GAUSSIAN ELIMINATION USING MPI

Dolly Gupta
Information Technology
National Institute of Technology,
Surathkal
Karnataka, India

Shraddha Gole
Information Technology
National Institute of Technology,
Surathkal
Karnataka, India

Naman Vijayvargiya
Information Technology
National Institute of Technology,
Surathkal
Karnataka, India

***Abstract*—This Gaussian Elimination is an algorithm used to solve a system of linear equations. The basic concept is to subtract some scalar of an equation from another equation in order to eliminate an unknown. This is done repeatedly until the set of equations is trivial to solve. Due to the nature of the computation, the inherent way of splitting the rows produces load imbalance and is not sufficient. A more efficient method called *cyclic-striped* mapping distributes the rows in a circular manner among the processors. Also, because of increased communication costs between iterations, *pipelining* is used so that some processors may be computing while others are communicating. Using these methods, this project will involve making an efficient and scalable parallel algorithm for Gaussian Elimination.**

Keywords—Gaussian Elimination, striped partitioning, cyclic-striped partitioning, pipelining.

I. INTRODUCTION

Carl Friedrich **Gauss** lived during the late 18th century and early 19th century, but he is still considered one of the most prolific mathematicians in history. His contributions to the science of mathematics and physics span fields such as algebra, number theory, analysis, differential geometry, astronomy, and optics, among others. His discoveries regarding matrix theory changed the way mathematicians have worked for the last two centuries. One of which is finding a solution of a system of linear equations.For the purpose of our project we have taken a consistent system of equations, which would guarantee a unique solution.

The method works by representing the system of equations as AX=B in matrix format.Then the matrix is converted into an upper triangular matrix by applying some elementary row operations. This is done to ensure elimination of unknown variables . Back substitution on the resulting set of equations gives us the solution.

But a major disadvantage and the very reason for us to take this up as a project is that this method is a very slow procedure because of which it takes time. We have implemented the method using MPI. The workload is balanced among several processors and techniques such as Blocked-Striped mapping and cyclic-striped mapping. Cyclic-striped partitioning has been used to avoid load imbalance. With the help of pipelined communication a row is sent to a processor who forwards it on to the next processor and then it can immediately begin its local computations. This quick point-to-point message allows a processor to do its local calculations *concurrently* while other processors are communicating.

II. LITERATURE SURVEY

A. Gaussian Elimination

Gaussian elimination, also known as **row reduction**, is an algorithm in linear algebra for solving a system of linear equations. It is usually understood as a sequence of operations performed on the corresponding matrix of coefficients. This method can also be used to find the rank of a matrix, to calculate the determinant of a matrix, and to calculate the inverse of an invertible square matrix.[1]

Gaussian elimination aims to transform a system of linear equations into an upper-triangular matrix in order to solve the unknowns and derive a solution. A pivot column is used to reduce the rows before it; then after the transformation, back-substitution is applied.

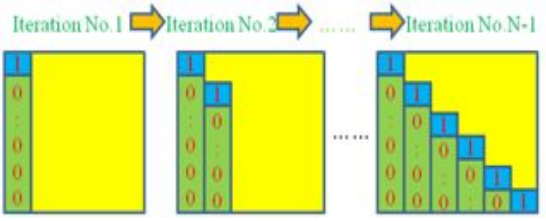

Fig 1: Gaussian Elimination Method[2]

Here is the serial algorithm for this method:

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

The algorithm descends a row of the equation matrix at each iteration. At this row it computes the new values of the row and its corresponding value in the solution vector. This new row and vector information is used to calculate all rows below it. Thus the difficulty in parallelizing Gaussian elimination is that the calculation of each row requires the calculation of all the rows that have come before it. Concurrency of operation is the overriding issue.

There are two different variables that are tested in our implementation: blocked vs. cyclic mapping and broadcast vs. pipelined communication.

B. Striped Mapping

This mapping refers to the method of dividing up a matrix between the processors in a geometric manner. Striped mapping is used throughout our implementation. An entire row or group of rows are issued to each processor. In striped mapping, a row is an atomic unit. As a result, there cannot be more processors than rows in the equation matrix.

A design decision was made not to implement the other type of mapping, checkerboard mapping. The justification for this is apparent by inspection of the serial pseudo-code above. Gaussian eliminations cascade in a row-by-row manner. Our mapping should logically follow the algorithm's topology in order to make a more intuitive and less communicative parallel design.

Checkerboard mapping divides the matrix into equal sized rectangles. The communications that update the preceding rectangles have to be very organized and do not flow intuitively from the algorithm. Also, checkerboard mapping has the advantage to allow more processors than there are rows. This was a feature that was not needed because of the relatively few processors available.

C. Pipelined Communication

Two different communication implementations are tested. They are broadcast and pipelined communication. In the broadcast method, when a row has completed the division step, it broadcasts its newly calculated values to all members. This means all processors must wait until all others receive the data.

With pipelined communication a row is sent to a processor who forwards it on to the next processor and then it can immediately begin its local computations. This quick point-to-point message allows a processor to do its local calculations *concurrently* while other processors are communicating.

In the pipelined version, there are three steps - normalization of a row, communication, and elimination. These steps are performed in an asynchronous fashion. A processor Pk waits to receive and eliminate all rows prior to k. Once it has done this, it forwards its own row to processor Pk+1.[3]

The total number of steps in the entire pipelined procedure is Θ(n). In any step, either O(n) elements are communicated between directly connected processes, or a division step is performed on O(n) elements of a row, or an elimination step is performed on O(n) elements of a row. n The parallel time is therefore O(n2) . n This is cost optimal.

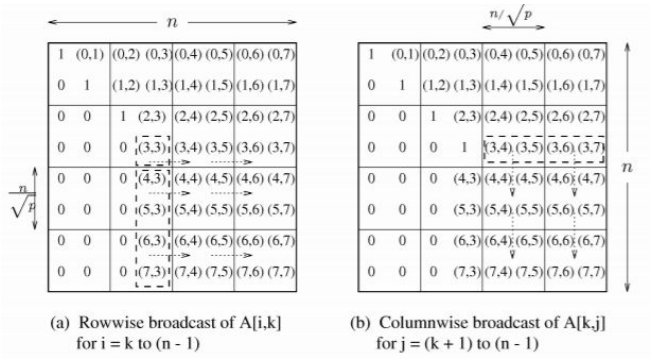

Fig. 2: 2-D Mapping with Pipelining. The communication steps in the Gaussian elimination iteration corresponding to k = 3 for an 8 x 8 matrix on 16 processes of a two dimensional mesh.[3]

D. Blocked vs. Cyclic Mapping

Blocked mapping is the intuitive method for assigning rows to processors. In blocked mapping, a processor is assigned a number of *contiguous* rows. The problem with this type of mapping can be seen if one views the characteristics of the computation. There are far more calculated values on the top rows than the bottom. This is a cause for a great deal of load-imbalance and processor idling time. The solution to this load-balancing problem is cyclic mapping.

Cyclic mapping divides the matrix rows into n/p sections, and then assigns a processor one row from each section. This greatly improves the load-balancing problem. Unfortunately, it complicates pipelined communication since the processor to start the chain is different at each iteration.

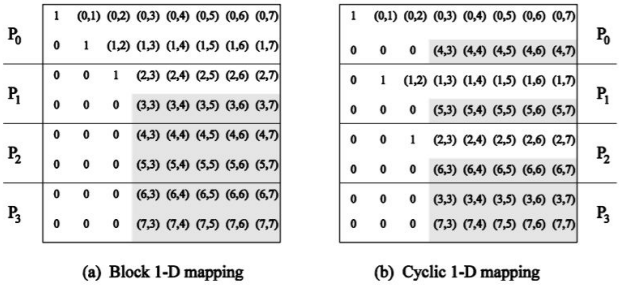

Fig 3: Computation load on different processes in block and cyclic 1-D partitioning of an 8 x 8 matrix on four processes during the Gaussian elimination iteration corresponding to k = 3. [3]

E. Identifying the parallel region

We have three possible candidate loops to parallelize.

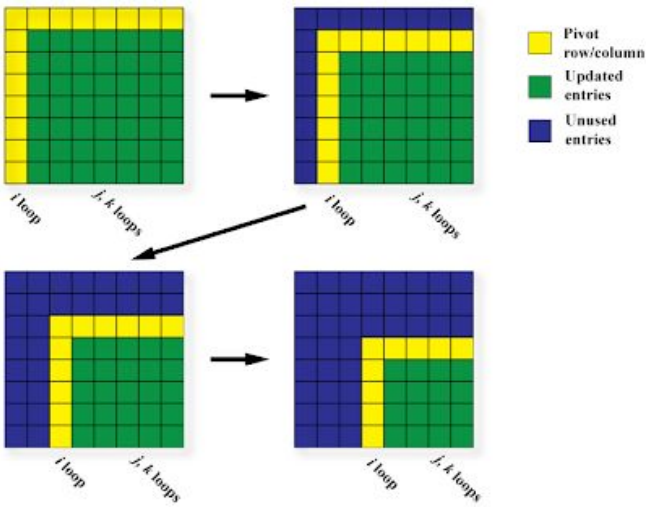

Fig 4: Shows the working of Gaussian Elimination[4]

1. 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.

2. 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.

3. 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.

References

[1] https://en.wikipedia.org/wiki/Gaussian_elimination/

[2] https://www.cs.rutgers.edu/~venugopa/parallel_summer2012/ge.html

[3] http://web.cse.msstate.edu/~luke/Courses/fl13/CSE4163/Slides/DPA_Matrix.pdf

[4] By Suriya Gharib, Syeda Roshana Ali, Rabia Khan, Nargis Munir & Memoona Khanam, "System of Linear Equations, Gaussian Elimination.", Global Journal of Computer Science and Technology: Software & Data Engineering ,Volume 15 Issue 5 Version 1.0 Year 2015. Online ISSN: 0975-4172 & Print ISSN: 0975-435