

Shared Memory Implementation of Gaussian Elimination

Gabriel Dimitriu

1 Introduction

Let the system be

$$Ax = b \quad (1)$$

Here A is a dense $n \times n$ matrix of coefficients such that $A[i][j] = a_{i,j}$, b is an $n \times 1$ vector and x is the desired solution vector.

A system of equations is solved in two stages. First, through a series of algebraic manipulations, the original system of equations is reduced to an upper triangular system of the form $Ux = y$, where U is a unit upper-triangular matrix. In the second stage of solving a system of linear equations, the upper-triangular system is solved for the variables in reverse order from x_{n-1} to x_0 by a procedure known as back-substitution.

Algorithm 1 *gaussian_elimination*

1. procedure gaussian_elimination(A, b, y)
2. begin
3. for $k := 0$ to $n-1$ do
4. for $j := k+1$ to $n-1$ do
5. $A[k][j] := A[k][j] / A[k][k]$; //division step
6. $y[k] := b[k] / A[k][k]$;
7. $A[k][k] := 1$;
8. for $i := k+1$ to $n-1$ do
9. for $j := k+1$ to $n-1$ do
10. $A[i][j] := A[i][j] - A[i][k] \times A[k][j]$; //elimination step
11. $b[i] := b[i] - A[i][k] \times y[k]$;
12. $A[i][k] := 0$;
13. endfor;
14. endfor;
15. end gaussian_elimination;

2 Pipeline Communication and Computation

The matrix is scattered to all processor so two consecutive row are in two consecutive processors.

During the k^{th} iteration processor P_k broadcast part of the k^{th} row of the matrix to processors P_{k+1}, \dots, P_{p-1} . Assume that the processors $P_0 \dots P_{p-1}$ are connected in a linear array, and P_{k+1} is the first processor to receive the k^{th} row from processor P_k . Then the processor P_{k+1} must forward this data to P_{k+2} . However, after forwarding the k^{th} row to P_{k+2} , processor P_{k+1} need not wait to perform the elimination step until all the processors up to P_{p-1} have received the k^{th} row. Similarly, P_{k+2} can start its computation as soon as it has forwarded the k^{th} row to P_{k+3} , and so on. Meanwhile, after completing the computation for the k^{th} iteration, P_{k+1} can perform the division step, and start the broadcast of the $(k+1)^{th}$ row by sending it to P_{k+2} . In this case of shared memory the receive, send and broadcast are made with mutex for pthread and lock for OpenMP, there is a m dimensional mutex vector who locked the rows and when a row is send the mutex is unlock, the original state of the mutex is locked.

3 Results

The Gaussian Elimination without load balancing and pivoting is compiled with Omni 1.6, an OpenMP compiler, and GCC with original pthread library from Fedora Core 1. Both are compiled with optimization "-O3" for maximum performance.

The programs ran on a dual pentium II at 500MHz with 256MB RAM and LINUX Fedora Core 1 and resulted the following graph. In the graph with red is plotted the results from Omni and with blue is plotted the results from pthread.

