Solving System of Linear Equations in Parallel using Multithread

K. Rajalakshmi
Sri Parasakthi College for Women, Courtallam-627 802, Tirunelveli Dist., Tamil Nadu, India

Abstract - This paper deals with the efficient parallel algorithms for solving linear equations using multithread. Since using Multi thread, each thread represent one processor, execution time is minimized. Here the compare Time Complexity and Processor Complexity. Finally we conclude that the parallel algorithm performance better than the sequential algorithm.

Key Words - Parallel, Multithread, Crout’s Reduction, Sequential, Algorithm

I. INTRODUCTION
Solving the system of linear equations is probably one of the most visible applications of Linear Algebra. The system of linear equations appears in almost every branch of Science and Engineering. Many scientific and engineering problems can take the form of a system of linear equations. The system of linear equations has many applications such as Digital Signal Processing, Geometry, Networks, Temperature Distribution, Heat Distribution, Chemistry, Linear Programming, Games, Estimation, Weather Forecasting, Economics, Image Processing, Video Conferencing, Oceanography and many Statistical analysis (for example in Econometrics, Biostatistics and Experimental Design). The applications of specialized types are Transportation problems and Traffic Flow problems. These enquire special techniques for setting up the system of equations. The Linear equations can be used to find out the best method of Maximizing Profits. Other applications are Design and Computer analysis of Circuits, Power System Networks, Chemical engineering processes, Macroeconomics models, Queueing systems, Polynomial curve fitting, Medical etc. Parallel Algorithm as opposed to a traditional sequential algorithm, is one which can be executed a piece at a time on many different processing devices, and then put back together again at the end to get the correct result. Java is a popular language and has support for parallel execution that is integrated into the language. Hence it is interesting to investigate the usefulness of Java for executing Scientific Programs in Parallel. I have described one Method of Linear equations which can be solved in parallel using the thread mechanism of Java. Java let programs interleave multiple program steps through the use of threads. For example, one thread controls an animation, while another does a computation. On a multiprocessor or multi-core system, threading can be achieved via multiprocessing, wherein different threads and processes can run literally simultaneously on different processors or cores. Threads and processes differ from one operating system to another but, in general, a thread is contained inside a process and different threads in the same process share some resources while different processes do not.

We consider implementations of linear equation method and compare the result of performance. As platform, we use a Windows XP system. The rest of the paper is organized as follows: Solving linear equations by Crout’s Reduction Method, both Sequential and parallel Programming in Section II. Comparison of result in sequential execution and parallel execution of programs in Section III. Conclusion in Section IV.

II. LITERATURE REVIEW
An efficient Parallel Algorithm for the solution of a Tridiagonal linear system of equations was formed by H. S. Stone (1973). A new class of iterative methods for the numerical solution of linear Simultaneous equations which were applicable for use on parallel computers was introduced. Cetin K. Koc (1990) formed a Parallel Algorithm for exact solution of linear equations via Congruence technique. New Technique for efficient parallel solution of very large linear systems of equations on a SIMD processor was arrived by Okon Hanson Akpan in 1994. Which was implemented in parallel computer? His result was N > P. Where N is the problem size and P is the Number of Processor.

Solving Linear Equations Systems using parallel processing was attempted by Jorge C. pais, Rainundo M. Delgado. He made use of 2 to 16 processors to compute parallel execution because this work was developed in a parallel computer with 16 transputers IMS T800-20 every one with 2 Mega Bytes of RAM and also compared the efficiency.

A. CROUT’S Reduction METHOD

Let the system of equations to be solved be

\[ AX = B \]

In this method also, we decompose the coefficient matrix A as LU, where L is general lower triangular matrix and U is a unit upper triangular matrix with each of the leading diagonal elements equal to unity.

\[
L = \begin{bmatrix}
l_{11} & 0 & 0 & \ldots & 0 \\
l_{21} & l_{22} & 0 & \ldots & 0 \\
l_{31} & l_{32} & l_{33} & \ldots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
l_{n1} & l_{n2} & l_{n3} & \ldots & l_{nn}
\end{bmatrix}
\]

\[
U = \begin{bmatrix}
1 & u_{12} & u_{13} & \ldots & u_{1n} \\
0 & 1 & u_{23} & \ldots & u_{2n} \\
0 & 0 & 1 & \ldots & u_{3n} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \ldots & 1
\end{bmatrix}
\]

Using \( A = LU \) in (1), we get

\[
LUX = B
\]

\[
L^{-1} LUX = L^{-1} B
\]

\[
UX = C
\]

System (3) can easily solved for \( X \) by back substitution.

Thus Crout’s Reduction method expresses \( A \) in the form \( LU \) and solves the equivalent system \( UX = C \), instead of solving the given system \( AX = B \). the same operation of pre-multiplying by \( L^{-1} \), which reduces \( A \) to \( U \), also transforms \( B \) to \( C \). The procedure to decompose \( A \) to \( LU \), to get the elements of \( L \) and \( U \).

\[
\begin{bmatrix}
l_{11} & 0 & 0 & \ldots & 0 \\
l_{21} & l_{22} & 0 & \ldots & 0 \\
l_{31} & l_{32} & l_{33} & \ldots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
l_{n1} & l_{n2} & l_{n3} & \ldots & l_{nn}
\end{bmatrix}
\begin{bmatrix}
1 & u_{12} & u_{13} & \ldots & u_{1n} \\
0 & 1 & u_{23} & \ldots & u_{2n} \\
0 & 0 & 1 & \ldots & u_{3n} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \ldots & 1
\end{bmatrix}
= \begin{bmatrix}
a_{11} & a_{12} & a_{13} & \ldots & a_{1n} \\
a_{21} & a_{22} & a_{23} & \ldots & a_{2n} \\
a_{31} & a_{32} & a_{33} & \ldots & a_{3n} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
a_{n1} & a_{n2} & a_{n3} & \ldots & a_{nn}
\end{bmatrix}
\]

In this method, the order in which the elements of the triangular matrices computed is as follows:
First column element of \( L \), first row elements of \( U \), second elements of \( L \), second row elements of \( U \) and so on.

B. Implementation of Sequential Algorithm

Compute the elements of the first column of \( L \) (Lower Triangular) matrix.

For \( i = 1 \) to \( n \)

\[ \text{Aim} = b_i \]

\[ l_{ij} = a_{ij} \]

Next \( i \)

Compute the elements of the first row of \( U \) (Upper Triangular) matrix.

For \( j = 1 \) to \( n \)

\[ u_{ij} = a_{ij}/l_{jj} \]

Next \( j \)
Compute the elements of the successive columns of \( L \) (Lower Triangular Matrix) and those of successive rows of \( U \) (Upper Triangular Matrix) alternatively.

\[
\begin{pmatrix}
l_{11} & 0 & 0 & \ldots & 0 \\
l_{21} & l_{22} & 0 & \ldots & 0 \\
l_{31} & l_{32} & l_{33} & \ldots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
l_{n1} & l_{n2} & l_{n3} & \ldots & l_{nn}
\end{pmatrix}
\begin{pmatrix}
1 & u_{12} & u_{13} & \ldots & u_{1n} \\
0 & 1 & u_{23} & \ldots & u_{2n} \\
0 & 0 & 1 & \ldots & u_{3n} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \ldots & 1
\end{pmatrix}
\]

For \( m = 1 \) to \( n-1 \)
For \( i = m \) to \( n-1 \)
For \( k = 1 \) to \( m \)
\( \text{Aim} = a_{im} - l_{ik} \times u_{mp} \)
Next \( k \)
Next \( i \)

For \( j = m+1 \) to \( n \)
For \( k = 1 \) to \( m \)
\( \text{Am} = a_{m} - l_{ik} \times a_{uk} \)
Next \( k \)
Next \( j \)
Next \( m \)

Compute the values of the unknowns \( x_n, x_{n-1}, x_{n-2}, \ldots, x_1 \) by backward substitution.

for \( k = n \) to \( 1 \)
\( i = n-k+1 \)
\( x_i = u_{in} \)
next \( k \)

Sequential Algorithm (CRM)

Input: Size of the equation is \( n \)
Given matrix is \( a[1:n,1:n] \) and right hand side value is \( b[1:n] \)
Output: \( x[1:n] \)

1. for \( i = 1 \) to \( n \)
2. \( a_{in} = b_i \)
3. \( l_{ii} = a_{i1} \)
4. next \( i \)
5. for \( j = 1 \) to \( n \)
6. \( u_{ij} = a_{ij} / l_{jj} \)
7. next \( j \)
8. for \( m = 1 \) to \( n-1 \)
9. for \( i = m \) to \( n-1 \)
10. for \( k = 1 \) to \( m \)
\( \text{Aim} = a_{im} - l_{ik} \times u_{im} \)
11. next \( k \)
12. \( l_{im} = a_{im} \)
13. next \( i \)
14. for \( j = m+1 \) to \( n \)
15. for \( k = 1 \) to \( m \)
\( \text{Am} = a_{mj} - l_{mk} \times u_{kj} \)
16. next \( j \)
17. next \( m \)
18. next k
19. \( u_{mj} = a_{mj} / b_{mm} \)
20. next j
21. next m
22. for \( k = 1 \) to \( n \)
23. \( i = n - k \)
24. \( x_i = u_{im} \)
25. for \( j = i + 1 \) to \( n - l \)
26. \( x_i = x_i - u_{ij} \times x_j \)
27. next j
28. next k
29. end

Implementation of Parallel Algorithm
Compute the elements of the first column of Lower triangular matrix and the first row of Upper triangular matrix.
Compute the elements of the successive columns of Lower Triangular matrix and the elements of the successive rows of Upper triangular matrix are calculated in parallel.
Compute the values of unknowns by back Substitution.

Parallel Algorithm (CRM)
Input: Size of the equation is \( n \)
Given Matrix is \( a[1:n,1:n+1] \)
Output: \( x \) values \( x[1:n] \)

1. for \( i = 1 \) to \( n \)
2. \( b_{i1} = a_{i1} \)
3. next i
4. for \( j = 1 \) to \( n \)
5. \( u_{ij} = a_{ij} / b_{11} \)
6. next j
7. for \( m = 1 \) to \( n - l \) do in parallel
8. for \( i = m \) to \( n - l \) do in parallel
9. for \( k = 1 \) to \( m \)
10. \( a_{im} = a_{im} - b_{ik} \times u_{km} \)
11. next k
12. \( b_{im} = a_{im} \)
13. end parallel
14. for \( j = m + 1 \) to \( n \)
15. for \( k = 1 \) to \( m \)
16. \( a_{mj} = a_{mj} - b_{mk} \times u_{kj} \)
17. next k
18. \( u_{mj} = a_{mj} / b_{mm} \)
19. next j
20. end parallel
21. for \( k = 1 \) to \( n \)
22. \( i = n - k - 1 \)
23. \( x_i = u_{im} \)
24. for \( j = i + 1 \) to \( n - l \)
25. \( x_i = x_i - u_{ij} \times x_j \)
26. next j
27. next k
28. end

III. RESULTS
The analysis of the algorithms will take into consideration the following issues
- The time complexity of the Sequential algorithm.
- The time Complexity of the parallel algorithm.
IV. CONCLUSION

From the above results and observations, we come to the following conclusions. The Crout’s Reduction Method developed in this study is a very effective tool for efficient parallel execution of linear systems. The system size (size of the Equations) N is equal to the number of processor. That is, N = P, where P is the Processors. The system performance of parallel execution, more effective than a serial execution of a system.

REFERENCES