Parallel Algorithm for Solving Large System of Simultaneous Linear Equations

K. RAJALAKSHMI

Sri Parasakthi College for Women, Courtallam – 627 802, Tirunelveli Dist. Tamil Nadu, India

1. Introduction

Solving systems of linear equation is probably one of the most visible applications of Linear Algebra. System of linear equations appear in almost every branch of Science and Engineering. Many scientific and engineering problems can take the form of a system of linear equations.

System of linear equations have 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 (example in Econometrics, Biostatistics and Experimental Design).

Java is a popular language and has support for a 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 two methods of linear equations which can be solved in parallel using the thread mechanism of Java.

We consider implementations of the linear equation methods and compare the result of performance. As platform, we use a windows XP system.

The rest of the paper is organized as follows: section I Solving linear equations by Gauss Elimination method , both sequential and parallel programming and Comparison of the results in sequential execution and parallel execution of program. Section II Solving linear equations by Gauss Jordan method both sequential and parallel programming and Comparison of the results in sequential execution and parallel execution of program. Section III Conclusion.

2. Gauss Elimination Method

This method is based on the elimination of the unknowns by combining equations such that the n equations in n unknowns are reduced to an equivalent upper triangular system which could be solved by back substitution.

Consider the n linear equations in n unknowns

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n &= a_{1,n+1} \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n &= a_{2,n+1} \\ a_{31}x_1 + a_{32}x_2 + \dots + a_{3n}x_n &= a_{3,n+1} \\ \dots \quad \dots \quad \dots \quad \dots \quad & \dots \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n &= a_{n,n+1} \end{aligned}$$

Where a_{ij} and $a_{i,j+1}$ are known constant and x_i's are unknowns.

The system (1) is equivalent to

$$AX = B$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \dots \\ x_n \end{pmatrix} = \begin{pmatrix} a_{1,n+1} \\ a_{2,n+1} \\ a_{3,n+1} \\ \dots \\ a_{n,n+1} \end{pmatrix}$$

Step 1 : Store the coefficients in an augmented matrix. The superscript on a_{ij} means that this is the first time that a number is stored in location (i, j).

$$\left(\begin{array}{ccccc|c} a_{11} & a_{12} & a_{13} & \dots & a_{1n} & a_{1,n+1} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} & a_{2,n+1} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} & a_{3,n+1} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} & a_{n,n+1} \end{array}\right)$$

Step 2 : If necessary, switch rows so that $a_{11} \neq 0$, then eliminate x_1 in row2 through n . In this process m_{i1} is the multiple of row1 that is subtracted from row i .

```
for i = 2 to n
        m_i1 = a_i1 / a_11
        a_i1 = 0
        for j = 2 to n+1
                a_ij = a_ij – m_i1 * a_1j
        end for
end for
```

The new elements are written a_{ij} to indicate that this is the second time that a number has been stored in the matrix at location (i , j) . The result after step 2 is

$$\left(\begin{array}{ccccc|c} a_{11} & a_{12} & a_{13} & \dots & a_{1n} & a_{1,n+1} \\ 0 & a_{22} & a_{23} & \dots & a_{2n} & a_{2,n+1} \\ 0 & a_{32} & a_{33} & \dots & a_{3n} & a_{3,n+1} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & a_{n2} & a_{n3} & \dots & a_{nn} & a_{n,n+1} \end{array}\right)$$

Manuscript received July 5, 2009
Manuscript revised July 20, 2009

Step 3 : If necessary , switch the second row with some row below it so that $a_{22} \neq 0$, then eliminate x_2 in row 3 through n. In this process u_{i2} is the multiple of row 2 that is subtracted from row i .

```
for i = 3 to n
        u_i2 = a_i2 / a_22
        a_i2 = 0
        for j = 3 to n+1
                a_ij = a_ij – u_i2 * a_2j
        end for
end for
```

The new elements are written a_{ij}'' indicate that this is the third time that a number has been stored in the matrix at location (i , j) . The after step 3 is

$$\left(\begin{array}{ccccc|c} a_{11} & a_{12} & a_{13} & \dots & a_{1n} & a_{1,n+1} \\ 0 & a_{22} & a_{23} & \dots & a_{2n} & a_{2,n+1} \\ 0 & 0 & a_{33} & \dots & a_{3n} & a_{3,n+1} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & a_{n3} & \dots & a_{nn} & a_{n,n+1} \end{array}\right)$$

Step k+1 : This is the general step. If necessary, switch row k with some row beneath it so that a_{kk} != 0; then eliminate x_k in rows k+1 through n . Here u_{ik} is the multiple of row k that is subtracted from row i .

```
for i = k +1 to n
        u_ik = a_ik / a_kk
        a_ik = 0
        for j = k + 1 to n+1
                a_ij = a_ij – u_ik * a_kj
        end for
end for
```

The final result after x_{n-1} has been eliminated from row n is

$$\left(\begin{array}{ccccc|c} a_{11} & a_{12} & a_{13} & \dots & a_{1n} & a_{1,n+1} \\ 0 & a_{22} & a_{23} & \dots & a_{2n} & a_{2,n+1} \\ 0 & 0 & a_{33} & \dots & a_{3n} & a_{3,n+1} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & a_{nn} & a_{n,n+1} \end{array}\right)$$

The upper trianglarization process is now complete $x_n = a_{n,n+1} / a_n$

```
for i = n to 1 step -1
      sum = 0
      for j = i+1 to n
            sum = sum + a_ij * x_j
      end for
      x_i = ( a_i,n+1 - sum )/a_ii
end for
```

Perform the back substitution , get the values of x_n, x_{n-1}, x_{n-2}, . . . x_1.

Sequential Algorithm – Gauss Elimination Method

Input : Given Matrix a[1 : n, 1: n+1]
Output : x[1 : n]

1. for k = 1 to n-1
2. for i = k+1 to n
3. u = a_{ik}/a_{kk}
4. for j = k to n+1
5. $a_{ij} = a_{ij} – u * a_{kj}$
6. next j
7. next i
8. next k
9. $x_n = a_{n,n+1}/a_{nn}$
10. for i = n to 1 step -1
11. sum = 0
12. for j = i+1 to n
13. sum = sum + $a_{ij} * x_j$
14. next j
15. $x_i = (a_{i,n+1} - sum)/a_{ii}$
16. next i
17. end

Parallel Algorithm for Gauss Elimination Method

In the Parallel execution using the Multi thread mechanism . If the size of the linear equation is n, n processors are used. Each thread represent one processor. In the Parallel execution processing time is less compared to sequential execution.

```
for k = 1 to n-1
        for i = k+1 to n do in parallel
 u = a_ik/a_kk
for j = k to n+1 do in parallel
a_ij = a_ij – u * a_kj
end parallel
end parallel
next k
```

In the Gauss elimination method lower triangular matrix elements are zero which are calculated in parallel.

Parallel Algorithm – Gauss Elimination Method

Input : Given Matrix a[1 : n, 1 : n+1]
Output : x[1 : n]

1. for k = 1 to n-1
2. for i = k+1 to n do in parallel
3. u = a_{ik}/a_{kk}
4. for j = k to n+1 do in parallel
5. $a_{ij} = a_{ij} – u * a_{kj}$
6. end parallel
7. end parallel
8. next k
9. $x_n = a_{n,n+1}/a_{nn}$
10. for i = n to 1 step -1
11. sum = 0
12. for j = i+1 to n

13. sum = sum + a_{ij} * x_j
14. next j
15. x_i = ($a_{i,n+1}$ - sum)/a_{ii}
16. next i
17. end

Compare the Execution Time in Gauss Elimination Method

Size of the Equations	Execution Time (ms) in Sequential	Execution Time (ms) in Parallel
5	15	2
10	26	3
15	29	5
20	45	7

3. Gauss Jordan Method

This method based on the elimination of the unknowns by combining equations such that the n equations in n unknowns are reduced to an equivalent upper triangular system which could be solved by back substitution.
Consider the n linear equations in n unknowns

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n &= a_{1,n+1} \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n &= a_{2,n+1} \\ a_{31}x_1 + a_{32}x_2 + \dots + a_{3n}x_n &= a_{3,n+1} \\ \dots \quad \dots \quad \dots \quad \dots \quad & \dots \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n &= a_{n,n+1} \end{aligned}$$

where a_{ij} and $a_{i,j+1}$ are known constant and x_i's are unknowns.
The system (1) is equivalent to

$$AX = B$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \dots \\ x_n \end{pmatrix} = \begin{pmatrix} a_{1,n+1} \\ a_{2,n+1} \\ a_{3,n+1} \\ \dots \\ a_{n,n+1} \end{pmatrix}$$

Step 1 : Store the coefficients in an augmented matrix. The superscript on a_{ij} means that this is the first time that a number is stored in location (i, j).

$$\left(\begin{array}{ccccc|c} a_{11} & a_{12} & a_{13} & \dots & a_{1n} & a_{1,n+1} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} & a_{2,n+1} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} & a_{3,n+1} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} & a_{n,n+1} \end{array}\right)$$

step 2 : If necessary . switch rows so that $a_{11} \neq 0$, then eliminate x_1 in row2 through n . In this process mi_1 is the multiple of row1 that is subtracted from row i .

```
for i = 2  to  n
        u_i1 a_i1 / a_11
        a_i1  = 0
        for j = 2 to n+1
            a_ij  = a_ij – u_i1 * a_1j
        end for
    end for
```

The new elements are written a_{ij} to indicate that this is the second time that a number has been stored in the matrix at location (i , j) . The result after step 2 is

$$\left(\begin{array}{ccccc|c} a_{11} & a_{12} & a_{13} & \dots & a_{1n} & a_{1,n+1} \\ 0 & a_{22} & a_{23} & \dots & a_{2n} & a_{2,n+1} \\ 0 & a_{32} & a_{33} & \dots & a_{3n} & a_{3,n+1} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & a_{n2} & a_{n3} & \dots & a_{nn} & a_{n,n+1} \end{array}\right)$$

step 3 : If necessary , switch the second row with some row below and above it so that $a_{22} \neq 0$, then eliminate x_2 in row 3 through n and also eliminates row 1. In this process u_{i2} is the multiple of row 2 that is subtracted from row i .

```
for i = 3 to n
    for j  =  i-2  to  n+1
        if ( i ≠  j ) then
            u_i2 = a_i2 / a_22
            a_ij  = a_ij – u_i2 * a_2j
        end if
    end for
end for
```

The new elements are written $a_{ij'}$ indicate that this is the third time that a number has been stored in the matrix at location (i , j) . The after step 3 is

$$\left(\begin{array}{ccccc|c} a_{11} & 0 & a_{13} & \dots & a_{1n} & a_{1,n+1} \\ 0 & a_{22} & a_{23} & \dots & a_{2n} & a_{2,n+1} \\ 0 & 0 & a_{33} & \dots & a_{3n} & a_{3,n+1} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & a_{n3} & \dots & a_{nn} & a_{n,n+1} \end{array}\right)$$

step k+1 : This is the general step. If necessary, switch row k with some row beneath it so that $a_{kk} \neq 0$; then eliminate x_k in rows 1 . . . k-1 and k+1 through n except k . Here u_{ik} is the multiple of row k that is subtracted from row i .

```
for  i  =  k +1  to  n
    for j  =  i+1   to   n+1
        if ( i ≠  j ) then
            u_ik = a_ik / a_kk
            a_ij  = a_ij – u_ik * a_kj
        end if
    end for
end for
```

The final result after x_n has been eliminated from row n is

$$\left(\begin{array}{ccccc|c} a_{11} & 0 & 0 & \dots & 0 & a_{1,n+1} \\ 0 & a_{22} & 0 & \dots & 0 & a_{2,n+1} \\ 0 & 0 & a_{33} & \dots & 0 & a_{3,n+1} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & a_{nn} & a_{n,n+1} \end{array}\right)$$

The diagonal matrix is now complete , get the values of $x_n, x_{n-1}, x_{n-2}, \dots x_1$.

Sequential Algorithm – Gauss Jordan Method

Input : Given Matrix a[1 : n, 1 : n+1]
Output : x[1 : n]
1. for i = 1 to n
2. for j = 1 to n+1
3. if ($i \neq j$) then
4. u = a_{ik}/a_{kk}
5. for k = 1 to n+1
6. $a_{jk} = a_{jk} - u * a_{ik}$
7. end if
8. next k
9. next j
10. next i
11. for i = 1 to n
12. $x_i = a_{i,n+1}/a_{ii}$
13. end

Parallel Algorithm for Gauss Jordan Method

In the Gauss Jordan Method, if the size of the linear equations is n, n processors are used. In the Multi thread mechanism, Each thread represent one processor.

for i = 1 to n do in parallel
for j = 1 to n+1 do in parallel
if ($i \neq j$) then
u = a_{ik}/a_{kk}
for k = 1 to n+1
$a_{jk} = a_{jk} - u * a_{ik}$
end if
next k
end parallel
end parallel

Parallel Algorithm – Gauss Jordan Method

Input : Given Matrix a[1 : n, 1 : n+1]
Output : x[1 : n]
1. for i = 1 to n do in parallel
2. for j = 1 to n+1 do in parallel
3. if ($i \neq j$) then
4. u = a_{ik}/a_{kk}
5. for k = 1 to n+1
6. $a_{jk} = a_{jk} - u * a_{ik}$
7. end if
8. next k
9. end parallel
10. end parallel
11. for i = 1 to n
12. $x_i = a_{i,n+1}/a_{ii}$
13. end

Compare the Execution Time in Gauss Jordan Method

Size of the Equations	Execution Time (ms) in Sequential	Execution Time (ms) in Parallel
5	16	3
10	27	5
15	31	6
20	47	8

Conclusion

The solving system of linear equation method described in this paper deals with sequential algorithm and parallel algorithm. Parallel algorithm has good speedups and less time complexity than sequential algorithm. Similarly Gauss Elimination method is better than Gauss Jordan method.

Acknowledgments

I would like to record my heartful thanks to Dr. C. Xavier, my guide in rendering full support for the submission of the paper.

References

[1] John H. Mathews and Kurtis D. Fink, Numerical Methods using MATLAB, Fourth Edition, Pearson Education (Singapore) Private Limited (2005).
[2] S.R.K. Iyengar and R.K. Jain, Mathematical Methods, Narosa Publishing House Private. Limited (2006).
[3] M.K.Jain, S.R.K. Iyengar and R.K. jain , Numerical Methods for Scientific and Engineering Computation, Third Edition, New age International (P) Limited. (1995).
[4] Jaan Kiusalaas , Numerical methods in Engineering ith Python, Cambridge University Press.
[5] C.Xavier and S.S. Iyengar , Introduction to Parallel Algorithms, A Wiley Inter-Science publication (1998).
[6] Doug Lea , Concurrent programming in Java – Design Principles and Patterns, Second Edition, Addison-Wesley (2000).
[7] Paul Hyde , Java Thread Programming – The Authoritative Solution , A Division of Macmillan Computer Publishing (1999).
[8] Michael J. Quinn , Parallel Computing – Theory and Practice, Second Edition, Tata McGraw-Hill Publishing Company Limited .
[9] Michael J. Quinn, Designing Efficient Algorithms for Parallel Computers, McGraw-Hill Book Company (1987).
[10] http://www.tandf.co.uk/journals/titles/10637192.html
[11] http://www.cs.emu.edu/~guyb/papers/BM04.pdf