Gaussian Elimination in Serialized and Parallel Programming

Ramdan Own Ali[1]*, Nor Adin A M Yahia*[2]*, Mohamed Said S' Ahmed*[3]

[1,2] ***Higher Institute of Science and Technology, Alhraba***

[3] ***Higher Institutes of Science and Technology, Kabaw***

1.Introduction.

The solving of dense linear systems is an important scientific problem that is used as a nature in many scientific disciplines ranging from computer science, data analysis to computational economics[1], There are many methods for solving a linear system such as direct and iterative methods, Gaussian elimination is method to solve linear equations such as LU decomposition or LU (factorization) that can be used In numerical analysis where 'LU' stands for 'lower upper'[2], that is not like the method of Gaussian elimination which means the concept of upper-triangular matrix.

In this work, we have presented a definition of Gaussian elimination, technique of date decomposition, explanation for LU decomposition, programming definition, kinds of programming computing, in addition we represented graphically the execution time in the sterilized and parallel using OpenMP by C language, and we calculated the average time speeds in the different cases.

2. Algorithm of Gaussian Elimination

The Gaussian elimination algorithm is transforming of linear equations system into an upper-triangular matrix in order to solve the unknowns and derive a solution [2]. A pivot column is used to reduce the rows before it, then after the transformation, back- substitution is applied.

2.1 Gaussian Elimination Explanation

It is easy to see the method of Gaussian Elimination with an example. Let's consider the system questions

$$\begin{cases} x - 3y + z = 4 \\ 2x - 8y + 8z = -2 \\ -6x + 3y - 15z = 9 \end{cases}$$

To solve for x, y, and z, we must eliminate some of the unknowns from some of the equations. Consider adding -2 times the first equation to the second equation and also adding 6 times the first equation to the third equation[2]:

$$\begin{cases} x - 3y + z = 4 \\ 0x + 2y + 6z = -10 \\ 0x + 15y - 9z = 33 \end{cases}$$

We have now eliminated the x term form the last two equations, now simplify the last two equations by 2 and 3, respectively:

$$\begin{cases} x - 3y + z = 4 \\ 0x + y + 3z = -5 \\ 0x + 5y - 3z = 11 \end{cases}$$

To eliminate the y term in the last equation, multiply the second equation by -5 and add it to the third equation:

$$\begin{cases} x - 3y + z = 4 \\ 0x + y + 3z = -5 \\ 0x + 0y - 18z = 36 \end{cases}$$

From the third equation, we can get Z= -2, substituting this into the second equation yields Y= -1 Using both of these results in the first equation gives X= 3. This process of progressively solving for the unknowns is back-substitution [2].

3. Data Decomposition:

In this method, the decomposition of computations is done in two steps. In the first step, the data on which the computations are performed is partitioned, and in the second step, this data partitioning is used to induce a partitioning of the computations into tasks [2]. The operations that these tasks perform on different data partitions are usually similar or are chosen from a small set of operations as it will be seen next in Lu factorization as an example.

4. LU Decomposition

LU decomposition can be used In numerical analysis (where 'LU' stands for 'lower upper', and we can called it LU factorization) factors a matrix as the product of a lower triangular matrix and an upper triangular matrix[3].

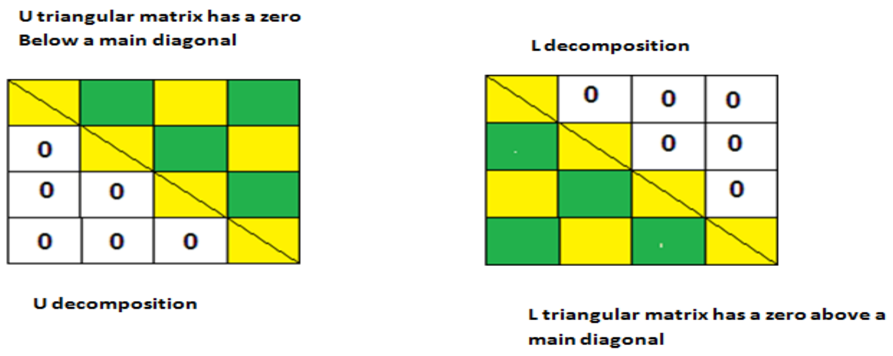
U triangular matrix has a zero Below a main diagonal
L decomposition
0
U decomposition
L triangular matrix has a zero above a main diagonal

Fig1: Lower Upper Decomposition

The LU decomposition can be viewed as the matrix form of Gaussian elimination. Computers usually solve square systems of linear equations using the LU decomposition.

5. Programming Definition

Process of developing and implementing sets of various instructions to enable a computer to do a certain task. These instructions are considered computer programs and help the computer to operate smoothly. The language used to program computers is not understood by an untrained eye. Computer programming continues to be a necessary process as the Internet continues to expand.

6. Kinds of Programming Computing:

6.1 Serialization Computing

Serialization Computing is the process of converting an object into a stream of bytes in order to store the object or transmit it to memory, a database, or a file. Its main purpose is to save the state of an object in order to be able to recreate it when needed.

6.2 Parallel Computing

Parallel Computing is a type of an architecture computing in which several processors execute or process an application or computation simultaneously [4]. Parallel computing helps in performing large computations by dividing the workload between more than one processor, all of which work through the computation at the same time. Most supercomputers employ parallel computing principles to operate, parallel computing is also known as parallel processing.

7. Representation of Execution Time Using Serialized programming

In this case every line in the program will be executed separately, that means the execution will be lined from top to bottom.

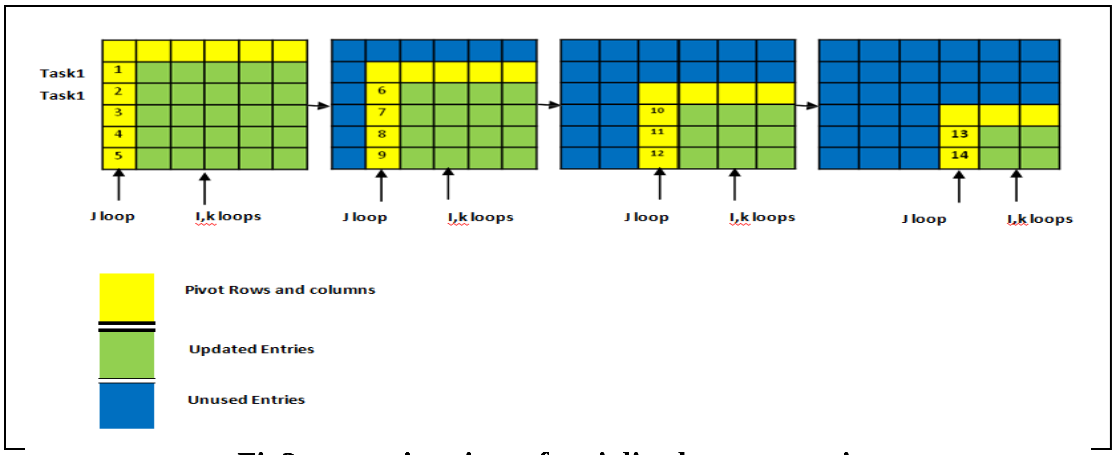

Fig2: execution time of serialized programming

8. Representation of Execution Time Using OpenMP

We can't use OpenMP to parallelize loop (J) because of data dependence [5]. The (I) loop has many iterations that varies with (j), and we know the number of iterations every time we are entering the loop. None of the later iterations depend on the earlier ones and the iterations can be computed in any order. So the (I) loop is parallelizable. The k loop, like the (i) loop, has a number of iterations that varies, but is calculable for each (I) [6]. There is no depend between later and earlier iteration and they can all be computed in any order. Therefore the k loop is also parallelizable.

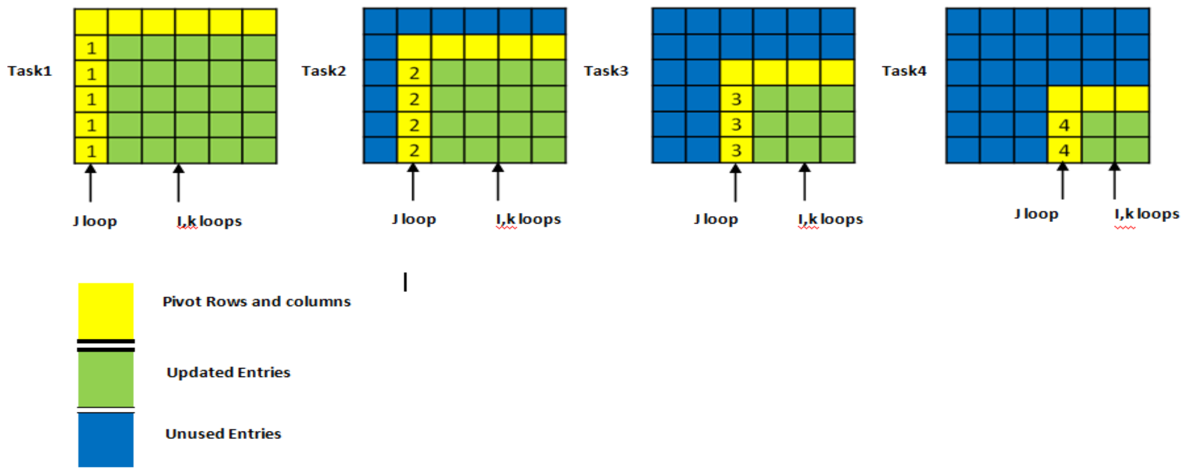

Fig: 3 execution time of openMP

```
for(j=1; j<=n; j++)
{  #pragma omp parallel for
for(i=1; i<=n; i++)
{   If (i>j)
            { …………..
#pragma omp parallel for
for(k=1; k<=n+1; k++)
           {                                                    …………..
```

9. Different Of CPU Time Speeds:

In our program we applied the above concepts represented in the different between the Serialized and parallel programming , and in the table of all the different execution time with the average will be shown:

Table1: execution average of different programs

Reading Def Kinds	**1**	**2**	**3**	**4**	**5**	**Average**
Serialized	Real,0m40,598s user 0m0.536s sys 0m0.084s	Real, 0m25.337s user, 0m0.352s sys 0m0.012s	Real,0m26.982s user 0m0.032s sys 0m0.000s	Real, 0m28.834s user 0m0.032s sys 0m0.008s	Real,0m31.026s user 0m0.024s sys 0m0.008s	30.5554
One loop parallel	Real, 0m21.516s user 0m0.180s sys 0m0.056s	real 0m20.826s user 0m0.480s sys 0m0.004s	real 0m19.546s user 0m0.040s sys 0m0.020s	real 0m17.596s user 0m0.024s sys 0m0.048s	real 0m17.401s user 0m0.412s sys 0m0.128s	19.377
Two loops Parallel	real 0m15.151s user 0m0.072s sys 0m0.268s	real 0m15.073s user 0m0.080s sys 0m0.036s	real 0m14.672s user 0m0.444s sys 0m0.128s	real 0m13.992s user 0m0.032s sys 0m0.004s	real 0m12.117s user 0m0.032s sys 0m0.012s	14.201

The size of matrix that we used was 10, and the entering operation was random.

9.1 Graphic of Different Averages CPU Time Speeds

This graph represented the difference in the average of execution times in three kinds of programs serialized, parallel program with one loop, and parallel program with two loops.

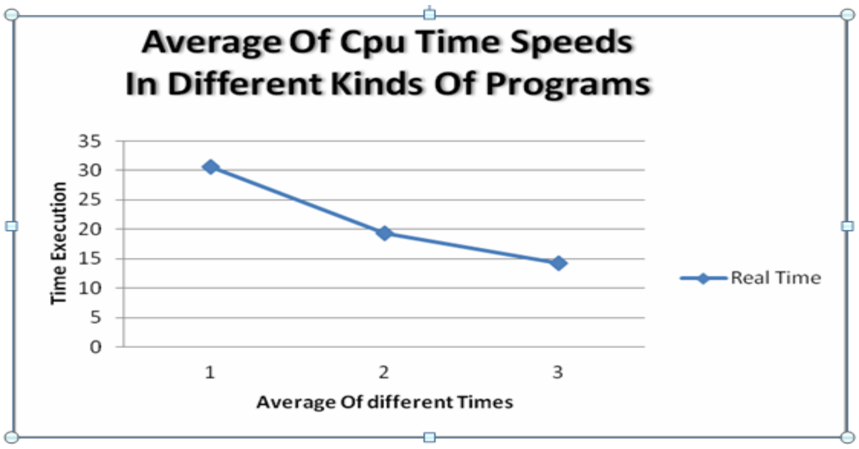

Fig4: execution average of different programs

10. Different Of CPU Time Speeds with Different (Threads)

Here we applied the concept of threads in our program, and we let all two loops worked parallel with different threads represented in threads number 4, 8, and 16, and in the table of all the different execution times in different threads with the average will be shown:

Table2: execution average of different threads

Reading Dif threads	**1**	**2**	**3**	**4**	**Average**
Two loops Parallel with thread (4)	real 0m3.882s user 0m0.428s sys 0m0.012s	real 0m3.290s user 0m0.412s sys 0m0.136s	real 0m2.506s user 0m0.044s sys 0m0.008s	real 0m2.269s user 0m0.064s sys 0m0.012s	2.98675
Two loops Parallel with thread (8)	real 0m2.410s user 0m0.000s sys 0m0.004s	real 0m2.051s user 0m0.000s sys 0m0.096s	real 0m2.130s user 0m0.008s sys 0m0.156s	real 0m2.020s user 0m0.000s sys 0m0.004s	2.15275
Two loops Parallel with thread (16)	real 0m1.905s user 0m0.000s sys 0m0.060s	real 0m1.787s user 0m0.000s sys 0m0.008s	real 0m1.654s user 0m0.000s sys 0m0.008s	real 0m1.594s user 0m0.000s sys 0m0.092s	1.735

10.1 Graphic of Different Average of CPU Time speeds Using Two Loops Parallel In Different Threads:

In this graph we represented the difference in the average of execution times in three different threads 4, 8, and 16 with two loops parallel in the program

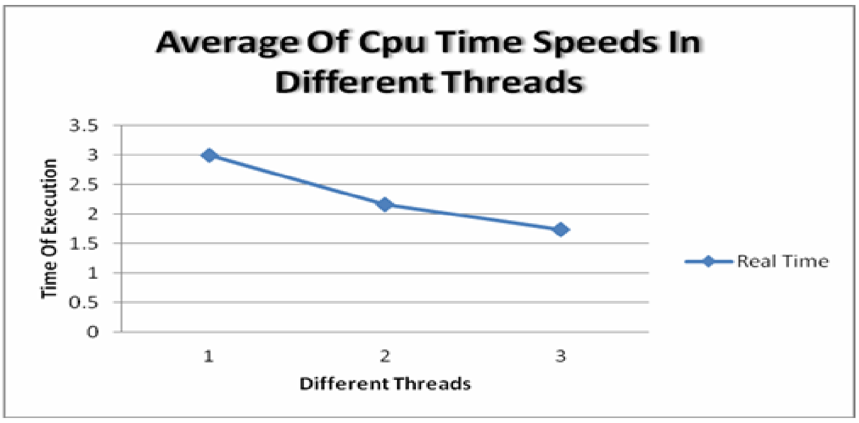

Fig5: execution average of different thread

11. Conclusion

In this work we used the Gaussian elimination method to solve a linear equations using C++ environment in a system of Ubuntu 14.04 LTS under oracle VM virtual box manager with processor Intel core i3 CPU M 330 @2,13 GHz ×4, so we proved that there is a difference between the execution time in case serialized, and parallel, because of using the openMP which let the computer used more than two processors, this thing let all the processors distributed the tasks between them, and sharing the memory simultaneously, that is why the execution time in parallel is faster than in case of serialized. Also we applied the concept of threads 4, 8, 16 in our program in the case of parallel that let the execution time increased while we increased the number of threads, the result was exploiting the efficiency of the processers in our computer to perform the tasks fast, addition to that the size of matrix that we used was (10X10), and the entering operation was random, also the times of reading was 5 times between serialized and parallel program, and was 4 times between the different threads with parallel programming, we calculated the average in all the cases above, which shown in the tables and represented in the graphics. Finally programming parallel is faster than serialized, because of executing all tasks together is faster than executing all tasks task by task.

11. References:

[1] Robert van Engelen, Parallel Gaussian Elimination (2012).

[2] Gary D. Knott,Gaussian Elimination and LU-Decomposition (2014),Civilized Software Inc.

[3] Bindel,Gaussian elimination in matrix terms (2012),Intro to Scientific Computing

[4] Jan Verschelde, Introduction to Supercomputing, Parallel Gaussian Elimination (2014)

[5] S.F.McGinn and R.E.Shaw, Parallel Gaussian Elimination Using OpenMP and MPI (2002),University of New Brunswick.

[6] P. D. Michailidis; K. G. Margaritis,Open Multi Processing (OpenMP) of Gauss-Jordan Method for Solving System of Linear Equations (2011) ,University of Western Macedonia.