



# GAUSSIAN ELIMINATION USING MPI

*Submitted by*  
**Dolly Gupta (181IT115)**  
**Naman Vijayvargiya (181IT129)**  
**Shraddha Gole (181IT145)**  
**Group no. 8**



# Introduction

- Gaussian Elimination is an algorithm used to solve a system of linear equations.
- This algorithm aims to transform a system of linear equations into augmented matrix and then convert that into an upper-triangular matrix and then apply back-substitution method to get the solution.
- It is a way which makes it very easy to solve linear equations in multiple variables and find a lot of advantage in linear algebra used in various deep learning models.
- Applications - Determinant of matrix, inverse of a matrix etc.
- This project implements the algorithm for a set of equations (represented by matrices) distributed across multiple processors. The matrix is split up using *striped partitioning*.



# Objective

- To implement Gaussian Elimination in an efficient way.
- Can be achieved by an improved version on multi-core computers with increasing number of cores.
- In serial approach, elementary calculation of each row depends on all row before it and thus it is time costly.
- This can be avoided by dividing rows among processors and hence this reduces time.
- Each processor contains a subset of rows from the original matrix.
- Rows will be mapped among processors using:
  - Block striped mapping
  - Cyclic striped mapping



# Literature Survey

## 1. *[“System of Linear Equations, Gaussian Elimination”](#)*

In this paper linear equations are discussed in detail along with Gaussian elimination method. The purpose of this paper is to revise an introductory concept of linear equations, matrix theory and forms of Gaussian elimination.

## 2. *[“cs.rutgers.edu/~vengopa/parallel\\_summer2012”](http://cs.rutgers.edu/~vengopa/parallel_summer2012)*

This paper tell us about the main perspective of using gaussian algorithm. This depicts the scope of parallel gaussian algorithm using mapping methods and various types of communications.

# Serial Algorithm For Gaussian Elimination

1. Start
2. Read Number of Unknowns:  $n$
3. Read Augmented Matrix (A) of  $n$  by  $n+1$  Size
4. Transform Augmented Matrix (A) to Upper Triangular Matrix by Row Operations.
5. Obtain Solution by Back Substitution.
6. Display Result.
7. Stop

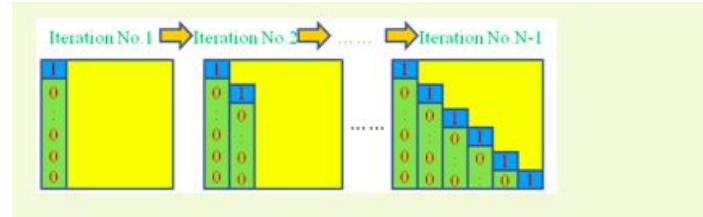


Fig 1: Gaussian Elimination Method



# Pseudo Code

```
1. Start
2. Input the Augmented Coefficients Matrix (A):
    For i = 1 to n
        For j = 1 to n+1
            Read Ai,j
        Next j
    Next i
3. Apply Gauss Elimination on Matrix A:
    For i = 1 to n-1
        If Ai,i = 0
            Print "Mathematical Error!"
            Stop
        End If
        For j = i+1 to n
            Ratio = Aj,i/Ai,i
            For k = 1 to n+1
                Aj,k = Aj,k - Ratio * Ai,k
```

```
                Next k
            Next j
        Next i
4. Obtaining Solution by Back Substitution:
    Xn = An,n+1/An,n
    For i = n-1 to 1 (Step: -1)
        Xi = Ai,n+1
        For j = i+1 to n
            Xi = Xi - Ai,j * Xj
        Next j
        Xi = Xi/Ai,i
    Next i
5. Display Solution:
    For i = 1 to n
        Print Xi
    Next i
6. Stop
```



# Implementation

Gaussian Elimination will be implemented in the following ways:

- Serial
- Parallel (using methods of striping)
  - Block Striped Mapping
  - Block Striped Mapping with Pipelining
  - Cyclic Striped Mapping



# Mapping

- Refers to the method of dividing up a matrix between the processors in a geometric manner.
- An entire row or group of rows are issued to each processor.
- **Striped mapping** -
  - Each row is an atomic unit.
  - There cannot be more processors than rows in the equation matrix.
    - Blocked mapping
    - Cyclic Mapping

## Blocked mapping

- A processor is assigned a number of *contiguous* rows in  $n/p$  sections.
- Disadvantage - Load Imbalance Problem
- How to solve? -Cyclic Mapping

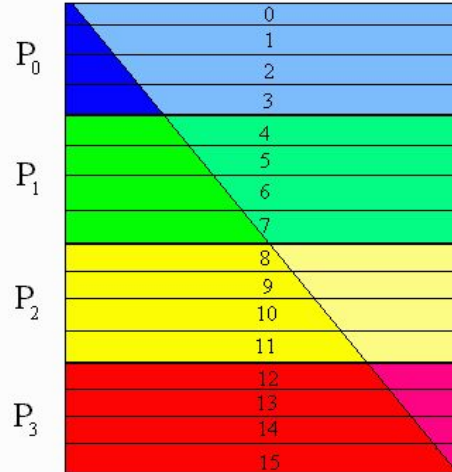


# Mapping

- **Cyclic mapping-**
  - Divides the matrix rows into  $n/p$  sections.
  - Assigns a processor one row from each section.
  - Advantage- Improves the load-balancing problem.
  - Disadvantage - complicates pipelined communication since the processor to start the chain is different at each iteration.

## Block-striped mapping

$P_3$  must do much more computation than the other processors. Other processors become idle.



## Cyclic-striped mapping

Each processor has nearly the same load so idling is reduced to a minimum

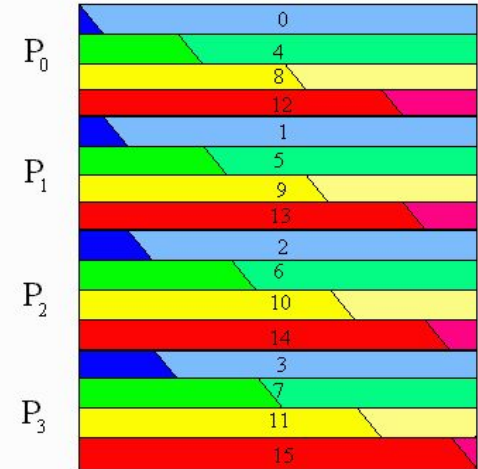


Fig : Blocked vs. Cyclic Stripping

# Types of communication between the processes

- **Broadcast Communication:**
  - A row broadcasts its newly calculated values to all members after completing division step.
  - This means all processors must wait until all others receive the data.
- **Pipelining:**
  - A row is sent to a processor who forwards it on to the next processor and then it can immediately begin its local calculations.
  - This quick point-to-point message allows a processor to do its local calculations *concurrently* while other processors are communicating.

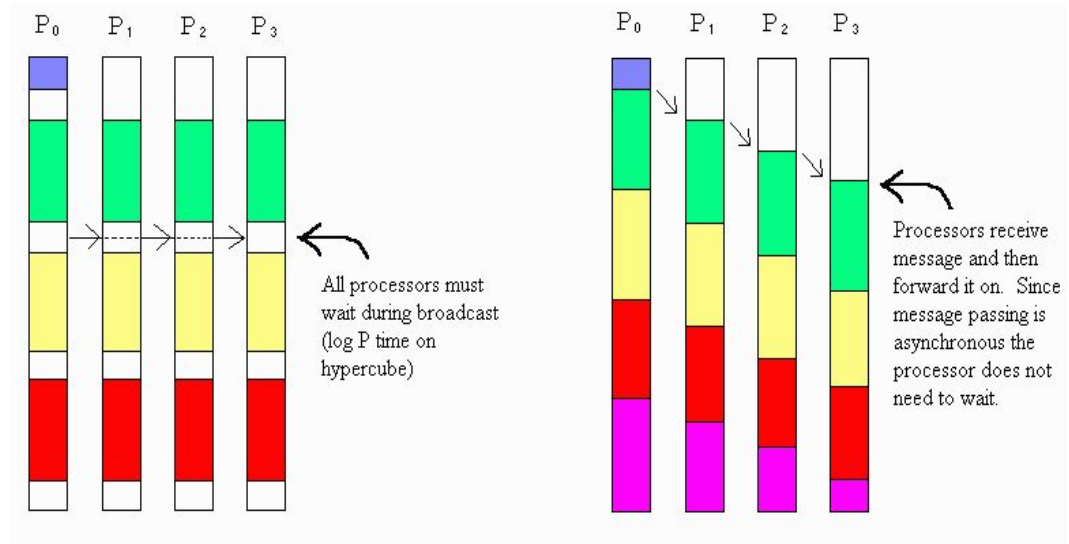
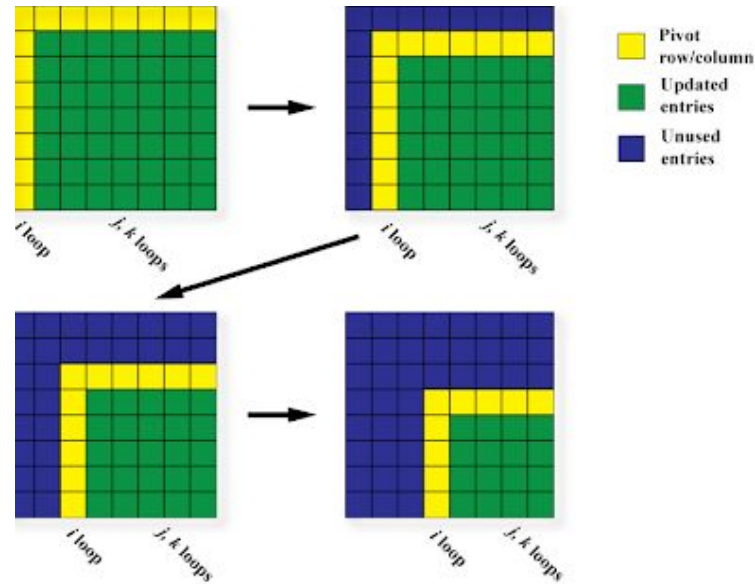



Fig: Broadcast vs. Pipelined Communication

# Identifying the parallel Region

We have three possible candidate loops to parallelize.



- 
- The  $i$  loop is represented by the yellow row and column. The entries in the yellow row and column are being used to update the green sub matrix before going on to row/column  $i+1$ , meaning the values of the entries in the  $(i+1)$ st yellow area depend on what operations were performed on them at previous values of  $i$ . Therefore we can't use OpenMP to parallelize this loop because of data dependence.
  - The  $j$  loop has a number of iterations that varies with  $i$ , but we do know the number of iterations every time we are about to enter the loop. None of the later iterations depend on the earlier ones and the iterations can be computed in any order! So the  $j$  loop is parallelizable.
  - The  $k$  loop, like the  $j$  loop, has a number of iterations that varies but is calculable for every  $i$ . None of the later iterations depend on earlier ones, and they can all be computed in any order. Therefore the  $k$  loop is also parallelizable.



# Individual Contribution

**Dolly Gupta:** Analysed the Gaussian elimination and broke down into simpler parts in form of an pseudocode. Further figuring out how pipelining can be integrated with the blocked striping implementation to make it faster and smoother..

**Naman:** Implemented the fully working sequential version of the pseudocode into a cpp program. Validated by testing with several inputs. Have to come up with a full fledged MPI version of the sequential code based on blocked striping.

**Shraddha:** Identified the regions in the cpp code that could be parallelized, identified the possible data dependency involved among loops .Have to replace the blocked striping with pipelining strategy with the cyclic implementation of the same.



## Conclusion

- Providing a good parallel algorithm for Gaussian elimination is tricky as there are communication dependencies as well as load-balance problems.
- We can see that cyclic-striped mapping provides a vast improvement over block-striped mapping, although it does not completely eliminate load imbalance.
- This algorithm involves more complex communication, but can prove to be more scalable since more processors can be assigned to a given matrix.