PARALLEL TRI-DIAGONAL MATRIX SOLVER USING MPI

Final Presentation –

CSE 708 – Dr. Russ Miller

Paramveer Singh

University at Buffalo The State University of New York

Introduction

Many technical and scientific problems involve the solution of linear systems of equations:

$$QX = Y,$$

where Q is structured as a block tri-diagonal matrix of order n:

$$Q = \begin{bmatrix} a_1 & b_1 & 0 & \dots & 0 & 0 & 0 \\ c_2 & a_2 & b_2 & \dots & 0 & 0 & 0 \\ & & & \ddots & & & \\ 0 & 0 & 0 & \dots & c_{n-1} & a_{n-1} & b_{n-1} \\ 0 & 0 & 0 & \dots & 0 & c_n & a_n \end{bmatrix}, \qquad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}, \qquad y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix}.$$

Introduction – What is a tri-diagonal Matrix ??

A Tri-diagonal matrix is a matrix that has non zero elements on the main diagonal, the sub-diagonal and super-diagonal.

Considering a 4 X 4 Matrix

$$\begin{bmatrix} a_1 & b_1 & 0 & 0 \\ c_2 & a_2 & b_2 & 0 \\ 0 & c_3 & a_3 & b_3 \\ 0 & 0 & c_4 & a_4 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 4 & 0 & 0 \\ 3 & 4 & 1 & 0 \\ 0 & 2 & 3 & 4 \\ 0 & 0 & 1 & 3 \end{bmatrix}$$

Source:Wikipedia

Sequential Method

The Thomas algorithm is an efficient way of solving tridiagonal matrix systems.

It involves following steps :-

1) Forward Elimination
2) Backward Substitution

Sequential Method

1) Forward Elimination

Iterate through each row in a forward elimination phase, eliminating x1, x2 …. xn-1. The last equation will just involve one unknown xn. The coefficients at each iteration are calculated as :-

$$a'_i = 0 \qquad b'_i = 1 \qquad c'_1 = \frac{c_1}{b_1}$$

$$c'_i = \frac{c_i}{(b_i - c'_{i-1}a_i)} \qquad y'_1 = \frac{y_1}{b_1} \qquad y'_i = \frac{y_i - y'_{i-1}a_i}{(b_i - c'_{i-1}a_i)}$$

$$\begin{bmatrix} b_1 & c_1 & 0 & 0 & 0 \\ a_1 & b_2 & c_2 & 0 & 0 \\ 0 & a_2 & b_3 & c_3 & 0 \\ 0 & 0 & a_3 & b_4 & c_4 \\ 0 & 0 & 0 & a_4 & b_5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix}$$

Sequential Method

2) Backward Substitution

$$\begin{bmatrix} b_1 & c_1 & 0 & 0 & 0 \\ a_1 & b_2 & c_2 & 0 & 0 \\ 0 & a_2 & b_3 & c_3 & 0 \\ 0 & 0 & a_3 & b_4 & c_4 \\ 0 & 0 & 0 & a_4 & b_5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix}$$

Iterate back through the matrix solving for x at each row

$$x_i + c\prime_i x_{i+1} = y\prime_i \qquad i = 1 \ldots n-1$$

$$x_n = y\prime_n \qquad i = n$$

The complexity of is O(n)

Simple Example

Let us consider the system of equations

$$3x_1 - x_2 + 0x_3 = -1$$
$$-x_1 + 3x_2 - x_3 = 7$$
$$0x_1 - x_2 + 3x_3 = 7$$

Matrix form is

$$\begin{bmatrix} 3 & -1 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ 7 \\ 7 \end{bmatrix}$$

- Row 1

$$\mathbf{3x_l - x_2 = -1}$$

Divide the Equation by a_1, in this case $a_1 = 3$

$$\Rightarrow \mathbf{x_1 - \frac{1}{3}x_2 = \frac{-1}{3}}$$

Assuming the coefficient of x_2 as γ_1 and the remaining constants as ρ_1.
Now the equations converts to,

$$\Rightarrow \mathbf{x_1 + \gamma_1 x_2 = \rho_1}$$

Simple Example

- Row 2

Multiplying a_2 (-1) in Row 1 and eliminating x_1 Row 2

Row 2	$-x_1 + 3x_2 - x_3 = 7$
a_2 x Row 1	$-x_1 - \gamma_1 x_2 - 0x_3 = -\rho_1$
Subtracting	$x_2(3+\gamma_1) - x_3 = 7 + \rho_1$

$$x_2(3+\gamma_1) - x_3 = 7 + \rho_1$$

Divide by $(3+\gamma_1)$,

Equation becomes $\boldsymbol{x_2 + \gamma_2 x_3 = \rho_2}$

$$\boldsymbol{\gamma}_2 = \frac{-1}{3+\gamma_1} = 0.375 \qquad \boldsymbol{\rho}_2 = \frac{7+\rho_1}{3+\gamma_1} = 2.5$$

Simple Example

- Row 3

Multiplying a_3 (-1) in Row 1 and eliminating x_2 Row 3

Row 3	$-x_2 + 3x_3 = 7$
a_3 x Row 2	$-x_2 - \gamma_2 x_3 = -\rho_1$
Subtracting	$(3+\gamma_2)x_3 = 7 + \rho_2$

$$x_3 = \frac{7+\rho_2}{3+\gamma_2} \Rightarrow \rho_3$$

$$\boldsymbol{\rho}_3 = \frac{7+\rho_2}{3+\gamma_2} = 3.619 \Rightarrow x_3 = 3.619$$

Simple Example

STAGE II -> Backward Substitution

$$\begin{bmatrix} 1 & \gamma_1 & 0 \\ 0 & 1 & \gamma_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \rho_1 \\ \rho_2 \\ \rho_3 \end{bmatrix}$$

Row 2:

$$x_2 + \gamma_2 x_3 = \rho_2$$

$$x_2 = \rho_2 - \gamma_2 x_3$$

Substituting

$$\rho_2 = 2.5$$

$$\gamma_2 = -0.375$$

$$x_2 = 3.857$$

Simple Example

STAGE II -> Backward Substitution

$$\begin{bmatrix} 1 & \gamma_1 & 0 \\ 0 & 1 & \gamma_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \rho_1 \\ \rho_2 \\ \rho_3 \end{bmatrix}$$

Row 1:

$$x_1 + \gamma_1 x_2 = \rho_1$$

$$x_1 = \rho_1 - \gamma_1 x_2$$

Substituting

$\rho_1 = -0.333$

$\gamma_1 = -0.333$

$x_1 = 0.952$

Pseudo Code -

Input : matrix size: N, Vectors $a[]$, $b[]$, $c[]$ representing the sub-diagonal, main-Diagonal and super-Diagonal of the matrix A and the forcing vector $d[]$ here $i = 0, 1, ..., N-1$

Output: row vector $d[i]$ here $i = 0, 1, 2, ..., N-1$

$d_0^* \longleftarrow d_0 \,/\, b_0$

$c_0^* \longleftarrow c_0 \,/\, b_0$

for $i = 1, 2...N-1$ **do**

  $r \longleftarrow 1 \,/\, (b_i - a_i c_{i-1})$

  $d_i^* \longleftarrow r(d_i - a_i d_{i-1})$

  $c_i^* \longleftarrow r c_i$

end

for $i = (N-2)...1$ **do**

  $d_i \longleftarrow d_i^* - c_i^* d_{i+1}$

end

$$\begin{bmatrix} b_1 & c_1 & 0 & 0 & 0 \\ a_1 & b_2 & c_2 & 0 & 0 \\ 0 & a_2 & b_3 & c_3 & 0 \\ 0 & 0 & a_3 & b_4 & c_4 \\ 0 & 0 & 0 & a_4 & b_5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix}$$

source: wikipedia

Parallel Method

Given a tri-diagonal system of equations of a given size N. We divide this system among P cores such that each core stores a system of size m (m = N/P).

$$
\left(\begin{array}{cccc|cccc|cccc}
b_1 & c_1 & & & & & & & & & & \\
a_2 & b_2 & c_2 & & & & & & & & & \\
 & a_3 & b_3 & c_3 & & & & & & & & \\
 & & a_4 & b_4 & c_4 & & & & & & & \\
\hline
 & & & a_5 & b_5 & c_5 & & & & & & \\
 & & & & a_6 & b_6 & c_6 & & & & & \\
 & & & & & a_7 & b_7 & c_7 & & & & \\
 & & & & & & a_8 & b_8 & c_8 & & & \\
\hline
 & & & & & & & a_9 & b_9 & c_9 & & \\
 & & & & & & & & a_{10} & b_{10} & c_{10} & \\
 & & & & & & & & & a_{11} & b_{11} & c_{11} \\
 & & & & & & & & & & a_{12} & b_{12}
\end{array}\right)
\left(\begin{array}{c}
u_1 \\ u_2 \\ u_3 \\ u_4 \\ \hline u_5 \\ u_6 \\ u_7 \\ u_8 \\ \hline u_9 \\ u_{10} \\ u_{11} \\ u_{12}
\end{array}\right)
=
\left(\begin{array}{c}
d_1 \\ d_2 \\ d_3 \\ d_4 \\ \hline d_5 \\ d_6 \\ d_7 \\ d_8 \\ \hline d_9 \\ d_{10} \\ d_{11} \\ d_{12}
\end{array}\right)
$$

$$0 \leq j \leq P - 1.$$

Parallel Method

Step 1 –

This step takes place at a particular core j such that $0 \leq j \leq P-1$

$d_1^* \longleftarrow d_1 \,/\, b_1;\; c_1^* \longleftarrow c_1 \,/\, b_1;\; a_1^* \longleftarrow a_1 \,/\, b_1$

$d_2^* \longleftarrow d_2 \,/\, b_2;\; c_2^* \longleftarrow c_2 \,/\, b_2;\; a_2^* \longleftarrow a_2 \,/\, b_2$

for $i = 3 \ldots m$ **do**

- $r \longleftarrow 1 \,/\, (b_i - a_i c_{i-1})$
- $d_i \longleftarrow r(d_i - a_i d_{i-1})$
- $c_i \longleftarrow r c_i$
- $a_i \longleftarrow -r(a_i a_{i-1})$

end

for $i = (m-2) \ldots 2$ **do**

- $d_i \longleftarrow d_i - c_i d_{i+1}$
- $c_i \longleftarrow -c_i c_{i+1}$
- $a_i \longleftarrow a_i - c_i a_{i+1}$

end

$d_1 \longleftarrow r(d_i - a_i d_{i-1})$; $c_1 \longleftarrow -r c_1 c_2$; $a_1 \longleftarrow r a_1$

$$
\left(\begin{array}{cccc|cccc|cccc}
\mathbf{1} & & & \mathbf{c}_1^* & & & & & & & & \\
a_2^* & 1 & & c_2^* & & & & & & & & \\
a_3^* & & 1 & c_3^* & & & & & & & & \\
\mathbf{a}_4^* & & & \mathbf{1} & \mathbf{c}_4^* & & & & & & & \\
\hline
 & & & \mathbf{a}_5^* & \mathbf{1} & & & \mathbf{c}_5^* & & & & \\
 & & & & a_6^* & 1 & & c_6^* & & & & \\
 & & & & a_7^* & & 1 & c_7^* & & & & \\
 & & & & \mathbf{a}_8^* & & & \mathbf{1} & \mathbf{c}_8^* & & & \\
\hline
 & & & & & & & \mathbf{a}_9^* & \mathbf{1} & & & \mathbf{c}_9^* \\
 & & & & & & & & a_{10}^* & 1 & & c_{10}^* \\
 & & & & & & & & a_{11}^* & & 1 & c_{11}^* \\
 & & & & & & & & \mathbf{a}_{12}^* & & & \mathbf{1}
\end{array}\right)
\left(\begin{array}{c}
\mathbf{u}_1 \\ u_2 \\ u_3 \\ \mathbf{u}_4 \\ \hline \mathbf{u}_5 \\ u_6 \\ u_7 \\ \mathbf{u}_8 \\ \hline \mathbf{u}_9 \\ u_{10} \\ u_{11} \\ \mathbf{u}_{12}
\end{array}\right)
=
\left(\begin{array}{c}
\mathbf{d}_1^* \\ d_2^* \\ d_3^* \\ \mathbf{d}_4^* \\ \hline \mathbf{d}_5^* \\ d_6^* \\ d_7^* \\ \mathbf{d}_8^* \\ \hline \mathbf{d}_9^* \\ d_{10}^* \\ d_{11}^* \\ \mathbf{d}_{12}^*
\end{array}\right)
$$

Parallel Method

Step 2 – Forward Communication:

Collect coefficients a_i, b_i, c_i and d_i for $i = 1$ and m from each core

Step 3 – Solving the reduced system :

Obtain The solution for the reduced system with the help of the Thomas Algorithm

$$
\left(\begin{array}{cccc|cccc|cccc}
\mathbf{1} & & & \mathbf{c}_1^* & & & & & & & & \\
a_2^* & 1 & & c_2^* & & & & & & & & \\
a_3^* & & 1 & c_3^* & & & & & & & & \\
\mathbf{a}_4^* & & & \mathbf{1} & \mathbf{c}_4^* & & & & & & & \\
\hline
 & & & \mathbf{a}_5^* & \mathbf{1} & & & \mathbf{c}_5^* & & & & \\
 & & & & a_6^* & 1 & & c_6^* & & & & \\
 & & & & a_7^* & & 1 & c_7^* & & & & \\
 & & & & \mathbf{a}_8^* & & & \mathbf{1} & \mathbf{c}_8^* & & & \\
\hline
 & & & & & & & \mathbf{a}_9^* & \mathbf{1} & & & \mathbf{c}_9^* \\
 & & & & & & & & a_{10}^* & 1 & & c_{10}^* \\
 & & & & & & & & a_{11}^* & & 1 & c_{11}^* \\
 & & & & & & & & \mathbf{a}_{12}^* & & & \mathbf{1}
\end{array}\right)
\left(\begin{array}{c}
\mathbf{u}_1 \\ u_2 \\ u_3 \\ \mathbf{u}_4 \\ \hline \mathbf{u}_5 \\ u_6 \\ u_7 \\ \mathbf{u}_8 \\ \hline \mathbf{u}_9 \\ u_{10} \\ u_{11} \\ \mathbf{u}_{12}
\end{array}\right)
=
\left(\begin{array}{c}
\mathbf{d}_1^* \\ d_2^* \\ d_3^* \\ \mathbf{d}_4^* \\ \hline \mathbf{d}_5^* \\ d_6^* \\ d_7^* \\ \mathbf{d}_8^* \\ \hline \mathbf{d}_9^* \\ d_{10}^* \\ d_{11}^* \\ \mathbf{d}_{12}^*
\end{array}\right)
$$

Parallel Method

Step 4 – Backward Communication:

Distribute the solution of the reduced system, d_i for $i = 1$ and m back to each core

Step 5 – Update the other solutions:

for $i = 2...m-1$ **do**
  $d_i \longleftarrow d_i - a_i d_1 - c_i d_m$
end

$$\left(\begin{array}{cccccc} 1 & c_1^{*j-1} & & & & \\ a_4^{*j-1} & 1 & c_4^{*j-1} & & & \\ \hline & a_1^{*j} & 1 & c_1^{*j} & & \\ & & a_4^{*j} & 1 & c_4^{*j} & \\ \hline & & & a_1^{*j+1} & 1 & c_1^{*j+1} \\ & & & & a_4^{*j+1} & 1 \end{array}\right) \left(\begin{array}{c} u_1^{j-1} \\ u_4^{j-1} \\ \hline u_1^{j} \\ u_4^{j} \\ \hline u_1^{j+1} \\ u_4^{j+1} \end{array}\right) = \left(\begin{array}{c} d_1^{*j-1} \\ d_4^{*j-1} \\ \hline d_1^{*j} \\ d_4^{*j} \\ \hline d_1^{*j+1} \\ d_4^{*j+1} \end{array}\right)$$

Parallel Method

Summary:-

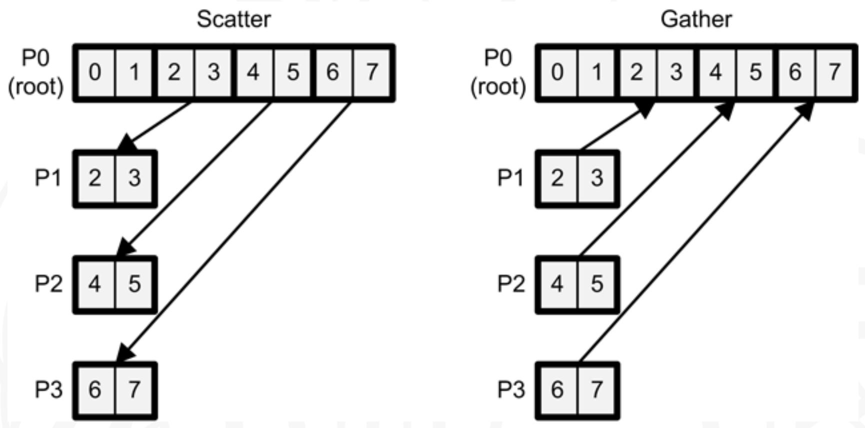
1. Every core transforms the partitioned sub-matrices in the tridiagonal systems of equations into the modified forms.

2. Construct a reduced tridiagonal system of equations by collecting the first and last row of every modified sub-matrix,using MPI_Gather

3. The reduced tridiagonal system constructed in step 2 is solved using the Thomas Algorithm.

4. The solutions of reduced tridiagonal systems in Step 3 are distributed to each core, using MPI_Scatter

5. Finally, we solve for the remaining unknowns of the modified sub-matrices (Step 1) by using the solutions obtained in Steps 3 and 4.

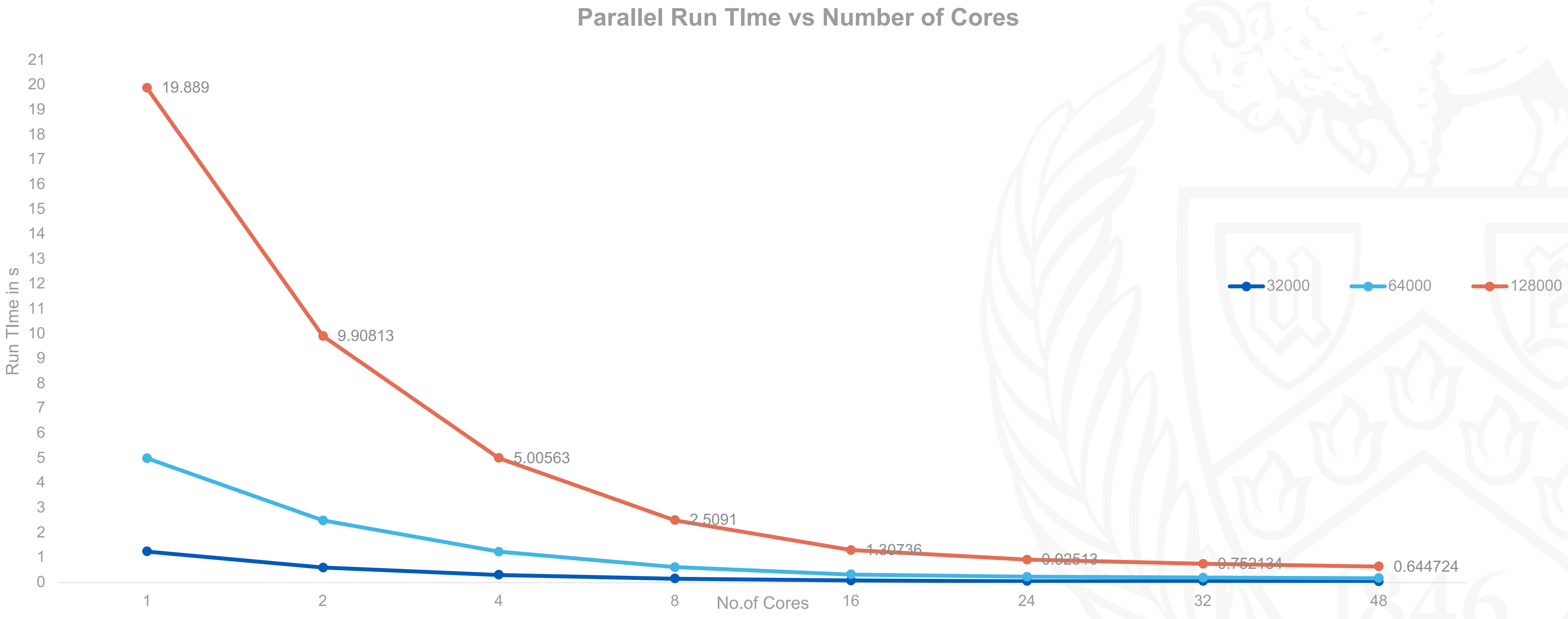


source- codeproject.com

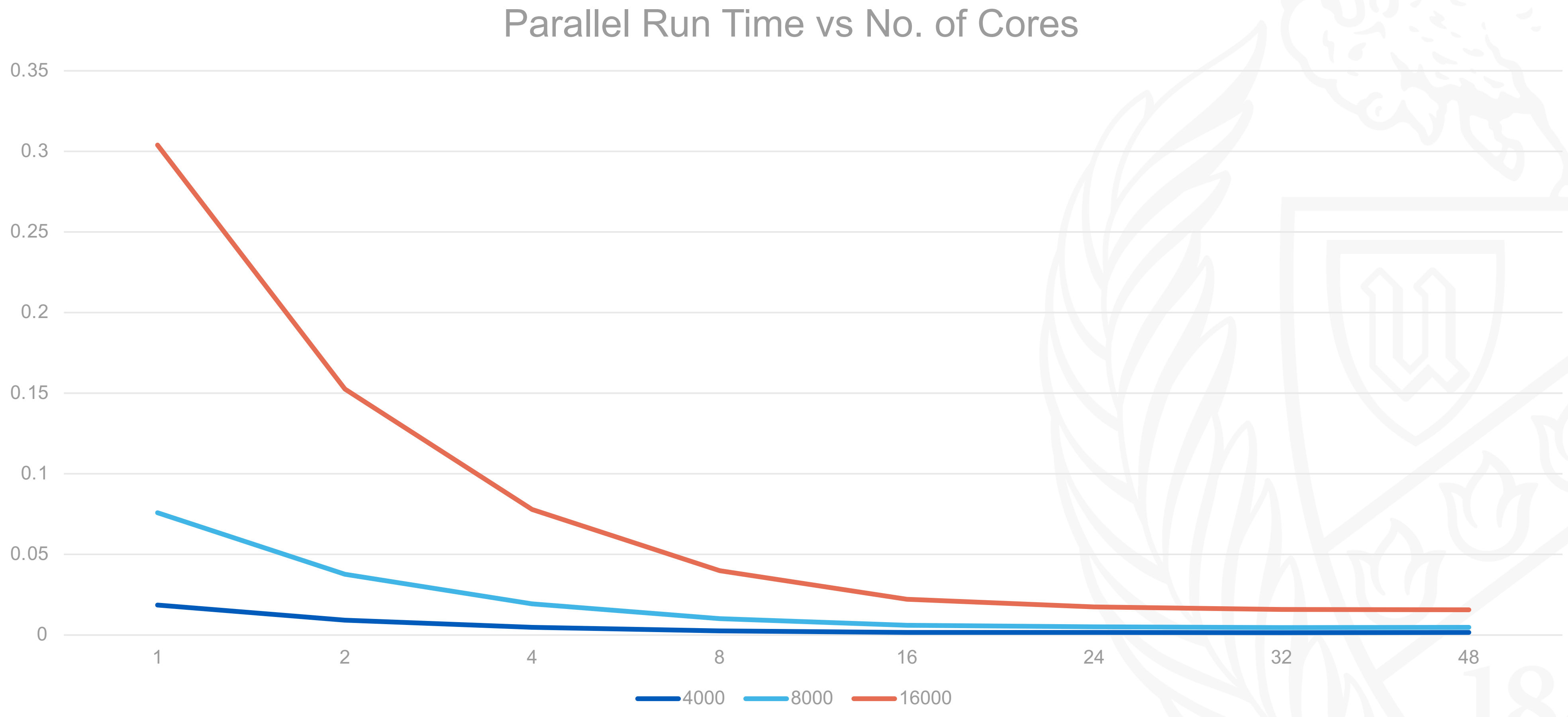
Results: The average of time(in seconds) recorded for four runs

N*N(size)	Serial	Procs=2	Procs==4	Procs=8	Procs=16	Procs=24	Procs=32	Procs=48
500	0.00242	0.000204	0.000129	0.000113	0.000137	0.000169	0.000551	0.000315
1000	0.003194	0.000602	0.00034	0.000208	0.000191	0.000229	0.000304	0.000411
2000	0.005477	0.002355	0.001223	0.000703	0.000525	0.001181	0.000602	0.000838
4000	0.0186	0.009305	0.004789	0.00255	0.001605	0.001484	0.001471	0.001763
8000	0.075866	0.037791	0.019273	0.010203	0.006001	0.004951	0.004636	0.004818
16000	0.303937	0.152449	0.077777	0.039812	0.022207	0.017308	0.015984	0.015659
32000	1.24737	0.608305	0.30757	0.156505	0.08398	0.062848	0.054981	0.050052
64000	5.00169	2.49914	1.2428	0.621809	0.325638	0.235615	0.197465	0.173918
128000	19.889	9.90813	5.00563	2.5091	1.30736	0.92513	0.752134	0.644724

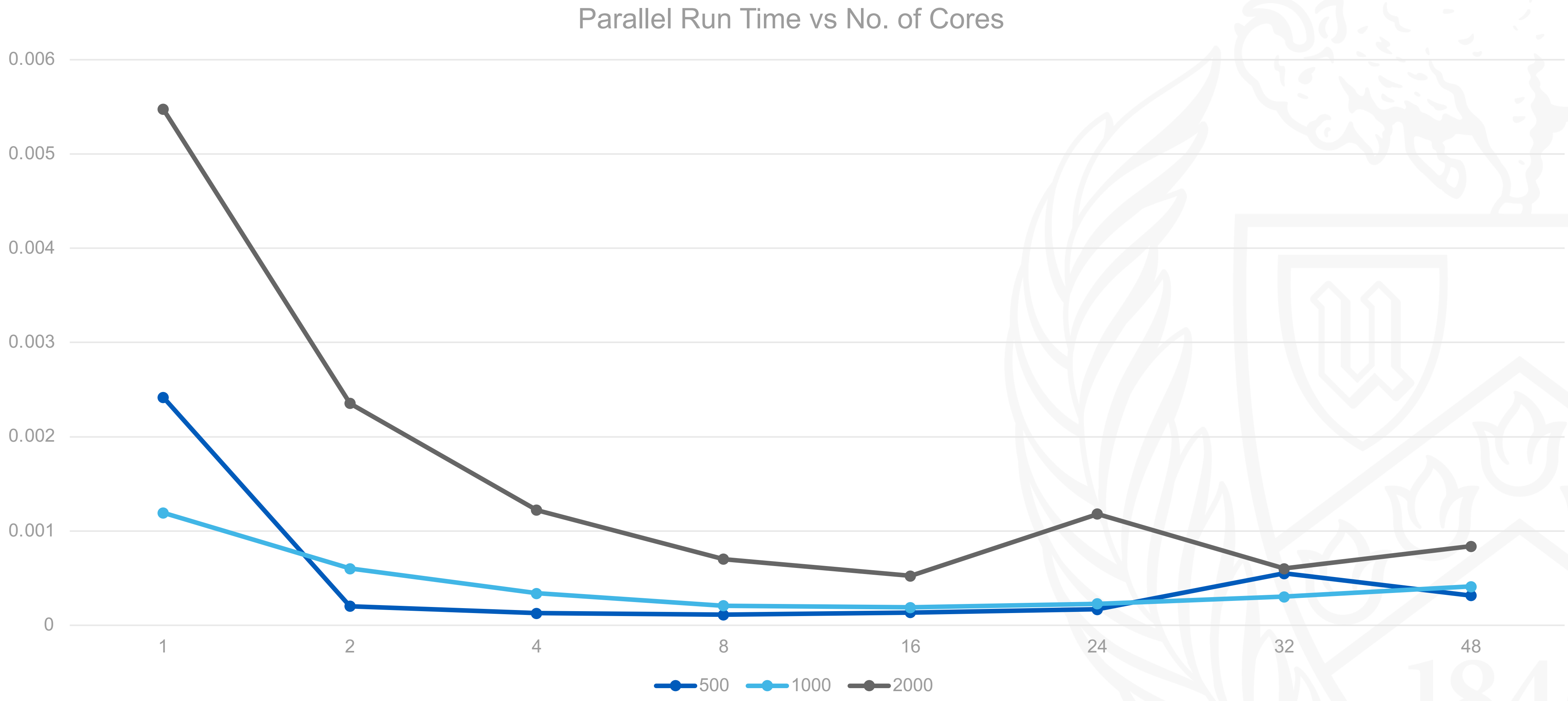
Graph


Parallel Run TIme vs Number of Cores
Run TIme in s
21
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
19.889
9.90813
5.00563
2.5091
1.30736
0.92513
0.752134
0.644724
1
2
4
8
16
24
32
48
No.of Cores
32000
64000
128000

Graph


Parallel Run Time vs No. of Cores
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
1
2
4
8
16
24
32
48
4000
8000
16000

Graph



Parallel Run Time vs No. of Cores
0.006
0.005
0.004
0.003
0.002
0.001
0
1
2
4
8
16
24
32
48
500
1000
2000

Future Work -

- Calculate Efficiency and Speedup
- Explore more ways to solve Tri-diagonal System

Reference -

- Algorithms Sequential and Parallel: A Unified Approach by Russ Miller, Laurence Boxer
- Stefan Bondeli, Divide and conquer: a parallel algorithm for the solution of a tridiagonal linear system of equations, Parallel Computing, Volume 17, Issues 4–5, 1991
- L. Brugnano, A parallel solver for tridiagonal linear systems for distributed memory parallel computers, Parallel Computing, Volume 17, Issue 9, 1991
- Implementation of a Parallel Tridiagonal Solver for Linear system of Equations arising in Physicell-BioFVM, by Shardool Kulkarni
- Parallel Numerical Algorithms - Chapter 9 – Band and Tridiagonal Systems, Michael T. Heath University of Illinois.

Thank You !!!