IMPLEMENTING THE GRAPH MODEL OF THE SPREAD OF A PANDEMIC ON GPUS

Vladimir Sudakov[1,2,a], Nikita Yashin[1]

[1]*Plekhanov Russian University of Economics*

[2]*Keldysh Institute of Applied Mathematics (Russian Academy of Sciences)*

E-mail: [a]sudakov@ws-dss.com

Modeling the spread of viruses is an urgent task in modern conditions. In the created model, contacts between people are represented in the form of the Watz and Strogatz graph. We studied graphs with tens of thousands of vertices with a simulation period of six months. The paper proposes methods for accelerating computations on graph models using graphics processors. In the considered problem, there were two resource-intensive computational tasks: generating an adjacency matrix of a graph that simulates the presence of contacts between people and traversing this graph in order to simulate infection. The calculations were carried out in sequential mode and with acceleration on GPUs. The modeling system software is implemented using the Cuda, CuPy, PyTorch libraries. The calculations were carried out using the Tesla T4 graphics accelerator. Compared to computations without using graphics accelerators, their application gave a 7-fold increase in speed.

Keywords: graph, pandemic, graphics accelerator, simulation, small world

Vladimir Sudakov, Nikita Yashin

1. Introduction

Computing on graphs is a large cluster of problems with a wide range of applications. Modeling interactions between objects in a group is one such challenge. However, this type of problem is resource-intensive in the case of a graph with a huge number of vertices. For computations on graphs, an adjacency matrix is used, which in full form takes $O(n^2)$ time to complete a round trip, where n is the number of vertices. There are several ways to improve the execution speed of the algorithm - using the multi-threading approach on the CPU or porting the calculations to CUDA. This study explored the second approach. Behavioral comparison and speed of execution of the algorithm at the CPU and the GPU. Effectiveness of the technology was demonstrated on model of the interaction among people and the spread of the virus among them.

2. Literary review

In 2016, the GraphOps library was written [1]. This library performs its calculations on FPGA, which made it possible to accelerate computations, but the basis of this work was the creation of a library that accelerates work with graphs for the end user. There have also been several studies on speeding up certain algorithms that run on graphs. Improvement of the GlimmerHHM algorithm [2] consisted in parallelization and transfer of computations to GPU and FPGA, which led to acceleration up to 34 and 200 times, respectively. Semantic search [3] has been successfully accelerated on both GPU and FPGA. It was study of the possible benefits FGPA front of CPU and GPU, but the results show accelerating work on the GPU in 5.6 times in comparison with a CPU. Most of these papers investigate the use of FGPA. However, the use of these devices requires a lot of labor costs for the development and availability of computing equipment with FGPA, while graphics accelerators are more common, it is easier to start working with them, and there are many different libraries in different languages that support using of GPU. Regarding the issues of modeling pandemics using graphic processors, it is worth highlighting the work [4]. There are offered effective approaches to the treatment of co-occurrence matrices for large graphs on the GPU, but there are no algorithms that are designed to prepare these graphs. In the context of the threatening spread of COVID-19, approaches based on SEIR models are actively developing [5]. Basically, such models presuppose a description of pandemics through a system of differential equations that describe the situation well under typical conditions, but do not allow assessing the consequences of certain decisions on isolating the population at the micro level.

3. Method

The constructed model of virus propagation works on graphs. The vertices of the graph describe people, and the edges describe the presence of contacts between them through which the virus can be transmitted. Simulation is carried out with a fixed time step. Every unit of time, all the vertices of the graph are traversed and checked for a change in health status - whether the person has become infected, if he was not infected earlier, or whether the disease has passed to a new stage if the person is already infected.

In total, a person can have four conditions, which are taken from the SEIR model [5]. The first condition (Susceptible) - the person is healthy, he is not a distributor, but can be infected. Most people have this condition initially. The second state (Exposed) - a person has become infected, and the disease is in the incubation stage. Such a person cannot infect anyone yet and cannot be more infected. This condition is obtained after a successful test for transmission of infection from carriers with whom there were contacts. The third state (Infectious) - a person's disease has passed into the active stage. Such a person can infect others and is the only way to directly transmit the infection, but he himself cannot get the disease more. The state is obtained after a certain time has elapsed in the second state. The fourth and final state of a person (Recovered) is recovered or passed away. A sick person is no

longer contagious and cannot transmit the infection, and it is also believed that he has immunity and he himself can no longer become infected at a given simulation period. This state is obtained after a certain time in the third state.

In the proposed model, it is proposed to separate the types of contacts. A separate graph is built for each type of contacts. Next, the union of the resulting graphs is constructed. Testing model was carried out on two types of contacts. The first is constant contact that does not appear or disappear over time, for example, communication with relatives. Such contacts are described in a permanent graph that is created once before the state updates start. The second is an accidental contact, it can appear and disappear absolutely by accident, for example, an occasional meeting in a transport. Such contacts are described in a graph that is rebuilt every unit of time. Each time unit, these two graphs are combined, and the spread of the virus is calculated from the resulting graph.

For the software implementation of the model, the programming language Python3 was chosen. This language is very easy to learn, and has a huge variety of libraries, which allows you to write programs with a minimum of knowledge in the shortest possible time. Of the libraries used in the work – CuPy [6] and NetworkX [7], which were used to build graphs, and PyTorch [8], which can traverse graphs using their adjacency matrices.

First, you need to build a graph of the first type contacts. The so-called "small world" concept proposed in work [9] is well suited for solving this problem. It is built using the Watts-Strogatz algorithm, in two steps.

The first step is to build N vertices and build edges between each i-th and j-th vertices, if they satisfy the condition:

$$0 < |i - j| \bmod \left(N - 1 - \frac{K}{2}\right) \leq \frac{K}{2},$$

where K is the number of edges at each vertex.

The second step - each edge (i, j) with a chance β changes to (i, k), where $k \neq j$, edge (i, k) does not exist yet, k is a random vertex.

Thereafter must renew states. The first step in updating states is building a random graph. In this graph, each vertex has the same degree. There are two ways to construct such a graph. The first, using the CPU, is the fast_gnp_random_graph [10] function of the NetworkX library. The second, using the GPU, is the sparse.rand [11] function of the CuPy library. NetworkX, unlike CuPy, does not support using the power of graphics accelerators, so two different libraries are used. CuPy also creates directly an adjacency matrix, not a graph.

The second stage of updating states is directly traversing all vertices. It is done with the help of library Pytorch, which can perform calculations as the CPU, as well as on the GPU. If a specific person is not yet infected, the number N of his infectious contacts is considered and according to the formula

$$P = 1 - (1 - \alpha)^N,$$

where α is the probability of infection from one contact.

If a person has just passed to the second or third stage of infection, then for him, using the Monte Carlo method, the time spent in this state is modeled using the Weibull distribution. The applicability of the Weibull distribution to this problem was substantiated in [12].

4. Results

To check the results of the program, four programs were written with different levels of use of the graphics accelerator. Each of the programs was wrapped in a separate function, and the execution time of each function was measured. For measurement accuracy, each function was run 50 times and the average running time was taken. To compare the efficiency on different sizes of matrices, only one variable changed in the functions - the number of people in the model. Efficiency was compared for

different population sizes, and, accordingly, for different graph sizes. This value was measured in the range [1000, 17000] with a step of 1000.

Первая функция, no_cuda, не использовала графического ускорителя, все вычисления проводились исключительно на процессоре. Вторая функция, torch_cuda, использовала графический ускоритель только для моделирования непосредственно распространения вируса. Матрица смежности строилась на обычном процессоре. Третья функция, graph_cuda, использовала графический ускоритель для построения матрицы смежности, а распространение между людьми вычисляется на обычном процессоре. Четвертая функция, all_cuda, использовала графический ускоритель для обоих задач – и для построения матрицы смежности, и для расчетов распространения вируса.

The first function, no_cuda, did not use a graphics accelerator, all calculations were performed exclusively on the processor. The second function, torch_cuda, used a graphics accelerator only to simulate the actual spread of the virus. The adjacency matrix was built on a conventional processor. The third function, graph_cuda, used a graphics accelerator to build an adjacency matrix, and the spread between people was computed on a regular processor. The fourth function, all_cuda, used a graphics accelerator for both tasks - both for building an adjacency matrix and for calculating the spread of the virus.

Computing devices used: CPU: Intel Xeon CPU @ 2.00GHz; GPU: NVIDIA Tesla T4 16GB. The calculation results are shown in Tab. 1. A comparative graph is also built based on these data (Fig.1).

Table 1. Results of measurements of the speed of functions execution

Population size	**no_cuda**	**torch_cuda**	**graph_cuda**	**all_cuda**
1000	3.200957	3.351706	0.710170	0.912640
2000	7.914960	7.242812	1.510238	0.848995
3000	13.984233	12.124245	3.181422	1.496793
4000	21.029639	17.249926	5.338851	2.455780
5000	28.600931	22.572831	8.708607	3.728163
6000	37.634567	28.611021	11.137490	4.775039
7000	47.314849	35.154100	15.447932	6.286974
8000	57.854738	41.903814	19.260485	7.980263
9000	69.675968	50.306944	25.033169	9.963441
10000	84.130254	57.087363	29.785555	12.199988
11000	97.206411	65.925612	38.764165	14.762600
12000	112.067288	73.430345	42.834090	17.492042
13000	126.610486	80.314935	51.497435	20.050080
14000	141.328987	87.860316	58.368204	23.021266
15000	176.068139	97.052003	68.845642	26.358311
16000	215.929255	109.848109	76.498892	29.992050
17000	244.363839	119.475280	88.965665	33.416545

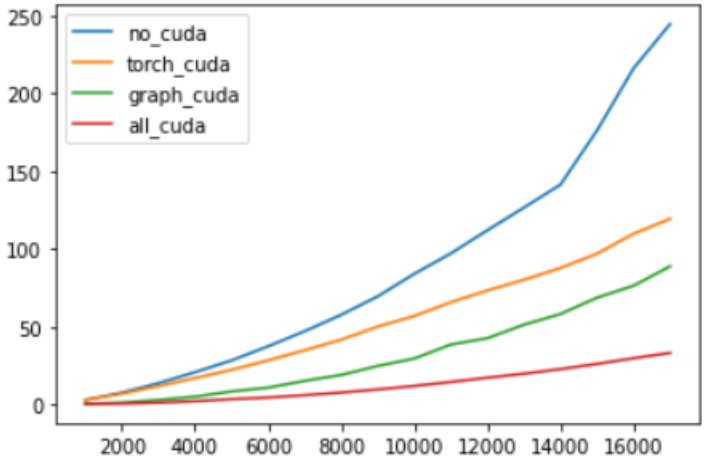

Figure 1. Comparing the change in the execution time of different functions with an increase in the size of the graph

As a result, it can be seen that the use of a graphics accelerator gives a significant gain in computing speed. This gain becomes more significant as the graph size increases. How many times a full cuda program is faster than a full CPU program is shown in Tab. 2.

Table 2. Effect of GPU computing for different graph sizes

Population size	Population size
1000	3.507361
2000	9.322746
3000	9.342799
4000	8.563324
5000	7.671588
6000	7.881520
7000	7.525854
8000	7.249728
9000	6.993163
10000	6.895929
11000	6.584640
12000	6.406758
13000	6.314712
14000	6.139062
15000	6.679796
16000	7.199550
17000	7.312660

On average, this value is approximately 7.152. Thus, the use of a graphics accelerator can give a 7-fold improvement in efficiency.

5. Conclusions

The advantage of using graphics accelerators to increase the speed of computations on graphs was investigated. A new approach to modeling contacts in pandemics based on combining graphs of different types is proposed. The use of the “small world” graph for modeling permanent contacts is investigated. The premium rate of different libraries of working on graphics processors for solving problems of this type has been investigated. Result showed approbation evident an advantage of using GPU, on graphs of high dimensionality. The proposed model of the spread of the virus makes it possible to assess the consequences of decisions to limit the number of contacts of different types.

6. Acknowledgement

The reported study was funded by RFBR and CNPq, FASIE, DBT, DST, MOST, NSFC, SAMRC according to the research project No. 20-51-80002.

References

[1] Tayo Oguntebi, Kunle Olukotun. GraphOps: A Dataflow Library for Graph Analytics Acceleration // FPGA '16: Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. Feb 2016. pp. 111–117 - DOI: 10.1145/2847263.2847337

[2] Nafsika Chrysanthou, Grigorios Chrysos, Euripides Sotiriades, Ioannis Papaefstathiou. Parallel accelerators for GlimmerHMM Bioinformatics Algorithm // Design, Automation and Test in Europe, Grenoble, France, March 14-18, 2011 - DOI: 10.1109/DATE.2011.5763024

[3] Abhinandan Majumdar, Hari Cadambi, Srimat T. Chakradhar, H.P. Graf. A parallel accelerator for semantic search // SASP '11: Proceedings of the 2011 IEEE 9th Symposium on Application Specific Processors, June 2011, pp. 122–128 - DOI: 10.1109/SASP.2011.5941090

[4] Peng Zou, Ya-shuai Lu, Ling-da Wu, Li-li Chen, Yi-ping Yao. Epidemic simulation of a large-scale social contact network on GPU clusters // SIMULATION, vol. 89, no. 10, Oct. 2013, pp. 1154–1172 - DOI:10.1177/0037549713482026.

[5] Wilfredo Angulo, José M. Ramírez, Dany De Cecchis, Juan Primera, Henry Pacheco, Eduardo Rodríguez-Román. A modified SEIR model to predict the behavior of the early stage in coronavirus and coronavirus-like outbreaks // Scientific reports vol. 11,1 16331. 11 Aug. 2021, – DOI:10.1038/s41598-021-95785-y

[6] CuPy software available at: https://cupy.dev/ (accessed 20.08.2021)

[7] NetworkX software available at: https://networkx.org/ (accessed 20.08.2021)

[8] PyTorch software available at: http://pytorch.org/ (accessed 20.08.2021)

[9] Watts, D. J.; Strogatz, S. H. Collective dynamics of 'small-world' networks // Nature 393. 1998. 440-442 - DOI:10.1038/30918.

[10] Networkx function fast_gnp_random_graph, documentation available at: https://networkx.org/documentation/stable/reference/generated/networkx.generators.random_graphs.fast_gnp_random_graph.html (accessed 20.08.2021)

[11] Cupy function sparse.rand, documentation available at: https://docs.cupy.dev/en/stable/reference/generated/cupyx.scipy.sparse.rand.html (accessed 20.08.2021)

[12] Jing Qin, Chong You, Qiushi Lin, Taojun Hu, Shicheng Yu, Xiao-Hua Zhou. Estimation of incubation period distribution of COVID-19 using disease onset forward time: a novel cross-sectional and forward follow-up study // SCIENCE ADVANCES, 14 Aug 2020, Vol 6, Issue 33 - DOI: 10.1126/sciadv.abc1202