UDC 004.932
DOI: 10.15587/1729-4061.2020.197095

IMPLEMENTATION OF A PARALLEL ALGORITHM OF IMAGE SEGMENTATION BASED ON REGION GROWING

Jesús Antonio Álvarez-Cedillo
PhD. Advance Technology*
E-mail: jaalvarez@ipn.mx
Mario Aguilar-Fernández
PhD. Industrial Engineering*
E-mail: maguilarfer@ipn.mx
Teodoro Álvarez-Sánchez
PhD Informatic Sciences
Section of postgraduate studies and research- IPN
Instituto Politécnico Nacional – CITEDI
Av. Instituto Politécnico Nacional No. 1310, Nueva Tijuana, Tijuana, Baja California, 22435
E-mail: talvarezs@citedi.mx
Raúl Junior Sandoval-Gómez
PhD Public relations*
E-mail: rsandova@ipn.mx
*Section of postgraduate studies and research of the interdisciplinary professional unit of social and administrative sciences of the IPN
Instituto Politécnico Nacional – UPIICSA, México
Av. Te 950, Col. Granjas México, Iztacalco, 08400

В області комп'ютерного зору і обробки зображень сегментація зображень залишається актуальною областю досліджень, в якій міститься багато частково вирішених питань. Одна з найбільш значущих областей в цифровій обробці зображень відноситься до сегментації, процесу, який розбиває зображення на різні складові компоненти. Метод, який широко використовується в літературі, називається нарощуванням областей. Однак він має високий рівень обчислювальної складності. Традиційні методи нарощування областей засновані на порівнянні рівнів сірого сусідніх пікселів і зазвичай непридатні, коли сегментована область містить інтенсивності, подібні до сусідніх областей. Однак, якщо в порогових значеннях вказано широкий допуск, то виявлені межі перевищуватимуть область ідентифікації; і навпаки, якщо граничний допуск занадто сильно зменшується, то ідентифікована область буде менше бажаної. При аналізі текстур кілька сцен можна розглядати як композицію різних текстур. Візуальна текстура відноситься до враження шорсткості або гладкості, яке створюють деякі поверхні шляхом зміни відтінків або повторення візуальних візерунків. Методи аналізу текстур засновані на призначенні одного або декількох параметрів, що вказують на характеристики текстури кожній області зображення. В даній статті показано, як був реалізований паралельний алгоритм для вирішення відкритих проблем в області дослідження сегментації зображень. Нарощування областей – це вдосконалений підхід до сегментації зображень, при якому сусідні пікселі досліджуються один за іншим і додаються до відповідного класу областей, якщо межа не виявлена. Цей процес є ітераційним для кожного пікселя в межах області. Якщо виявлені суміжні області, то використовується алгоритм злиття областей, в якому слабкі краї розчиняються, а тверді залишаються недоторканими, що вимагає багато часу обробки на комп'ютері, щоб уможливити паралельну реалізацію.

Ключові слова: комп'ютерний зір, обробка зображень, методи сегментації, нарощування областей, паралельна обробка, паралельні алгоритми, графічний процесор, ОКМД, аналіз текстур, цифрова обробка зображень

*Received date 08.01.202
Accepted date 17.02.2020
Published date 29.02.2020*

1. Introduction

Region Growing offers several advantages over classic segmentation techniques. Unlike the gradient and Laplacian methods, the edges of the regions that are found by the growing region are perfectly thin and connected. The algorithm is also very stable concerning noise [1–3].

The main objective of image segmentation is to obtain the independent division of the domain of an image and convert it into a set of disjoint regions that are visually different concerning some characteristics or properties calculated as the level of grey, texture or colour, with them, it is possible to perform simple image analysis [4].

Segmentation, in concept, is a straightforward idea [5]. Only by looking at an image, one can say which regions are contained in an image [6].

The relevance of our work here presented is to apply the classic parallel processing techniques used to the growing region techniques to process large and high-quality images with the biggest number of pixels.

2. Literature review and problem statement

Image segmentation based on growing regions begins with the selection of a pixel, often referred to as a seed, that is within the object of interest. Usually, the seed is chosen manually. Hsiao and Tynan present the results of research where the seed pixel (first point of the region) will begin to extend the region by processing its neighbours and adding them using a predefined criterion [7, 8].

Felzenszwalb showed that there were unresolved issues related to the insertion criterion uses common characteristics; these are the intensity, the colour, the texture of the seed and the points that belong to the region [9, 11].

Each time a new point is inserted into the region, the characteristic that is being used to perform the insertion will be recalculated, for example, if the grey level is used, the average value of the grey levels that will be recalculated there is in the region generated so far. In this way, the region will expand by adding neighbours until it finds one that does not meet the insertion condition imposed by the criteria, this approach was used in [10].

If a point has not been added to any region, it can be added to a region near if the difference between, for example, the grey level of this point and the average grey level of the region does not exceed a given threshold value.

This technique is useful when the intensity of the background and the object are very similar but are separated by an edge or another region.

The segmentation of any image would be straightforward if factors such as image noise and the number of pixels in the process were not involved. In the last ten years, the power of computing has increased with the use of GPU.

A way to overcome these difficulties was proposed by [10, 11]. Their research shows that the use of this type of cards has become accessible for research, and their uses are in different fields.

The GPU has doubled its performance every six months on average. At present, the computing power of high-performance GPUs can reach Teraflops, which is much higher than the central processing unit of a computer [12], all this suggests that it is advisable to conduct a study on the architectures and behaviour of the memory and improve all techniques of computer vision.

3. The aim and objectives of the study

The aim of the study is to implement a parallel algorithm to solve general problems in the area of image segmentation research. In our first approximation, the computing power is applied to Region growing technique, which is an advanced approach to image segmentation in which neighbouring pixels are examined one by one and added to an appropriate region class if no border is detected, this algorithm is viral and very used in the literature.

However, to achieve this aim, the following objectives are accomplished:

– understand and apply all the techniques of parallel computing to the computer vision field;

– establish a parallel strategy to implement the Region growing technique;

– validate the correct function of the algorithm;

– show the measures of performance and improvements.

4. GPU and their programmation

NVIDIA proposed a free CUDA (Compute Unified Device Architecture) GPU program development tool in 2007, which allowed programming any type of GPU and turned the technology into general-purpose [13, 14].

It has contributed to high-speed scientific operations, so the scientific fields benefited are: statistical engineering, Monte Carlo statistical simulation, financial engineering, global climate change simulation, 3D multimedia, biomedicine, national defence science, oil exploration, civil engineering, CAM, CAE, CAD [15, 16].

From an architectural point of view, the first generations of GPUs had a relatively small number of cores, but it quickly increased until today, when we talk about many-core devices with hundreds of cores on a single chip [17].

This increase in the number of cores caused that in 2003 there was a significant jump in the capacity of calculation in a floating-point of the GPUs concerning the CPUs, as shown in Fig. 1. Fig. 1 shows the evolution of the floating-point performance of Intel and NVIDIA based technology during the last decade.

It can be seen that GPUs are much more ahead than CPUs in terms of performance improvement, especially since 2009, when the ratio was approximately 10 to 1 [18, 19].

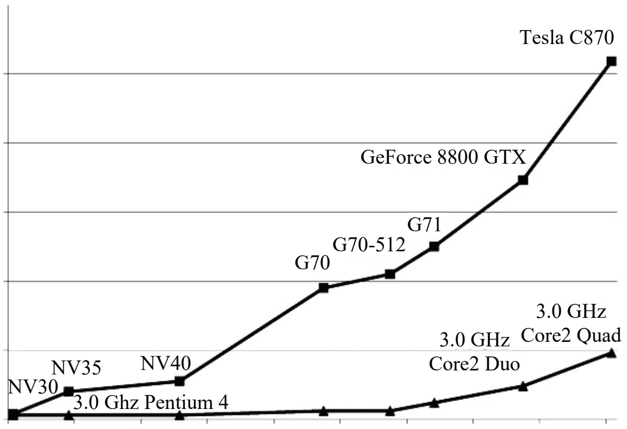

Fig. 1. Performance comparison (theoretical peak) between CPU and GPU technologies, Source: NVIDIA [13]

The significant differences between CPU and multi-core GPU performance are mainly due to a design philosophy issue. While GPUs are designed to exploit parallelism data level with mass parallelism and relatively simple logic, the design of a CPU is optimized for efficient sequential code execution [20].

The CPUs use sophisticated control logic that allows instructional and out-of-order parallelism and use quite large cache memories to reduce data access time in memory.

There are also other issues, such as power consumption or memory access bandwidth. Current GPUs have memory bandwidths around ten times higher than those of CPUs, among other things because CPUs must meet requirements inherited from operating systems, applications or output-input devices.

There has also been a very rapid evolution from GPU programming that has changed the purpose of these devices. GPUs in the early 2000s used programmable arithmetic units (shaders) to return the colour of each pixel on the screen.

Since the programmer could fully control the arithmetic operations that were applied to the input colours and textures, the researchers observed that the input colours could be any type of data.

Thus, if the input data were numerical data that had some meaning beyond a colour, the programmers could execute any of the calculations they needed on that data through the shaders [21].

Despite the limitations that programmers had to develop high-performance applications with arithmetic operations, many efforts were devoted to developing general-purpose application programming interfaces and environments for GPUs. Some of these programming interfaces have been widely accepted in various sectors, although their use still requires some specialization [22].

OpenMP has been successfully implemented in small to medium scale shared memory systems and large scale systems. OpenMP was developed by the Architecture Review Board (ARB) in November 2004. Its main evolution allowed the use of the C, C++ and FORTRAN languages. As computer components decrease in size, architects have begun to consider different strategies to exploit space on a chip. A recent trend is to implement Multi-Threading Chip (CMT) in hardware [23, 24].

This term refers to the simultaneous execution of two or more threads within a processor. It can be implemented through several cores of physical processors on a chip, a single-core processor with feature replication to maintain the status of multiple threads simultaneously or the combination of CMP and SMT. OpenMP support for these new microarchitectures needs to be evaluated and possibly improved [25].

5. Parallel implementation

Region-based segmentation consists of dividing an image into similar and equal areas delimited by connected pixels through standard criteria between sets of sample pixels. Each pixel of a region is similar in some features usually like colour, intensity and texture.

If these criteria are not adjusted, the results will be incorrect and undesirable. The following problems are:

1) the segmented region is smaller or larger than the current one;

2) pseudo objects arise;

3) fragmentation.

As the main objective of segmentation is to divide an image into regions, segmentation methods such as threshold achieve this objective partially by seeking boundaries between regions based on discontinuities. However, region-based segmentation is a technique that works well to determine the region directly [2].

The basic formulation is defined as follows:

Image segmentation partitions the set X into the subsets $R(i)$, where i=1, 2, 3, ..., N having the following properties:

$\bigcup i=1\ nR\ i=R.$ (1)

$R\ i$ is a connected region, i=1, 2, ..., n. (2)

$R\ i \cap R\ j=\emptyset,\ i\neq j.$ (3)

$P(Ri)$ is a logical predicate defined over the points in set $R\ i$:

1) segmentation must be completed; for this to happen, each pixel must be in a region;

2) the points in a region must always be connected;

3) regions should be disjoint;

4) the properties that pixels in a segmented region must meet must be clearly defined;

5) indicates that regions $R\ i$ are different in the sense of predicate P.

With this information, the algorithm was developed shown in Listing 1.

As shown in Listing 1, all the instructions within the while statement correspond to the structure that can be parallelized, because this block of sentence can be sent to the different GPUs.

With the previous information, it is possible to parallelize the algorithm using OpenMP. Fig. 2 shows how the parallelization strategy is applied. Parallelization.

Listing 1. Region Growing Algorithm

```
WHILE (NUM) {
IF (MASK_VALUE (X, Y, Z)==255) {
IF (THRESHOLD_MASK (X, Y, Z)==255) {
MASK_VALUE (X–1, Y, Z)
THRESHOLD_MASK (X–1, Y, Z);
MASK_VALUE (X+1, Y, Z)
THRESHOLD_MASK (X+1, Y, Z);
MASK_VALUE (X, Y–1, Z)
THRESHOLD_MASK_ (X, Y–1, Z);
MASK_VALUE (X, Y+1, Z)
THRESHOLD_MASK (X, Y+1, Z);
MASK_VALUE (X, Y, Z–1)
THRESHOLD_MASK (X, Y, Z–1);
MASK_VALUE (X, Y, Z+1)
THRESHOLD_MASK (X, Y, Z+1);
THRESHOLD_MASK (X, Y, Z)=0;
NUM=NUM+1;
}}}
```

Fig. 2 shows the implementation strategy, using OpenMP, the central part of the code is sent in different threads independently.

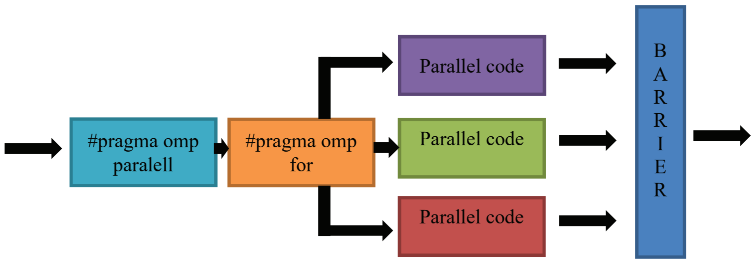

Fig. 2. The strategy of the algorithm shown in Listing 1

In the end, there is a synchronization called the barrier, which expects the tasks to be completed and then accumulate the sub-results obtained.

6. Discussion of experimental results

The results presented in this work cover three critical stages:

1) verification of algorithm operation;

2) execution and parallel performance and parallel speedup.

6. 1. Checking the operation of the algorithm

The algorithm performed was tested by different professional tomography and x-ray images. In the image analyzed by tomography of a slice of the brain, Fig. 3 shows the original image, Fig. 4 and the segmented image and finally, in Fig. 5 both overlapping images. Fig. 6 shows the resulting histogram.

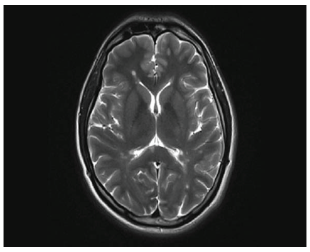
Fig. 3. Brain tomography

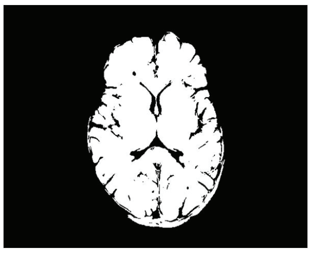
Fig. 4. Segmentation performed of the image, x=198; y=359, max*dist*=0.2

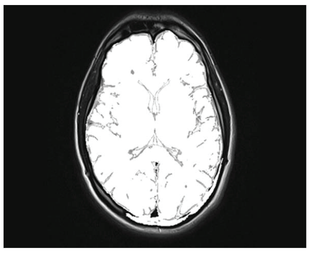
Fig. 5. Complete result of the algorithm

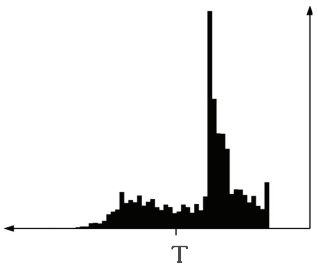

Fig. 6. Resulting histogram

Another test was performed directly on an X-ray plate on the side of a human head. Fig. 7 shows the original image; Fig. 8 shows the original and segmented image, both overlapping images.

Two fundamental problems that were observed are the selection of the appropriate seeds to define the regions of interest; and the choice of properties that allow adding pixels.

The selection of the starting points depends on the image to be segmented.

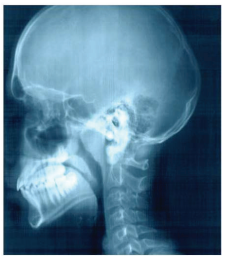
Fig. 7. X-ray plate analysed

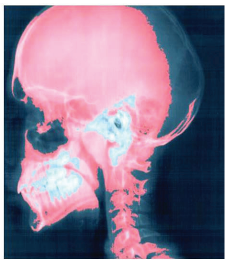
Fig. 8. Complete result of the algorithm application. Segmentation performed of the image, x=198; y=359, max*dist*=0.2

The selection of similarity criteria depends on the problem and the type of image available.

Another test was performed directly on a stream of 4k video to know the real-time parameters of the parallelized processing. Fig. 9 shows the stream images and the processed images.

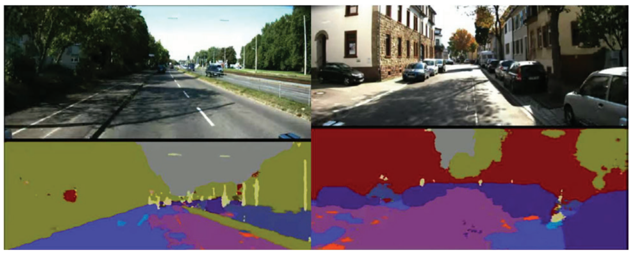
Fig. 9. Complete result of the algorithm application on video stream. Segmentation performed of the image, x=4096; y=2160, max*dist*=0.2

6. 2. Execution and parallel performance

The CT images from cancer imaging archive with contrast and patient age dataset obtained from kaggle.com is used to perform the parallel performance tests.

The sequential program developed in C language and its equivalent developed in OpenMP with GPU interface NVIDIA Tegra was executed. See Fig. 9. Table 1 shows the execution time of GPU+OpenMP time vs. Sequential time.

The speedup is defined as the linear relationship between the execution time of serial processing over the best parallel algorithm used to solve a problem. It is necessary to emphasize that there will always be a serial time that cannot be parallelized.

The equation of the speedup is defined in (4) where S is the speedup, Ts the processed sequential time and Tp is the processed parallel time. Using this equation is shown in Table 2.

$$S = \frac{Ts}{Tp}. \tag{4}$$

Table 1

Executing time

Image	GPU+openmp time	Secuential time
ID_0000_AGE_0060	0.513	16.023
ID_0001_AGE_0069	1.245	12.034
ID_0002_AGE_0074	0.876	13.234
ID_0003_AGE_0075	0.923	9.234
ID_0004_AGE_0056	1.022	10.345
ID_0005_AGE_0048	0.945	14.750
ID_0006_AGE_0075	1.023	12.345
ID_0007_AGE_0061	0.456	15.879
ID_0008_AGE_0051	0.678	13.056
ID_0009_AGE_0048	1.230	10.456

Table 2

Speedup

Image	Parallel time	Sequential time	Speedup
ID_0000_AGE_0060	0.513	16.023	31.2339181
ID_0001_AGE_0069	1.245	12.034	9.66586345
ID_0002_AGE_0074	0.876	13.234	15.1073059
ID_0003_AGE_0075	0.923	9.234	10.0043337
ID_0004_AGE_0056	1.022	10.345	10.1223092
ID_0005_AGE_0048	0.945	14.750	15.6084656
ID_0006_AGE_0075	1.023	12.345	12.0674487
ID_0007_AGE_0061	0.456	15.879	12.0674487
ID_0008_AGE_0051	0.678	13.056	19.2566372
ID_0009_AGE_0048	1.230	10.456	8.50081301

As can be seen, in Table 1, 2, the calculation of the performance limits of the parallelization is critical since it represents how the software will manifest when the hardware is scaled. The previous performance considerations indicate that it has many chances of success in the application with multi-core hardware.

Our studio is limited to 4K images with maximum modifications of 4,096×2,160. However, it is possible to use pictures and video with more resolution by scaling the proposed hardware architecture. Our method is necessary to remember that it is local and does not take into account the global vision of the problem. It is necessary to emphasize that the algorithm is sensitive to noise, for this it is necessary to apply a threshold function to the image, there can also be a continuous route of points related to colour, which connects any of the two points of the image which allow affecting the performance.

7. Conclusions

1. Understand and apply all the techniques of parallel computing to the computer vision field: In past times, the supercomputers were the most significant system that used precious resources and the parallel computing only could be processed in a computer cluster. Now, using a GPU system to do computing parallel the image analysis to the most significant image can be processed using a sophisticated algorithm. In our experience, 256-core Maxwell in 4 CPU core arm Cortex A57 was used and each process was processed in 20ns.

2. Establish a parallel strategy to implement the Region growing technique: The proposed study showed that the improvement of technologies and mechanisms for obtaining data from images has led to the obtaining of more detailed raster information and with a higher level of resolution and spatial precision, which has generated a gradual growth in the volume of the models. However, each model in their analysis has the most significant computational complexity, and with this, the processing time could be computable or hard computable. Our strategy works fine, and it is possible to reach scalability for the application of the algorithm in clusters.

3. Validate the correct function of the algorithm: The Complete result of the algorithm applied in the segmentation applied was performed of the image, x=198; y=359, max-*dist*=0.2 in an image kind DICOM in the first algorithm and excellent result was obtained in grey images and colour image, but the algorithm is robust to be implemented in satellite images and ecological images of most significant dimensions.

4. Show the measures of performance and improvements: Table 2 shows the excellent results of the functionality test of the proposed algorithm. The number of cores 256 was sufficient for executing the images of any size, and it is possible to use the hardware architecture proposed for another algorithm more complexes in all the fields of computer vision.

Acknowledgements

We appreciate the facilities granted to carry out this work to the INSTITUTO POLITECNICO NACIONAL through the Secretary of Research and Postgraduate with the SIP project 20180023. To the Interdisciplinary Unit of Engineering and Social and Administrative Sciences, Center for Research and Development of Digital Technology. Likewise, the Program for Stimulating the Performance of Researchers (EDI) and the Program for Stimulating Teaching Performance.

(EDD) and COFAA.

References

1. Keely, C. C., Hale, J. M., Heard, G. W., Parris, K. M., Sumner, J., Hamer, A. J., Melville, J. (2015). Genetic structure and diversity of the endangered growling grass frog in a rapidly urbanizing region. Royal Society Open Science, 2 (8), 140255. doi: https://doi.org/10.1098/rsos.140255
2. Ashburner, J., Friston, K. J. (2005). Unified segmentation. NeuroImage, 26 (3), 839–851. doi: https://doi.org/10.1016/j.neuroimage.2005.02.018
3. Bandler, R., Tork, I. (1987). Midbrain periaqueductal grey region in the cat has afferent and efferent connections with solitary tract nuclei. Neuroscience Letters, 74 (1), 1–6. doi: https://doi.org/10.1016/0304-3940(87)90041-3
4. Patel, N. H., Liu, P. Z. (2009). Segmentation. Encyclopedia of Insects, 909–912. doi: https://doi.org/10.1016/b978-0-12-374144-8.00240-x
5. Taylor, J. R. A., deVries, M. S., Elias, D. O. (2019). Growling from the gut: co-option of the gastric mill for acoustic communication in ghost crabs. Proceedings of the Royal Society B: Biological Sciences, 286 (1910), 20191161. doi: https://doi.org/10.1098/rspb.2019.1161

6. Chen, D. (2008). Image Segmentation. User Centered Design for Medical Visualization, 258–279. doi: https://doi.org/10.4018/978-1-59904-777-5.ch013
7. Hsiao, Y.-T., Chuang, C.-L., Jiang, J.-A., Chien, C.-C. (2005). Robust Multiple Targets Tracking Using Object Segmentation and Trajectory Estimation in Video. 2005 IEEE International Conference on Systems, Man and Cybernetics. doi: https://doi.org/10.1109/icsmc.2005.1571289
8. Tynan, A. C., Drayton, J. (1987). Market segmentation. Journal of Marketing Management, 2 (3), 301–335. doi: https://doi.org/10.1080/0267257x.1987.9964020
9. Felzenszwalb, P. F., Huttenlocher, D. P. (2004). Efficient Graph-Based Image Segmentation. International Journal of Computer Vision, 59 (2), 167–181. doi: https://doi.org/10.1023/b:visi.0000022288.19776.77
10. Hu, R., Dollar, P., He, K., Darrell, T., Girshick, R. (2018). Learning to Segment Every Thing. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. doi: https://doi.org/10.1109/cvpr.2018.00445
11. Owens, J. D., Houston, M., Luebke, D., Green, S., Stone, J. E., Phillips, J. C. (2008). GPU Computing. Proceedings of the IEEE, 96 (5), 879–899. doi: https://doi.org/10.1109/jproc.2008.917757
12. Stuart, J. A., Owens, J. D. (2011). Multi-GPU MapReduce on GPU Clusters. 2011 IEEE International Parallel & Distributed Processing Symposium. doi: https://doi.org/10.1109/ipdps.2011.102
13. Nickolls, J., Dally, W. J. (2010). The GPU Computing Era. IEEE Micro, 30 (2), 56–69. doi: https://doi.org/10.1109/mm.2010.41
14. Bergstra, J., Breuleux, O., Bastien, F. F., Lamblin, P., Pascanu, R., Desjardins, G. et. al. (2010). Theano: a CPU and GPU math compiler in Python. Proceedings of the Python for Scientific Computing Conference (SciPy).
15. Sanders, J., Kandrot, E. (2010). CUDA by Example: An Introduction to General-Purpose GPU Programming. NVIDIA Corporation, 311.
16. Pratx, G., Xing, L. (2011). GPU computing in medical physics: A review. Medical Physics, 38 (5), 2685–2697. doi: https://doi.org/10.1118/1.3578605
17. Sengupta, S., Harris, M., Zhang, Y., Owens, J. D. (2007). Scan primitives for GPU computing. Proceedings of the SIGGRAPH/Eurographics Workshop on Graphics Hardware, 97–106.
18. Che, S., Boyer, M., Meng, J., Tarjan, D., Sheaffer, J. W., Skadron, K. (2008). A performance study of general-purpose applications on graphics processors using CUDA. Journal of Parallel and Distributed Computing, 68 (10), 1370–1380. doi: https://doi.org/10.1016/j.jpdc.2008.05.014
19. Power, J., Hestness, J., Orr, M. S., Hill, M. D., Wood, D. A. (2015). gem5-gpu: A Heterogeneous CPU-GPU Simulator. IEEE Computer Architecture Letters, 14 (1), 34–36. doi: https://doi.org/10.1109/lca.2014.2299539
20. Feng, W., Xiao, S. (2010). To GPU synchronize or not GPU synchronize? Proceedings of 2010 IEEE International Symposium on Circuits and Systems. doi: https://doi.org/10.1109/iscas.2010.5537722
21. Lee, V. W., Hammarlund, P., Singhal, R., Dubey, P., Kim, C., Chhugani, J. et. al. (2010). Debunking the 100X GPU vs. CPU myth. Proceedings of the 37th Annual International Symposium on Computer Architecture - ISCA ’10. doi: https://doi.org/10.1145/1815961.1816021
22. Zhou, Y., Tan, Y. (2009). GPU-based parallel particle swarm optimization. 2009 IEEE Congress on Evolutionary Computation. doi: https://doi.org/10.1109/cec.2009.4983119
23. Shah, S., Bull, M. (2006). OpenMP---OpenMP. Proceedings of the 2006 ACM/IEEE Conference on Supercomputing - SC ’06. doi: https://doi.org/10.1145/1188455.1188469
24. Hermanns, M. (2002). Parallel programming in Fortran 95 using OpenMP. School of Aeronautical Engineering.
25. Chapman, B., Jost, G., Van Der Pas, R. (2008). Using OpenMP. Cluster Computing.