Proect Assignment 2: Pthreads Programming
Project Report

Denim Deshmukh - dede19@student.bth.se
and
Koyyada Sai Pranav - saky19@student.bth.se

11'th May 2020

1 Introduction

The program use Pthreads (POSIX threads) for Implement a parallel version of Gaussian elimination.Gauss Elimination is a method for solving Equation like
Ax=B(1)
Gauss Elimination takes a system of equation and transform the system of equation into a upper triangular matrix and the back substitution is used to solve for unknown variable and get solution.

$$a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n = b_1$$

$$a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n = b_2$$

$$.$$

$$.$$

$$a_{n1}x_1 + a_{n2}x_2 + ... + a_{nn}x_n = b_n$$

From above Equation we we create a matrix as

$$\text{A=}\begin{vmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ . & & & \\ . & & & \\ a_{n1} & a_{n2} & ... & a_{nn} \end{vmatrix} \text{X=}\begin{vmatrix} x_1 \\ x_2 \\ . \\ . \\ x_n \end{vmatrix} \text{B=}\begin{vmatrix} b_1 \\ b_2 \\ . \\ . \\ b_n \end{vmatrix}$$

By performing elementry operation as given in Figure 1 we transfor the matrix to upper triangular matrix

$$\text{A}^{'} = \begin{vmatrix} a^{'}_{11} & a^{'}_{12} & ... & a^{'}_{1n} \\ 0 & a^{'}_{22} & ... & a^{'}_{2n} \\ . & & & \\ . & & & \\ 0 & 0 & ... & a^{'}_{nn} \end{vmatrix} B = \begin{vmatrix} b^{'}_1 \\ b^{'}_2 \\ . \\ . \\ b^{'}_n \end{vmatrix}$$

We get solution for system of linear equations by back substitution by using given equation(1) as

$$x_i = \frac{1}{a^{'}_{ii}}\Big(b^{'}_i - \sum_{=i+1}^{n} a^{'}_{ij}x_j\Big) \tag{1}$$

2 Implementation

```
Gaussian Elimination (no pivoting):
for (int i = 0; i < N-1; i++) {  /* Outer loop */
  // parallel part
  for (int j = i; j < N; j++) {
  //j loop is parallelizable.
   //  later iterations  do not depend on earlier ones, j iterations be
      computed in any order
    double ratio = A[j][i]/A[i][i]; /* Division step */
    for (int k = i; k < N; k++) {  //k loop is also parallelizable.
       // its number of iterations varies but is calculable for every i.
       // None of the later iterations depend on earlier ones, and they can
          all be computed in any order
      A[j][k] -= (ratio*A[i][k]);
      b[j] -= (ratio*b[i]);
    }
  }
}
```

Fig:1

The Gauss Elimination method is to solve system of equations in two major part first is to get the upper triangular form of matrix by doing row wise elimination ,following the algorithm in Figure 1 on give matrix and second is back substitution step.
The implementation is based on what part of code can be parallelized and what part cannot be parallelized.From fig.1 The inner loop is parallelizable since later iterations in the loop are not dependent on earlier ones and order of calculation does not matter in this loop.
The most inner loop k is also parallelizable , it's number of iterations varies but is calculated for every i and also later iterations in the loop are not dependent on earlier ones and order of calculation does not matter in this loop.[2][3]. The back substitution(equation-1) part is not parallelizable due to data dependency.
For data distribution Guided-self scheduling[1] as a base is used in program to determine the chunk ,each thread gets successively smaller chunks .

3 Measurements

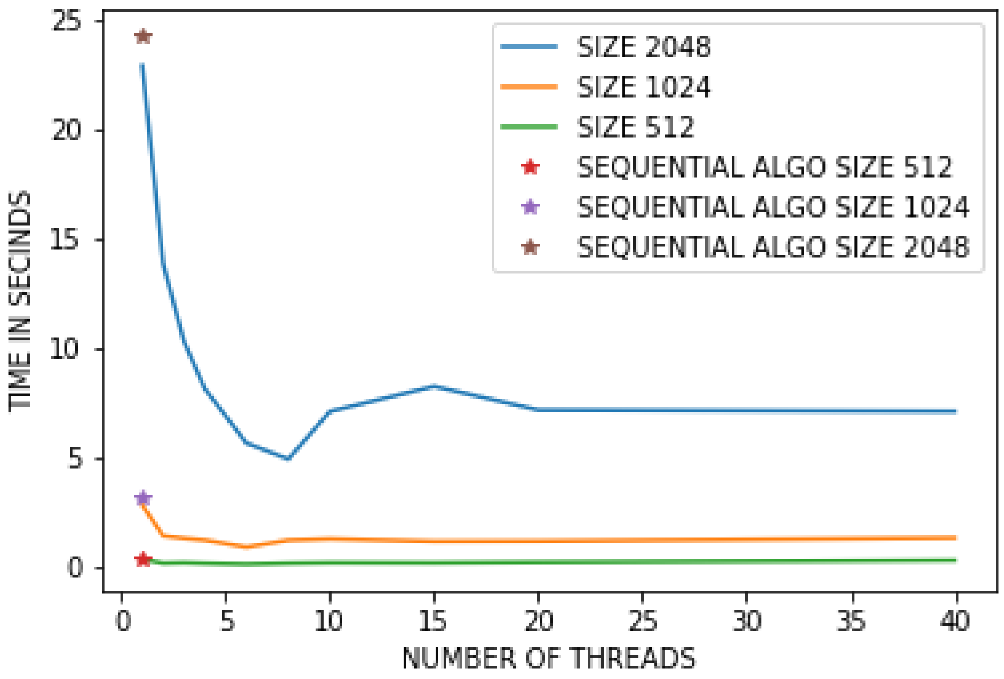

Figure 2:Time to execute V/S Number of Threads

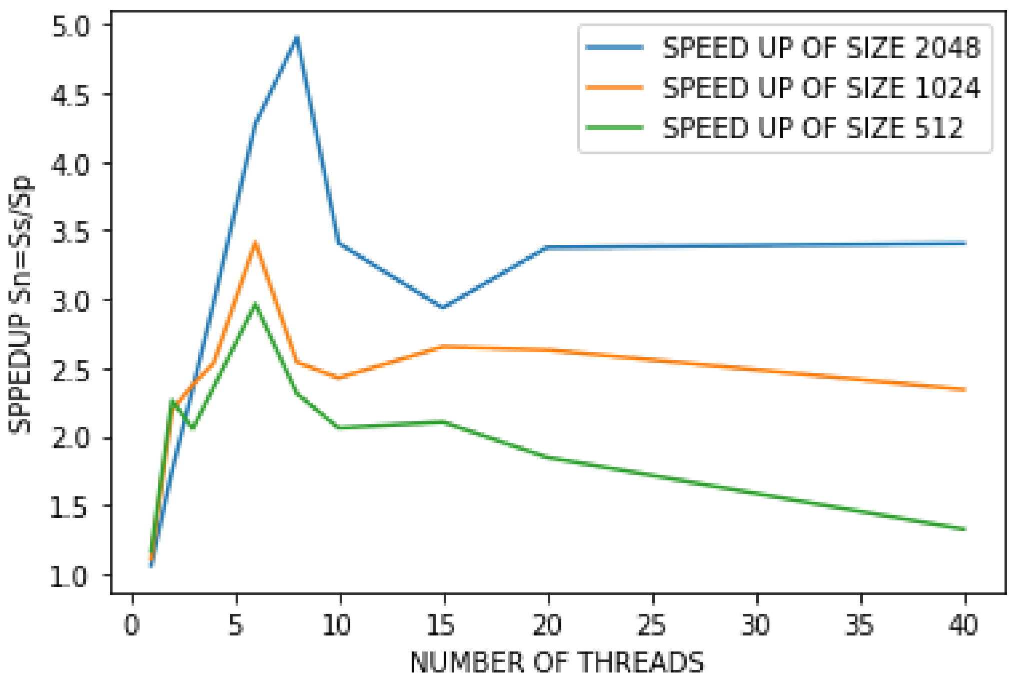

Figure 3:Speed Up V/S Number of Threads

As in Figure 2 we the program was executed on various size of 2048, 1024 and 512 size matrix and on various number of threads as 1, 2, 3, 4, 6, 8, 10, 15, 20 and 40 and the extracted data is shown in figures and it was notices that there is a optimal point for every size which give best time of execution , that point was around 8 threads and 6 threads.
For speed up was calculate by equation 2[6] given below which is ration of time to execute serially to time to execute parallelly. The speed up of positive sub-linear and have crossed 1.5 speedup mark for optimal number of threads according to size. The speed up can be seen to increase till the 8 number of threads and then a flat or decrease curve is noticed.

$$S = \frac{S_s}{Sp}$$

No of Threads	Time(Sec),SIZE=2048	Time(Sec),SIZE=1024	Time(Sec),SIZE=512
1	22.892943	2.833853	0.357798
2	13.861803	1.431275	0.184023
3	10.313798	1.318918	0.202564
4	8.146708	1.237064	0.176454
6	5.666787	0.917756	0.140396
8	4.938299	1.233493	0.180112
10	7.108071	1.291891	0.202032
15	8.253593	1.180779	0.197832
20	7.175234	1.191410	0.225453
40	7.114234	1.338384	0.314419

Table 1: Time for execution for various size and for number of threads

Size	Time in Seconds
512	0.416125
1024	3.132290
2048	24.223247

Table 2: Size V/s linear Execution time

No of Threads	SpeedUp,SIZE=2048	SpeedUp,SIZE=1024	SpeedUp ,SIZE=512
1	1.0581097851857668	1.1053113905343714	1.1630165624179007
2	1.7474816948415730	2.1884613369198790	2.2612662547616330
3	2.3486253075734080	2.3748936628357487	2.0542890148298810
4	2.9733785720563450	2.5320355292854693	2.3582633434209486
6	4.2745998746732500	3.4129877658113920	2.9639377190233342
8	4.9051803060122525	2.5393658496643270	2.3103679932486454
10	3.4078510189332665	2.4245776152941696	2.0596984636097253
15	2.9348729698690010	2.6527317982450564	2.1034261393505600
20	3.3759521989108650	2.6290613642658696	1.8457283779767848
40	3.4048988267746045	2.3403522456933135	1.3234728181184980

Table 3: Speed Up for various size and for number of threads

4 References

[1] C. D. Polychronopoulos and D. J. Kuck, "Guided Self-Scheduling: A Practical Scheduling Scheme for Parallel Supercomputers," IEEE Transactions on Computers, vol. C-36, (12), pp. 1425-1439, 1987.
[2]https://riebecca.blogspot.com/2008/12/supercomputing-course-openmp-syntax-and$_1$5.*html*
[3]*https* : //*www.cs.rutgers.edu/ venugopa/parallel*$_s$*ummer*2012/*ge.html*
[4]*https* : //*computing.llnl.gov/tutorials/pthreads/*
[5]*A.Grama*, *A.Gupta*, *G.Karypis*, *andV.Kumar*, *Introductiontoparallelcomputing*, 2*ndedition*, *Addison−Wesley*, 2003.
[6]*https* : //*www.cs.uky.edu/ jzhang/CS*621/*chapter*7.*pdf*