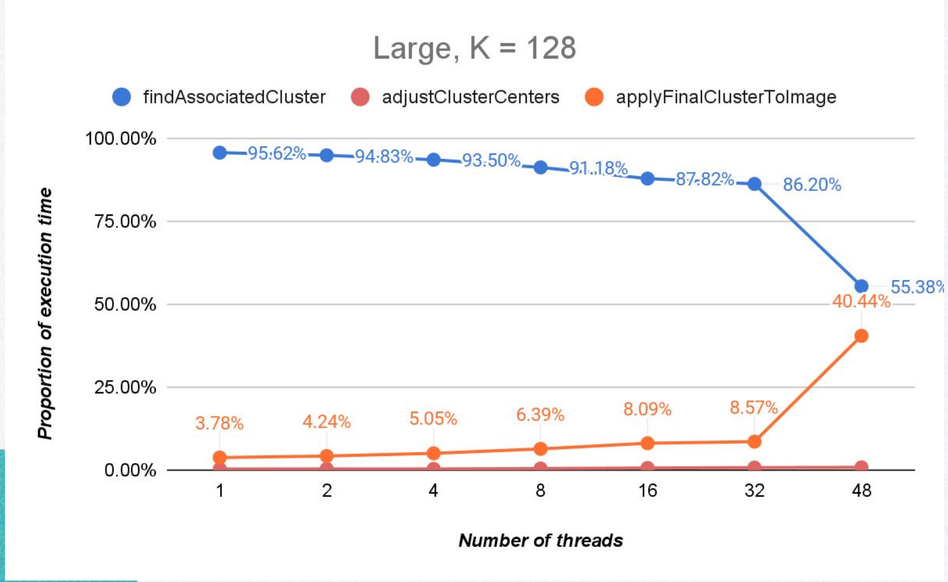
1. The main workload depends on the number of pixels.
2. The cost of reconstruction will become more and more apparent as the number of threads increases.

Reconstruction cost : Merge ***“k”*** local_image together.

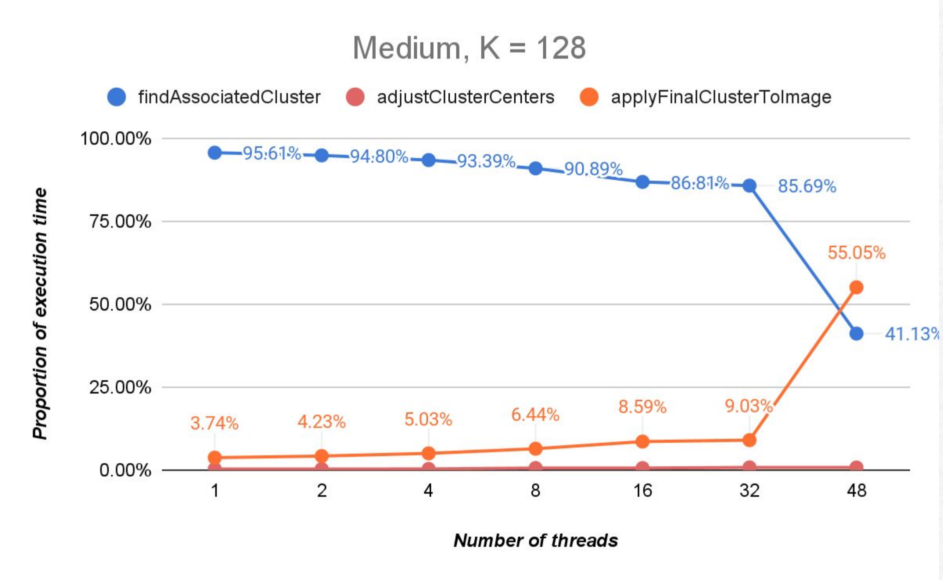
Evaluation

- Computation time decreases as image size decreases.
- Speedup decreases when the number of threads increases to 32.
- Efficiency drops significantly when the number of threads increases to 32.
- The time proportion of ***"findAssociatedCluster"*** decreases as the number of threads increases, but the time proportion of ***"applyFinalClusterToImage"*** increase as the number of threads increases.

Evaluation - Proportion of execution time



size = 1920 * 1330



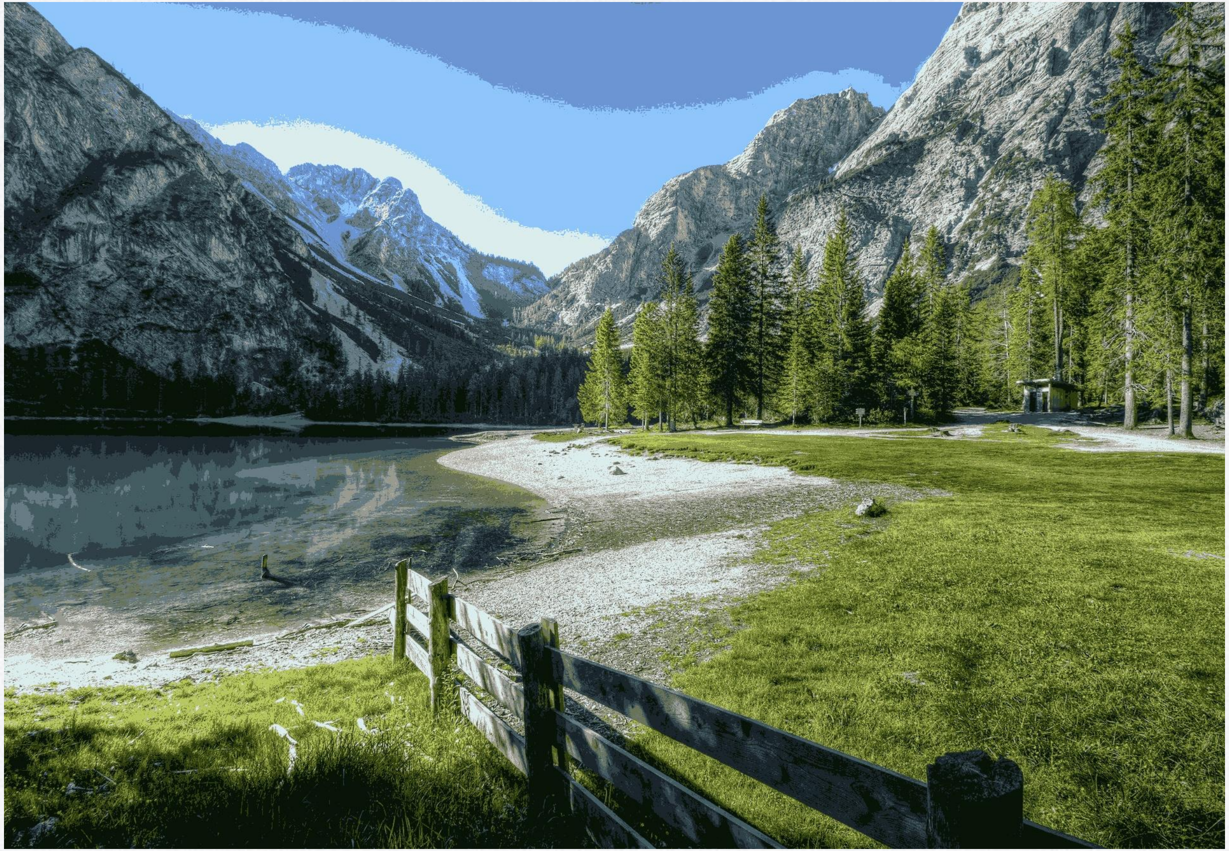
size = 1280 * 887

Conclusion

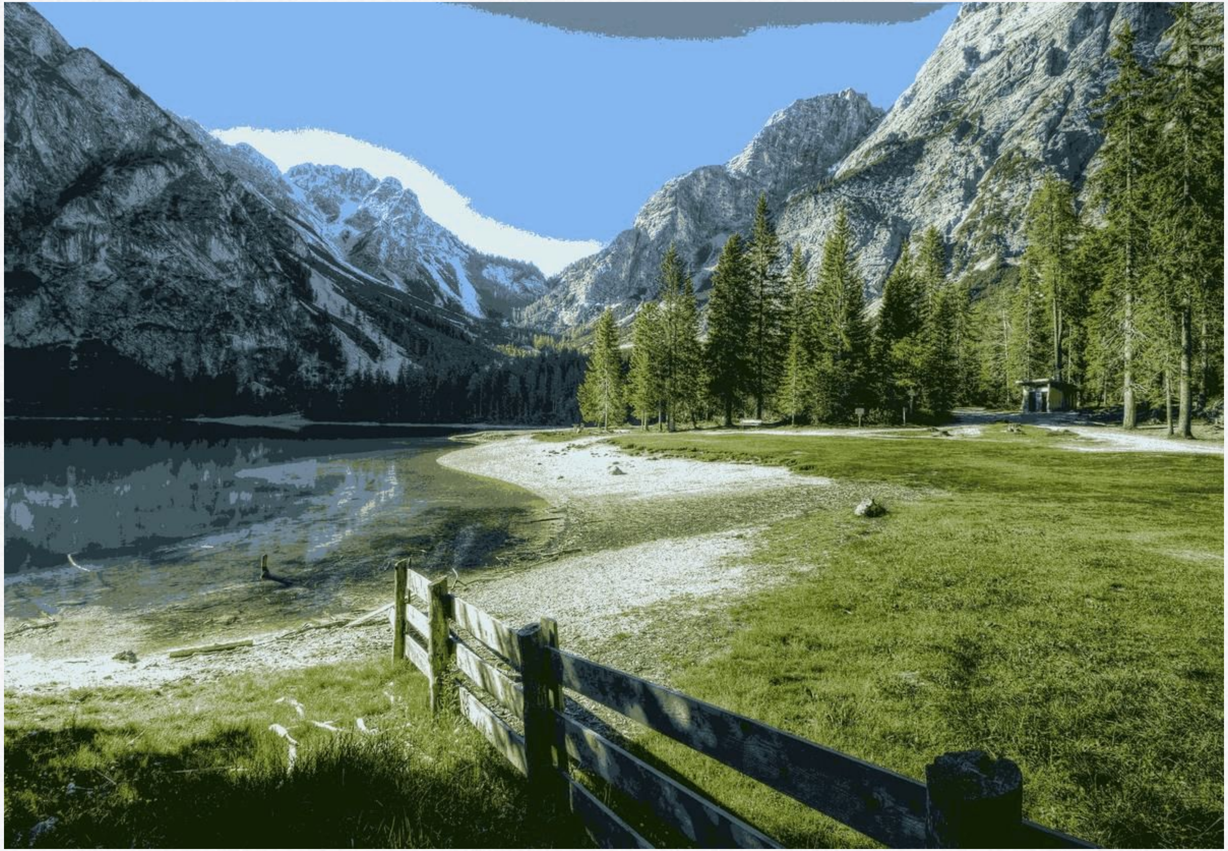
1. The main workload depends on the number of pixels.
2. The cost of reconstruction will become more and more apparent as the number of threads increases.
3. Using the parallelization method with split rows can achieve better performance when increasing the number of threads.
4. The cost of reconstruction will become more and more apparent as the number of pixels decreases.

Source

Result k = 32



Animated GIF k =32



Related work

Contributions of each member

- 310551154 林子恒 33%
- 310554047 張方華 33%
- 310552059 曾宇廷 33%

Q&A