Nasikan, Dmytro; Yaremenko, Vadym

Article

Performance evaluation of LU matrix decomposition using the SYCL standard

Reference: Nasikan, Dmytro/Yaremenko, Vadym (2023). Performance evaluation of LU matrix decomposition using the SYCL standard. In: Technology audit and production reserves 3 (2/71), S. 6 - 9.
https://journals.uran.ua/tarp/article/download/284518/278782/656353.
doi:10.15587/2706-5448.2023.284518.

UDC 004.42
DOI: 10.15587/2706-5448.2023.284518

Dmytro Nasikan,
Vadym Yaremenko

PERFORMANCE EVALUATION OF LU MATRIX DECOMPOSITION USING THE SYCL STANDARD

The object of this study is the performance of the SYCL standard tools when solving the LU matrix decomposition problem. SYCL is a fairly new technology for parallel computing in heterogeneous systems, so the topic of evaluating the performance of the standard on specific tasks in the field of parallel computing is relevant. In the study, the algorithm of parallelized LU decomposition of a square matrix was implemented by means of the SYCL standard and standard C++, and an experiment was conducted to test the implementation in a heterogeneous system with several types of processors. During testing, the program received square matrices of various dimensions as input, and the output was the execution time of the LU schedule on the selected processor. The obtained results, presented in the form of tabular and graphic data, show the advantage of the implementation of the SYCL standard over ordinary C++ by more than 2 times when using a graphics processor. It was experimentally shown that the implementation on SYCL is almost not inferior in speed to the implementation on ordinary C++ when executed on a central processor. Such results are caused both by the high possibility of parallelizing the LU schedule algorithm itself, and by the great work of the developers of the standard on its optimization.

The obtained results indicate the possibility of speeding up the solution of the LU decomposition of the matrix and similar algorithms by means of SYCL when using heterogeneous systems with processors optimized for data parallelism. The results of the study can be used in justifying the choice of technology for solving LU matrix decomposition problems or problems with a similar parallelization scheme.

Keywords: *SYCL standard tools, parallel computing, LU decomposition, SYCL performance, numerical methods.*

Received date: 02.05.2023
Accepted date: 12.06.2023
Published date: 30.06.2023

How to cite
Nasikan, D., Yaremenko, V. (2023). Performance evaluation of LU matrix decomposition using the SYCL standard. Technology Audit and Production Reserves, 3 (2 (71)), 6–9. doi: https://doi.org/10.15587/2706-5448.2023.284518

1. Introduction

The SYCL standard is beginning to gain more and more attention from both application developers and researchers [1]. Regardless, SYCL is a relatively new standard of parallelizing and hardware-accelerating C++ applications on heterogeneous systems. Questions often arise regarding the niche and optimality of using the standard to solve certain classes of problems in the field of parallel computing, one of which is the *LU* decomposition of a matrix.

Existing literature and research on this topic do not address the topic of evaluating the performance of *LU* scheduling using SYCL. For example, this problem solving was not included to the «SYCL-bench» test suite [2]. Also, similar studies have been conducted for other technologies, such as OpenCL, CUDA or HIP [3–5]. Despite the presence of a certain relations with them, the results of these works cannot be extended to SYCL. Given this situation, it can be considered that the research topic is relevant.

The aim of this research is to implement the *LU* matrix decomposition algorithm using SYCL to evaluate its effectiveness on different backends in comparison with the classical parallel implementation. This will help to make a more thorough choice of technologies for solving problems in the future and will provide an implementation example of the algorithm using SYCL.

2. Materials and Methods

In this study the performance of the *LU* decomposition of the matrix using the SYCL standard for parallel computing in heterogeneous systems is evaluated.

As is known from the basic course of matrix algebra, any non-degenerate square matrix A can be represented as the product of two triangular matrices L and U, where the matrix L is lower triangular, and the matrix U is upper triangular. At the same time, the dimensions of the matrices L and U coincide with the dimension of the matrix A [6]:

$$A = L \cdot U. \quad (1)$$

Therefore, the goal of the *LU* decomposition problem is to find the matrices L and U. Classical algorithms for finding the *LU* decomposition of a matrix are based on the well-known Gaussian method or are its partial modifications [7].

For example, in the algorithm that uses elementary transformations of matrix rows, the decomposition matrices are found according to recurrence formulas (2) and (3) [8]:

$$U_{si} = A_{si} - \sum_{k=1}^{s-1} L_{sk} \cdot U_{sj},\ j = 1, n; \quad (2)$$

$$L_{is} = A_{is} - \sum_{k=1}^{s-1} L_{ik} \cdot U_{ks}, j = s+1, n. \quad (3)$$

In this work, two parallelized algorithms were used to find the *LU* decomposition of the matrix. The first algorithm, the pseudocode of which is described on the snippet 1, is a classic implementation of a general algorithm for parallel execution on a computer. This implementation involves the parallel computation of each row of the matrix *A* during its decomposition in parallel [9].

```
LU Decomposition (A):
n=number of rows in matrix A
Let L be an n×n lower triangular matrix initialized
as identity matrix
Let U be an n×n upper triangular matrix initialized
as zero matrix

for k=1 to n:
U[k][k]=A[k][k]
for i=k+1 to n:
L[i][k]=A[i][k]/U[k][k]
U[k][i]=A[k][i]

parallel for j=k+1 to n:
A[i][j]=A[i][j]–L[i][k]·U[k][j]

return L, U
```

Snippet 1: classic parallel LU decomposition algorithm pseudocode [9].

The second algorithm is a slightly modified variation of the first one and allows using massive parallelism and the SIMD architecture of hardware accelerator cores.

```
LU Decomposition (A):
n=number of rows in matrix A
Let L be an n×n lower triangular matrix initialized
as identity matrix
Let U be an n×n upper triangular matrix initialized
as zero matrix

for k=1 to n:
U[k][k]=A[k][k]

parallel for i=k+1 to n:
L[i][k]=A[i][k]/U[k][k]
U[k][i]=A[k][i]

parallel for i=k+1 to n:
parallel for j=k+1 to n:
A[i][j]=A[i][j]–L[i][k]·U[k][j]

return L, U
```

Snippet 2: data-parallel *LU* decomposition algorithm pseudocode [9].

By separating the operations of finding the elements of matrices *L* and *U* and calculating the next state of the matrix in different cycles, it is possible to speed up the execution of the algorithm on certain hardware accelerators that have light SIMD cores [9].

During the practical part of the work, both algorithms were implemented in the C++ programming language. The first algorithm was implemented using standard C++11 threading tools. The second was implemented using SYCL 2020 rev. 3 [10].

The following software infrastructure was used to conduct the experiment:

– C++11;
– gcc;
– SYCL ComputeCpp-2.11.0;
– SYCL DPC++2023.1.0.

The following backends were used to execute the second algorithm:

– OpenCL NVIDIA CUDA 11.8 backend;
– OpenCL Intel backend;
– NVIDIA CUDA 11.8 backend.

The experiment was conducted on a heterogeneous system with the following configuration:

– CPU – x86_64 11th Gen Intel(R) Core(TM) i5-11400H@2.70GHz;
– GPU – NVIDIA GeForce RTX 3050 To Laptop GPU;
– RAM – 16GB;
– OS – Manjaro Linux, kernel – 5.15.108-1-MANJARO.

During the experiment, the algorithms were using the heterogeneous system and software tools listed above. During execution, each of the programs received as input a square matrix *A* of different dimensions and its *LU* decomposition was found. The time taken to perform was measured for further analysis.

3. Results and Discussion

The result of the study is the time evaluation of the implemented algorithms for matrices of different dimensions on each of the platforms. These results are presented in the Table 1.

As it is mentioned, 15 tests were performed for each platform with an initial matrix size of 1000×1000 and a step of 1000. At the same time, the minimum number of elements is 1 million, and the maximum is 225 million. Execution time varied from a few tens of milliseconds to 20 minutes. For clarity, the results were presented graphically in the Fig. 1.

The graph in Fig. 1 illustrates the dependence of execution time on the number of matrix elements. As can be seen, when executed on a central processor (CPU) both using the SYCL standard and the usual C++11 parallelization tools give almost the same result. This is not surprising, since the tested processor has 12 physical cores, which during testing are 100 % occupied with useful work.

As expected, the NVIDIA GPU execution using DPC++'s Native CUDA backend performed the best, outperforming the CPU execution by almost a factor of two.

However, ComputeCpp's OpenCL CUDA driver has the worst performance, averaging 1/3 times the execution time on a CPU. Such results are explained by serious gaps in driver optimization – its support was stopped several years ago.

In general, analyzing the obtained data, it can be concluded that the use of the SYCL standard to solve the LU decomposition problem can increase the performance of the solution when performed in a heterogeneous system, if it has hardware accelerators that support massive parallelism in the data parallelism paradigm. However, when executed on the central processor, both implementations show approximately the same speed.

ISSN 2664-9969

Table 1

LU decomposition benchmark results

Matrix dimension	Number of elements	Processing time			
		ComputeCpp OpenCL CUDA Nvidia 3050 Ti, c	ComputeCpp OpenCL Intel Core i5-11400H, c	Native C++ threading (12 threads), c	DPC++ CUDA Nvidia 3050 Ti, c
1000	1.00E+06	0.287	0.115	0.398	0.095
2000	4.00E+06	2.941	0.798	2.482	0.926
3000	9.00E+06	9.26	3.891	7.472	3.314
4000	1.60E+07	21.186	10.528	16.197	8.104
5000	2.50E+07	40.256	20.868	30.684	16.188
6000	3.60E+07	69.839	36.975	52.407	28.145
7000	4.90E+07	107.531	59.728	81.326	45.181
8000	6.40E+07	157.045	90.011	127.99	67.18
9000	8.10E+07	225.308	140.701	170.823	96.204
10000	1.00E+08	303.839	207.852	232.319	132.295
11000	1.21E+08	402.758	287.888	308.394	175.12
12000	1.44E+08	513.328	383.379	399.074	226.099
13000	1.69E+08	659.079	497.839	504.891	287.8
14000	1.96E+08	825.754	631.822	629.679	359.183
15000	2.25E+08	997.38	783.543	772.416	438.459

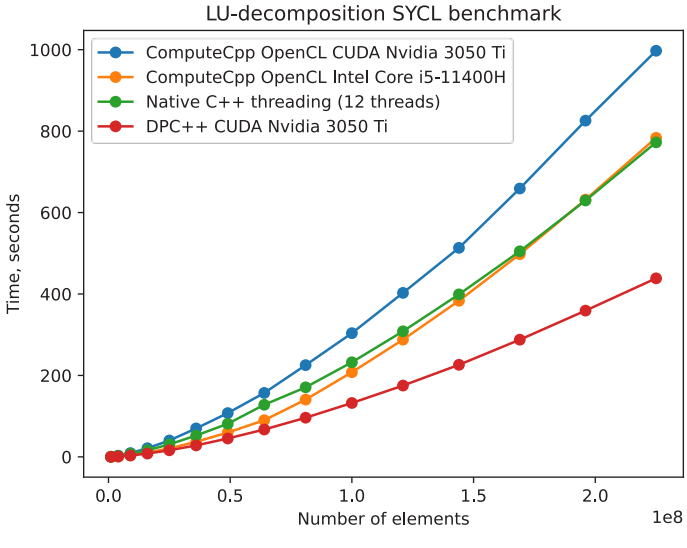

Fig. 1. *LU* decomposition benchmark graphical results

The results of a practical experiment and its analysis can be practically used when choosing a technology at the stage of planning a software tool for solving problems of a similar class that require high parallelization. Also, developments in the form of adaptation of the algorithm and its implementation by means of SYCL in the form of raw code can be used for integration into existing systems.

The limitations of this study are testing the implementation of the algorithms on one heterogeneous system. Additional testing on a larger number of hardware accelerators would allow for a more objective assessment of the results and form a list of platforms on which execution will provide a guaranteed acceleration.

Also, the conditions of martial law in Ukraine have a certain negative influence on this research. Thus, during the experimental part of the research, problems arose in while trying to use the GPU hardware of the university department – certain parts of the functionality are temporarily unavailable for use due to the difficulties of their maintenance in the conditions of martial law. That's why the testing is done using our personal computers.

4. Conclusions

In this work, the performance of the SYCL standard tools for solving the matrix *LU* decomposition problem was investigated. During the study, the implementation of the algorithm on SYCL was implemented and its execution was tested on several heterogeneous systems. The results were also compared with the classic C++11 implementation. As a result of the research, it can be concluded that using SYCL to solve the *LU* decomposition problem can speed up execution by more than two times, when selecting a heterogeneous system containing suitable hardware accelerators.

It is possible to suggest the extension of this rule to other tasks similar to the *LU* decomposition, from the point of view of the organization of the parallel part. However, this assumption needs further verification, which is beyond the scope of this work.

Conflict of interest

The authors declare that they have no conflict of interest in relation to this research, whether financial, personal, authorship or otherwise, that could affect the research and its results presented in this paper.

Financing

The research was performed without financial support.

Data availability

The manuscript has no associated data.

References

1. Alpay, A., Heuveline, V. (2020). SYCL beyond OpenCL: The architecture, current state and future direction of hipSYCL. *Proceedings of the International Workshop on OpenCL*. doi: https://doi.org/10.1145/3388333.3388658
2. Lal, S., Alpay, A., Salzmann, P., Cosenza, B., Hirsch, A., Stawinoga, N. et al. (2020). SYCL-Bench: A Versatile Cross-Platform Benchmark Suite for Heterogeneous Computing. *Lecture Notes in Computer Science*. Cham: Springer, 629–644. doi: https://doi.org/10.1007/978-3-030-57675-2_39
3. Diop, T., Gurfinkel, S., Anderson, J., Jerger, N. E. (2013). DistCL: A Framework for the Distributed Execution of OpenCL Kernels. *2013 IEEE 21st International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems*, 556–566. doi: https://doi.org/10.1109/mascots.2013.77
4. Ozcan, C., Sen, B. (2012). Investigation of the performance of LU decomposition method using CUDA. *Procedia Technology, 1*, 50–54. doi: https://doi.org/10.1016/j.protcy.2012.02.011
5. Ghysels, P., Synk, R. (2022). High performance sparse multifrontal solvers on modern GPUs. *Parallel Computing, 110*, 102897. doi: https://doi.org/10.1016/j.parco.2022.102897
6. Mittal, R. C., Al-Kurdi, A. (2002). LU-decomposition and numerical structure for solving large sparse nonsymmetric linear systems. *Computers & Mathematics with Applications, 43 (1-2)*, 131–155. doi: https://doi.org/10.1016/s0898-1221(01)00279-6
7. Lambers, J. (2021). *«The LU Decomposition» in MAT 610 – Numerical Linear Algebra, Sec. 3.2*. Available at: https://www.math.usm.edu/lambers/mat610/class0125.pdf
8. Yang, A., Liu, C., Chang, J., Guo, X. (2020). Research on Parallel LU Decomposition Method and its Application in Circle Transportation. *Journal of Software, 5*, 1250–1255. doi: https://doi.org/10.4304/jsw.5.11.1250-1255
9. Peng, S., Tan, S. X.-D. (2020). GLU3.0: Fast GPU-based Parallel Sparse LU Factorization for Circuit Simulation. *IEEE Design & Test, 37 (3)*, 78–90. doi: https://doi.org/10.1109/mdat.2020.2974910
10. *SYCL Working Group, «SYCL™ 2020 Specification (revision 7)»* (2023). The Khronos Group. Available at: https://registry.khronos.org/SYCL/specs/sycl-2020/pdf/sycl-2020.pdf

Dmytro Nasikan, *Department of System Design, National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute», Kyiv, Ukraine, ORCID: https://orcid.org/0009-0007-1840-4344*

✉ ***Vadym Yaremenko,*** *Postgraduate Student, Assistant, Department of System Design, National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute», Kyiv, Ukraine, e-mail: yaremenko.v.s@gmail.com, ORCID: https://orcid.org/0000-0001-8557-6938*

✉ *Corresponding author*