Applications of Parallel Processing In Linear Algebra

Vibhor Jaunsari[1], Shyamal Goel[2], Mridul Tripathi[3], Prof. Narayanan Prasanth[4]
[1, 2, 3, 4] Vellore Institute Of Technology

Abstract- *Direct factor based math is the piece of number juggling concerning straight conditions and direct limits, for instance, and their depictions through systems and vector spaces. Straight factor based math is vital to essentially all regions of number juggling. For instance, straight factor based math is basic in current presentations of geometry, including for describing basic things, for instance, lines, planes and turns. Furthermore, utilitarian examination may be in a general sense seen as the use of direct factor based math to spaces of limits. Straight factor based math is moreover used in numerous sciences and building domains, since it licenses exhibiting various ordinary wonders, and capably figuring with such models. For nonlinear systems, which can't be exhibited with direct polynomial math, straight factor based math is consistently used as a first-demand surmise.*

Keywords- Linear algebra, Leontief Economic Models, Upper triangular matrix, Lower triangular matrix, Do-Little Algorithm.

I. INTRODUCTION

There are various employments of straight polynomial math in various fields of science, material science and number- crunching, for instance, Constructing Curves and Surfaces through Specified Points, Cubic Constructing Curves and Surfaces Through Specified Points , Cubic Spline Interpolation , Markov Chains ,Graph Theory , Games of Strategy , Leontief Economic Models ,Forest Management ,Computer Graphics , Equilibrium Temperature Distributions ,Computed Tomography ,Fractals , Cryptography , Genetics , Age-Specific Population Growth ,Harvesting of Animal Populations, A Least Squares Model for Human Hearing , Warps and Morphs ,Internet Search Engines

A huge amount of the extended and repetitive calculations of incorporating lattices in straight factor based math can be taken care of stunningly snappier by using parallel computations.

In the field of numerical assessment and straight factor based math, a cross section is considered or rotted along with a lower and an upper triangular matrix. A phase system is similarly joined into the thing. Square structures of direct conditions are as often as possible comprehended by PCs by applying LU factorization. PCs similarly use this methodology while preparing various numerical computations like calculating the determinant or in reverse of the system. A cross section wherein the segments over the guideline corner to corner of the system are zero is known as a lower triangular structure while frameworks in which the parts underneath the crucial inclining are zero is known as the upper triangular lattice.

For example, in case we consider a 3*3 network, the LU breaking down of the cross section is according to the accompanying

Consider An as a n*n system , with x being a vector is size n which contains cloud factors which is to be lit up and vector b of size n being the yield. This is for the condition Ax=b. We factorize network An into upper and lower triangular grids which are U and L. By and by we use substitution to appreciate 2 triangular systems Ly=b, to calculate the vector y which is then in this manner used to settle Ux=y to finally enlist the estimation of x.

When preparing the LU rot of a system, consider each cycle, which has been apportioned into 3 phases. The underlying advance incorporates us searching for the turn segment among the essential line segments of A(k). The accompanying antic- ipates that us should separate each segment in the essential area of A(k) by a(k,k). The last advance anticipates that us should reviving and preparing the segments of A(k) in like manner.

All the 3 phases of each accentuation referenced above can be parallelised sufficiently, and the test remains in understand- ing the correct figuring to consider the data conditions present in the more than 3 phases t circled the remarkable jobs that needs to be done subject to the number and availability of enlisting or planning units open to us.

Consider An as a n*n system , with x being a vector is size n which contains cloud factors which is to be lit up and vector b of size n being the yield. This is for the condition Ax=b. We factorize network An into upper and lower triangular grids which are U and L. By and by we use substitution to appreciate 2 triangular systems Ly=b, to calculate the vector y which is then in this manner used to settle Ux=y to finally enlist the estimation of x.

When preparing the LU rot of a system, consider each cycle, which has been apportioned into 3 phases. The underlying advance incorporates us searching for the turn segment among the essential line segments of A(k). The accompanying antic- ipates that us should separate each segment in the essential area of A(k) by a(k,k). The last advance anticipates that us should reviving and preparing the segments of A(k) in like manner.

All the 3 phases of each accentuation referenced above can be parallelised sufficiently, and the test remains in understand- ing the correct figuring to consider the data conditions present in the more than 3 phases t circled the remarkable jobs that needs to be done subject to the number and availability of enlisting or planning units open to us.

II. SEQUENTIAL LU FACTORIZATION ALGORTIHM

We have 3 made due with hovers in the back to back count. We have 2 phases, one division and one end for each accentuation of the outer circle. K*K sub grids become

$$\begin{bmatrix} a11 & a12 & a13 \\ a21 & a22 & a23 \\ a31 & a32 & a33 \end{bmatrix} = \begin{bmatrix} l11 & 0 & 0 \\ l21 & l22 & 0 \\ l31 & l32 & l33 \end{bmatrix} \begin{bmatrix} u11 & u12 & u13 \\ 0 & u22 & u23 \\ 0 & 0 & u33 \end{bmatrix}$$

Fig. 1. Matrix Representation

dynamic on the lower straightforwardly as the estimation executes. This causes a computational weight which is non uniform, as the figuring augmentations for the parts which are accessible in the lower right corner of the structure. As the count proceeds, we can see that there are(n-k-1) and (n-k- 1)^2*(2) figuring exercises for division similarly as end. There, this prompts a runtime of O(n^3).

```
for k ← 0 to n - 1 do
/* Division step */
for i ← k + 1 to n − 1 do
a[i][k] ← a[i][k]/a[k][k]
/* Elimination step */ for i ← k + 1 to n − 1 do
for j ← k + 1 to n − 1 do
a[i][j] ← a[i][j] − a[i][k] ¯ a[k][j]
```

III. PROPOSED METHODOLOGY

LU DECOMPOSITION PARALLEL ALGORITHM->

We take a n*n coefficient system called An and we square stripe it among p strings and focuses with [n/p] circumscribing lines being doled out to each middle. We can use various arranging philosophies which OpenMP demonstrates using the timetable condition. Along these lines, we can consign unmistakable subtasks to different strings the two of which can be made statically or continuously. Designers using OpenMp gadgets have assorted booking proclamations accessible to them. We can use different pieces to allocate different courses of action of cycles to different strings which incredibly in- fluences the execution of the count. In this figuring, we use static arranging anyway don't show the protuberance size which causes OpenMP to disengage the cycles into p squares which are of proportional size of [n/p] and in a blockwise apportionment, they are statically consigned to the strings. Exactly when the circle begins, this task or designation is passed on to be executed by different strings.

```
for k ← 0 to n - 1 do
/* Division step */ #pragma omp parallel for for i ← k + 1
to n - 1 do
a[i][k] ← a[i][k]/a[k][k]
/* Elimination step */ #pragma omp parallel for for i ← k
+ 1 to n 1-do
for j ← k + 1 to n - 1 do
a[i][j] ← a[i][j] - a[i][k] ¯ a[k][j]
```

ALGORITHM WITH ROW CYCLIC DATA DISTRI-BUTION

1.

Gaussian transfer is typically used to unravel direct straight factor based math structures and is all things considered utilized in sensible and arranging models . LU factorization schedules are intertwined into in every way that really matters all prominent direct factor based math libraries, for example, the Linear Algebra Package (LAPACK), the Scalable Linear

,Variable based math Package (ScaLAPACK), and Matrix Variable put together math with respect to GPU ,what's more, Multicore Architectures (MAGMA). As clear standard programming for world class thick direct polynomial math checks, LAPACK and ScaLAPACK have been made for shared-memory and passed on memory structures . LAPACK is an item library for getting a handle on structures of syn-chronous direct conditions, least-squares strategies of straight frameworks of conditions, eigenvalue issues, and single worth issues. It is proposed to be gainful on a wide degree of present

day unmatched PCs and gives the related framework factorizations. ScaLAPACK is a continuation of the LAPACK experience for scattered memory message-passing MIMD PCs and structures of workstations supporting PVM and MPI. Regardless, they don't have any tremendous bearing to heterogeneous structures with GPUs. MAGMA is a social event of direct polynomial math libraries for cross breed many focus and GPU frameworks. MAGMA utilizes a hybridization framework where tallies of intrigue are part into assignments of differentiating granularity, and their execution is organized over the open rigging parts. To improve the presentation of the check, insignificant nonparallelizable errands are much of the time pushed toward the CPU, and more prominent consistently parallelizable assignments are routinely set up for the GPU. Their maltreatment of parallelism depends upon the transparency of parallel BLAS. CUBLAS and CULA have understood the standard BLAS and LAPACK subroutine libraries, only, on various GPUs. For the handles execution im- provement for cross area factorization, Volkov et al. Finished the right-looking calculation for LU, Cholesky, and QR for GPUs. Van De Velde indicated a LU breaking down estimation for multicomputers with certain rotating. Kurzak et al. utilized halfway rotating for multicore frameworks to breath life into LU factorization. A halfway turning structure knows about LU factorization in standard straight polynomial math libraries, for example, LAPACK.

APPLICATION 1:

LU DECOMPOSITION SERIAL ALGORITHM DOOLITTLE ALGORITHM

It is constantly conceivable to consider a square lattice a lower triangular grid and an upper triangular network. That is,

[A] = [L][U]

Doolittle's technique gives an elective method to figure An a LU decay without experiencing the problem of Gaussian Elimination.

x

For a general n n grid A, we accept that a LU decay exists, and compose the type of L and U expressly. We at that point methodicallly unravel for the sections in L and U from the conditions that outcome from the increases vital for A=LU.

For each i = 0,1,2 . . . n-1

U(i,k) = A(i,k) – sum(j :0 –>i) {L[i,j] * U[j,k]}

L(i,k) = (A[i,k] – sum(j:0->i) {L[i,j]*U[j,k]}) / U(k,k)

PARALLEL ALGORITHM

We section the framework 'An' as: A1A2

A3A4

We section the framework 'L' as : L10

L3L4

We section the framework 'U' as: U1U2

0U4

Stage 1: Compute LI and U1 by applying Doolittle calculation on A1

A1 = [L1 U1]

Stage 2: Compute U2 A2 = [L1 U2]

U2 = inv(L1) * A2 Step 3: Compute L3 A3 = [L3 U1]

L3 = A3 * inv(U1)

Stage 4: Compute L4 and U4 by applying Doolittle Algorithm as pursues [L4 U4] = A4 – [L3 *U2]

To parallelize the calculation, we can process stage 2 and stage 3 together in light of the fact that there is no information reliance .

IV. TEST RESULTS AND FINDINGS

LU FACTORIZATION SERIAL ALGORITHM ->

MATRIX 1

3	9	13	22	18	24	21	22
25	21	12	7	36	27	12	13
12	24	27	8	24	16	26	17
9	4	9	17	26	22	14	19
21	29	28	30	10	31	28	17
13	11	12	23	13	20	25	34
10	13	14	22	7	24	17	25
15	17	18	25	12	26	20	26

Fig. 2. Matrix 1

In order to get information into the display of the com- putation presented, a couple of models have been decided for diagram. Since we don't have a parallel processor, we impersonated it on a consecutive PC (VAX I l/780). In this multiplication each processor is brought into the VAX 1 l/780 each thusly, and its program is executed.

LOWER TRIANGULAR

Exactly when this is done, it is then turned out, and the processor that is most remote back in its execution time is

3	0	0	0	0	0	0	0
25	-54	0	0	0	0	0	0
12	-12	-3.592594	0	0	0	0	0
9	-23	11.030864	-99.21475	0	0	0	0
21	-34	-2.345678	13.673553	-36.18142	0	0	0
13	-28	5.617283	-44.71819	-19.22596	-4.327602	0	0
10	-17	0.993826	-7.111668	-19.86618	2.795475	6.738060	0
15	-28	2.950615	-27.08931	-24.11506	0.117135	0.281463	9.825930

Fig. 3. Lower Triangular

gotten straightaway. The time spent by a processor includes the program execution time and the data

correspondence time. Here the synchronization time was unimportant.

UPPER TRIANGULAR

1	3	4.333	7.333	6.0	8.0	7.0	7.3333
0	1	1.783951	3.265	2.1111	3.203704	3.018518	3.154321
0	0	1.0	11.360823	6.309276	11.567006	6.061854	9.226801
0	0	0	1.0	0.494294	1.047313	0.468091	0.768335
0	0	0	0	1.0	0.421820	0.236355	0.514514
0	0	0	0	0	1.0	+2.297744	-4.484860
0	0	0	0	0	0	1.0	3.612786
0	0	0	0	0	0	0	1.0000

Fig. 4. Upper Triangular

In our first investigation we have taken a 5 X 5 lattices utiliz- ing a different number of processors. The parallel arrangement time TP, the holding up time TW, the productivity EFF, and the accelerate proportion

TIME TAKEN –

EXECUTION 1	46678 microseconds
EXECUTION 2	50567 microseconds
EXECUTION 3	50556 microseconds
EXECUTION 4	49672 microseconds
EXECUTION 5	47817 microseconds

Fig. 5. Time Comparison

The arrangement of a direct arrangement of conditions lies at the core of numerous projects for logical calculation. With the ongoing improvement and accessibility of different parallel PCs, new calculations have showed up for settling tridiagonal frameworks of conditions reasonable for these mama chines.

AVERAGE = 49,058 microseconds

LU FACTORIZATION PARALLEL ALGORITHM -> MATRIX 1

3	9	13	22	18	24	21	22
25	21	12	7	36	27	12	13
12	24	27	8	24	16	26	17
9	4	9	17	26	22	14	19
21	29	28	30	10	31	28	17
13	11	12	23	13	20	25	34
10	13	14	22	7	24	17	25
15	17	18	25	12	26	20	26

Fig. 6. Matrix 1-1

So as to get data into the presentation of the calculation displayed, a few models have been chosen for chart. Since we don't have a parallel processor, we imitated it on a back to back PC (VAX I l/780). In this augmentation every processor is brought into the VAX 1 l/780 each along these lines, and its program is executed.

TIME TAKEN –

Precisely when this is done, it is then turned out, and the processor that is most remote back in its execution time is got- ten straightaway. The time spent by a processor incorporates

EXECUTION 1	9453 microseconds
EXECUTION 2	9010 microseconds
EXECUTION 3	8838 microseconds
EXECUTION 4	9671 microseconds
EXECUTION 5	8672 microseconds

Fig. 7. Time Taken

the program execution time and the information correspon- dence time. Here the synchronization time was irrelevant.

AVERAGE = 9128.8 microseconds

TIME IMPROVEMENT FINAL RESULT

ALGORITHM	INITIAL TIME (microseconds)	FINAL TIME (microseconds)	IMPROVEMENT (microseconds)
LU FACTORIZATION (MATRIX 1)	49,058	9128.8	39,929.2
LU FACTORIZATION (MATRIX 2)	50,071	8837.6	41,233.4

Fig. 8. Final Time Comparison

A huge amount of the extended and repetitive calculations of incorporating lattices in straight factor based math can be taken care of stunningly snappier by using parallel computations.

V. CONCLUSION

1. The productivity goes down when the quantity of pro- cessors increments. The effectiveness is the most elevated when the grid is thick.
2. The accelerate proportion is an expanding capacity of the quantity of processors and the data transmission.
3. The productivity increments with the request: along these lines, this calculation is useful for huge networks.

REFERENCES

[1] S. C. Chen, D. J. Kuck and A. H. Sameh, Practical parallel band triangular framework solvers. ACM Trans. Math. Programming 4, 270-277 (1980).

[2] S. C. Chen and A. H. Sameh, On parallel triangular frame- work solvers. Proc. 1975 Sagamore Computer Conf. on Parallel Processing, pp. 237-238, August (1975).

[3] J. W. Huang and 0. Wing, Optimal parallel triangulation of a scanty lattice. /EEE Trans. Circuirs Sysf. CAS26, 726-732 (1979).

[4] E. Isaacson and H. B. Keller, Analwis of Numerical Mrrh- ods. John Wiley, New York (1966).

[5] J. A. G. Jess and H. G. M. Kees, An information structure for parallel L/U decay. fEEE Trans. Comput. C-31, 23 l-239 (1982).