The Advantage Analysis of GPUs Based on CUDA

Yinuo Xu *

Royal Grammar School Guildford Nanjing, Nanjing, 211800, China

* Corresponding Author Email: xuyinuo0312@gmail.com

Abstract. The development of computing technology has increasingly exposed the limitations of traditional central processing units (CPUs) and their architectures, especially handling data-intensive and highly paralleled computing tasks. This paper aims to analyze the advantages presented by graphic processing units (GPUs) based on Nvidia compute unified device architecture (CUDA). First, introduce the development background of GPUs, explaining the architecture conversion from specialized graphic processors to general-purposed parallel processors. The core research work takes a deep dive into the CUDA programming model, explaining its layered thread organization, memory structure, and software ecosystem. Then, a comparative analysis is conducted, detailing the outstanding advantages of CUDA-based GPUs over CPUs and other GPU methods in terms of performance, energy efficiency, programming flexibility, and ecosystem. In the end, the paper summarizes that the synergy between GPU hardware and CUDA software is the key driving force to break through modern computing bottlenecks and push forward the development of high-performance computing.

Keywords: CUDA, GPU, Parallel computing, Performance acceleration.

1. Introduction

In recent years, people have become increasingly reliant on the Artificial Intelligence (AI) technology in every aspect of daily life, for example, asking for a traveling plan, auto driving by Advanced Driving Assisting System (ADAS), and summarize an academic research paper. These AI-based technology applications are gradually changing everyday lives. The foundation of AI technology is machine deep learning. It is a dynamic function to get the key feature of the input, enabling computer to understand, compute with, and perform tensor operations to train themselves for certain functions like image recognition.

The intermediate step of tensor operation requires extremely high computing power, and the cost of achieving such power is nearly unacceptable by using traditional generalized computer hardware. However, using specified hardware is a way for a more affordable cost and a better efficiency. Then it leads to a concept, generalized and specialized architectures.

The very foundation of computer technology can be traced back to World War II, with the invention of the Turing machine. Initially designed to decrypt German military telegraph codes, the Turing machine is also regarded as the origin of generalized hardware [1]. As a generalized hardware, the Turing machine has the main feature: the scope of operations is a mile wide but an inch deep. It can perform almost all the operations, but with relatively low efficiency. Modern computers use the Von Neumann architecture instead of Turing machine architecture because the Von Neumann architecture optimized the operating process to achieve better energy efficiency ratio.

Specialized architecture is another different idea, and lots of hardware employ specified architectures. For instance, Tensor Processing Unit (TPU) from Google use the systolic array architecture, which breaks the regular of the Von Neumann architecture. The Von Neumann architecture has a fixed process of fetch, decode, execute, and write back, while a systolic array acquires the data once and perform all the calculations before write the result back. Compares to Von Neumann architecture, which only one of its four steps involves actual computation, and the big part of energy is consumed by getting access with the Random Access Memory (RAM), the systolic array directs most of the energy toward the calculation part, and significantly enhanced the efficiency [2]. This shows the feature of specified architecture.

The CUDA is the tool that Nvidia turns GPU to a generalized hardware but specified in certain fields. This paper will explain the main principle of GPU and CUDA, and the superiority of GPUs using CUDA.

The paper will introduce the principle of GPU firstly, then introduce CUDA and the analysis of the advantage of GPUs that based on CUDA. Finally, conclude this paper and discuss the future prediction of GPUs based on CUDA.

2. The Introduction of GPU

2.1. Background Introduction

2.1.1. The market driving forces: the evolution of 3D (Three-Dimensional) games

The game Quake released in 1996 marked that the 3D gaming has already converted to the mainstream of the gaming market from just a concept. However, the 3D gaming brought 3 major technical problems: the geometric complexity, the lighting calculation, and the texture filling. First, regarding geometric complexity, the number of polygons to be rendered in the scene has leaped from the thousands to the millions [3]. Second, lighting calculations became vastly more demanding. Each frame in the game needs to process hundreds of millions of floating-point operations to perform the Phong shading. Third, texturing filling requires an extremely high bandwidth where far exceeding the limit at that time, making it almost impossible to run smoothly just using a CPU. A fundamental flaw of the CPU in the rendering process is the parallel bottleneck. A typical CPU merely has 1 to 2 floating-point units, while needing to process more than 500 thousand pixels under the resolution of 1024*768. Therefore, the hardware accelerating was needed, and it is here that the GPU emerges as the critical solution.

2.1.2. The main definition of GPU

A GPU is a specialized parallel processor that processes graphic operations. It has a vast number of cores to accelerate the image generating and numerical calculations. When compared to a CPU, a GPU has significantly higher throughput albeit with higher computing latency. This is because GPU is using Single Instruction Multiple Threads (SIMT) architecture, whereas CPUs rely on the Von Neumann architecture. The SIMT architecture is throughput-oriented, and using more than 70% of transistors to computing instead of being cache or control [4]. Thus, GPUs typically have a better computational power than the same generation CPU. According to the specific parameters, Nvidia A100 GPU has about 100 times more cores than the Intel Xeon Platinum 8380 CPU, and roughly 50 times greater of floating-point computing power.

2.1.3. The milestone Nvidia GeForce 256

In the development of GPUs, Nvidia's GeForce 256 stands as a key milestone. It established the definition of the GPU: "A GPU is not just a 3D accelerator, but a complete computing architecture for visual computing." [5]. It innovatively integrated a hardware called T&L (Transform & Lighting) engine that firstly solved the geometric complexity and lighting calculation problem in 3D game. The T&L engine took over transform and lighting operations that a CPU used to do individually, releasing the CPU for other tasks [6]. To compare the performance of GeForce 256 with CPU Pentium III, the rate of performing transform and lighting operations of GPU is 12 times and 13 times better respectively. The T&L engine has 4 parallel ALUs (Arithmetic and Logic Unit), and made the vertex processing speed increased by 12 times [7].

2.2. The Architecture and Working Principle

The main feature of GPU that differs from CPU is parallel. This is because the performance of a single core is limited by physical properties. The main concept of SIMT is that, if a single core operates a task takes a certain time, then multiple cores operate it simultaneously takes less time. In this analogy, the single core one is just like the CPU and multi-core model is like the GPU.

The architectural evolution of GPUs can be divided into two stages. The first is the fixed function pipeline, whose operational principle is illustrated in figure 1.

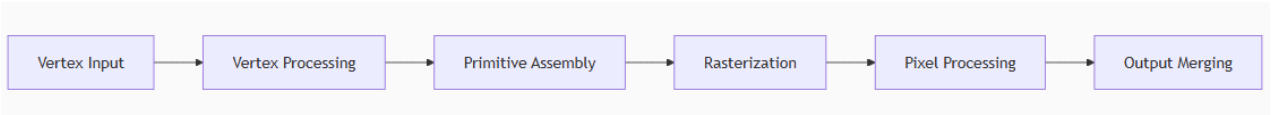

Fig. 1 The Principle of Fixed Function Pipeline (Picture credit: Original)

As suggested by its name, this pipeline is not programmable, which is its fatal shortcoming. If a developer wants to add a new visual effect, it will be impossible using this architecture. Additionally, it has resource waste to certain extent, for instance when performing complex geometric operations, the usage of the vertex unit will be high, while the pixel unit remains underused.

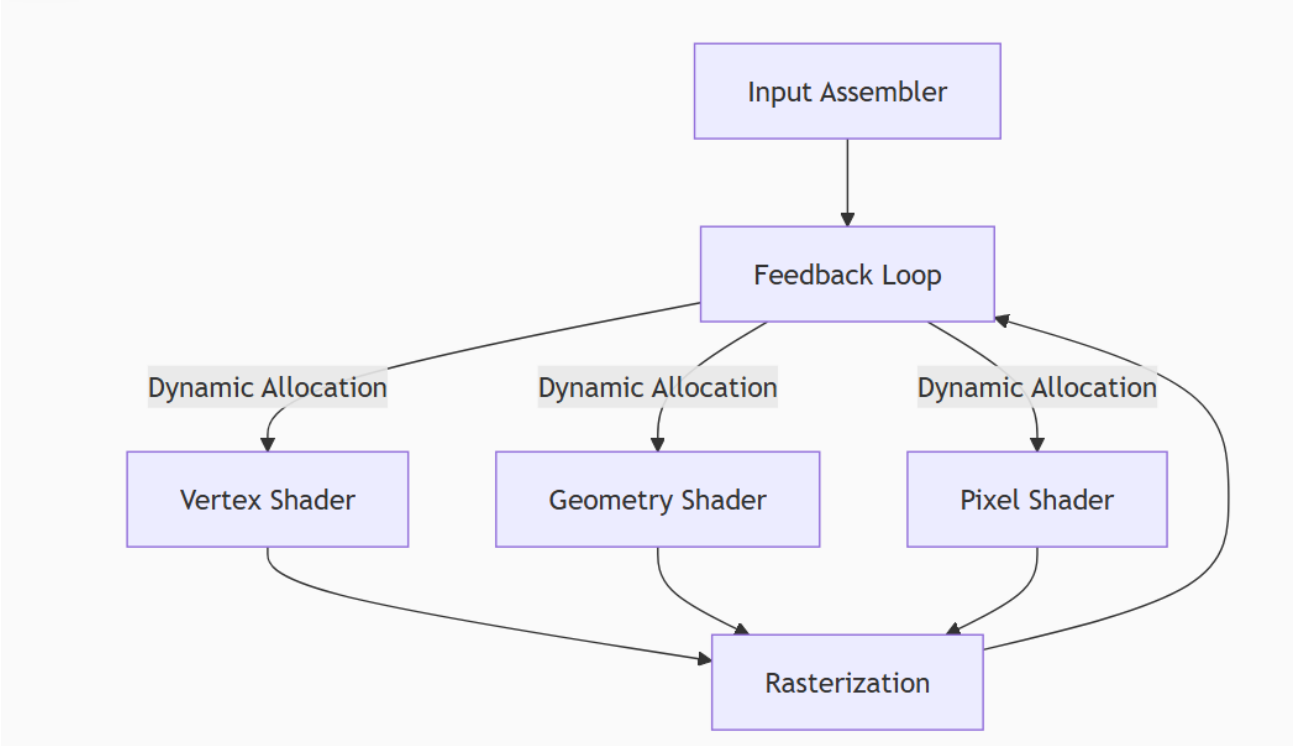

Fig. 2 The Principle of Unified Shader Architecture (Picture credit: Original)

The second stage is the unified shader architecture. As illustrated in this figure 2, a unified shader decides which components should be involved in operation. This architecture supports programmability because its progress is flexible. The unified shader architecture also has better efficiency and lower latency, because the length of the whole progress is shortened. Introduced in 2006, the unified shader architecture is no less significant than the transition of CPUs from single-core to multi-core. It breaks the physical division of graphics pipelines, reconstructs the rendering paradigm with general-purpose computing units, and lays the decisive groundwork for GPUs to conquer the field of scientific computing and AI [8].

GPUs are widely used in game rendering, professional visualization, movie industry, and virtual reality, all those fields are having a high degree of dependence.

They are also mostly used in AI training, science calculations, and financial calculations, where the usage of GPU is almost 100%

3. CUDA

3.1. Background Introduction

The development of CUDA mainly stems from three driving forces: first, Moore's Law gradually fails, and the growth of CPU computing power slows down, unable to meet the growing demand for computing efficiency in big data and intelligent applications; Secondly, the GPU itself has massively parallel computing capabilities, and its multi-core architecture is good at handling high-throughput tasks, making general-purpose GPU computing (GPGPU) possible. Finally, before the advent of CUDA, GPU programming was difficult and inefficient, and the market urgently needed an efficient programming model that could directly express the intention of parallel computing. CUDA transforms latency-sensitive tasks into throughput optimization problems through the SIMD architecture, significantly improving computational efficiency.

On November 8, 2006, Nvidia officially announced the launch of the CUDA technology. This is a new architecture for computing on Nvidia GPUs and the first development environment for C language using GPU in the industry. This milestone marked the start of GPUs transforming from specialized image processors to general-purpose parallel processors [9].

CUDA was released accompanied by the Nvidia G80 architecture (GeForce 8800 series). It introduced a unified programming model enabling developers to program directly in C language. The first-generation CUDA provided a direct programming interface, including the parallel computing engine and instruction set architecture (ISA) allowing developers use the parallel computing ability of GPU directly without learning the image API.

Additionally, CUDA enabled GPU and CPU to connect with a high efficiency data exchange way to enable GPUs to handle complex and time-consuming computing tasks in fields of science computing, machine learning, and cryptology.

The main innovation of CUDA is that it provided the layer of abstraction between the hardware and the software. Different from the traditional way that need to transform the computing task to the image processing task, this design enables GPUs to operate simultaneously as a flexible threaded processor, using thousands of threads, invoked by computing programs, to cooperate on complex tasks, and as a stream processor for specific applications.

Nvidia has built a complete ecosystem around CUDA. In the aspect of toolchain, it provides the Nvidia CUDA Complier (NVCC), Nsight Visual Studio Edition, and similar tools. For software stack, some highly optimized calculation libraries are offered, like CUDA basic linear algebra subprograms (cuBLAS), CUDA fast Fourier transform (cuFFT), and CUDA deep neural network library (cuDNN), and covered frequent compute-intensive tasks. CUDA also support multiple programming languages, and greatly lowered the barrier of developers to use GPU for general proposing.

3.2. The Architecture and Working Principle

CUDA adopts a layered parallel computing model, organizing calculation tasks into the Grid to Block to Thread three-layer structure. When a task starts, a grid will be created, containing multiple blocks, each of which contains several threads [10]. This layered design ensures both programming flexibility and mapping efficiency to the GPU's hardware architecture. The main feature of CUDA is the SIMT model. In this model, 32 threads make up a warp, and all threads in the same warp receive same instructions but process different data, significantly improving the instruction throughout.

A CUDA device contains several memory types, forming a hierarchical memory architecture. The global memory has the biggest volume but the highest latency, accessible to all the threads, and often used to store massive data. The shared memory is the high-speed memory shares by every block thread, with moderate latency, often used to store reused data or cooperating data between threads. The register is the fastest memory, used for storing intermediate computation results. There are also constant memory and texture memory, both designed for special uses.

The CUDA software stack contains several layers. The bottom layer is the hardware driving programs, above which lies the CUDA Runtime API and Driver API. The intermediate layer includes

various libraries, and the top layer are the applications and frameworks, such as the deep learning frameworks TensorFlow and the PyTorch.

The toolchain involves NVCC, which complies the CUDA code into parallel thread execution (PTX) intermediate code before converting it to the machine code for a certain GPU architecture. The Nsight environment provides debugging, performance analysis, and optimization tools, helping developers to compile more efficient code.

3.3. The Advantage Analysis of GPU Based on CUDA

CUDA presents key advantages in performance, programmability, ecosystem, and cost, underpinning its dominance in parallel computing.

The first advantage is performance. GPUs based on CUDA provides extreme parallel computing ability. Modern GPUs contain thousands of computing cores and can execute tens of thousands of threads at the same time, providing TFLOPS-level computing power. For instance, the Nvidia H100 GPU offers 1979 Tflops of FP16 computing power, far beyond contemporary CPUs [11]. In practice, CUDA often can provide the performance acceleration of 10-100 times, and over 1000 times for highly parallel tasks. Such acceleration not only reduces computing time but also enables tasks cannot accomplish possible previously. The energy efficiency ratio is another important superiority. GPUs have a higher level of energy efficiency ratio than CPUs most the times, meaning that GPUs can provided more computing power per unit of energy consumed.

The second advantage is the coding advantage. CUDA is expanded based on standard C language, and it can express the parallel computing intention with minimal additional words and structures. This significantly reduced learning cost.

CUDA further simplified the programming model through a shared memory between CPU and GPUwith automatic transfer data. This feature allows GPUs to start a new task while running, increased the coding flexibility.

The third advantage is the ecological advantages. Decades of development have built up a mature CUDA ecosystem. Almost all mainstream deep learning frameworks are based on CUDA- enabled GPUs. Rich documentation resources, an active developer community, and abundant learning resources facilitate developers to quickly solve problems encountered, creating significant competitive barriers. Although there are some other alternative ways, CUDA still keeps the leading status of the market.

The last is the cost advantage. Compares to constructing CPU compute clusters, GPUs can reduce the cost for the same performance. GPU also takes less space and lower energy consumption, lowering the total cost of owning.

In science computing, CUDA has already made a great effect. For instance, astrophysicists are using CUDA to speed up cosmological simulations, allowing them to model larger-scale structures of the universe.

In the field of AI, CUDA has almost become the factual standard. Most frameworks are accelerated by GPUs based on CUDA. In the field of natural language processing, large Transformer models like the generative pre-trained transformer (GPT)PTX series rely heavily on CUDA and GPU computing power [12]. Using multi-GPU clusters can cut training time from weeks down to just days.

4. Conclusion

This article systematically analyzes the key advantages of GPUs based on CUDA in modern computing environment. The analysis reveals that the architecture difference between CPUs and GPUs, that are sequential execution and massive parallel processing respectively. CUDA acts like a key bridge connecting GPU hardware with software developers. It unlocks the parallel computing potential of GPUs with its layered thread model, Grid-Block-Thread, detailed memory hierarchy, global memory, shared memory, registers, and a well-established software ecosystem, like cuBLAS, cuDNN, Nsight. This study summarizes the advantages of CUDA-based GPUs into four core aspects:

ultimate performance with a noticeable acceleration, excellent energy efficiency ratio, the flexibility of coding and developing efficiency, and the mature ecosystem with the support of community.

While CUDA currently leads the field, it faces future challenges and opportunities. In conclusion, with its incomparable parallel computing capabilities, GPUs based on CUDA architecture have become the core engines driving artificial intelligence, scientific discovery, and industrial innovation. Their advantages stem from Nvidia's long-term collaborative investment in hardware architecture, programming models, and software ecosystems. Nevertheless, Nvidia needs to maintain the creativity to handle challenges incoming to keep its current status.

References

[1] De Mol, Liesbeth. Turing Machines. The Stanford Encyclopedia of Philosophy, 2021.

[2] Armoni, Marco. Beyond Von Neumann. The Systolic Array Architecture. IEEE Computer Architecture Letters, 2020, 19(2): 45–48.

[3] Peddie, Jon. The History of the GPU—New Developments. Springer, 2022.

[4] Hennessy, John L., and David A. Patterson. Computer Architecture: A Quantitative Approach. 6th ed., Morgan Kaufmann, 2019.

[5] NVIDIA. SIGGRAPH 1999 Keynote. Proceedings of SIGGRAPH, 1999.

[6] NVIDIA. GeForce 256: the World's first GPU. Press Release, 1999.

[7] Peddie, Jon. The History of Visual Magic in Computers. Springer, 2022.

[8] ACM Transactions on Graphics. The Evolution of GPU Architectures: From Fixed-Function to Unified Shaders. ACM TOG, 2021, 40(4): 38.

[9] NVIDIA. CUDA: A Parallel Computing Platform and Programming Model. IEEE Micro, 2008, 28(2): 50–57.

[10] NVIDIA. CUDA Toolkit Documentation. Version 11.0, 2020.

[11] NVIDIA. NVIDIA H100 Tensor Core GPU Architecture. White Paper, 2022.

[12] Vaswani, A., et al. Attention Is All You Need. Advances in Neural Information Processing Systems (NeurIPS), 2017, 30.