

Convolution of Images with MPI & OMP

By: Priyanka
Parimelazhagan
Year 3, AI & DS B
21011101090

THEORY

- A very useful technique learnt in Image Processing is the convolution of a 2D image with a convolution filter.
- The discrete convolution of an input image 'I_I' with the filter 'h' to produce an output image 'I_O' is defined as follows:

$$I_O(i, j) = \sum_{p=-s}^s \sum_{q=-s}^s I_I(i-p, j-q) * h(p, q)$$

- Here the filter 'h' needs to be normalized to avoid color distortions in the image.

$$h = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \longrightarrow h' = \begin{bmatrix} \frac{1}{16} & \frac{2}{16} & \frac{1}{16} \\ \frac{2}{16} & \frac{4}{16} & \frac{2}{16} \\ \frac{1}{16} & \frac{2}{16} & \frac{1}{16} \end{bmatrix}$$

PROJECT SUMMARY

- *To implement the convolution of a 2D image using parallel processing techniques like OMP and MPI to accelerate the computation*
- *By leveraging parallelism, the goal is to accelerate the blurring process for large images.*

OBJECTIVES

- *Implement image blurring algorithm using OpenMP and MPI.*
- *Achieve significant speedup compared to sequential processing.*
- *Analyze the scalability of the parallel implementations.*
- *Identify challenges and limitations of parallel image blurring.*

ASPECTS OF PARALLELISM

- *Parallelism will be exploited at both pixel-level (using OpenMP) and task-level (using MPI)*
- *OpenMP will parallelize the computation of individual pixels within an image*
- *MPI will distribute the workload across multiple nodes in a cluster*

MPI IMPLEMENTATION

- *MPI is a library that allows us to write parallel programs that run on multiple processors.*
- *Communication between the processors is done by sending messages to each other, without using shared memory.*
- *In this case, original image is split into smaller blocks, as many as the processes.*
- *Each one undertakes a particular process and apply convolution to one of the blocks of the image.*
- *In this way an equal division of workload and consequently a reduction of the total processing time is achieved.*

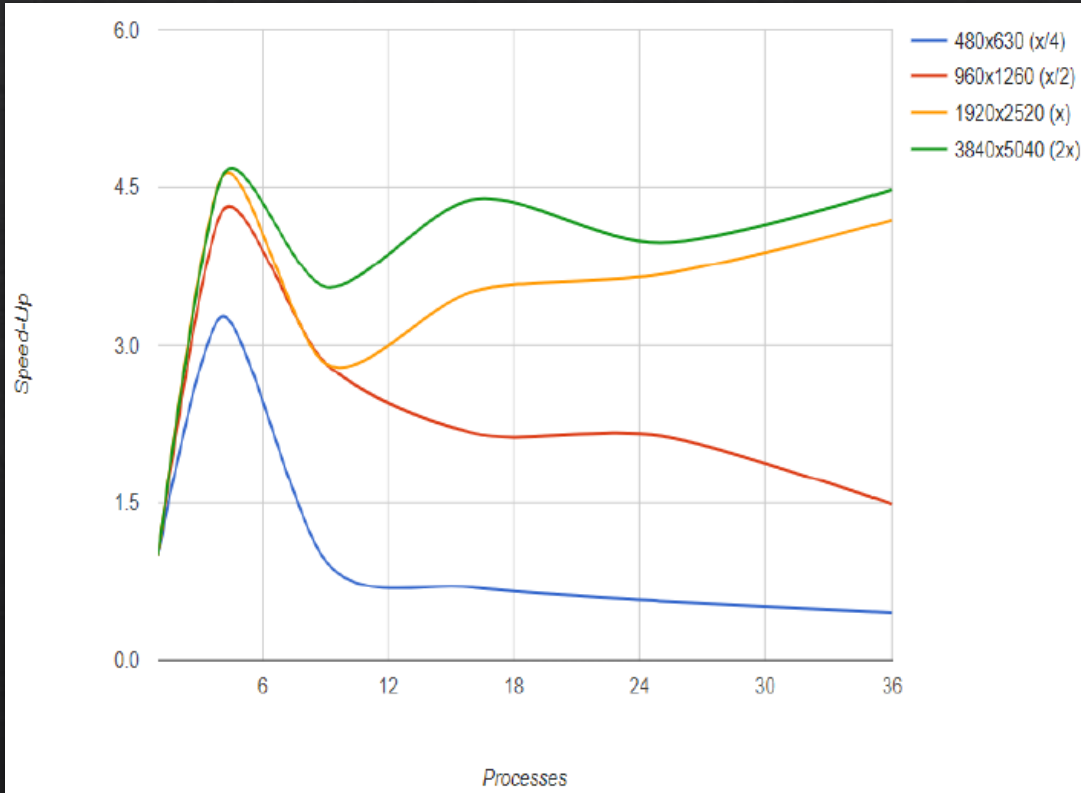
MPI IMPLEMENTATION

- For the sending and receiving of messages, nonblocking functions, *lsend()* and *lrecv()* of the MPI library are used
- This way convolution of the inner pixels of the block is done first and then the calculation of the extreme points
- This enables a process to deal with a task while waiting to receive data from neighboring processes.
- To store the data, special structures of the MPI library, *Datatypes*(vector, contiguous) are used for the rows and columns of the image respectively.

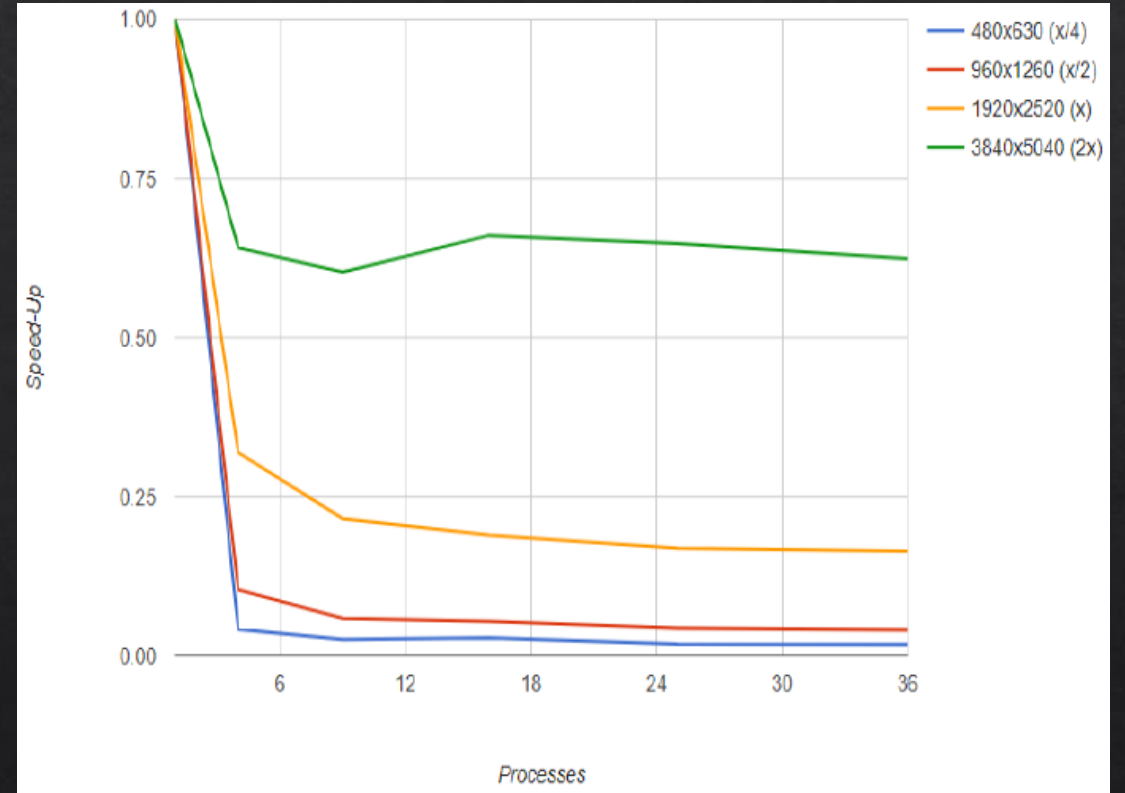
OPENMP IMPLEMENTATION

- *OpenMP is a method to parallelize a program using shared memory, at the instruction level to the compiler.*
- *Through a set of instructions, which are integrated into the code of a program, the compiler can achieve parallelism in the program.*
- *Achieved using threads through OpenMP, where local chunks of code can run in parallel.*
- *When combined with MPI, in a hybrid program, an even greater improvement in processing time is observed.*

PERFORMANCE METRICS: SPEED-UP PLOT

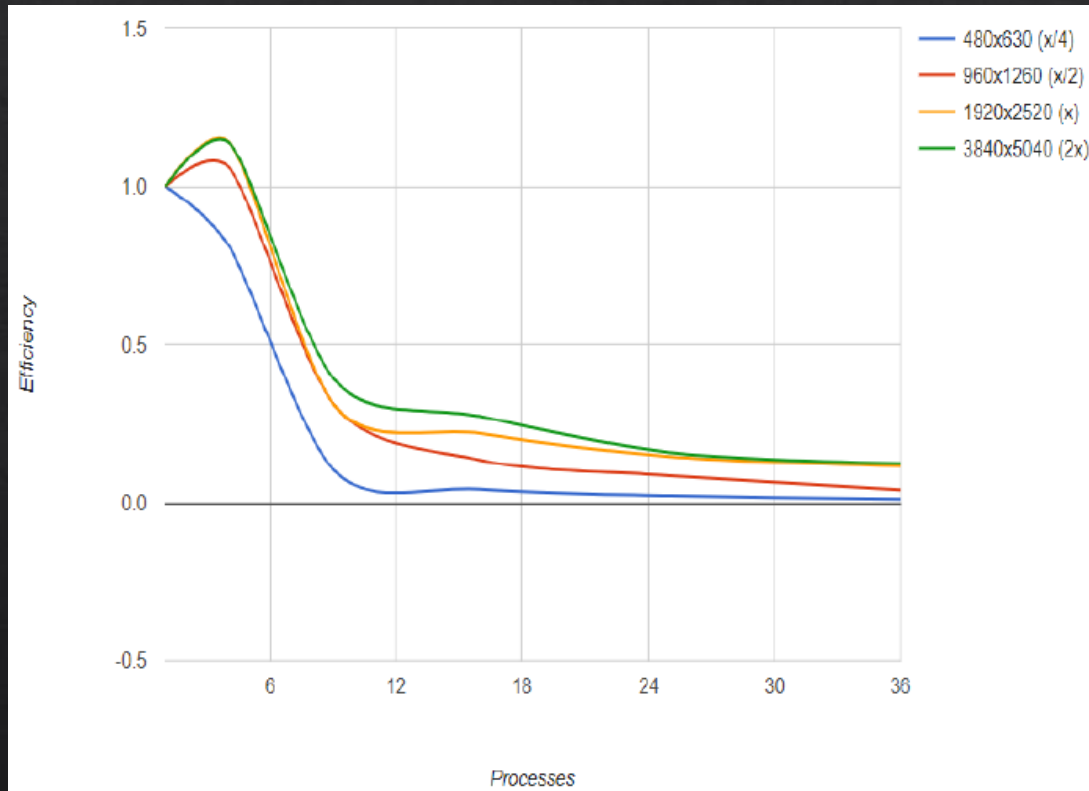


MPI

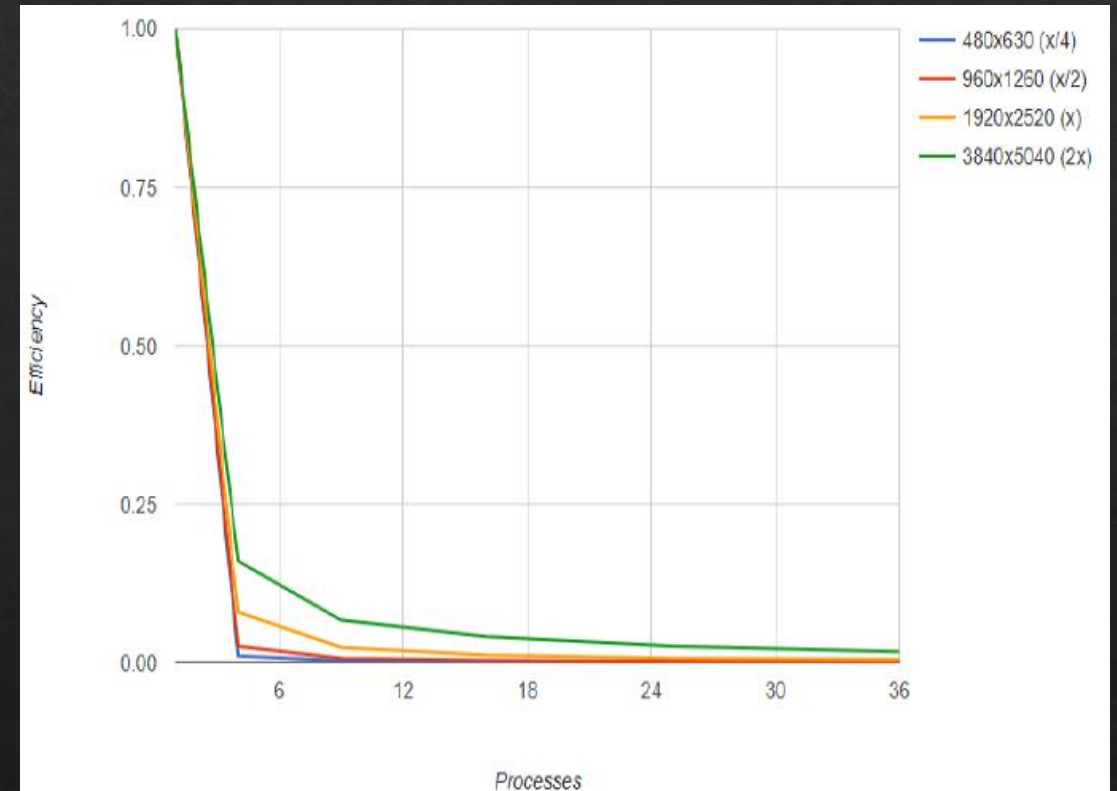


OMP & MPI

PERFORMANCE METRICS: EFFICIENCY PLOT



MPI



OMP & MPI

CHALLENGES

- *Communication overhead: Minimizing the overhead associated with inter-process communication in MPI (delay in communication)*
- *Load balancing: Ensuring an even distribution of workload among processing elements.*
- *Memory management: Efficient utilization of memory resources, especially for large images.*
- *Scalability: Ensuring that the parallel implementations scale well with increasing problem sizes and processing elements.*