**Technical Report – Project 9**

**GPU-Based Parallel Computing**

Yakshita B Rakholiya

CS\_610 Introduction to Parallel Computing

Pace University – NYC

Instructor: Dr. Lixin Tao

Spring 2023

Contents

Introduction ……………………………………………………………. 3

GPU …………………………………………………………… 3

System Architecture …………………………………………………….5

Definition of GPU-based parallel computing ………………….5

Types of GPU architecture …………………………………….5

Integrated GPU ………………………………………..5

Dedicated GPU ………………………………………..5

Hardware Architecture ……………………………………….. 6

Memory hierarchy and data management …………………… 8

Comparison with CPU architecture ………………………….. 8

Programming Environment …………………………………………… 9

CUDA model ………………..……………………………….. 9

OpenCL model ……………………….…………………….... 10

OpenACC model ………………………….…………………. 11

Tools and libraries for GPU programming ……………………11

Applications ……………………………………………………………12

High-performance computing (HPC) ………………………... 12

Cloud computing …………………………………………….. 12

Machine learning and deep learning …………………………. 12

Cryptography and security……………………………………. 13

GPU Uses ………………………………………………………………14

What I learned in this course…………………………………………....15

Conclusion …………………………………………………………......16

References ……………………………………………………………...17

**Introduction**

Graphics processors began as fixed-function graphics pipelines. These graphics circuits got increasingly programmable over time, prompting NVIDIA to produce the first GPU. Computer scientists, along with academics in disciplines such as medical imaging and electromagnetics, began employing GPUs to speed a many of scientific application in the 1999-2000 timeframe. This marked the beginning of the General-Purpose GPU Computing, movement.

NVIDIA saw the opportunity to provide this performance to the broader scientific community and invested in improving the GPU to make it completely programmable for scientific purposes. It also had support for high-level languages such as C, C++, and Fortran. This gave rise to the CUDA parallel computing platform for GPUs.

The GPU speeds CPU-based applications by offloading part of the compute-intensive and time-consuming code. The remainder of application continues to operate on the CPU. From the user, the program runs quicker because it makes use of the GPU's massively parallel processing capacity to increase speed. This is referred to as "hybrid" or "heterogeneous" computing.

A CPU has four to eight CPU cores, but a GPU has hundreds of smaller cores. They work together to process the data in the application. The GPU's great computation capability is due to its massively parallel design. A variety of GPU-accelerated apps make it simple to access high-performance computing (HPC).

**GPU**

A graphics processing unit GPU is a type of computer processor that creates images and graphics by performing a sequence of extremely rapid mathematical calculations. Graphics processing units are utilized in both commercial and household computers. Graphics processing units GPUs have historically been responsible for the rendering of 2D and 3D images, animations, and films. But overtime, GPUs application have been increasing in different areas as well like crypto mining.

These computations were carried out by the central processing unit in the early stages of computer development. However, as more graphics-intensive programs were designed and developed, the requirements of these apps put additional burden on the central processing unit, resulting in a reduced performance. GPUs were developed to relieve CPUs of the stress of accomplishing these tasks and to improve 3D graphics rendering. GPUs work by employing a technique known as parallel processing, in which many processors combined do a small fraction of task parallelly for whole operation.

A GPU is a specialized co-processor that, like many other things, has advantages and disadvantages, excelling at some tasks while failing badly at others. It collaborates with a CPU to boost data operations and the number of concurrent calculations inside an application.

GPUs, or graphics processing units, are commonly utilized in PC gaming because they allow for smooth, high-quality graphics rendering. Furthermore, programmers began to use GPUs to accelerate workloads in fields such as artificial intelligence (Al) and Crypto mining.

**System Architecture**

GPU-based parallel computing is a type of parallel computing that takes advantage of the processing capability of a GPU to do activities that need a large amount of data and calculation. Originally, GPUs were designed to produce graphics and pictures for video games and computer simulations. As the development of General-Purpose Computing on Graphics Processing Units got better, a wide range of parallel computing applications can now be run on GPUs.

**Types of GPU architecture:**

**Integrated GPU:**

A GPU that is incorporated into the same packaging as the CPU is referred to as integrated graphics. Although integrated graphics is often derided in enthusiast computer circles, this approach to GPU design has numerous significant advantages.

Except for high-end CPUs, almost all computer CPUs nowadays include an integrated GPU. It's safe to assume that the integrated GPU model is the most frequent GPU type in use. There are a few decent reasons why it's so popular, but as always, the list is quite short.

The first is price. Etching a GPU into your silicon real estate does not add much cost to a CPU. Incorporating a GPU into each GPU reduces costs in other parts of the system far mostly it raises the cost of the GPU itself. Base on result, systems with integrated GPUs are much less expensive than those with a specialized solution.

The intricacy is the second key factor. This is especially true with laptop computers, where every cubic millimeter counts. Laptops may be significantly smaller by incorporating the GPU into the CPU package since you don't need all the extra supporting gear to cool, power, and connect an altogether different chip package.

**Dedicated GPU:**

Discrete graphics refers to a GPU that is independent of the CPU. Because discrete graphics is separate from the main CPU processor chip, it consumes more power and generates huge amount of heat. Still, because discrete graphics have their own memory and power, they mostly outperform integrated graphics. Desktop PCs and Gaming PCs are the most popular place to find discrete graphics cards. Discrete graphics cards can be found in laptops and tiny form factor PCs.

Integrated GPUs are less powerful than dedicated GPUs since they are part of the CPU and share the same memory and data bus as the CPU. They are ideal for applications needing high processing power because they can do complex computations more rapidly and effectively.

Dedicated GPUs are frequently used in gaming, video editing, and other graphics-intensive activities that need excellent visuals and fast processing rates. They can speed up the processing of enormous data quantities and complicated algorithms in artificial intelligence, machine learning, and scientific and engineering simulations.

**Hardware Architecture:**

A graphics processing unit, or GPU, is an integrated circuit that improves performance through data management and manipulation. Deep learning involves a huge number of mathematical calculations and other procedures, which may be considerably parallelized and hence expedited by using GPUs (Graphics Processing Units). A graphics processing unit (GPU) can contain hundreds of cores, but a central processing unit (CPU) may only have twelve. Long training times and restricted GPU memory, which has a significant impact on the amount of data that can be stored on the GPU, limit the practical usage of GPU. This chapter introduces readers to a variety of distributed massively parallel systems that reduce training time and increase memory economy.

These steps have been taken to address the previously mentioned issues. The most recent Tesla graphics processing unit (GPU) has just 1 GB of RAM. This is because graphics processing units (GPUs), which are frequently used for deep learning, have a lower memory capacity than central processing units (CPUs). As a result, GPU memory cannot be easily expanded to that level; so, networks must be designed to fit within the memory that is available. It's probable that this is a barrier to progress and overcoming it would be extremely advantageous to the area of computational intelligence.

**Diagram

Description automatically generated**

A cross-platform parallel programming model called OpenCL enables programmers to create applications for a variety of compute accelerators, such as GPUs, CPUs, and FPGAs. There are many programming languages are supported by OpenCL, which offers a consistent programming interface on many hardware platforms.

The interplay between hardware generalization and specialization is fascinating to investigate and comprehend. The graphics processing unit was originally designed for graphics processing, which is more parallelized than other types of computing. Designers later discovered that GPUs could be used for more general computing tasks, which led to the development of CUDA and OpenCL. GPUs have their own set of graphics-specific resources.

**Memory hierarchy and data management:**

The memory structure of the GPU architecture is complex, consisting of registers, shared memory, and global memory. Registers are the quickest kind of memory and are used to store temporary data. Shared memory is a type of memory that is shared by all threads in a block, allowing for good synchronization and communication. Data that is shared by all threads is stored in global memory, which is the largest type of memory. In GPU-based parallel computing, robust data management is required for high-performance processing. GPUs handle data concurrently using the Single Instruction Multiple Data (SIMD) technique.

**Comparison with CPU architecture:**

CPUs and GPUs have many similarities. Both are important computer engines which are both silicon-based microprocessors. And both handles data, but CPUs and GPUs have distinct architectures and are designed for distinct purposes. The CPU is well-suited to a wide range of tasks, particularly those requiring low latency or high per-core performance. The CPU is a strong execution engine that concentrates its lesser number of cores on tasks and completing them rapidly. This makes it ideally suited to tasks ranging from serial computing to database administration. Following diagram compares the architectures of a traditional CPU and a GPU:

Diagram

Description automatically generated

comparing the architecture of a typical CPU with a GPU

The CPU is made up of billions of transistors that are linked together to form logic gates, which are subsequently linked together to form functional blocks. On a bigger scale, the CPU is made up of three major components: The Arithmetic and Logic Unit (ALU) is a collection of circuits that execute arithmetic and logic operations. The Control Unit reads instructions from the input and routes them to ALUs, cache, RAM, or peripherals. Cache contains interim values required for ALU computations or aids in the tracking of functions in the program being performed.

**Programming Environment**

Because of the expanding popularity of GPUs, high-level programming paradigms and tools are now available. GPU programming has always required the use of low-level languages and packages such as CUDA or OpenCL. An introduction to GPU computing programming models: There is some program paradigms available for GPU computation.

**CUDA model:**

The CUDA programming approach is founded on the concept of a kernel, which is a function that may be performed in parallel on the GPU. The global property specifies whether the kernel should be run on the GPU. C or C++ is used to write the kernel. The kernel is divided into threads, each of which executes the same code on a different portion of the data. The CUDA runtime system automatically maps the threads to the GPU's available processing units and arranges their concurrent execution.

A collection of libraries that support the CUDA programming model provide additional capabilities for GPU-based parallel programming. These include the CUDA BLAS Library, CUDA FFT Library, and CUDA Math Library, all of which provide optimized implementations of common mathematical operations. The CUDA FFT Library also includes efficient versions of the Fourier transform.

Overall, the CUDA programming style makes building parallel programs that operate on NVIDIA GPUs straightforward and effective. It provides a collection of tools and APIs that make it easier to write efficient and scalable parallel code, allowing programmers to take use of the GPU's high bandwidth and massive parallelism.

**OpenCL model:**

Although OpenCL programs can be compiled and linked into binary objects using traditional off-line compilation methods, OpenCL also supports run-time compilation, which allows OpenCL programs to run natively on target hardware, even on platforms that the original software developer did not create. Run-time compilation removes instruction set dependencies, allowing hardware makers to make large modifications to instruction sets, drivers, and supporting libraries from one hardware generation to the next. Applications that employ OpenCL's run-time compilation features will immediately take use of the newest hardware and software features of the target device, eliminating the requirement for the primary application to be recompiled.

The OpenCL programming model encapsulates CPUs, GPUs, and other accelerators as "devices" with one or more "compute units" consisting of one or more SIMD "processing elements" that execute instructions in lockstep. OpenCL defines four types of memory systems that devices may include: global memory, small low-latency read-only constant memory, shared local memory accessible from multiple PEs within the Each PE has its own computing unit and "private" memory or device registers.

However, managing memory transfers between the host and device as well as writing effective parallel code that can make use of the available processing resources make programming for OpenCL challenging. The OpenCL programming model is still a valuable tool for accelerating computationally demanding applications on a variety of hardware.

**OpenACC model:**

The NVIDIA HPC SDKTM with OpenACC provides scientists and researchers with a fast path to accelerated computing with minimal programming effort. By putting compiler "hints" or directives into your C11, C++17, or Fortran 2003 code, you may offload and run your code on the GPU and CPU using the NVIDIA OpenACC compiler.

In addition to the NVIDIA OpenACC compilers, the HPC SDK contains GPU-enabled libraries and developer tools to aid in your GPU acceleration efforts.

**Tools and libraries for GPU programming:**

There are several tools and packages available for GPU programming. OpenCL SDKs from various designers is among them, as is NVIDIA's CUDA Toolkit, which provides a set of libraries and tools for developing CUDA applications. GPU-optimized versions that are common operations also available in other libraries, such as cuBLAS for linear algebra and cuDNN for deep learning. In addition, many commonly used programming languages, such as Python and MATLAB, have libraries that allow programmers to use GPU hardware for specific tasks.

**Applications**

GPU-accelerated computing has been widely used to speed up computations in a variety of applications. This section will look at a variety of programs that have improved their performance by utilizing GPU processing. Financial modeling, cloud computing, machine learning and deep learning, cryptography, and high-performance computing (HPC) are examples of these.

1. High-performance computing (HPC):

The method of tackling complicated computer problems employing resilient equipment and parallel processing strategies is known as high-performance computing (HPC). HPC has substantially improved as a consequence of the development of GPU-accelerated computing, allowing for faster simulation and analysis of complex systems. The behavior of fluids, such as air flow around aircraft, has been examined using GPU-accelerated simulations, which can assist improve aircraft design and reduce fuel use.

1. Cloud computing:

Cloud computing has changed the way businesses and organizations store and manage data. Because of the speed and efficiency benefits brought about by GPU-accelerated cloud computing, large datasets may now be processed more rapidly and correctly. Massive volumes of genetic data, for example, have been evaluated in genomics research using GPU-accelerated cloud computing, allowing disease-causing mutations to be identified more quickly.

1. Machine learning and deep learning:

GPUs are still used to drive and improve gaming graphics, as well as to improve PC workstations, but their adaptability has led them to play an essential part in modern supercomputing.

They are an ideal contender for machine learning, which is the study of teaching computers to learn and act like humans, due to their high-performance computing (HPC) capabilities. This is also known as deep learning.

The goal of machine learning and deep learning is for computers to improve their learning over time on its own. This is gained via the provision of data and information in the form of observations and real-world interactions.

GPUs are important for machine learning because they can offer the bandwidth required to accept big datasets and allow for the spread of training processes, both of which may considerably speed up machine learning procedures.

GPUs, because of their capacity to do multiple computations parallelly, it has become accelerators for speeding up a wide range of operations, from encryption to networking to AI.

1. Cryptography and security:

Security and cryptography are critical topics that require significant computational resources to process and analyze data. GPU-accelerated computing has been used to improve the speed and accuracy of security-related computations, as well as the performance of encryption and decryption procedures. GPU-accelerated cryptography, for example, has been used to expedite the processing of large datasets for security research and to improve the security of online transactions.

**GPU Uses**

Graphics processing units (GPUs) were created to accelerate the process of graphic rendering. As a result, they are well suited for high-level 2D and 3D graphics, where the rendering of lighting, textures, and forms must all take place concurrently to keep images moving quickly across the screen.

GPUs can handle the ever-increasing visual demands because games are getting more computationally demanding, with hyper-realistic images and massive, complex in-game worlds; and screens require higher resolution and quicker refresh rates; GPUs can meet these needs.

**GPUs for Gaming:**

Graphics processing units (GPUs) were created to accelerate the process of graphic rendering. As a result, they are well suited for high-level 2D and 3D graphics, where the rendering of lighting, textures, and forms must all occur concurrently in order to keep images moving quickly across the screen.

GPUs can handle the ever-increasing visual demands because games are getting more computationally demanding, with hyper-realistic images and massive, complex in-game worlds; and screens require higher resolution and quicker refresh rates; GPUs can meet these needs.

This means gamers now being able to play games with higher frame rates or both.

Graphics processing units have been incredibly expensive from the inception of gaming servers, but that value will only increase as virtual reality gaming gets more widespread.

**GPUs for Video Editing and Content Creation:**

Long rendering times have plagued video editors, graphic designers, and other creative professions for years, tying up computational resources and stifling creative flow.

Graphic designers, video editors, and other professionals involved in the creation of visual material had to wait painfully long periods of time for their videos to be rendered prior to the introduction of GPUs.

Because graphics processing units (GPUs) are capable of parallel computation, rendering an excellent-quality movie or image today takes significantly less time and resources than it did previously.

**GPU for Machine Learning:**

AI and machine learning are the most interesting use case for GPU technology. Because GPUs have a huge amount of computational capability, they can facilitate incredible acceleration in workloads that benefit from GPUs' highly parallel nature, such as image recognition. Today's deep learning technologies do not command on GPUs collaborating with CPUs.

GPUs are still utilized to drive and create gaming graphics, as well as to improve PC workstations; nevertheless, due to their broad variety of applications, GPUs have also become a significant component of modern supercomputing.

The high-performance computing capabilities make them excellent choice for the machine learning.

The goal of machine learning, also known as deep learning, is to educate computers to improve their knowledge and abilities on their own over time. To accomplish this goal, we must provide facts and knowledge to youngsters through observations and interactions with the actual world.

**What I learned in this course**

I realized that good GPU-based parallel programming is required for fast, large-scale cloud computing and machine learning applications. By knowing the GPU hardware architecture, memory structure, and programming paradigms such as CUDA, OpenCL, and OpenACC, programmers may design and create highly parallel programs that fully leverage the GPU's capabilities.

Furthermore, by utilizing the tools and libraries available for GPU programming, programmers can speed up the development process. After that they achieve higher levels of optimization for their applications. Learning GPU-based parallel programming is becoming increasingly important for programmers as the need for data-intensive and computationally expensive applications in cloud computing and machine learning grows.

**Conclusion:**

Finally, GPU-based parallel computing has emerged as a powerful tool for large-scale, rapid calculations, mainly in fields like as machine learning, cryptography, financial modeling, and others. GPU hardware and programming frameworks like as CUDA, OpenCL, and OpenACC have enabled faster processing of complex data. However, several issues remain in the areas of memory management, scalability, portability, and performance optimization. Future GPU-based parallel computing advancements are expected to introduce advanced powerful and efficient hardware, improving programming environments and tools, and its unique algorithms and applications.

Overall, GPU-based parallel computing has accomplished record to be a useful tool for high-performance computing and will keep playing a big part in the development of science and innovation in the years to come.

**Reference:**

[1] Barros RC, Basgalupp MP, De Carvalho AC, Freitas AA (2012) A survey of evolutionary algorithms for decision-tree induction. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 42(3):291-312

[2] Breiman L, Friedman JH, Olshen RA, Stone CJ (1984) Classification and regression trees. Wadsworth

[3] Beyer HG, Finck S, Breuer T (2014) Evolution on trees: on the design of an evolution strategy for scenario-based multi-period portfolio optimization under transaction costs.

Swarm and Evolutionary Computation 17:74-87

[4] Biswal **B,** Behera H, Bisoi **R,** Dash **P** (2012) Classification of power quality data using decision tree and chemotactic differential evolution based fuzzy clustering. Swarm and Evolutionary Computation 4:12-24

[5] Munshi Aaftab. OpenCL Specification Version 1.0. Dec, 2008

[6] Nickolls John, Buck Ian, Garland Michael, Skadron Kevin. Scalable parallel programming with CUDA. ACM Queue. 2008;6(2):40–53.

[7] Cao J, Yin B, Lu X, Kang Y, Chen X (2018) A modified artificial bee colony approach for the 0-1 knapsack problem. Appl Intell 48(6): I 582-1595

[8] Yang Z., Zhou Q., Lei L., Zheng K., Xiang W. (2016). An IoT-cloud based wearable ECG monitoring system for smart healthcare. J. Med. Syst. 40:286. 10.1007/s10916-016-0644-9

[9] Chitty D (2016) Improving the performance of GPU-based genetic programming through exploitation of on-chip memory. Soft Comput 20(2):661—680

[10] Cano A, Zafra A, Ventura S (2015) Speeding up multiple instance learning classification rules on GPUs. Knowl Inf Syst 44(1): 127- 145

[11] Candel F, Petit S, Sahuquillo J, Duato J (2018) Accurately modeling the on-chip and off­ chip GPU memory subsystem. Futur Gener Comput Syst 82:510-519