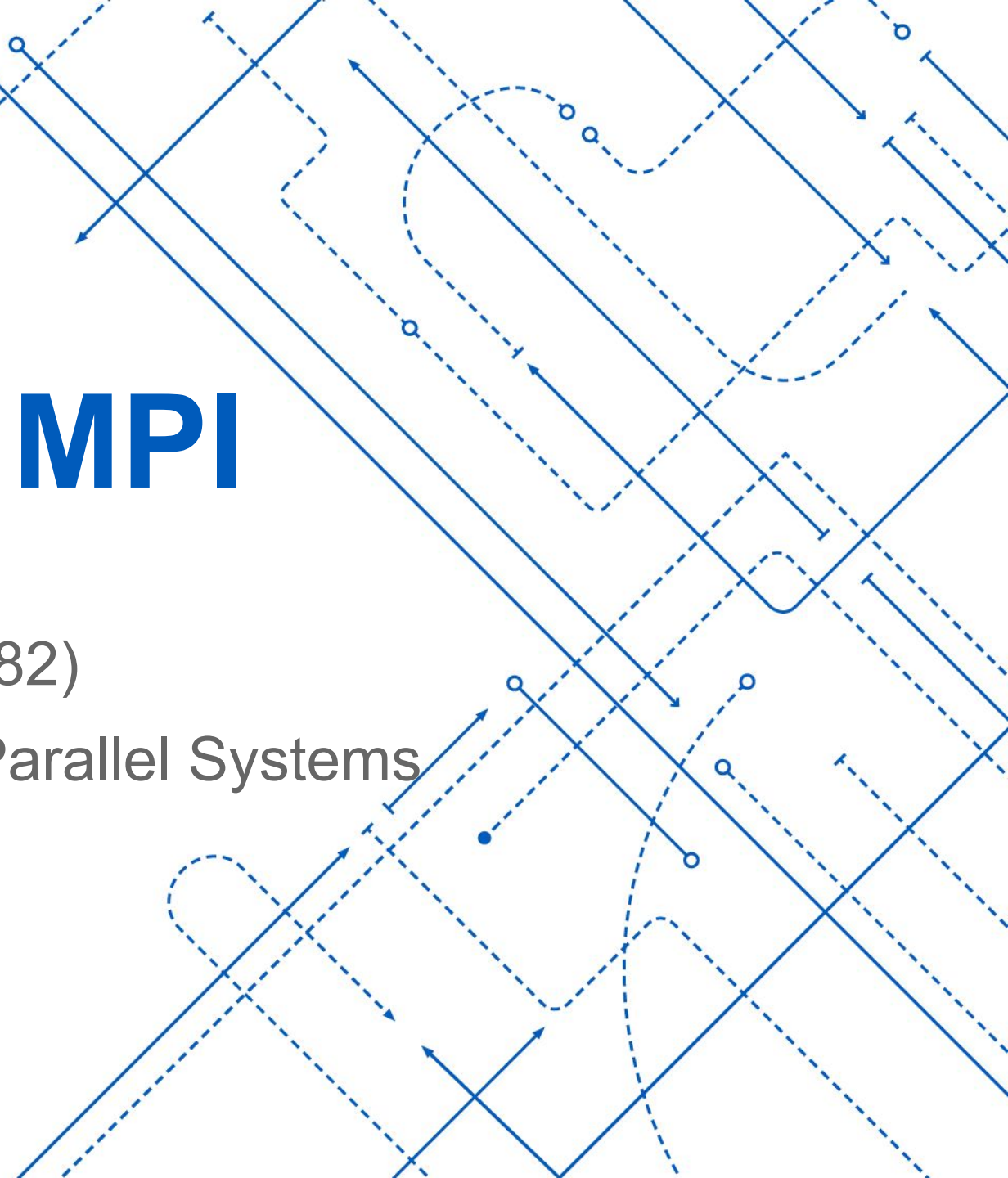
# Connected Component Labelling using MPI

Final presentation

By Swapnil Sanjay Satpute (50365582)

CSE702 - Programming Massively Parallel Systems

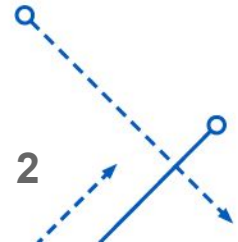
Instructor - Dr. Russ Miller



University at Buffalo The State University of New York

# Table of Content

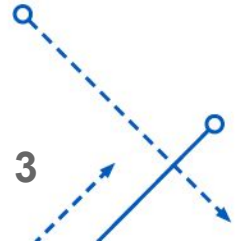




# Problem Description

Creation of a labeled image in which the positions associated with the same connected component of the binary input image have a unique label.

- Goal is to detect unique region in a binary image where each unique region is given a unique label.
- Each foreground pixel can be considered as a vertex.
- Vertices are neighbors if they’re one pixel space away.
- We could have four-connected neighbors or eight connected neighbors.
- For easier implementation, we will assume a four-connected neighbors system.

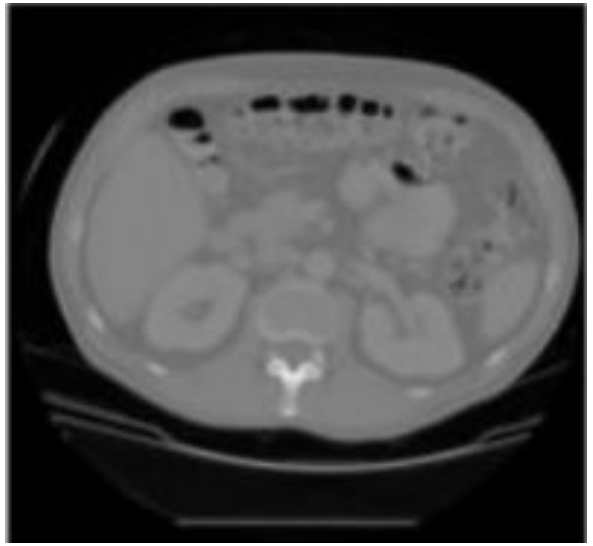


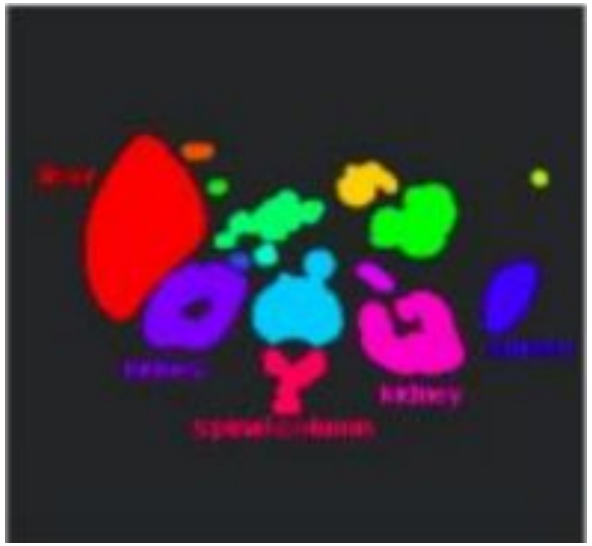
# Application

Labelling vehicle number plate



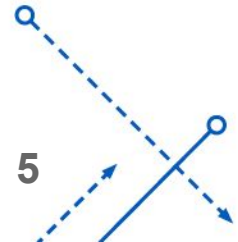
Labelling CT cross-section





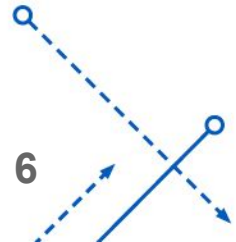
# Methods

- Recursive Tracing
- Row-by-row (or Two-pass)
- Parallel



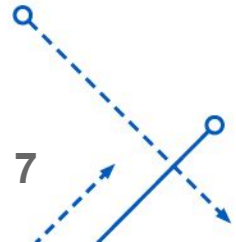
# Row-by-row (or Two-Pass)

- Classic model used for connected component labelling.
- First pass :
  - Scan each element in the first row.
  - Assign a temporary label to the first foreground pixel (x).
  - Continue labelling until a break occurs.
  - Moving forward, if the next pixel (x+i) is not connected to the element before or above it, increment the labelling counter and assign the value to it.
  - In case another label is connected to the pixel, we take the minimum of the value, but set up an equivalence list.



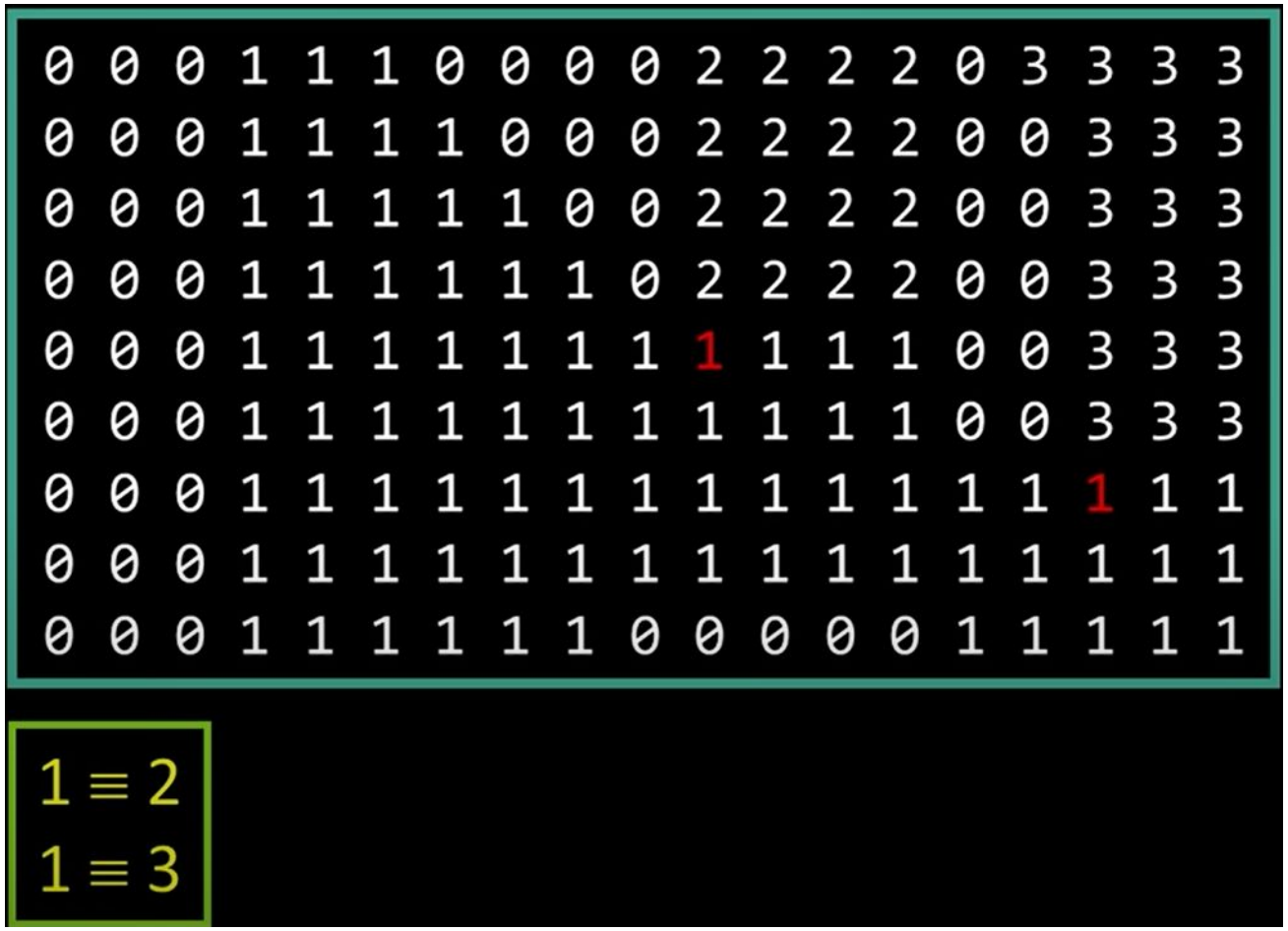
# Row-by-row (or Two-Pass)

- Second Pass :
  - Scan the image again and re-label each foreground pixel based on the lowest value in the equivalence list.



# Row-by-row (or Two-Pass)

Sample image ⟹

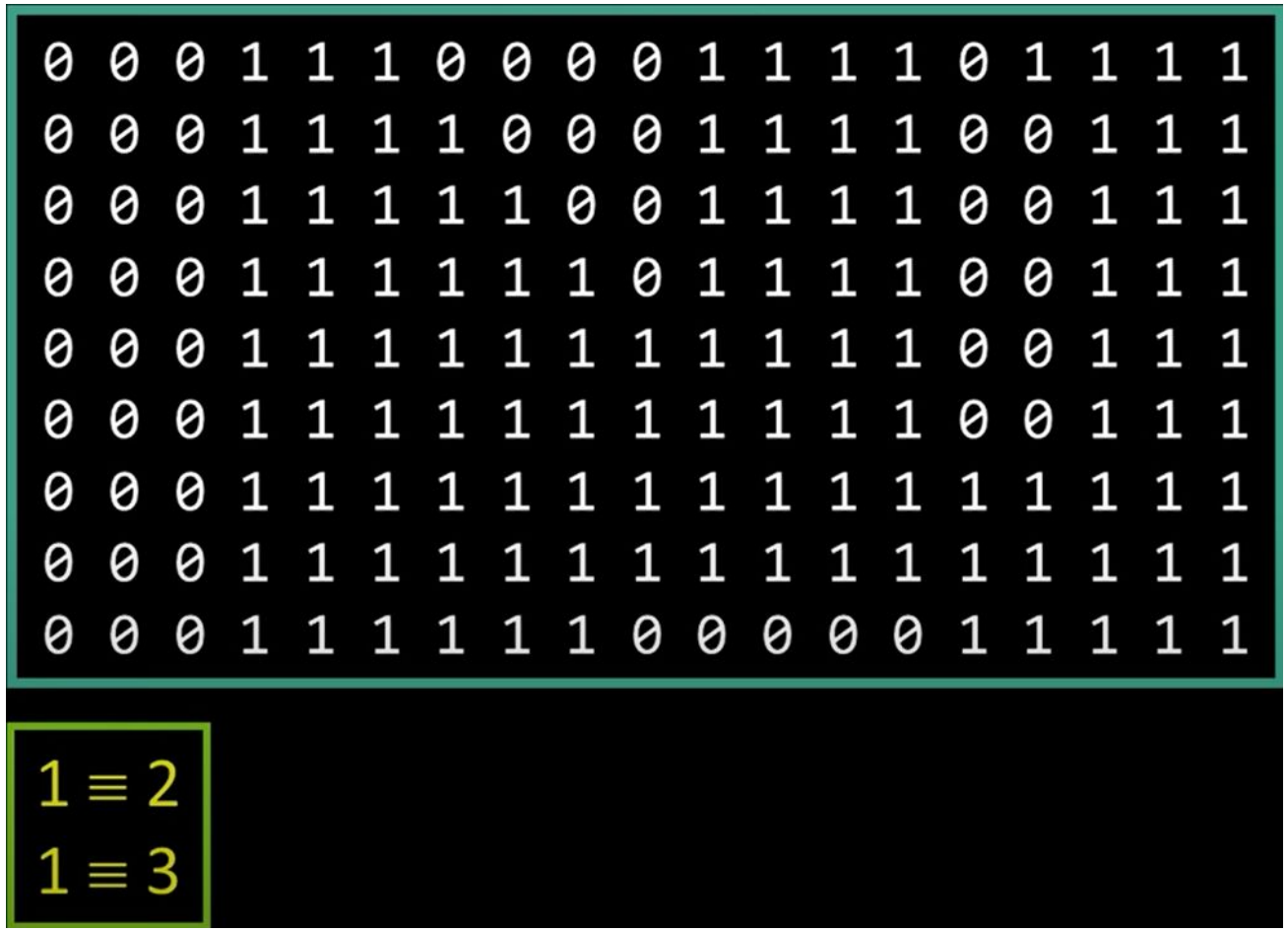



Equivalence List ⟹

$1 \equiv 2$

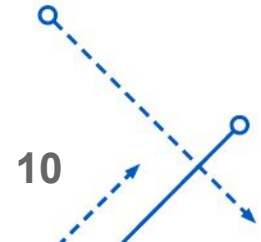
$1 \equiv 3$

# Row-by-row (or Two-Pass)



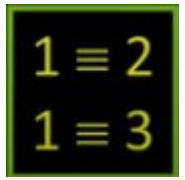
# Parallel Propagation

- Split the image into different sections by rows thereby obtaining multiple images.
- Each processor will receive an image.
- Compute the Two-pass algorithm locally on each processor.
- Each processor will then label based on neighbouring pixels of the image they received..
- The root node then establishes a global equivalence list and broadcast it to each processor based on border pixel labels.
- All processors will then perform Second-pass and re-label the pixels w.r.t the global equivalence list.



# Example


Global Equivalence List



# Why Parallel?

Disadvantages of non-parallel implementation :

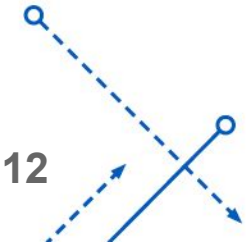
- Time Consuming.
- Need to maintain a large equivalence table.

Using parallel implementation, we divide the pixels and assign each part to a separate processor.

This reduces the computation time of the algorithm.

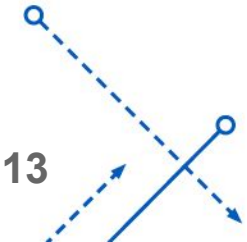
Advantages of parallel implementation :

- Lesser computation time.
- Eliminates need to maintain large equivalence table.



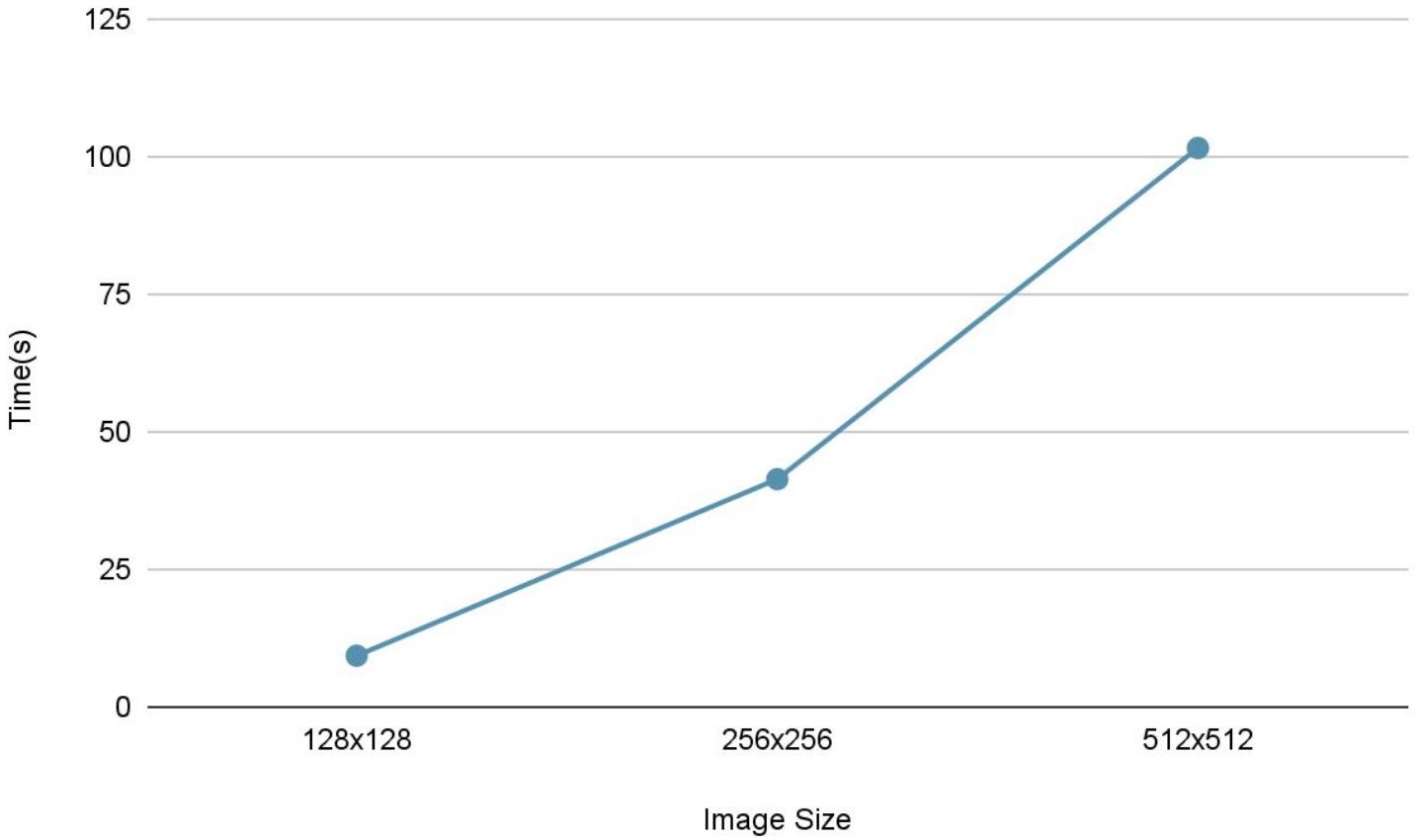
# Implementation

- Generate a 2D matrix on root processor (Simulated binary image)
- Distribute each part of the matrix to different processors using MPI_Scatter.
- Perform Two-pass algorithm on each processor locally.
- Gather the boundary label information from each processor using MPI_Gather.
- Relabel adjacent pixels on border.
- Relabel connected neighbour pixel.



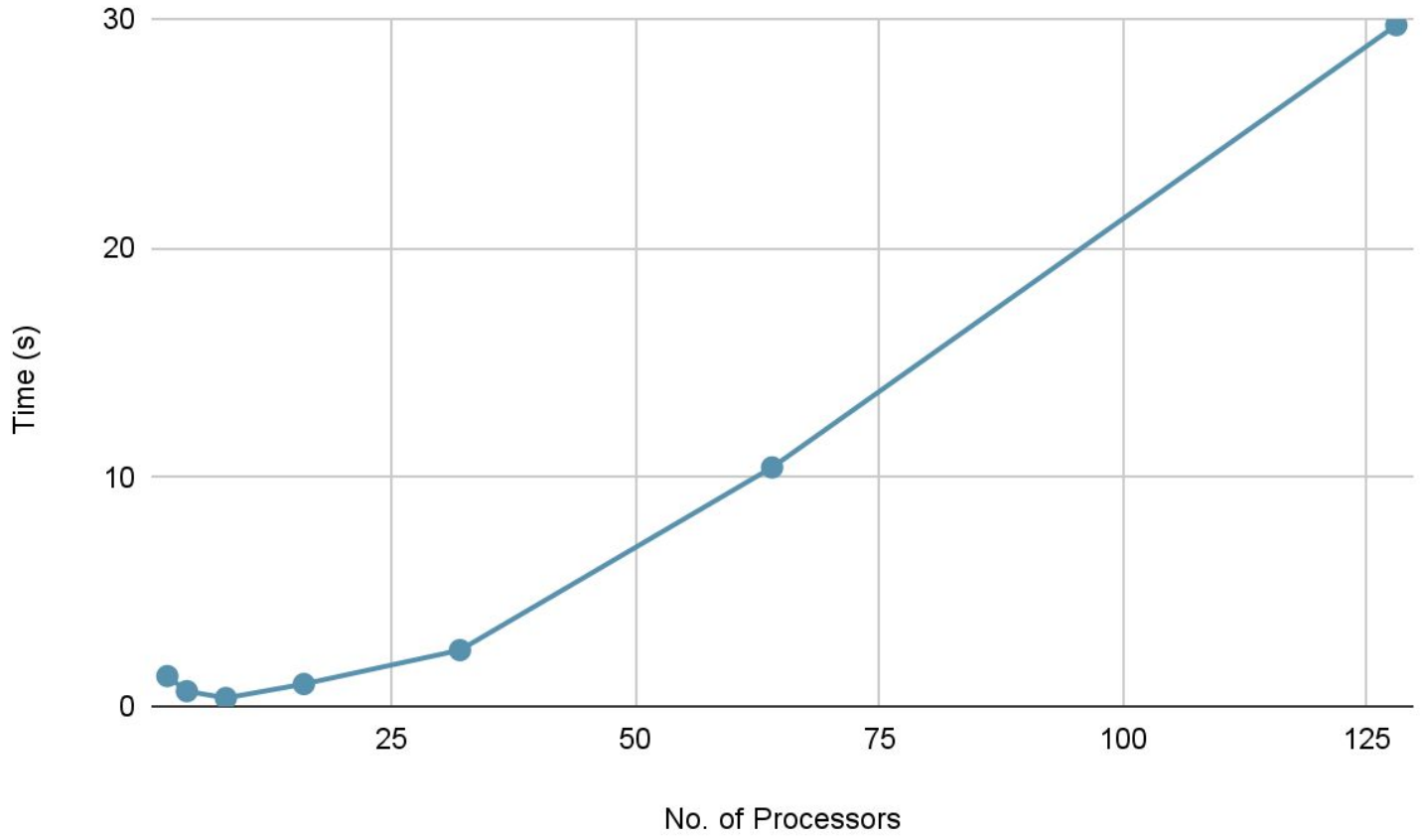
# Results (Sequential or row-by-row)

| Image Size | Time(s) |
|---|---|
| 128x128 | 9.32 |
| 256x256 | 41.42 |
| 512x512 | 101.626 |



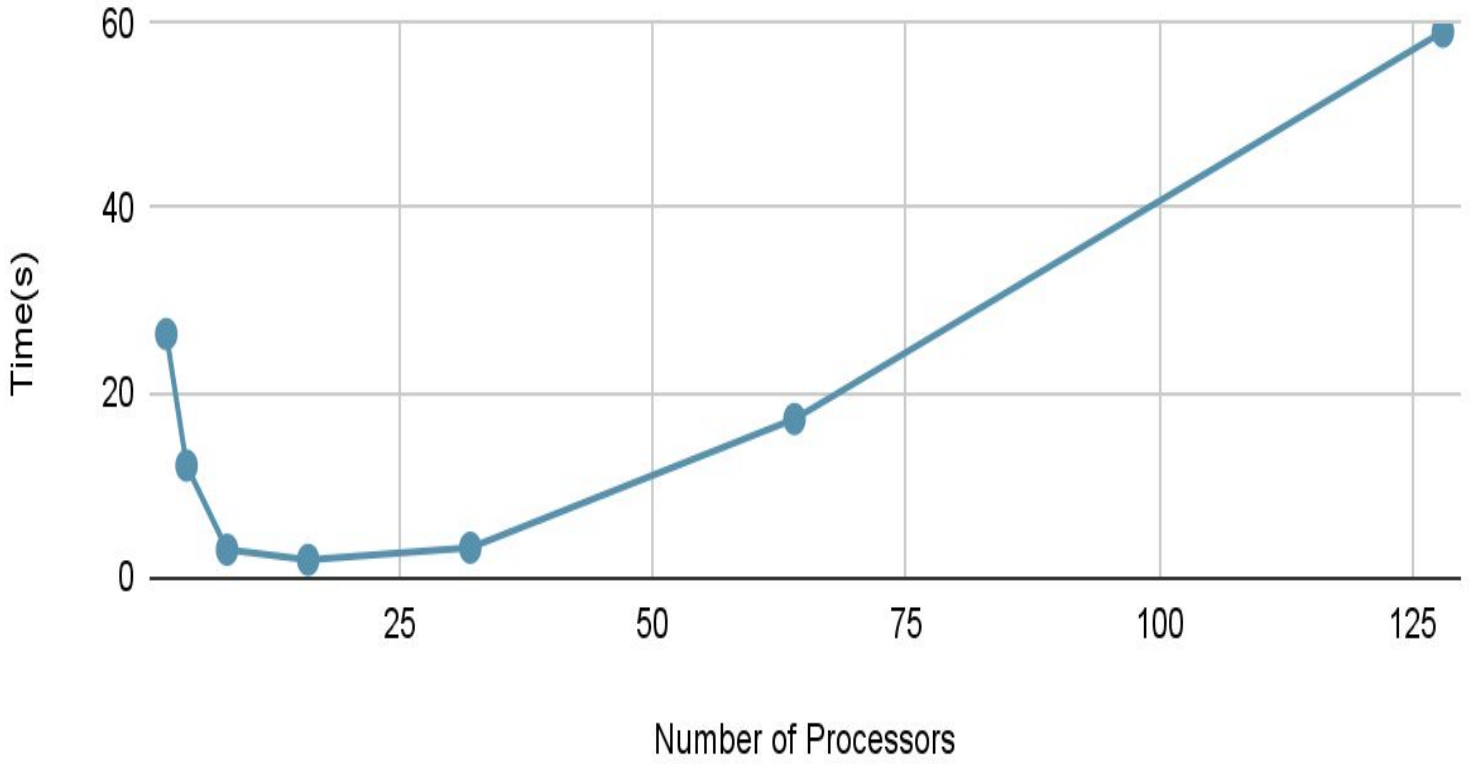
# Results (128x128 image)

| No. of Processors | Time(s) |
|---|---|
| 2 | 1.32 |
| 4 | 0.662 |
| 8 | 0.363 |
| 16 | 0.974 |
| 32 | 2.453 |
| 64 | 10.43 |
| 128 | 29.739 |



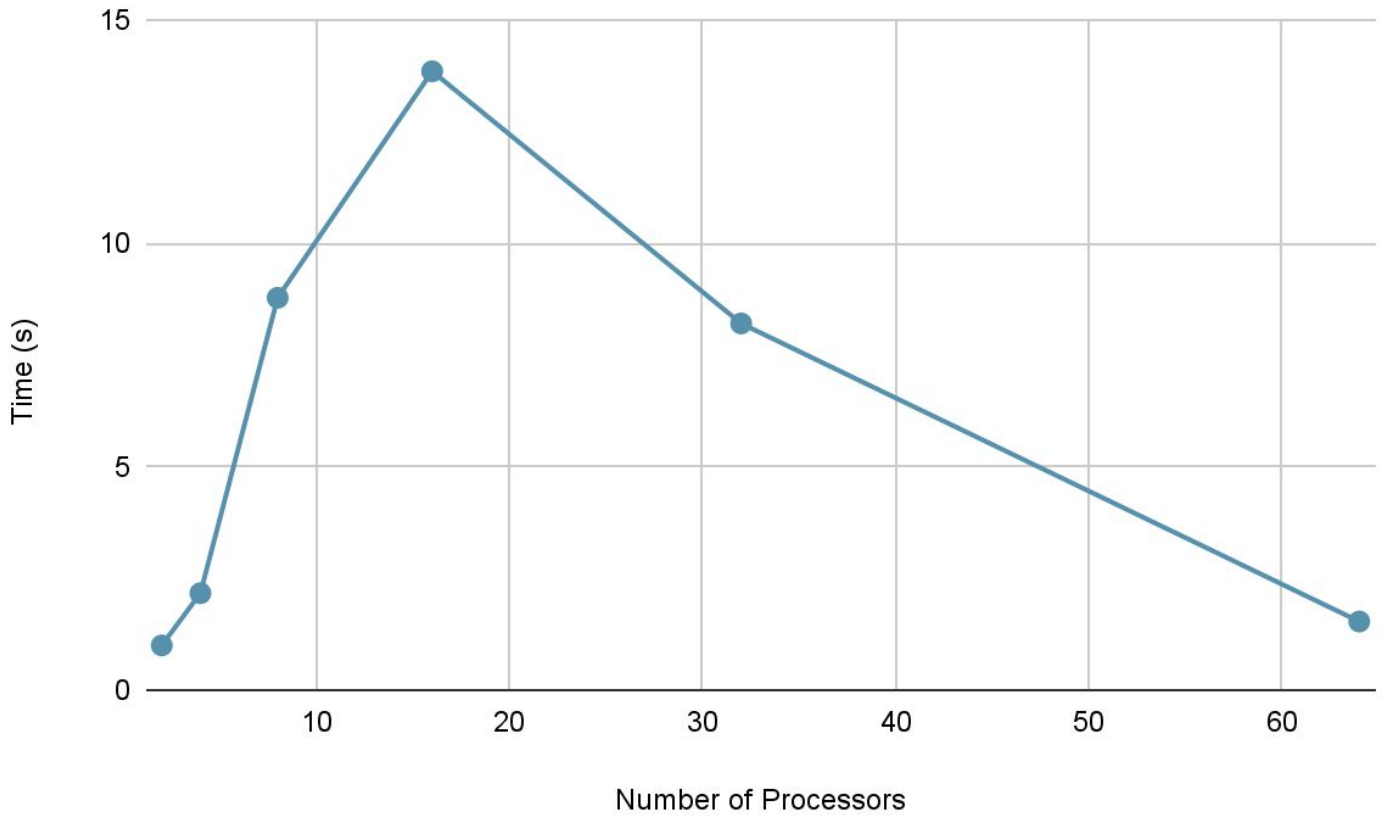
# Results (256x256 image)

| No. of Processors | Time(s) |
|---|---|
| 2 | 26.33 |
| 4 | 12.102 |
| 8 | 2.997 |
| 16 | 1.9 |
| 32 | 3.21 |
| 64 | 17.137 |
| 128 | 59.05 |



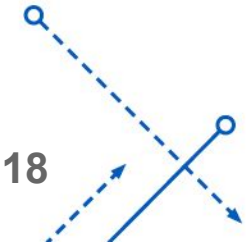
# Speed up

| No. of Processors | Scaling |
|---|---|
| 2 | 1 |
| 4 | 2.17 |
| 8 | 8.785 |
| 16 | 13.857 |
| 32 | 8.21 |
| 64 | 1.536 |



# Conclusion

- Parallel algorithm uses lesser time for computation compared to sequential algorithm
- A particular image has an optimal number of processors to be used after which the communication time exponentially increases the time taken



# Scope

- Use eight-neighbors system for better region labelling.
- Compare results with recursive tracing algorithm.
- Use actual image as input and convert it into a binary image.

# References


- R. Miller and L. Boxer, Algorithms Sequential and Parallel: A Unified approach
- Introduction to Computer Vision course - Udacity
- https://en.wikipedia.org/wiki/Connected-component_labeling
- A Parallel Graph Algorithm for Finding Connected Components - Hirschberg, D.S. (1975)
- https://pvs-studio.com/en/blog/posts/cpp/a0054/

# THANK YOU