OpenCL and CUDA Comparison of MapReduce Performance on Distributed Heterogeneous Platform through Integration with Hadoop Cluster

Vincent Karovič[a], Mateusz Kaźmierczak[b], Oleksandr Pankiv[b], Maciej Górkiewicz[b], Maryana Zakharchuk[c], Roksolyana Stolyarchuk[c]

[a] *Comenius University in Bratislava, Faculty of Management, 820 05 Bratislava, Slovakia*
[b] *Technical University of Lodz 90-924 Łódź, Poland*
[c] *Lviv Polytechnic National University, S. Bandera str., 12, Lviv, 79013, Ukraine*

Abstract

An effective processing of Big Data in various application areas is an important task today. Modern development of information technology provides the ability to calculate a large number of different tasks using a certain number of computers in distributed mode. MapReduce technology allows to perform distributed calculations on a huge amount of data by dividing them into parts, performing parallel calculations of each of them and combining the results. In this paper, experimental studies were performed to compare CUDA and OpenGL frameworks performance measurements for MapReduce operations on heterogeneous cluster. It has been found that CUDA is a more suitable framework that provide a significant advantage in this regard. It is determined that the greater the amount of processing data, the greater the delay caused by OpenCL. Further research will be conducted to determine the energy consumption of both technologies.

Keywords 1

OpenCL, CUDA, performance evaluation, Hadoop cluster

1. Introduction

Dramatic demand for usage of big data processing technologies has been observed during last few years. Simultaneously, general-purpose GPU accelerated computation frameworks develops. Both techniques are being used in similar fields eg. data science [1], data processing [2], data mining [3], machine learning [4], [5], solving various biological [6], [7], medical [8], [9], physical [10] and geographical [11] problems. Consequently, many research studies have been made to compose scalability of big data processing systems with maximization of resources usage and performance of GPU powered computations. Those attempts aim improving performance [12], power-usage [13], deal with low capacity of GPU memory [14] and improve programmability [2] of this approach

Computation using graphic processors can speed up any type of calculations. The technique is called GPGPU – general-purpose GPU [15]. It is possible to implement MapReduce's reduce operation using previously mentioned GPU computing frameworks in order to speed-up whole algorithm. GPGPU enables to run a code on way more cores than standard CPU. Although GPU is not designed for general usage and has significantly less memory that CPU [16], well designed algorithm can speed up computations of many different kinds

The main goal of this paper is to compare performance of basic algorithm which aggregates results by sum. We test both CUDA and OpenCL™ implementations. CUDA and OpenCL are frameworks

IT&AS'2021: Symposium on Information Technologies & Applied Sciences, March 5, 2021, Bratislava, Slovakia
EMAIL: vincent.karovicml@fm.uniba.sk (V. Karovič); maryana.zk@gmail.com (M. Zakharchuk); Roksoliana.R.Stoliarchuk@lpnu.ua (R.Stolyarchuk)
ORCID: 0000-0001-9946-7329 (V. Karovič); 0000-0002-2641-5947 (M. Zakharchuk); 0000-0003-3521-1425 (R. Stolyarchuk)
© 2021 Copyright for this paper by its authors.
Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).
CEUR Workshop Proceedings (CEUR-WS.org)

CEUR-WS.org/Vol-2824/paper20.pdf

which enable using GPU for non-graphic related computing [17]. They provide GPU acceleration for mass parallel computation that can potentially speed-up MapReduce algorithms execution [14].

To achieve our goal we use Apache™ Hadoop® [18] powered heterogeneous cluster. Algorithms are implemented according to MapReduce paradigm in CUDA and OpenCL. Apache Hadoop is a software framework for distributed computing dedicated for big data processing. It environment consist of connected workstations forming a cluster.

MapReduce is a paradigm for constructing algorithms dedicated to run on distributed environment [19]. It is supported by Apache Hadoop framework. It allows to spread process of computing among workstations connected into Hadoop cluster. It enables relatively fast processing of large amount of data by making the process parallel. Hadoop implements scheduling algorithm for optimal nodes resources usage and techniques for error recovery

We measure performance by measuring time taken to execute widely used MapReduce algorithms implemented with two leading GPU general purpose computation frameworks - CUDA and OpenGL. CUDA [17] is a framework dedicated for NVIDIA graphics cards while OpenCL [20] is multi-platform framework which computes using every fond CPU and GPU resource available on a host machine. This means that what we measure is an overhead introduced by OpenCL in the specific environment of Hadoop based heterogeneous cluster.

Other approaches includes using other distributed computation frameworks such as MPI. However, these are lesser fault-tolerant than Hadoop [1], [21]. On the other hand, consequence of using Hadoop and MapReduce paradigm forces redesigning data processing algorithm so as they fit into specific schema. Some techniques of visual programming may be applied to overcome this issue [2], but it does not ensure correctness of designed solution

There are many different systems that strive in the field of some kind of GPU powered MapReduce paradigm implementation. For example:

1. HAPI [22]

Hadoop combined with Aparapi – Java-to-OpenCL conversion tool developed and released by AMD. Proposes easy and ready to use API for designing and implementation of GPU MapReduce algorithms. By hiding complexity of GPU programming by system of annotations, allows programmers to focus on developing good algorithms.

2. HadoopCL [1]

Extension to Apache Hadoop, HadoopCL combines Hadoop and OpenCL by usage of Aparapi. Provides easy and flexible programming interface, guarantees reliability and low power consumption. States that it achieves nearly 3x overall speedup and 55x of computational sections of example MapReduce application algorithms.

3. GPMR [23]

GPMR is a stand-alone library for MapReduce that is supposed to use GPU clusters for large scale computing. By modifying MapReduce to combine large amounts of map and reduce items into chunks and partial reductions and accumulation, they better utilize power of GPU.

4. MITHRA [24]

An architecture that combines power of NVIDIA CUDA and Apache Hadoop to create scalable performance gains by utilizing MapReduce programming model. MITHRA was designed especially for executing computing tasks of massive and independent data.

5. MARS [25]

Mars is a MapReduce framework that is supposed to improve and ease programming complexity of GPU programming by a familiar MapReduce interface.

Authors of those systems compared their solutions with other works in terms of performance – however, reliable comparison of most popular GPGPU computation framework, that would be free of overhead ensuing from their framework usage, is still lacking

2. Proposed architecture

For the purpose of our research we have built a cluster containing 10 workstations, each of them hosting HortonWorks® Apache Hadoop framework implementation. Workstations uses two different physical configuration – differing with presence of GPU. Configuration consist of: Intel® Xeon® CPU E5-1630 v3 3.70 GHz, 16 GB RAM and optional NVIDIA Quadro K4200.

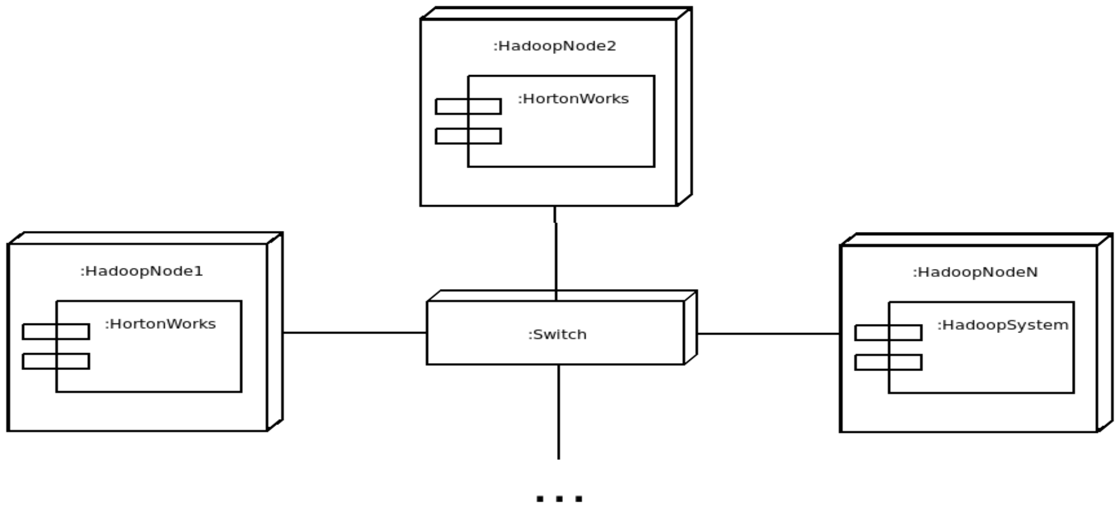

Figure 1: Architecture diagram. Own work

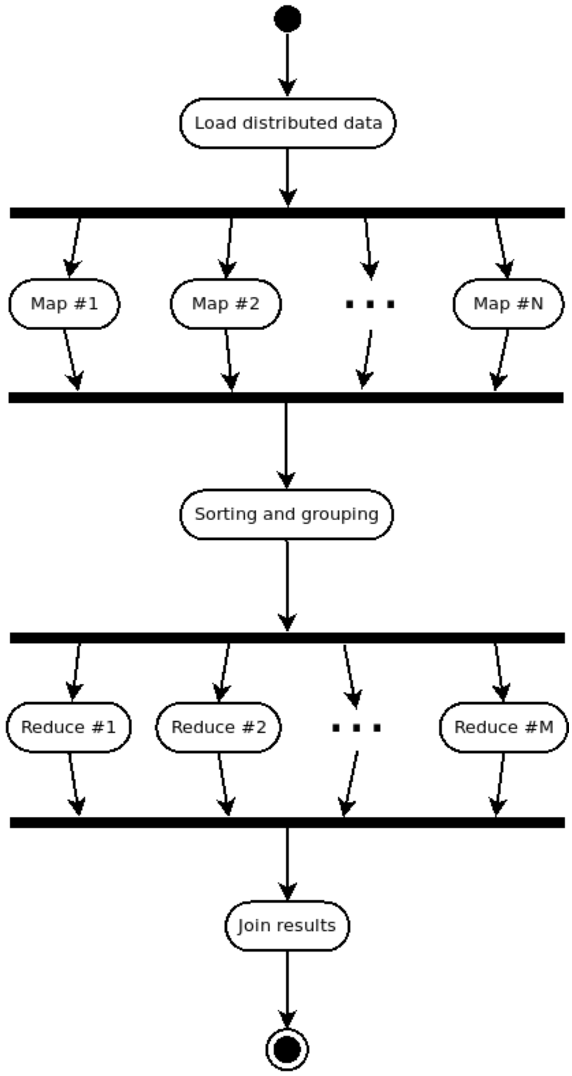

Figure 2: MapReduce data flow diagram. Own work.

Workstation are organized in star topology; they communicate with each other through a switch. Complete architecture can be found on Figure 1.

The area of work presented in this article are Apache Hadoop powered nodes. Hadoop schedules map and reduce tasks to be executed on a free node, preferably near to the place where input data are stored. In our approach this can lead to non-effective node usage because of a presence of GPU is irrelevant for scheduler when making a decision on which node certain task has to be executed. Some works have already been done in this field [26].

Our approach is to implement Reduce operation to move calculations and some algorithms from CPU to GPU.

3. Aggregation algorithm development

Basic idea behind our project is speeding up reduce operation (see Figure 2) using algorithms designed specifically for GPUs. As an example, we would use modified algorithm for calculation of exponent.

Exponent can be calculated using following formula [27]:

$$e^x = \sum_{n=0}^{\infty} \frac{x_n}{n!} \tag{1}$$

It is easy to notice that sum factors are independent, and as a result it is easy to calculate overall value by just summing up middle results (in any order).

When creating algorithm of fast sum calculation we inspired with NVIDIA publication [28] on parallel computation.

Basic idea behind it is following: having an array of numbers on length N, we can sum up pairs of elements, so complexity reduces from *O(N)* to around *O(log_2 (n))* (Figure 3).

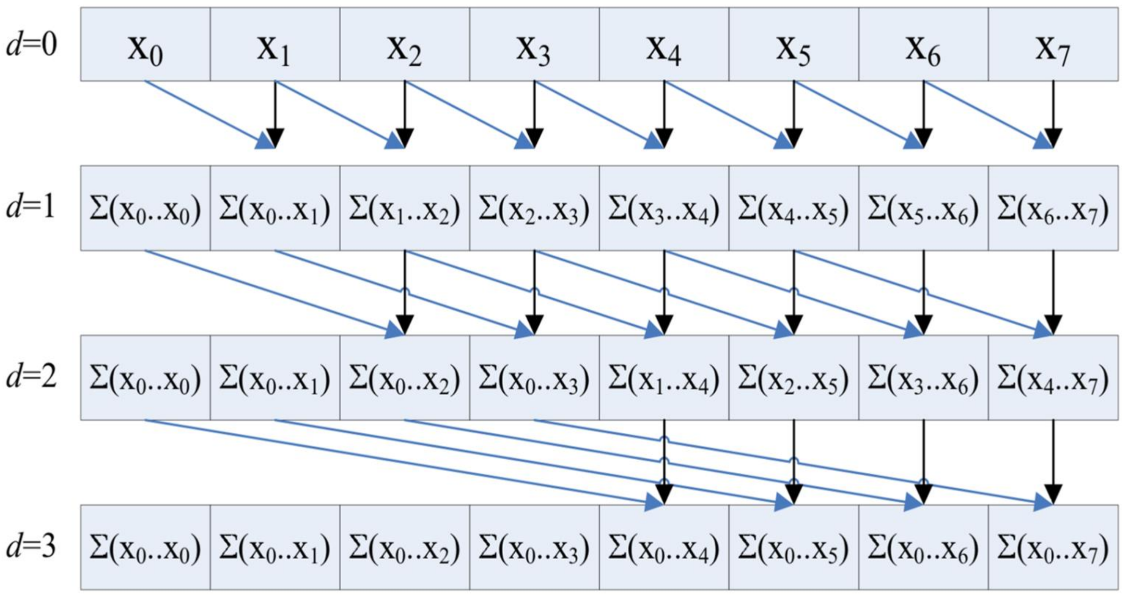

Figure 3: Illustration of naive summing algorithm. Source: [28]

As single operation is around simultaneous, speed-up is quite impressive. In order to achieve that with MapReduce in Hadoop we created modified implementations of map and reduce.

Map output was (key, value) pair where key was always 1, and value was n-th factor of the sum. As a result, after reduce operation we always got single key, and value which represents a result

4. Methodology

In this section, we describe methodology of our time performance analysis and comparison of MapReduce solution on both CUDA and OpenCl.

To measure a performance of the system we run several tests of exponent calculation, each of them differing number of jobs with and without GPU for both frameworks separately. The first test was run on only one GPU powered node. Then we have tested every number of nodes each kind starting from one to 10 total amount of nodes. All results were presented on charts below alongside with its discussion.

We measure performance, by measuring runtime of tests on all used nodes from sending data to each node, to receiving data from each node on master machine. Results often are shown as an improvement factor. This should be considered as execution time in certain case divided by execution time for non-GPU solution.

All the tests were run 5 times and presented results are average value. It was necessary due to the cache memory misses. Nevertheless, some small differences in similar measurements can be still observed.

5. Results

On the Figure 4 linear increase of computation time depending on the number of calculated elements can be observed. For our comparison the most significant information is difference between CUDA and OpenCL improvement. The results shows that usage of OpenCL generates latency in availability of computation results that linearly depends on the number of elements.

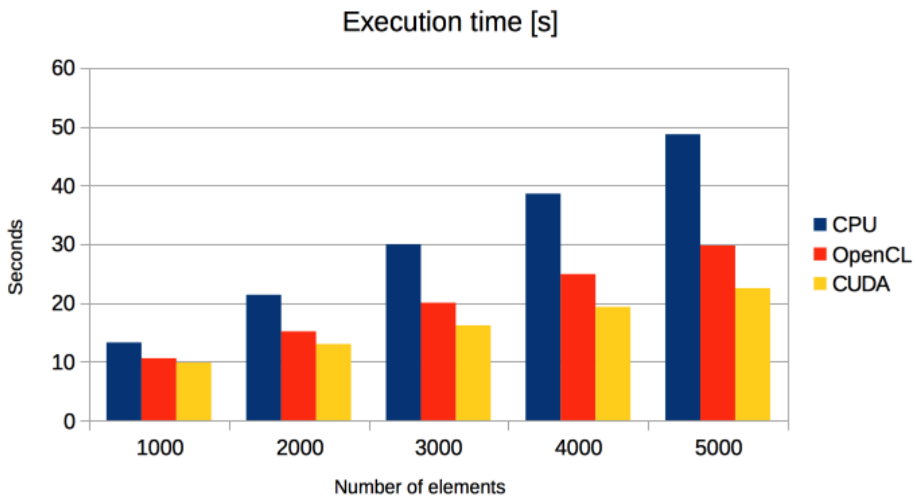

Figure 4: Execution time for various number of elements. Own work.

That means difference in improvement factor of execution time (Figure 5) between those two frameworks grows. These data were measured for very small – in terms of big data processing – number of elements. Big data algorithms usually work with millions of items. That implies that simple summation and memory move operations powered by OpenCL in big data analysis, when it comes to practical usage, can lead to significant superiority of CUDA over OpenCL

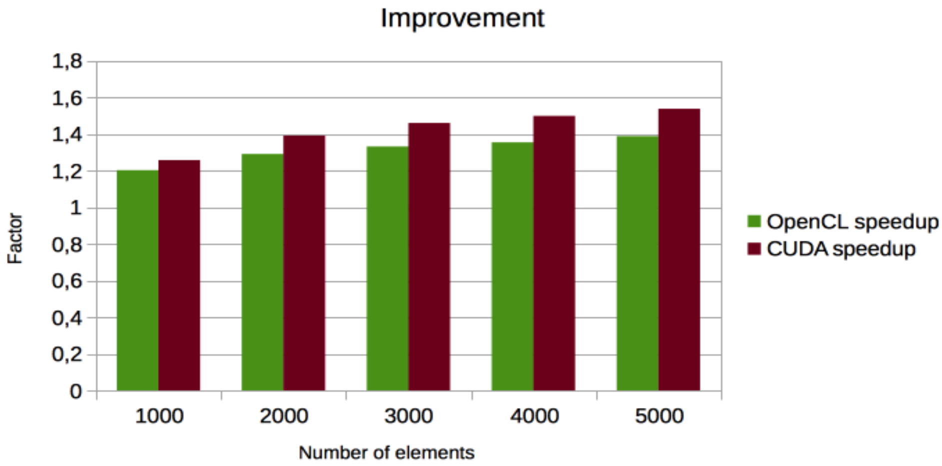

Figure 5: Improvement factor basing on execution time for each tested number of elements. Own work.

6. Conclusion

We have presented comparison of CUDA and OpenGL frameworks performance measurements for MapReduce operations on heterogeneous cluster. The results shows noticeable superiority of CUDA in this issue. Specifically, the biggest amount of is processed, the bigger OpenCL caused latency occurred to be. Nevertheless, it should be mentioned that CUDA is supported only by GPUs of the only one of two leading graphical cards manufacturers – namely NVIDIA. Therefore, this research does not solve the problem of the right framework choice for particular task on particular hardware resources available. Moreover, the comparison can be also done in the field of energy-saving [13].

7. References

[1] M. Grossman, M. Breternitz, and V. Sarkar, "HadoopCL: MapReduce on Distributed Heterogeneous Platforms through Seamless Integration of Hadoop and OpenCL," in 2013 IEEE International Symposium on Parallel Distributed Processing, 2013, pp. 1918–1927.

[2] Aneta Poniszewska-Maranda, Daniel Kaczmarek, Natalia Kryvinska, Fatos Xhafa, "Studying usability of AI in the IoT systems/paradigm through embedding NN techniques into mobile smart service system", Computing, November 2019, Volume 101, Issue 11, pp. 1661–1685.

[3] W. Ma, G. Agrawal, A Translation System for Enabling Data Mining Applications on GPUs, in Proceedings of the International Conference on Supercomputing, NY, USA, 2009, p. 400–409.

[4] R. Tkachenko, I. Izonin, N. Kryvinska, I. Dronyuk, K. Zub, "An Approach towards Increasing Prediction Accuracy for the Recovery of Missing IoT Data based on the GRNN-SGTM Ensemble", MDPI Sensors 2020, 20(9), 2625, https://doi.org/10.3390/s20092625.

[5] X. Li, M. Grossman, and D. Kaeli, "Mahout on Heterogeneous Clusters Using HadoopCL," in Proceedings of the 2Nd Workshop on Parallel Programming for Analytics Applications, New York, NY, USA, 2015, pp. 9–16.

[6] Z. Yin, et. al, "Computing Platforms for Big Biological Data Analytics: Perspectives and Challenges," Comput. Struct. Biotechnol. J., vol. 15, no. Supplement C, pp. 403–411, Jan. 2017.

[7] N. Khare, A. Khare, and F. Khan, "HCudaBLAST: an implementation of BLAST on Hadoop and Cuda," J. Big Data, vol. 4, no. 1, p. 41, Dec. 2017.

[8] E. Serrano, et. al., "Architecture for the Execution of Tasks in Apache Spark in Heterogeneous Environments," in Euro-Par 2016: Parallel Processing Workshops, 2016, pp. 504–515.

[9] O. Tymchenko, B. Havrysh, O. Khamula, B. Kovalskyi, S. Vasiuta, and I. Lyakh, 'Methods of Converting Weight Sequences in Digital Subtraction Filtration', in 2019 IEEE 14th International

Conference on Computer Sciences and Information Technologies (CSIT), Sep. 2019, vol. 2, pp. 32–36, doi: 10.1109/STC-CSIT.2019.8929750.
[10] A. Aji et al., "Hadoop GIS: A High Performance Spatial Data Warehousing System over Mapreduce," Proc VLDB Endow, vol. 6, no. 11, pp. 1009–1020, Aug. 2013.
[11] G. D. Guerrero et al., "A performance/cost model for a CUDA drug discovery application on physical and public cloud infrastructures," Concurr. Comput. Pract. Exp., vol. 26, no. 10, pp. 1787–1798, Jul. 2014.
[12] R. Tkachenko, et. al., 'Piecewise-linear Approach for Medical Insurance Costs Prediction using SGTM Neural-Like Structure', in Proceedings of the 1st International Workshop on Informatics & Data-Driven Medicine (IDDM 2018), Lviv, Ukraine, Nov. 2018, vol. 2255, pp. 170–179.
[13] W. Chen et al., "GPU Computations on Hadoop Clusters for Massive Data Processing," in Proceedings of the 3rd International Conference on Intelligent Technologies and Engineering Systems (ICITES2014), Springer, Cham, 2016, pp. 515–521.
[14] B. Zhao, et. al., "GPU-Accelerated Cloud Computing for Data-Intensive Applications," in Cloud Computing for Data-Intensive Applications, Springer, New York, NY, 2014, pp. 105–129.
[15] W. Ai, K. Li, C. Chen, J. Peng, and K. Li, "DHCRF: A Distributed Conditional Random Field Algorithm on a Heterogeneous CPU-GPU Cluster for Big Data," in 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), 2017, pp. 2372–2379.
[16] CUDA Zone, NVIDIA Developer. Available: https://developer.nvidia.com/cuda-zone. [2018].
[17] "Welcome to ApacheTM Hadoop®!" [Online]. Available: https://hadoop.apache.org/. [2018].
[18] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," Commun ACM, vol. 51, no. 1, pp. 107–113, Jan. 2008.
[19] A. Munshi, "The OpenCL specification," in 2009 IEEE Hot Chips 21 Symposium, 2009, p. 1-314.
[20] R. Searles, S. Herbein, and S. Chandrasekaran, "A Portable, High-Level Graph Analytics Framework Targeting Distributed, Heterogeneous Systems," in 2016 Third Workshop on Accelerator Programming Using Directives (WACCPD), 2016, pp. 79–88.
[21] Y. Lin, S. Okur, and C. Radoi, "Hadoop+Aparapi: Making heterogenous MapReduce programming easier," hgpu.org, Apr. 2012.
[22] J. A. Stuart and J. D. Owens, "Multi-GPU MapReduce on GPU Clusters," in 2011 IEEE International Parallel Distributed Processing Symposium, 2011, pp. 1068–1079.
[23] I. Dronyuk, O. Fedevych, N. Kryvinska, "Constructing of Digital Watermark Based on Generalized Fourier Transform", MDPI Electronics 2020, 9(7), 1108; ISSN 2079-9292; CODEN: ELECGJ, https://doi.org/10.3390/electronics9071108.
[24] B. He, W. Fang, Q. Luo, N. K. Govindaraju, and T. Wang, "Mars: A MapReduce Framework on Graphics Processors," in Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques, New York, NY, USA, 2008, pp. 260–269.
[25] P. Zhezhnych, O. Berezko, K. Zub, and I. Demydov, 'Analysis of Features and Abilities of Online Systems and Tools Meeting Information Needs of HEIs' Entrants', CEUR-WS.org, vol. 2616, pp. 76–85, 2020.
[26] N. Lotoshynska, I. Izonin, M. Nazarkevych, and S. Fedushko, "Consumer-centered design of the secondary packaging for industrial pharmaceuticals," CIRP Journal of Manufacturing Science and Technology, vol. 32, pp. 257–265, 2021, doi: https://doi.org/10.1016/j.cirpj.2021.01.001.
[27] H. Mohammadinejad and F. Mohammadhoseini, "Privacy Protection in Smart Cities by a Personal Data Management Protocol in Blockchain," IJCNIS, vol. 12, no. 3, pp. 44–52, Jun. 2020, doi: 10.5815/ijcnis.2020.03.05.
[28] M. Mitra and A. Chowdhury, "A Modernized Voting System Using Fuzzy Logic and Blockchain Technology," p. 9, 2020.
[29] Andrukhiv A, Sokil M, Fedushko S, Syerov Y, Kalambet Y, Peracek T. Methodology for Increasing the Efficiency of Dynamic Process Calculations in Elastic Elements of Complex Engineering Constructions. Electronics. 2021; 10(1): 40. https://doi.org/10.3390/electronics10010040
[30] M. Abramowitz and I. A. Stegun, Handbook of Mathematical Functions With Formulas, Graphs and Mathematical Tables. National Bureau of Standards, 1964.
[31] M. Harris, "Parallel Prefix Sum (Scan) with CUDA," NVIDIA Corporation, 2007.