SOLVING SYSTEM OF LINEAR EQUATIONS ON SHARED MEMORY COMPUTERS

Gabriel DIMITRIU, Felicia IONESCU
Politehnica University of Bucharest
gabriel@tech.pub.ro, fionescu@tech.pub.ro

Abstract: In this paper we present results of Jacobi with dominant rows and columns and Gaussian elimination. These implementations are done in OpenMP and pthdread for Linux. The algorithms implementation are a compromise between easier implementation and very powerful implementation like PLAPLAC.

Key words: shared memory, dual processors, OpenMP, thread, Omni

1. Introduction

In the recent years, dual processors with Intel processors have become the fastest growing choice for workstations not only in servers. The successes of dual processors have been mainly facilitated by the rapid advancement of microprocessor technologies; reduce costs and availability of dual processor motherboards. Typically, a workstation is composing of an Intel dual processors with Linux operating systems. To exploit this advantage of the workstations we can use OpenMP API and pthreads. We will present some high and low computational algorithms for solving a system of linear equations.

2. Background

Let be

$$AX = B \qquad (1)$$

our system of linear equations. Where A is a dense square matrix with dimension n, and let be A[i][j]=a_{ij} his elements. The solution of the system (X) is a vector with dimension n with x_i as components. The free term (B) is a vector with n dimension with b_j as components.

This system could be solved using one of the following methods: Gaussian Elimination if we don't have another information about the matrix A; Jacobi with dominant column if the matrix is column dominant and Jacobi with dominant rows if the matrix is row dominant.

Suppose $a_{ii} \neq 0$ for any $i \in \{1,\ldots,n\}$ then we can note

$$D = \begin{Bmatrix} a_{11} & \ldots & 0 \\ \vdots & \ldots & \vdots \\ 0 & \cdots & a_{nn} \end{Bmatrix}$$

and

$$D^{-1} = \begin{Bmatrix} 1/a_{11} & \cdots & 0 \\ \vdots & \cdots & \vdots \\ 0 & \cdots & 1/a_{nn} \end{Bmatrix}$$

2.1 Gaussian Elimination

The 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 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 know as back-substitution.

```
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]*A[k][j]; //elimination
    step
11. b[i]:=b[i]-A[i][k]*y[k];
12. A[i][k]:=0;
13. end for;
```

14. end for;
15. end procedure gaussian_elimination;

2.2 Jacobi with dominate rows

Using D and D^{-1} the system (1) could be transform in $D^{-1}AX = D^{-1}B$ or

$$(I-(I-D^{-1}A))X = D^{-1}B$$

This is equivalent with

$$(I-C)x = B \qquad (2)$$

Jacobi theorem

If

$$|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}|, \forall i \in 1,...,n \qquad (3)$$

Then the system (2) has the unique solution z and $\forall x^{(0)} \in \Re^n$ *the string* $(x^{(m)})_{m\in N}$ *with* $x^{(m+1)} = C * x^{(m)} + B$ *convert to z and it takes places the following relations:*

$$\left\|x^{(m)} - z\right\|_{\infty} \leq \frac{q}{1-q}\left\|x^{(m)} - x^{(m-1)}\right\|_{\infty} \qquad (4)$$

where

$$q = \max_{1\leq i\leq n} \sum_{j=1, j\neq i}^{n} \left|\frac{a_{ij}}{a_{ii}}\right| \qquad (5)$$

and

$$\|x\|_{\infty} = \max_{1\leq i\leq n} |x_i| \qquad (6)$$

2.3 Jacobi with dominate column

Using D and D^{-1} the system (1) could be transform in $AD^{-1}DX=B$ or $\left(I-\left(I-\underbrace{AD^{-1}}_{C}\right)\right)DX = B$ which is equivalent with

$$(I-C)\underbrace{DX}_{Y} = B \qquad (7)$$

Theorem

If $a_{jj} \neq 0$ *and*

$$a_{jj} > \sum_{i=1}^{n} a_{ij} \qquad (8)$$

For any $j \in \{1,\cdots,n\}$ *the w exists and is unique. So (I-C)w=B and the Jacobi method is convergent for the system (7) and the evaluation error for the original system is*

$$\left\|x^{(n)} - z\right\|_1 = \left\|D^{-1}(y^{(m)} - y)\right\|_1 \leq \frac{1}{\min_{1\leq j\leq n}|a_{jj}|}\frac{q}{1-q}\left\|y^{(m)} - y^{(m-1)}\right\|_1$$

Where

$$\|x\|_1 = \sum_{i=1}^{n} |x_i| \text{ and } q = \max_{1\leq j\leq m} \sum_{i=1, i\neq j}^{n} \left|\frac{a_{ij}}{a_{jj}}\right| < 1.$$

3. Design of the parallel algorithms

We want to obtain a high speedup defined by $\frac{Serial\ time}{Parallel\ time}$ in such type of problems. Because the system is dual processor we will use the pthread library from Linux and an OpenMP free compiler Omni 1.3.

3.1. Gaussian Elimination

The type of implemented algorithm is pipeline computation with communication presented in [1].

The matrix is cyclic striped scattered so two consecutive row are in two consecutive processors.

Processor P_k have to broadcast k^{th} row to all other processors. Assume that the processors $P_0,\ldots,P_{p-1}$ 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} doesn't need to 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 it is 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} row, 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}.

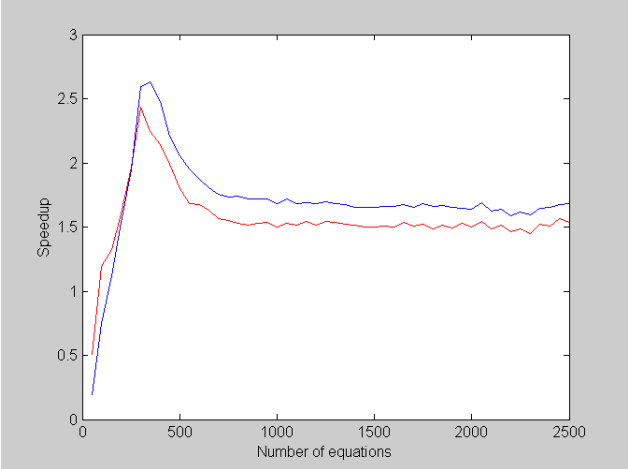
These facts conduct to the following behavior of for the each machine:

- If the machine has any data destined for other machine, it sends those data to the appropriate processor.
- If the machine can perform some computation, it does.
- Otherwise, the machine waits to receive data form other machines.

In this case of shared memory; the receive, send and broadcast are made with mutex for pthread and lock for

OpenMP. There is a n 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.

We present in the following figure the speedup from GCC with pthread with blue and with red the speedup from Omni.

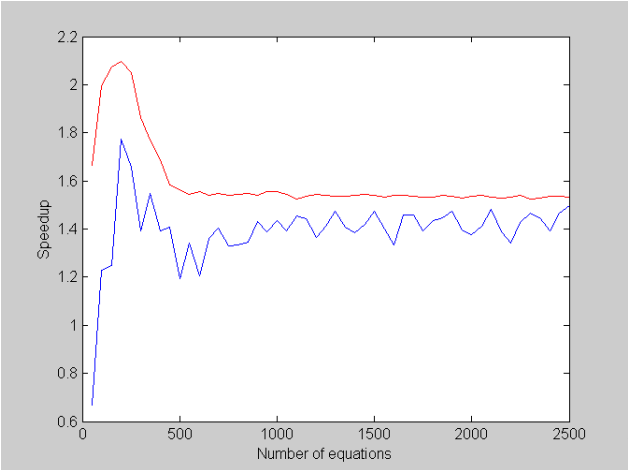


3.2. Jacobi with dominant rows

Because this problem doesn't have the data dependencies in the iterations except for computing the norm is not important haw we split the system, we chose the cyclic striped approach.

Each processor computes it's part of the maxim and we compute the last part in a single operation. We also compute the stop condition given by relation (4) in a single operation in each step.

We present in the following figure the speedup from GCC with pthread with blue and with red the speedup from Omni 1.3.

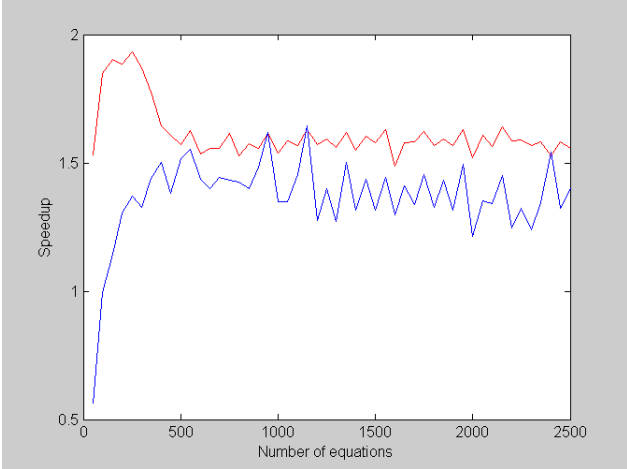


3.3. Jacobi with dominant column

Because this problem doesn't have the data dependencies in the iterations except for computing the norm is not important haw we split the system, we chose the cyclic striped approach.

Each processor computes it's part of the maxim and we compute the last part in a single operation. We also compute the stop condition given in the theorem in a single operation in each step.

We present in the following figure the speedup from GCC with pthread with blue and with red the speedup from Omni 1.3.



4. Measurements methodology and conclusions

The system consists in a dual Pentium II at 500 MHz with 256 MB RAM and Linux Suse 7.3 as Operating System.

The linear system is random generated only at startup of the benchmark and stored in a separate matrix. This is done because Gaussian Elimination modifies the matrix.

For the Gaussian Elimination the coefficient matrix is random generated with uniform distribution. We random generate only the coefficient matrix without the primary diagonal for the Jacobi methods. We generate the primary diagonal with relation (3) for Jacobi with dominant row and with relation (8) for Jacobi with dominant column.

If we randomly generate the free term, it is a chance that the system will be not compatible. So we chose the solution {1,2,3,…,n} and with this solution we generate the free term using the definition (1) of the linear system.

The time is given using operating system function gettimeofday before the start of the first distributions and after the tenth runs. The average time is then saved in a file for future analysis. We verify the correctitude of the solution with the desired solution after the tenth run.

The pthread implementations for Jacobi methods are slower than the OpenMP because of the implementation of the barrier function versus OMP_SINGLE.

It is seen that the speedup are acceptable for the Jacobi but is great for the Gaussian Elimination because this method is high computational.

REFERENCES

[1] V. Kumar, A. Grama, A. Gupta, G. Karypis, "Introduction to Parallel Computing. Design and Analysis of Algorithms", The Benjamin/Cummings Publishing Company, Redwood City, California, 1994.

[2] B.Demidovici, *Elements de calcul numerique,* Mir, 1973

[3] http://www.yl.is.s.u-tokyo.ac.jp/Omni