**LU Factorization using CUDA**

Van Quach Dinh

Andy Yang

**Abstract**

LU decomposition of a matrix is the factorization of a matrix into two triangular matrices, one lower triangular matrix and one upper triangular matrix. The product of these two matrices gives the original matrix. Converting a matrix into two triangular matrices makes it easy to solve linear systems of equations. Due to large processing power of the GPU that comes with grid of multiprocessors, solving LU factorization using GPU yields to great performance. In this work, LU factorization for solving systems of linear equations in CUDA has been implemented, and its performance has been examined.

**Introduction**

Given a linear systems of equations, LU factorization decomposes matrix into two triangular matrices. The lower triangular matrix would be, and the upper triangular matrix would be , and their product is the original matrix . By applying substitution, matrixcan be replaced with . Therefore, the equation becomes . By apply forward substitution using the lower triangular matrix , , where is the result of the forward substitution. Lastly, apply backward substitution by using upper triangular matrix to solve for . By keeping matrix constant, LU decomposition can be viewed as the matrix form of gaussian elimination. LU decomposition remembers the steps that are used to solve the system, and there is no need to calculate the steps from scratch when given a new vector .

Other methods such as gaussian elimination can also be used to solve linear systems of equations. It requires scaling, elimination, and swapping rows of both sides of the equations. However, the steps of gaussian elimination for solving matrix depend mainly on the structure of matrix rather than the values of vector . Therefore, it takes longer time to re-compute a solution. Alternatively, one can use the matrix inverse method that multiplies inverse of matrix on both sides; would simplifies to . LU decomposition can also be used in finding the inverse of a matrix. In this work, we will present the LU decomposition of matrix implemented on CUDA-enabled GPU, and evaluate its performance.

**Project Description**

This project studies LU decomposition using parallel algorithm that is run on GPU. Firstly, we solve for the standard LU decomposition. Then, we solve for the block LU decomposition.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | A00 | A01 | A02 | | A10 | A11 | A12 | | A20 | A21 | A22 | | |  |  |  | | --- | --- | --- | | L00 | 0 | 0 | | L10 | L11 | 0 | | L20 | L21 | L22 | | |  |  |  | | --- | --- | --- | | U00 | U01 | U02 | | 0 | U11 | U12 | | 0 | 0 | U22 | |

Initially, we set diagonals of to 1s, and solve for U00.  Given an example below:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | 8 | 1 | 7 | | 4 | 2 | 9 | | 6 | 3 | 10 | | |  |  |  | | --- | --- | --- | | 1 | 0 | 0 | | L10 | 1 | 0 | | L20 | L21 | 1 | | |  |  |  | | --- | --- | --- | | U00 | U01 | U02 | | 0 | U11 | U12 | | 0 | 0 | U22 | |

Step 1: Calculate the first row of in parallel given L00 = 1

A00 = L00U00  ➡ U00 = A00 / 1 =8

A01 = L00 U01 ➡ U01 = A00 / 1 =1

A02 = L00 U02 ➡ U02 = A00 / 1 =7

Step 2: Calculate the first column of in parallel given U00= 8

A10 = L10 U00 ➡ L10 = A10 / U00 = 4 / 8 = 0.5

A20 = L20 U00 ➡ L20 = A20 /U00 = 6 / 8 = 0.75

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Result:   |  |  |  | | --- | --- | --- | | 8 | 1 | 7 | | 4 | 2 | 9 | | 6 | 3 | 10 | | |  |  |  | | --- | --- | --- | | 1 | 0 | 0 | | 0.5 | 1 | 0 | | 0.75 | L21 | 1 | | |  |  |  | | --- | --- | --- | | 8 | 1 | 7 | | 0 | U11 | U12 | | 0 | 0 | U22 | |

Step 3: Find the sub-matrix A1 for the next iteration of LU decomposition

A0 = [0.5 0.75] T [1 7] + L1 U1

A1 = A0 - [ 0.5 0.75] T [1 7] = L1 U1

Result:

|  |  |
| --- | --- |
| 1.5 | 5.5 |
| 2.25 | 4.75 |

Step 4: Repeat the process of finding the first row and the first column of the sub-matrix.

A11 = L11U11 ➡ U11 = 1.5

A12 = L11U12 ➡ U12 = 5.5

A21 = L21U11 ➡ L21 = 1.5

Step 5: Find the sub-matrix A2 for the next iteration of LU decomposition.

A1 = [1.5] T [5.5] + L2U2

A2 = A1 - [1.5] T [5.5] = L2 U2

A2  = 4.75 - 8.25 = -3.5

Step 6: Terminate the LU decomposition when LnUn = 1 x 1.

A22 = L22U22 ➡ U22= -3.5

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| End Result:   |  |  |  | | --- | --- | --- | | 8 | 1 | 7 | | 4 | 2 | 9 | | 6 | 3 | 10 | | |  |  |  | | --- | --- | --- | | 1 | 0 | 0 | | 0.5 | 1 | 0 | | 0.75 | 1.5 | 1 | | |  |  |  | | --- | --- | --- | | 8 | 1 | 7 | | 0 | 1.5 | 5.5 | | 0 | 0 | -3.5 | |

For the standard LU decomposition, we repeatedly calculate the first row and the first column of matrix A and its sub-matrix until LnUn = 1 x 1. For block LU decomposition, given that matrix A has M rows and N columns, we choose a sub-matrix block of size: r x r, and treating that matrix as A00. By using the standard LU decomposition, we first calculate the LU decomposition for A00 block. Secondly, we calculate the first row of blocks for U and the first column of blocks for L. Thirdly, we update the sub-matrix of A for the next iteration of calculation until LnUn = 1 x 1. In the project, we will be using the block LU decomposition method on CUDA which incorporates the standard LU decomposition.

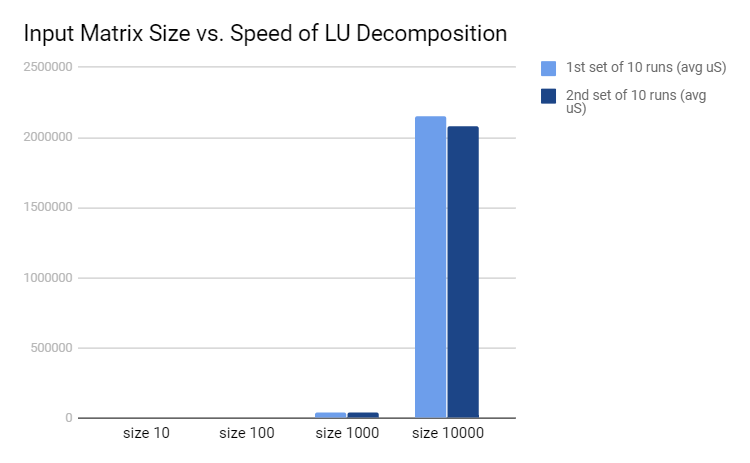
**Design alternatives**

One design alternative to the block LU decomposition we implemented would be using the just the standard LU decomposition to factorize the matrix. However, the standard LU decomposition algorithm applies to the block LU decomposition, and we are able to speed up our computation by processing more data at a time. Alternatively, we could use MPI with GPU to further speed up the computation or having multiple CPUs with GPUs.

**Performance Results:**

We ran two set of different matrix A. Each set would contain a matrix of size 10, 100, 1000, and 10,000. We ran each set 10 times, and gathers the average run time for each size on two different set. Below is the data:

|  |  |  |
| --- | --- | --- |
| Matrix Size (m x n) | 1st set of 10 runs (avg uS) | 2nd set of 10 runs (avg uS) |
| size 10 | 1220 | 1184 |
| size 100 | 5025 | 4380 |
| size 1000 | 40326 | 39414 |
| size 10000 | 2151017 | 2083625 |



As the input size is below 1000, the increase in the running time is almost close to O(n). As the input size increase from 1000 to 10000, there is a significant jump in the running time. This can happen due to that LU decomposition only run parallel on the part of the matrix at a time. As the input size increases, the ratio of parallelism part vs. sequencial part reduces. Further study would include gathering more sample data at a large size to better characterize the performance of the algorithm.

**Conclusion:**

Block LU Decomposition is an efficient way to solve linear systems of equations. Parallelism execution at standard LU decomposition, matrix multiplication speeds up the computational time. LU factorization is implemented on CUDA and its performance shows promising results on small sets of data. Future study would include benchmarking the algorithm with larger size of input test data, and improve the efficiency of the algorithm by experimenting using MPI and GPU.

REFERENCES

NOCTUA CG TOOLS, http://blog-noctua.rhcloud.com/index.php/2011/04/21/lu-matrix-decomposition-in-parallel-with-cuda/