International Journal of Trend in Scientific Research and Development (IJTSRD)
International Open Access Journal | www.ijtsrd.com

ISSN No: 2456 - 6470 | Volume - 3 | Issue – 1 | Nov – Dec 2018

Compressing of Magnetic Resonance Images with Cuda

Mahmut Ünver[1], Atilla Ergüzen[2]
[1]Lecturer, [2]Assistant Professor
Department of Computer Programming, Kırıkkale University, Kırıkkale, Turkey

ABSTRACT
One of the most important areas that use image processing is the health sector. In order to detect some diseases, the need to visualize a certain part of the patient's body using medical imaging devices has emerged. This field in the health sector is the Radiology department. MR, Tomography, Ultrasound, X-ray, Echocardiography. Because of the importance of time in the health sector, GPU technologies are a technology that should be used in hospitals. Medical MRI images showed that the unused areas (NON-ROI) occupy a large area and this unnecessary area in the image could reduce the image size significantly. In this method developed with CUDA, the ROI (Region of Interest) region within the Medical MR images is determined by sending a 3X3 Kirsch filter matrix to the CUDA cores, and the NON-ROI region is extracted with CUDA from the image. It is then compressed with a new compression method developed. As a result of this method; The parallel application with CUDA solves the problem 34 times faster than the sequential application for each image, while the compressed image takes up 90% less space than the original image size; it takes 40% less space than the compressed size of the original image.

Key Words: *CUDA, Medical Image Processing, Image Compression, Parallel Programming*

1. INTRODUCTION

Image processing is used in many areas of daily life and facilitates human life. With the image processing, we are able to choose the smallest detail on the image, match the objects with the image, count the objects, detect the faulty productions and do things very quickly.

The CPU is generally insufficient to be able to work on the image and make quick calculations. At this point, GPU technology has been developed which has carried out its structure on image processing.

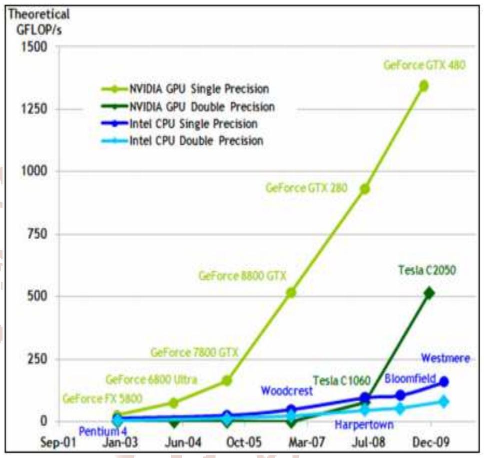

Figure 1 CPU-GPU GFLOP / s compared to years

In 2006, NVIDIA introduced the CUDA GPU programming platform. In 2008, Apple introduced the OpenCL GPU programming platform. With CUDA, only NVIDIA graphics cards can be developed, and with OpenCL, all GPU systems can be developed. With these developments, after the 2000s, the GPU has improved its computational power by increasing it every year [1]. Figure 1 shows the CPU-GPU comparison based on memory bandwidth over the years in Figure 2 on the basis of GFLOPS (GigaFloating Point Operations per Second) [2,3].

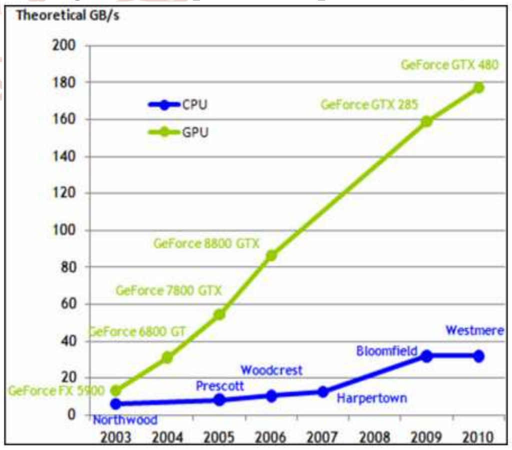

Figure2. Comparison of CPU-GPU memory bandwidth by years

2. MATERIALS AND METHODS

GPU command sets are required to perform GPGPU (General Purpose Computing) operations on the GPU. This programming is called heterogeneous programming. CUDA and OpenCL technologies are used for GPU programming.

2.1 CUDA

CUDA (Compute Unified Device Architecture) is a parallel computational architecture that enables significant increases in computational performance by using NVIDIA's GPU (Graphics Processing Unit) capability in 2006. Image and video processing, computational biology and chemistry, fluid dynamics, computed tomography, seismic analysis, beam monitoring can be performed with CUDA enabled GPU [2, 3].

CUDA can only operate on the GPU. It cannot directly access CPU and system resources. It simply sends the CPU and CPU back to the main memory and processing the data from the main memory [2,3].

CUDA; It can run on Linux, Windows and MAC OSX operating systems and support development in C, C ++, Python, Fortran and C # languages.

2.2 OpenCL

OpenCL (Open Computing Language) is a platform-independent, open platform that was developed by Apple in 2008, enabling both CPU and GPUs of graphics card manufacturers such as NVIDIA and ATI. OpenCL standards are developed to enable portable, manufacturer and device independent development across devices. OpenCL standards were introduced by a consortium called Khronos. Apple, Qualcomm, Intel, NVIDIA, AMD, Samsung are the member companies of this consortium.

The main purpose of OpenCL is to use resources in multi-core systems quickly and effectively. Since the number of parallel operations will increase as the number of cores increases, it will be necessary to manage these processes dynamically. OpenCL has been able to overcome this problem by constantly improving itself.

OpenCL allows developers to develop with C language (OpenCL C). There are differences according to classical C language. Function markers, recursion, variable-length arrays are removed, global, local, constant and private classes are added for memory management.

2.3 Using OpenCL with CUDA and Comparison

Advanced software that uses image processing has increased the performance of their applications using CUDA and OpenCL. As an example [4].

- ➢ Adobe Photoshop CC
- CUDA: 3D and Multi-GPU support
- OpenCL: No specific specifications
- ➢ Adobe After Effects CC
- CUDA: Mercury Graphics Engine in 30 effects
- OpenCL: No specific specifications
- ➢ Autodesk Maya
- CUDA: Enhanced model complexity and Wide scene
- OpenCL: Physics simulation
- ➢ Avid Media Composer
- CUDA: Faster video effects
- OpenCL: Unused

When CUDA and OpenCL developed the above programs, it was observed that the features developed with CUDA were more effective. These features are directly proportional to the platform being close to the hardware. To explain, the; CUDA and OpenCL utilizing Adobe CC and NVIDIA graphics cards in rendering processes (forums.adobe.com), processing times for CUDA in the order of 0:56, 0:36, 1:02 for OpenCL 3: 42,0.38, 7:23. This is an indication of how close the platform is to the equipment.

CUDA's advantages over OpenCL have the following advantages: The company's self-developed graphics card and GPU provides a platform for program developers to provide a better understanding and management of the GPU and graphics card structure. It makes it easy to use these additional features when the graphics card and GPU are separated from other graphics cards. It is also easier to manage any errors that may arise as the feature of each GPU is fully known when developing. The advantage of CUDA is that it directly receives information from a single partner about problems that may arise from a graphics card, GPU, or platform.

The advantages of OpenCL compared to CUDA are as follows: CUDA is not an open platform developed by a single company and can only be used on NVIDIA graphics cards. OpenCL is an open platform and can be used on all CPU and GPU systems. The developer is more flexible than a consortium. OpenCL is therefore portable and dynamic.

Considering the advantages and disadvantages of both platforms, you can conclude that if you have an NVIDIA graphics card, develop with CUDA or choose applications developed with CUDA; if it doesn't, you can do it with OpenCL or choose applications developed with OpenCL [2,3].

2.4 CPU Comparison with GPU

A CPU core is designed to execute a single stream of instructions at a time. GPUs are designed to quickly execute many parallel instruction streams [5].

The CPU uses the buffer to improve performance by reducing the overall memory access latency. GPU uses buffer (or shared memory) to increase bandwidth [5].

The CPU is managed with memory latency, wide buffers and prediction mechanisms. They take up a lot of space on design and consume a lot of power. The GPU manages the delay by supporting thousands of threads at a time. If any threads are waiting to load from memory, the GPU switches to another job without delay [5].

CPUs support one or two threads per core execution. CUDA-supported GPUs can support up to 1024 threads per stream processor (SM). The CPU's switching between applications requires hundreds of cycles, while the GPU does not have any loss when switching [5].

CPUs use SIMD (Single Instruction Multiple Data) units for vector operations. GPUs use SIMT (Single Thread Multiple Thread) for scale execution. SIMT doesn't require the programmer to edit data in vectors, it allows random distribution for threads [5].

2.5 CUDA Programming Structure

2.5.1 Kernel

A code in CUDA that will work on the GPU is called “Kernel code. A GPU is created as a Kernel copy for each element of the data set. These Kernel copies are called as Thread [6].

2.5.2 Grid

Grids are the structures that the Blocks come together. Each Kernel call creates a Grid. Grids can be 1B, 2B or 3B. These dimensions are expressed as ifade gridDim.x D, “gridDim.y D and“ gridDim.z Bu. The term io gridDim ıs refers to the number of blocks in the Grid [6].

2.5.3 Block

The block structure consists of threads that are capable of working in parallel.1B can be 2D or 3D. They are grouped together in the grids. Each Block has its own unique index in these 3 dimensions in the Grid. These indices are ”blockIdx.x“, ”blockIdx.y“ and ”blockIdx.z“. Thread meks are sized according to the number of rows and columns and these dimensions are called “blockDim.x.,“ BlockDim.y lar, ına blockDim.z İçer. The term io blockDim ith refers to the Thread in Block [6].

2.5.4 Thread

Thread is the smallest unit in CUDA architecture. They can be 1B, 2B or 3B in blocks. They run the same piece of code in parallel between them. Threads are grouped in blocks. The threads in different blocks cannot work together [6].

Each Thread has a Program Counter (PC), register and status register (StateRegister). Each Thread has its own unique index in the Block. These indices; Io threadIdx.x oz, (threadIdx.y ”and d threadIdx.z” [6].

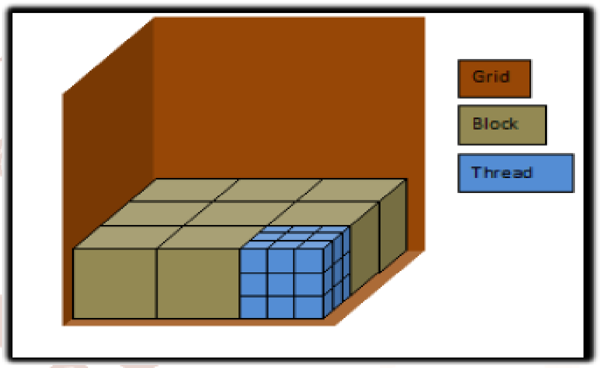

Figure3. Grid, Block and Thread structure [6].

In summary, as shown in Figure 3 Grid 3-dimensional blocks, each block consists of 3D threads.

The number of threads in a block is limited and varies according to the calculation capability. For example; For a GPU with a computational capability of 2.0, a Block can be up to 3B 1024 * 1024 * 6 4Thread.

2.6 CUDA Programming API

CUDA programming includes two APIs (Application Programming Interface). The first API is the high-level CUDA Runtime API, and the other is the low-level CUDA Driver API.

With the CUDA Driver API, module startup and memory management is handled in more detail. Detail and code is more. Particularly difficult to configure the kernels. Because the execution of the kernel parameters must be specified with explicit function calls instead of the execution configuration syntax. However, the driver for the kernel code offers better control and management than JIT (Just in Time) optimizations.

The CUDA Runtime API simplifies device code management and usage by providing implicit start, content management and module management. Runtime API has GPU memory allocation, memory evacuation, data copying and system management functions. For example; kernel <<<, >>> () "syntax is sufficient when calling a Kernel.

These two APIs are mutually exclusive, so one of the APIs should be preferred. In the next generation GPUs, these two APIs are said to be in peace, but one should be preferred. In terms of performance, there is no significant difference between them.

2.7 MR Image

Magnetic Resonance Imaging (MRI) is a diagnostic technique that does not use drugs that are painless and do not require any drugs to cause allergies and do not use harmful materials such as x-rays [7].

Mar; In the examination of brain lesions, lung, bronchial and trachea, detailed examination, kidney, urinary tract and bladder examination, joints and rheumatic findings, intestinal examination, soft tissue imaging and is frequently used in the examination [8].

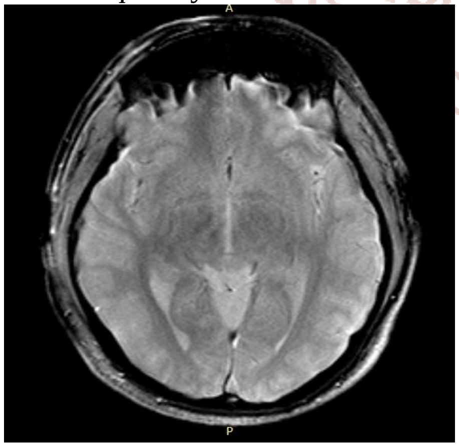

Figure4. Brain MR Image

2.8 Archiving of Medical Images

Archiving of medical images has an important place in medical documentation. In the information age, especially digital archiving is a topic that is more on the agenda. Because the cost of the film for the printing of the medical images on the films and the time of the archive scanning that will occur when the film is re-used for archiving is one of the disadvantages of physical archiving.

Today's medical imaging devices are produced in the IoT (Internet of Things) standard, and the medical image generated can be transferred to the digital environment over the network. Advantages of digital archiving; medical image on the instant image by connecting with the patient, image processing software and medical analysis on the medical image and to comment and to warn the user, the medical image with numerical data to dynamically provide statistics as a possibility, outside the server and storage space, you do not need a physical space ... disadvantages of archiving; The necessity of finding IoT-enabled devices is the establishment of network infrastructure, installation of server and storage systems, and image processing / imaging software needs.

As the image processing systems in the field of health develop, how to store the medical images together with the patient information and how to access this image together with the data are the subjects that the numerical system emphasizes. DICOM data set standard has emerged as a result of certain standard studies [9].

2.9 DICOM Standard

It has been developed in order to find solutions to the questions about how to use the archived medical image. DICOM (Digital Imaging and Communications) is a database of file structure. As in databases, both text data can be written in the file and raw image data can be added. There are labels to prevent confusion in retrieving the data in this standard, which stores all the data in a single file [9].

Having a standard in medical image archiving offers hospitals and doctors in different regions a common sharing on medical images. Since there are different devices and different companies in medical imaging, the dependence of the firm that produces the device will be required.

The DICOM file format is a format that can store more details on medical images, patient information, and hospital information, unlike all known formats. In the DICOM file, moving images and sound recordings can be stored with normal medical images. In addition, all tests and results of a patient and the fact that doctor diagnostics can be recorded on a common standard format will help to significantly reduce the diagnosis and treatment time of all patients in all hospitals.

Large storage areas are needed when medical images are archived. In addition, as the image size increases, it will use the bandwidth on the network more when the image is desired to be accessed on the medical system and this will lead to blockages.

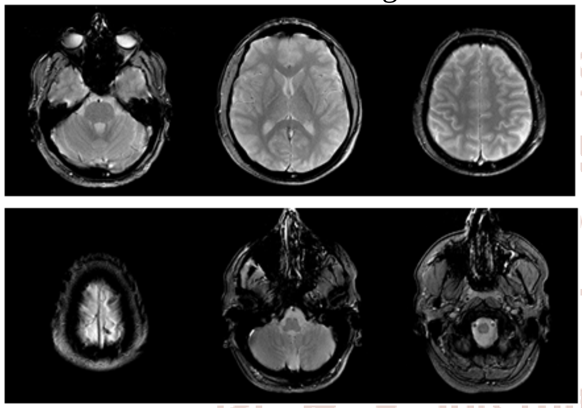
Figure5. Sample Brain MR Images

When the Figure 5 is examined, the part of the MRI that is to be considered on the MRI image, ie the black area outside the region of the region of interest (ROI), covers a large area on the image. In this thesis, the ROI region used on the image will be detected with CUDA and removed from the image.

3. IMPLEMENTATIONS AND ACHIEVEMENTS

3.1 Identification of the ROI Region

For each pixel when determining the ROI region on the MR image; The gradient based on the derivative is calculated. For best results, the most suitable gradient matrix should be chosen. The calculated Gradient value is then compared to the threshold value. If it is greater than the specified threshold value, the pixel is marked as the edge point. If not, move to the next pixel.

3.2 Removing the ROI Region

After the ROI zone is detected, the ROI region's coordinates are detected from top to bottom, from right to left, from top to bottom and from left to right, and the image frame is collapsed according to these coordinates.

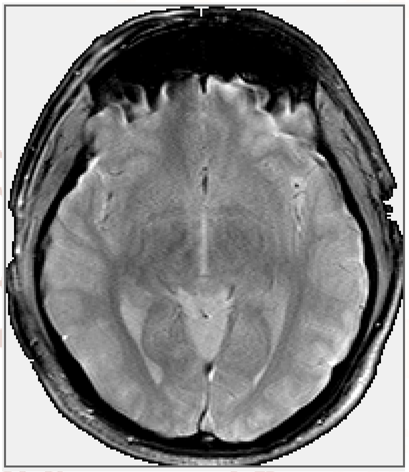
Figure 6 ROI area detected, and frame collapsed image

In Table 1, the 6 MR images shown in FIG. The data sizes of the ROI image obtained by the application are shown. The ROI image size is 40% less than the original MR image size. The compressed ROI image size is 15% less than the original size of the original image while it is 89% less than the original image size.

Table 2 shows the working times of the MR images on the CPU and GPU. Accordingly, the application running on the GPU with CUDA produced 34 times faster than the serial application running on the CPU.

Table1. Data dimensions before and after application

MR Image	Original MR Size	ROI Size	Compressed Original Image	Compressed ROI Size
1	256 KB	173 KB	41 KB	36 KB
2	256 KB	178 KB	36 KB	30 KB
3	256 KB	174 KB	36 KB	31 KB
4	256 KB	174 KB	43 KB	38 KB
5	256 KB	138 KB	35 KB	29 KB
6	256 KB	79 KB	21 KB	17 KB
Average	256 KB	153 KB	35 KB	30 KB

Table2. Working times obtained by running the application on the CPU and GPU

MR Image	Serial CPU Runtime	CUDA GPU Runtime
1	325 ms	9 ms
2	315 ms	10 ms
3	314 ms	12 ms
4	318 ms	9 ms
5	354 ms	13 ms
6	436 ms	8 ms
Average	343 ms	10 ms

4. RESULTS AND DISCUSSION

In this study, methods are used to store MR images with less memory space in the digital system and to achieve these images more quickly. Because the area of the area (NON-ROI) outside the area of interest (NI-ROI) of the MR image is significantly greater, it is thought that it would make more sense to store only the ROI area. The data size generated with this application is approximately 11% of the original image. An area saving of 89% for each MR image is a serious data space savings considering the number of MRIs recorded in hospitals. The storage of data in the archive system is a second possible situation. When the data generated by the developed application is compressed, the size of the data generated when the original image is compressed is about 85% of the data. In this case, an average space saving of 15% is achieved for each MR image. The application was run on both the CPU and CUDA on the GPU, and the run times were examined. According to this; It is seen that the parallel application written in CUDA runs faster than the application written on the CPU. In this study and literature studies, it is seen that GPU systems are quite fast in the image processing area compared to CPU performance. The CUDA-coded parallel GPU application used in this study yielded a result over the MRU GPU performance averages in the literature.

REFERENCES

1. Yıldız, E., (2011). NVIDIA CUDA ile yüksek performanslı görüntü işleme, Yüksek Lisans Tezi, İstanbul Üniversitesi Fen Bilimleri Enstitüsü, İstanbul.
2. http://developer.download.nvidia.com/compute/cuda/3_1/toolkit/docs/NVIDIA_CUDA_C_ProgrammingGuide_3.1.pdf, [Access Date: 25.11.2017].
3. http://www.nvidia.com.tr/object/cuda-parallel-computing-tr.html, [Access Date: 09.03.2018].
4. https://create.pro/blog/opencl-vs-cuda/, [Access Date: 10.03.2018)].
5. http://ugurkontel.net/cuda-cekirdegi-nedir/, [Access Date: 14.03. 2018].
6. http://nezihesozen.github.io/mydoc/cuda2.html [Access Date: 14.03.2018]
7. http://www.akcaabatdh.gov.tr/detay.php?id=598&cid=156, [Access Date: 25.03.2018].
8. http://www.medikalteknik.com.tr/saglik-hizmetlerinde-medikal-goruntuleme/, [Access Date: 25.03.2018].
9. Ulaş, M. ve Boyacı, A., (2007). PACS ve medikal görüntülerin sayısal olarak arşivlenmesi, Akademik Bilişim’07 - IX. Akademik Bilişim Konferansı Bildirileri, Dumlupınar Üniversitesi, Kütahya, s69-74.