INTERNATIONAL JOURNAL ON ORANGE TECHNOLOGY

https://journals.researchparks.org/index.php/IJOT e-ISSN: 2615-8140 | p-ISSN: 2615-7071

Volume: 4 Issue: 10 | October 2022

Using the Model in Cuda and Opencl for Medical Signals

Rakhimov Bakhtiyar Saidovich
Head of the Department of Biophysics and information technologies of Urgench branch of Tashkent Medical Academy, Uzbekistan
bahtiyar1975@mail.ru

Allayarova Asal Akbarovna
Senior teacher of the Department of Biophysics and information technologies of Urgench branch of Tashkent Medical Academy, Uzbekistan
asalakbarovna@gmail.com

Saidov Atabek Bakhtiyarovich
Student 4 – course Urgench branch of Tashkent University of Information Technologies named after Muhammad al Khwarizmi, Uzbekistan
ootabek2001@mail.ru

--***--

Annotation: The smallest computational unit in CUDA is a thread that runs on a scalar processor. This thread must be associated with one processor in the AGM. Further, the set of threads is combined into a computing unit, which is executed independently of other blocks on its multiprocessor. Ah, because AGM was developed on the basis of real graphic multiprocessors, then this computing unit must be associated with AGM. When developing a parallel computing kernel in CUDA, the processing of a single block is taken into account, because from one block it is not possible to change the calculation data from another.

Keywords: graph, basic functions, fast transformations.

In fact, the development of the kernel is the implementation of the PRAM algorithm for AGM, taking into account the block index and the index of each processor in this block. Thus, the number of threads in a block must be equal to the number of processors in the PRAM algorithm, and the shared memory of the multiprocessor is the shared memory for them. In addition, the kernel must provide for copying data from the global memory of the GPU to the shared memory of the multiprocessor and vice versa[1,6,7]. This type of operations is provided for by the proposed model. The number of blocks in CUDA is not limited and depends on the amount of data being processed, which must be presented in the form of a grid that unites all computing blocks. This concept of dividing data into blocks corresponds to dividing the processed data into AGMs, taking into account their independence from each other (see Section 2.2). In CUDA, all blocks are processed sequentially depending on the number of multiprocessors available. This factor is taken into account when estimating the time execution of the parallel algorithm on the GPU (see (2.5)). The proposed model of parallel computing on GPUs has a number of advantages over the CUDA model:

1. Allows you to estimate the execution time of a parallel algorithm depending on a particular GPU, taking into account only its essential features;
2. Simplifies the development of a parallel algorithm for a GPU using the already known PRAM parallel computing model, thereby hiding from the programmer the nuances associated with data parallelism.

Unlike OpenCL, the proposed parallel computing model is not common to all parallel computing devices, but is intended only for the CPU-GPU system. Accordingly, it does not consider job parallelism. Therefore, the

https://journals.researchparks.org/index.php/IJOT e-ISSN: 2615-8140 | p-ISSN: 2615-7071
Volume: 4 Issue: 10 | October 2022

proposed model can be considered a subset of the OpenCL model. The minimum computational unit in OpenCL is a work item that can be associated with one processor in an AGM (or one PRAM processor on a machine)[4.5]. Unlike OpenCL, the proposed model does not have private memory for each work item, which greatly simplifies the development of the program, relying on ready-made programming tools that solve issues with registers, addressing, etc. Naturally, the final program should fit in a limited amount of private memory in terms of volume for specific GPU. One OpenCL computational block must be associated with a data block that can be processed on one AGM. These blocks must be independent of each other so that their calculation can be scaled to many AGMs. The local memory of each computing unit is shared memory for the AGM. Also in OpenCL, as well as in the proposed model, from the computational unit there is access to the global memory of the GPU and the memory of constants. Unlike OpenCL, in the proposed model, these two types of memory are separated and caching of various levels is not taken into account, which makes it possible to simplify the estimation of the execution time of a parallel algorithm in the form of an upper limit on the execution time (a cache miss is always assumed when accessing the global memory of the GPU, constant memory - cache hit). Thus, the proposed parallel computing model is more simplified compared to OpenCL and more adapted to the CPU-GPU system, and also has the same advantages as over the CUDA model.

Results

Based on the analysis of architectures of graphic processors of various manufacturers, we can conclude that their main component is multiprocessors with shared memory. Therefore, the abstract GPU should also become the centerpiece of the GPU model.

A multiprocessor is a system containing several processors and one address space that is available for all processors [3]. This definition does not accurately reflect the specifics of GPUs. Therefore, we introduce the definition of an abstract GPU.

An abstract graphical multiprocessor is a system containing several scalar processors operating synchronously on the SIMD principle, and shared memory to which they have read and write access.

The proposed definition fits any graphics multiprocessor discussed in Chapter 1, while avoiding any particular characteristics, which is why the word "abstract" is included in the definition. The scalar processors that make up such a multiprocessor execute two types of instructions: arithmetic and memory access instructions. In turn, memory access instructions can be divided into local memory access instructions, which require a time comparable to arithmetic instructions, and GPU global memory (GRAM) access instructions, which require significantly more cycles to complete.

Reference

1. Rakhimov BS, Mekhmanov MS, Bekchanov BG. Parallel algorithms for the creation of medical database. J Phys Conf Ser. 2021;1889(2):022090. doi:10.1088/1742-6596/1889/2/022090
2. Rakhimov BS, Rakhimova FB, Sobirova SK. Modeling database management systems in medicine. J Phys Conf Ser. 2021;1889(2):022028. doi:10.1088/1742-6596/1889/2/022028
3. Rakhimov B, Ismoilov O. Management systems for modeling medical database. In: ; 2022:060031. doi:10.1063/5.0089711
4. Rakhimov BS, Khalikova GT, Allaberganov OR, Saidov AB. Overview of graphic processor architectures in data base problems. In: ; 2022:020041. doi:10.1063/5.0092848
5. RB Saidovich, SA Bakhtiyarovich, BB Farkhodovich, KDA Ugli, MMZ Qizi Analysis And Using of the Features Graphics Processors for Medical Problems Texas Journal of Medical Science 7, 105-110

https://journals.researchparks.org/index.php/IJOT e-ISSN: 2615-8140 | p-ISSN: 2615-7071
Volume: 4 Issue: 10 | October 2022

6. P. P. Kudryashov Algorithms for detecting a human face for solving applied problems of image analysis and processing: author. dis. Cand. tech. Sciences: 05.13.01. - M, 2007.
7. Tanenbaum E. Modern operating systems. 2nd ed. - SPb .: Peter, 2002 .-- 1040 p .: ill.
8. Forsyth DA, Pons, J. Computer vision. Modern approach / D.A. Forsyth, J. Pons: Trans. from English - M .: Publishing house "Williams", 2004. - 928 p .: ill. - Parallel. tit. English
9. Frolov V. Solution of systems of linear algebraic equations by the preconditioning method on graphic processor devices
10. Brodtkorb A.R., Dyken C., Hagen T.R., Hjelmervik J.M., Storaasli O.O. State-of-the-art in heterogeneous computing / A.R. Brodtkorb, C. Dyken, T.R. Hagen, J.M. Hjelmervik, O.O. Storaasli // Scientific Programming, T. 18, 2010. - S. 1-33.