UDC 004.451.44

Demchyk Valerii, Tyzun Vitalii,
Rusanova Olga, Korochkin Aleksandr

THE ORGANIZATION OF PARALLEL COMPUTATIONS IN HETEROGENEOUS COMPUTING SYSTEMS

Демчик Валерій, Тизунь Віталій,
Русанова Ольга, Корочкін Олександр

ОРГАНІЗАЦІЯ ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ В ГЕТЕРОГЕННИХ КОМП'ЮТЕРНИХ СИСТЕМАХ

The article deals with the method of computing in heterogeneous multicore CPU+GPU systems. The results of an investigation of the effectiveness of methods for the task of text recognition using the technology of neural networks.

Key words: CPU, GPU, core, thread, parallelism, granularity, fork-join.
Fig.: 4. Bibl.: 13.

В статті розглядається спосіб організації обчислень в багатоядерних гетерогенних CPU+GPU системах. Наводяться результати дослідження ефективності методики для задач розпізнавання тексту з використанням технологій нейронних мереж.

Ключові слова: CPU, GPU, ядро, потік, паралелізм, зернистість, fork-join.
Рис.: 4. Бібл.: 13.

Relevance of research topic. Heterogeneous computer systems (HCS) are computer systems that contain several heterogeneous computing elements. HCS with CPU / GPU architecture it's a type of HCS systems equipped with a graphic processor. In a simplified format, the GPU model can be described as a set of a large number of simple processing elements of the same type. Each of these elements is much cheaper than its CPU analog. This contributed to the significant development of the GPU in terms of increasing the number of cores in it, and at present, the average number of cores in the GPU reaches a half a thousand. However, as practice has shown, if a heterogeneous computer system, which includes both the CPU and the GPU, is not busy working with complex graphics, then the computing power, that is presented in the current GPU, is superfluous, calculations are conducted so fast that most of the time the GPU cores idle in waiting for the next task.

Using this powerful and cheap computing power is a clear and logical solution of the problem of increasing computing productivity. With the integration into the

GPU of programmable shader blocks, it became possible to program normal user computing on this device. This technique was called GPGPU (General-purpose Computing on Graphics Processing Units) and became an important breakthrough in the development of modern computer technology. The first Nvidia graphics processors that support CUDA technology (Compute Unified Device Architecture - the implementation of GPGPU technology from this company) were not cheap and affordable. However, other GPGPU implementations appeared quite quickly, including the OpenCL framework, which allows you to program for all types of GPU, not only for Nvidia, but, for example, AMD. Also, apart from the fact, that OpenCL supports more GPUs than CUDA, these GPUs are usually cheaper and simpler, this technology is usable even with regular graphic cards.

Formulation of the problem. When programming for the CPU+GPU HCS, the most important part is to understand in which cases it is advisable to connect the GPU calculations, and in which cases better performance can be gained only by the use of the CPU. It's important to understand, when the increase in GPU performance reduces delays due to the data transfers between it and the CPU. Also, since both components of such a heterogeneous system are capable of computing, a possible option for concurrent computing that a CPU may not wait all the GPU computing time, but process some of the calculations on its own computing power. Therefore, it is important to find a balance, an optimal division of the input data between tasks, which will provide minimizing of the idle time during heterogeneous computing.

Analysis of recent research and publications. In recent years, more and more scientific articles and diploma papers about a calculation using graphic processors have appeared [4-12]. However, an overwhelming majority of them consider this question only from the point of view of choosing the better device for calculations between CPU and GPU. And there is not so much works about using both of devices for calculations at the same time. Moreover, in most of them the problem is considered in the context of solving classical problems of linear algebra, cryptography, and implementations of hyperparallel test algorithms [5-8]. Features of the deployment of high-speed neural networks are considered [9] only on the examples of some typical problems for neural networks, among which there is no one example of the most popular tasks - recognition and classification of images. The solution to this problem within a heterogeneous computer system is presented in one work [10], but only for mobile systems, which are obviously less developed and productive, than classical stationary systems.

The question of finding an optimal distribution of computational load between the CPU and the GPU was also considered only in the context of solving typical mathematical problems and implementation of test parallel algorithms [11].

Identification of unexplored parts of the general problem. Research of the efficiency of parallel computing in heterogeneous computer systems is usually carried

out on the classical problems for such studies - vector-matrix operations and implementations of various test parallel algorithms [5-8]. However, the main area in which today such systems are actually used are systems of artificial intelligence and neural networks. This is explained by the fact that by itself the neural network involves the simultaneous execution of a large number of elementary tasks, its structure is similar to the structure of the GPU. It is necessary to show the described problem on one of the classical tasks of this sphere.

It is also important to consider, that the heterogeneity of the system, and, accordingly, the need for the exchange of data between its components can lead to a situation, where the time, spent on the preparation and transmission of data and the collection of results, can slow down the acceleration of distributed computing. Therefore, it is necessary to consider in more detail the search for an optimal distribution of the data between the components of the system in order to provide a minimal idle time of one computing processor relatively to the other.

Also, in this work an own method for increasing the efficiency of parallel computing in heterogeneous computer systems is proposed, which is based on the application of combined parallelism [1] [2].

Problem statement. The task is to develop a program for recognizing text on images based on a neural network and focused on parallel work in a heterogeneous computer system. The program has to implement combined parallelism, as well as the ability to divide the percentage of processing data between the CPU and the GPU. The input data for a task is an image with letters, numbers, characters in the PNG format, as well as the percentage of GPU loading. Output data is the text string and the time it took to receive result.

Developed program has to be tested in various heterogeneous computer systems and show conclusions about the effectiveness of the proposed approach.

Combined parallelism. The simplified version of the CPU / GPU interaction scheme looks like this: from the CPU through the interface (for the external GPU it is PCI - Peripheral component interconnect) a set of instructions that must be performed on each core of the GPU is sent; Through the GPU controller, each core is configured for these instructions; The CPU then sends a set of data with which the GPU has to work; The GPU controller distributes this data between available cores, and after completing computing, it collects the results and sends it to the CPU. The instruction sets for relatively simple GPUs are small in size (several rows of program code) and simple in their structure (usually elementary mathematical and logical operations). We can say that the cores of the graphics processor itself implements the fine-grained parallelism calculations. However, with the advent of PCI interfaces of 2.1 version and above in heterogeneous computer systems, the possibility of parallel data transfer between CPU and GPU has appeared, which allows you to organize combined parallelism in the system.

This approach is based on the simultaneous use in the program of two types of parallelism: medium-grained and fine-grained. At the same time, the parallel program includes a set of traditional threads along the number of cores of the CPU (parallelism of medium-grain size). Each of these threads implements fine-grained parallelism by creating sub-threads using appropriate Fork-Join tools [7]. These small threads are used only for calculations. Additionally, as noted above, each of the medium-grain threads can interact with the GPU without delay, sending the necessary data to the graphics processor and taking the results of its work. In the GPU, the calculations are transferred to a large number of small threads, each on a separate core.

Previous studies [1] [2] proved the effectiveness of combined parallelism in the organization of overloaded parallel computing.

Analysis of the means of implementation. The main part of the program is written using the C# language. This multiparadigm language allows you to write program code for necessary task quickly and conveniently. In addition, in previous studies [1] [2], the means of C# language demonstrated their high efficiency in the implementation of all types of parallelism, including the combined one.

Calculations on the graphics processor are organized by the OpenCL framework, because it integrates seamlessly with the C# language and supports a large lineup of conventional GPUs. In addition, existing research shows its high performance, at the level of Nvidia CUDA [12].

Test results. Testing of programs was carried out on two different heterogeneous computer systems, which had the following parameters:

1. CPU: Intel i5-7200u, 2 cores, 4 threads, maximum frequency 3.1 GHz. GPU: AMD Radeon R5 M420, 1030 MHz, 320 processors, 2 GB of memory;

2. CPU: Intel Core i7-7700HQ, 4 cores, 8 threads, maximum frequency 3.8 GHz. GPU: NVIDIA GeForce GTX 1050 Ti, 1030 MHz, 768 processors, 4 GB of memory;

Software: Windows 10, .NET Framework 4.7, OpenCL 2.2.

Graphs below showing the dependence of the program's running time on the number of threads involved in it, as well as the data distribution between the CPU and the GPU. The graphs are presented separately for the situation, when the program processed a large amount of data (text recognition of 1000 characters), and for the situation, where the program worked with a small amount of data (200-character text recognition). Graphics are shown for both systems (1-2) in which the testing was conducted.

The following two graphs show the dependence of the program's performance on the amount of text that is submitted for recognition for cases where all data processing is carried out only on one of the elements of the system. The point of intersection on these graphs shows the turning point of the dependence. That is, if you submit text that is larger than this volume, then the efficiency of the use of recognition calculations on the CPU is reduced, and the efficiency of the use of the graphics processor is increased.

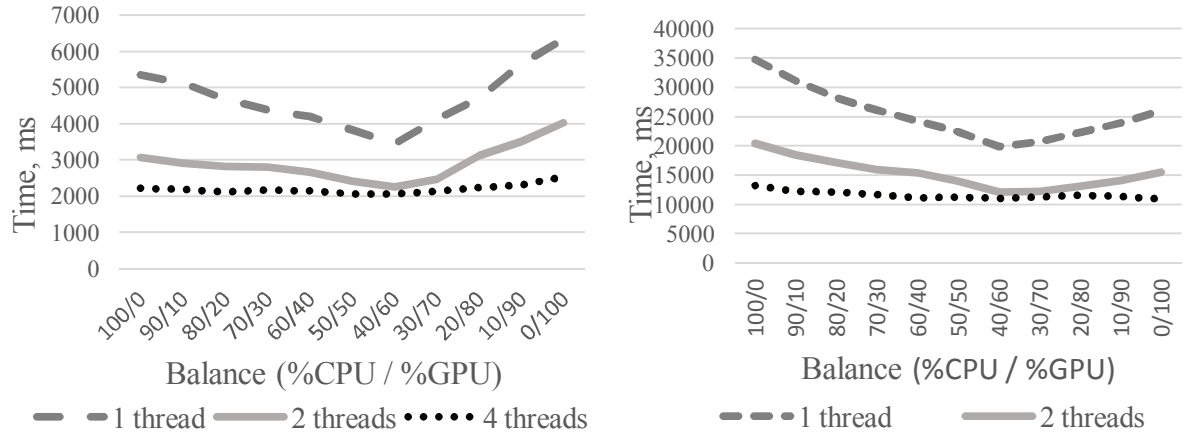

Fig. 1. Graphic of calculation speed depending on data distribution.
200-character text (left) and 1000 characters (right). System 1

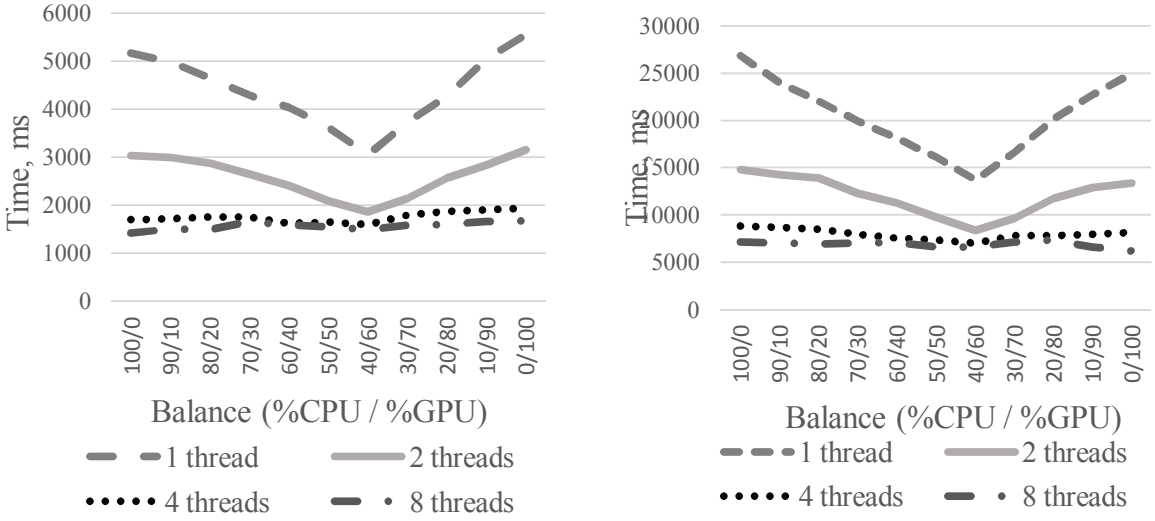

Fig. 2. Graphic of calculation speed depending on data distribution.
200-character text (left) and 1000 characters (right). System 2

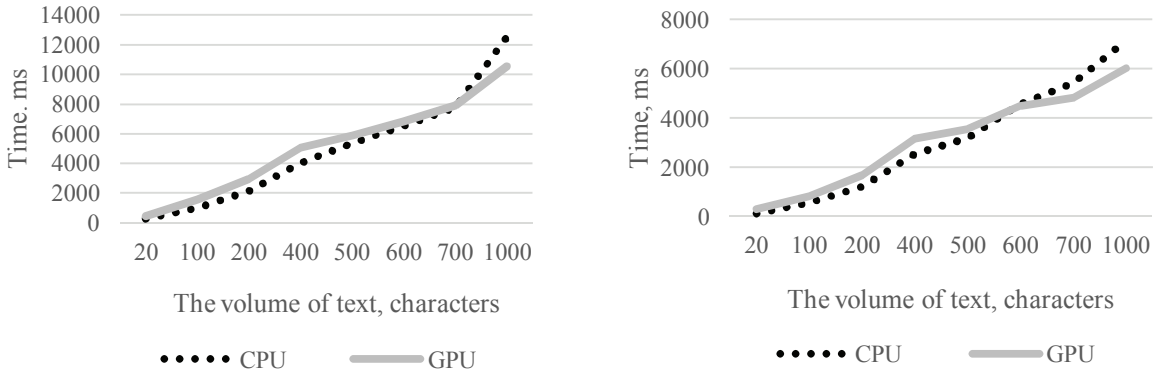

Fig. 3. Graph of the calculation time dependence of the program on the volume of text when recognizing on individual elements. Systems 1 (left) and 2 (right)

The following two graphs show the dependence of the program's performance on the amount of text that is submitted for recognition. The data in this case is distributed as follows: 40% for the CPU and 60 % for the GPU, since, as seen from the previous graphs, this kind of distribution achieves maximum program performance in all cases.

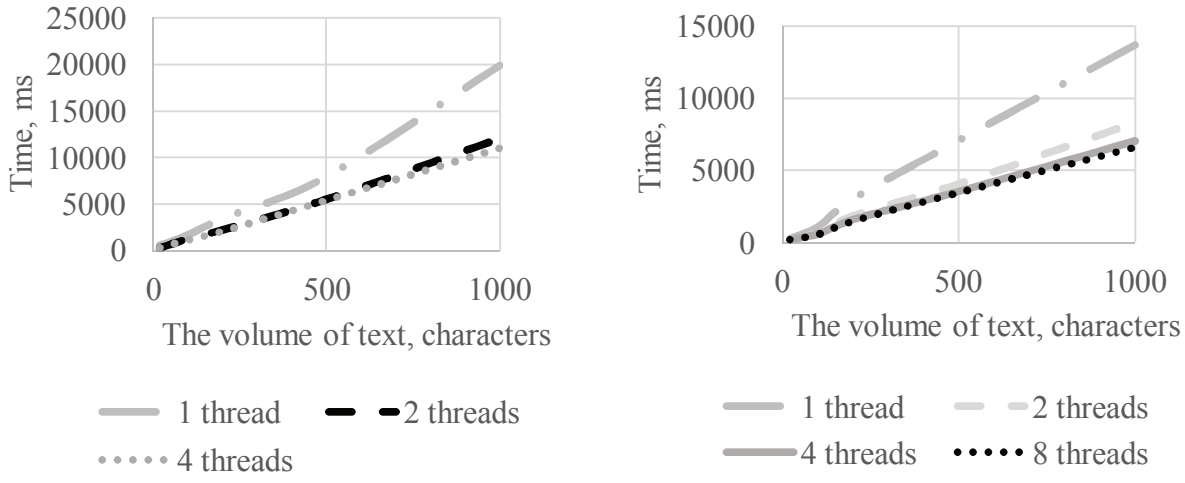

Fig. 4. Graph of calculation time according to the volume of text and number of threads. Balance point is 40/60. Systems 1 (left) and 2 (right)

Conclusions. The results of the test showed the effectiveness of heterogeneous computing systems in implementing the solution of the problem by means of C# and OpenCL. Applying the calculations to the graphics processor, along with the calculations at the CPU, allowed to significantly reduce the program execution time. In the process of research, in practice, the hypothesis of the existence of some of the most effective balance of data split between the CPU and the GPU has been confirmed.

Regarding the direct speed of the program work, the turning point can be considered as a text recognition of 600 characters. From the results it is clear that in the process of recognizing the text of less than 600 characters, the loss of time due to the exchange of data with the GPU is quite critical. It is because of sending a large percentage of data to the GPU nullifies the entire increase in performance from its use. It is desirable to send relatively a small amount of data to the graphics processor, so that it is accepted before the completion of computing on the central processor. In the case of text recognition over 600-700 characters, the situation is diametrically opposed. In this case, it is advisable to use the central processor only the role of administration and a small amount of computations, and the main part of data should be send to the graphics processor.

For all target systems considered, regardless of the amount of data processed by the program, the balance of 60/40 was the most optimal, that is, if the program

organizes the distribution of data, so that 60% of it is sent to the GPU, and the remaining 40% was left to process on the CPU, then it will provide the minimal idle, which allows you to get the most possible performance. Further offset of the balance in the direction of the GPU (for processing more than 600 characters) or CPU (for processing less than 600 characters) led to insignificant increases of performance, compared with the previous increases.

Additionally, it should be noted that the optimal balance obtained for the pattern recognition and classification problem is somewhat different from the optimal balance for linear algebra problems [11] in the direction of more GPU calculations.

The use of combined parallelism has also reduced the calculation time. With each subsequent added thread, you can observe a proportional decrease in the program's running time. It is also shows that the balance of calculations on the central and graphics processor remained unchanged with each subsequent added flow, since, on the one hand, the computing speed increased on the CPU, and on the other hand, the number of threads of interaction with the GPU increased, which turned into reduced the idle of GPU cores and idle due to data transfer between the CPU and the GPU.

So, in all cases, the most effective is the maximum possible use of cores and threads on the central processor, no matter how much we use the parallel calculations ob graphics processor.

Based on the foregoing, it can be admitted that the best approach to implement a text recognition system is to conduct preliminary testing on a target heterogeneous computer system, which will take extra time, but will ensure that the most effective proportion of the data distribution on this system is found.

References

1. Демчик В. В. Дослідження ефективності дрібнозернистого паралелізму в багатоядерних комп'ютерних системах / В. В. Демчик, О. В. Корочкін, О. В. Русанова // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка : зб. наук. праць. – К. : Век+, 2018. – № 66. – С. 56 – 61.

2. Демчик В. В. Застосування дрібнозернистого паралелізму для підвищення ефективності паралельних та розподілених обчислень / В. В. Демчик, О. В. Корочкін // Безпека. Відмовостійкість. Інтелект. Збірник праць міжнародної науково-практичної конференції ICSFTI2018. Київ, Україна, 10-12 травня 2018 р. / КПІ ім. Ігоря Сікорського –К. : КПІ ім. Ігоря Сікорського, Вид–во «Політехніка», 2018. – С. 362 – 368.

3. Жуков І.А., Корочкін О.В. Паралельні та розподілені обчислення: Навч. посібник [Текст]. – К.: Корнійчук, 2005. – 226 с. – ISBN 996-7599-36-1.

4. Hyesoon Kim, Richard Vuduc, Sara Baghsorkhi. Performance Analysis and Tuning for General Purpose Graphics Processing Units (GPGPU). — Morgan & Claypool Publishers, 2012. — 96 p.

5. Lin Cheng. Intelligent scheduling for simultaneous CPU-GPU applications by thesis // Graduate College of the University of Illinois at Urbana-Champaign, 2017 Urbana, Illinois [Електронний ресурс]. Режим доступу: http://rsim.cs.uiuc.edu/Pubs/Lin_thesis.pdf

6. Victor W Lee, Changkyu Kim, Jatin Chhugani, Michael Deisher, Daehyun Kim, Anthony D. Nguyen, Nadathur Satish, Mikhail Smelyanskiy, Srinivas Chennupaty, Per Hammarlund, Ronak Singhal and Pradeep Dubey. Debunking the 100X GPU vs. CPU Myth: An Evaluation of Throughput Computing on CPU and GPU // Throughput Computing Lab, Intel Corporation Intel Architecture Group, Intel Corporation [Електронний ресурс]. Режим доступу: https://www.academia.edu/36236172/Debunking_the_100X_GPU_vs._CPU_Myth_An_Evaluation_of_Throughput_Computing_on_CPU_and_GPU

7. T. Brandes, A. Arnold, T. Soddemann, D. Reith. CPU vs. GPU - Performance comparison for the Gram-Schmidt algorithm // Eur. Phys. J. Special Topics 210 – K.: EDP Sciences, Springer-Verlag, 2012. – № 210. – C. 73–88.

8. Haneesha H. K., Chandrashekhar B. N., Lakshmi H., Sunil. Performance Evaluation of CPU-GPU with CUDA Architecture Hybrid Computing, R&D // Nitte Meenakshi Institute of Technology, Bangalore-64.

9. Amr M. Kayid, Yasmeen Khaled, Mohamed Elmahdy. Performance of CPUs/GPUs for Deep Learning workloads // The German University in Cairo [Електронний ресурс]. Режим доступу: https://www.researchgate.net/publication/325023664_Performance_of_CPUsGPUs_for_Deep_Learning_workloads

10. Sipi Seppälä. Performance of Neural Network Image Classifcation on Mobile CPU and GPU // Aalto University MASTER'S THESIS 2018 [Електронний ресурс]. Режим доступу: https://pdfs.semanticscholar.org/946d/3f843ea93f22cc9c7e30af42a682139ad1e6.pdf

11. Ana Lucia Varbanescu. Heterogeneous CPU+GPU computing // University of Amsterdam. [Електронний ресурс]. Режим доступу: http://www.es.ele.tue.nl/~heco/courses/ASCI-schools/ASCI_springschool_2017/ASCI_HetCompCPU-GPU_part1.pdf

12. Kamran Karimi, Neil G. Dickson, Firas Hamze. A Performance Comparison of CUDA and OpenCL. - D-Wave Systems Inc. [Електронний ресурс]. Режим доступу: https://arxiv.org/ftp/arxiv/papers/1005/1005.2581.pdf

13. Lea, Doug. A Java Fork/Join Framework, In Proceedings of ACM Java Grande 2000 Conference (San Francisco, California, June 3-5, 2000)

Authors

Demchyk Valerii – student, Department of Computer Engineering, National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”.

E-mail: kirintor830@gmail.com

Демчик Валерій Валентинович – студент, кафедра обчислювальної техніки, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського».

Tyzun Vitalii – student, Department of Computer Engineering, National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”.

E-mail: vitaliy.tyzun@gmail.com

Тизунь Віталій Юрійович – студент, кафедра обчислювальної техніки, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського».

Rusanova Olga – docent, candidate of Technical Sciences, Department of Computer Engineering, National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”.

E-mail: olga.rusanova.v@gmail.com

Русанова Ольга Веніамінівна – доцент, кандидат технічних наук, кафедра обчислювальної техніки, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського».

Korochkin Aleksandr – docent, candidate of Technical Sciences, Department of Computer Engineering, National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”.

E-mail: avcora@gmail.com

Корочкін Олександр Володимирович – доцент, кандидат технічних наук, кафедра обчислювальної техніки, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського».

EXTENDED ABSTRACT

Demchyk Valerii, Tyzun Vitalii,
Rusanova Olga, Korochkin Aleksandr

THE ORGANIZATION OF PARALLEL COMPUTATIONS IN HETEROGENEOUS COMPUTING SYSTEMS

Relevance of the research. The method of computing in multi-core heterogeneous CPU+GPU systems using the technology of «combined parallelism» is considered. The results of research on the effectiveness of the method for text recognition task using the technology of neural networks are presented.

Target setting. The heterogeneity of the computational elements of a heterogeneous computer system leads to the problem of a long idle time, when one element of the system is waiting another to complete its task, which leads to total delay in the entire system.

Actual scientific researches and issues analysis. Over the past few years, there are more articles on this topic, but they simply choose between a CPU or a GPU, and only classical problems of linear algebra and hyperparallel test algorithms are considered.

Uninvestigated parts of general matters defining. The lack of research on the optimal distribution of computations between different elements of the system. Lack of research on solving real problems, such as text recognition with neural network technologies.

The research objective. The objective is to develop a program for text recognition based on a neural network and focused on parallel work in a heterogeneous computer system. The program has to implement combined parallelism, as well as the ability to adjust the percentage of processing on the CPU and GPU. It is necessary to test the developed program in various heterogeneous computer systems.

The statement of basic materials. A description of the main ideas and approaches that were implemented during the research. The extensive testing of the developed program in several different real heterogeneous computer systems has been carried out.

Conclusions. The proposed method for organizing calculations in heterogeneous computer systems has shown its effectiveness. Also, the hypothesis of the existence of an effective load distribution for the considered task between the elements of the target system was confirmed: 40% on the CPU and 60% on the GPU.

Key words: CPU, GPU, core, thread, parallelism, granularity, fork-join.