FINAL REPORT

Survey on GPU Accelerators

CS/ECE 570

HIGH PERFORMANCE COMPUTER ARCHITECTURE

By

Bharath Kumar Reddy Gangavaram

Mahendra Kumar Kodidala

Manikanta Ranganath

INTRODUCTION

The rapid evolution of technology has witnessed remarkable developments in processing power, significantly shaping various industries and applications. Graphics Processing Units (GPUs), which have transformed from simple video display controllers to powerful parallel processing powerhouses, are at the forefront of this revolution. This article provides a comprehensive overview of the history and evolution of GPUs, their architectural components, and their diverse applications. The article discusses the intricacies of GPU architectures like Nvidia's GTX 980 and the advancements offered by A100 and H100 GPU accelerators, which cater to artificial intelligence, high-performance computing, and data analytics. Furthermore, we explore the impact of GPU acceleration on distributed computing systems like Apache Spark, highlighting the performance boost they offer in machine learning tasks.

1.1 GPU Accelerators

A GPU, or graphics processing unit, is a specialized processor composed of numerous cores tailored to efficiently process large batches of data by performing identical operations repeatedly at high speeds. By leveraging parallel processing, these cores work in tandem to manage thousands of threads simultaneously. GPUs are a critical component of a wide range of computing devices, from personal computers to smartphones. Initially, GPUs were developed to expedite the rendering of 3D graphics. As their flexibility and programmability improved over time, their applications broadened to include accelerated video rendering, machine learning, financial simulations and risk modeling, as well as scientific computations.

GPU acceleration refers to the technique of utilizing a GPU to execute computing tasks that are conventionally managed by a CPU. This involves transferring some processing responsibilities from the CPU to the GPU, which excels in parallel processing and can simultaneously handle multiple tasks. By harnessing the power of GPUs, users can benefit from faster processing speeds, enhanced efficiency, and cost-effective solutions.

The two primary categories of GPU acceleration are Compute acceleration, that is employed for general-purpose computing tasks, such as scientific computations, data analysis, and machine learning and Graphics acceleration that is utilized for activities like gaming and video rendering.

1.2 Types of GPU Accelerators

i. iGPU

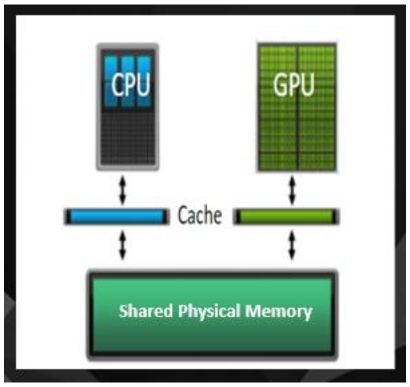

Fig.1 iGPU

An integrated GPU, or iGPU, is a graphics processing unit that is combined with the CPU on a single chip. This type of GPU is built into the motherboard and is commonly found in ultra-thin devices such as laptops and Intel® NUCs. The iGPU offers several advantages, including lower power consumption, which leads to reduced heat generation and potentially longer battery life. Additionally, devices with integrated graphics tend to be more affordable. While iGPUs can handle light gaming and editing tasks, their processing power is generally more limited compared to dedicated GPUs. Intel is among the companies that manufacture CPUs with integrated graphics. The performance of iGPUs is largely dependent on the capabilities of the other components within the system.

ii. dGPU

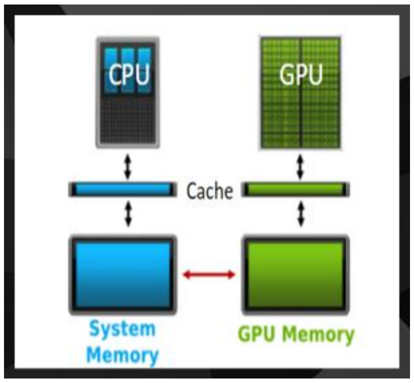

Fig.2 dGPU

Discrete graphics refers to a GPU that is separate from the processor. As an independent component, discrete graphics offer several advantages, including their own dedicated VRAM (video RAM or video random access memory). However, they also consume more power and generate substantial heat. Discrete graphics typically provide higher performance than integrated graphics and are predominantly found in desktop PCs, although some laptops and small form factor PCs may also feature discrete graphics cards. These GPUs connect to the motherboard and are larger and more power-hungry than their integrated counterparts. Discrete graphics are ideal for running resource-intensive graphics, effects, and videos, as well as for executing high-performance tasks. NVIDIA and AMD are the leading manufacturers in the discrete GPU market.

1.3 History and Evolution of GPU

The Graphics Processing Unit (GPU) has undergone a remarkable transformation over the past decades, impacting computer graphics, parallel computing, gaming experiences, and artificial intelligence applications. This article focuses the history and evolution of the GPU, from its humble beginnings as simple video display controllers to today's sophisticated parallel processing powerhouses.

The journey of the GPU began with the development of basic video display controllers in the 1970s and 1980s, which primarily focused on generating video signals. It wasn't until 1994 when Sony first used the term "Graphics Processing Unit" (GPU) with the release of their PlayStation console (Smith, 2015). In 1999, NVIDIA launched the GeForce 256, the first GPU to combine transformation and lighting calculations (T&L) into a single chip, leading to more intricate and real-time graphics rendering (NVIDIA, n.d.).

Throughout the early 2000s, GPU technology advanced rapidly, with NVIDIA and ATI (now part of AMD) competing to create increasingly powerful and efficient GPU architectures. By 2002, the concept of General-Purpose computing on Graphics Processing Units (GPGPU) emerged, broadening the scope of GPUs beyond graphics-related tasks. This period of growth saw NVIDIA ship 500 million GPUs by 2006 (Hruska, 2006). That same year, NVIDIA introduced the G80 architecture, which featured unified shaders and paved the way for general-purpose GPU computing. This shift transformed GPUs from specialized graphics processors into versatile parallel computing devices. The move towards general-purpose GPU computing unlocked new applications beyond graphics processing. NVIDIA unveiled CUDA (Compute Unified Device Architecture) in 2006, a parallel computing platform that enabled developers to use GPUs for various computing tasks (NVIDIA, 2006). This innovation allowed GPUs to be applied in areas such as artificial intelligence and high-performance computing (Owens et al., 2008).

OpenCL (Open Computing Language), launched in 2009, is an open standard for parallel programming of heterogeneous systems (Khronos Group, 2009). It allows developers to leverage the computing power of different hardware platforms, such as CPUs, GPUs, Digital Signal Processors (DSPs), and Field-Programmable Gate Arrays (FPGAs), using a single codebase. OpenCL's flexibility makes it possible for developers to optimize code for specific hardware architectures and create performance-portable applications.

Introduced in 2011, OpenACC is a high-level, directive-based programming model designed to simplify parallel application development on heterogeneous systems (OpenACC, 2011). OpenACC enables developers to offload compute-intensive tasks to accelerators like GPUs without extensive code modifications, improving productivity and allowing them to focus on algorithms and overall application performance. OpenACC is especially valuable for scientists and engineers who may not be experts in parallel programming, as it abstracts the complexities of hardware-specific optimization and parallelization.

By 2020, many of the world's top supercomputers integrated GPU accelerators to enhance their computational abilities. As highlighted by the June 2020 TOP500 list, roughly 25% of these high-performance systems used GPUs for faster processing and improved energy efficiency (TOP500, 2020). The inclusion of GPUs in supercomputers facilitated efficient handling of complex simulations, data analysis, and artificial intelligence tasks.

The growing use of GPU accelerators in supercomputers demonstrates the trend towards heterogeneous computing architectures, which blend different processor types for optimal performance and energy efficiency (Mittal & Vetter, 2015). Due to their parallel processing capabilities, GPUs can execute multiple threads simultaneously, making them ideal for accelerating a diverse range of applications in science, engineering, and machine learning.

1.4 CPU vs GPU

Over the past few decades, the field of computing has witnessed significant advancements in processing technology. Two primary types of processors have dominated this landscape: The Central Processing Unit (CPU) and the Graphics Processing Unit (GPU).

i. Architectural differences between CPUs and GPUs

A primary distinction between CPUs and GPUs stems from their architectural design. The CPU consists of a limited number of cores equipped with substantial cache memory, allowing it to manage a few software threads concurrently. These CPU cores are optimized for handling intricate operations, including data transfer between local registers, various cache levels (L1, L2, and L3), RAM, and hard disk storage, while efficiently switching between application tasks. CPUs are typically designed for sequential processing, excelling in a broad array of tasks and emphasizing single-threaded performance.

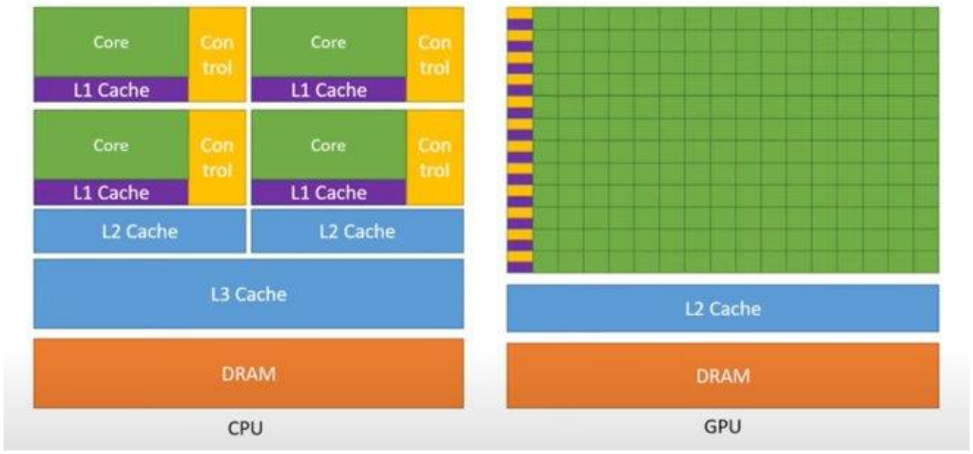

Fig.3 CPU vs GPU

Conversely, a GPU boasts a significantly larger number of cores than a CPU, enabling it to process thousands of threads simultaneously. The multitude of cores in a GPU allows it to fetch data from memory, conduct parallel computations, and return the data for further use. GPUs are engineered for parallel processing and focus on executing multiple threads concurrently. This design makes them well-suited for handling data-intensive tasks such as rendering graphics and conducting complex mathematical operations.

ii. Performance comparison in general-purpose computing

The growing interest in using GPUs for general-purpose computing (GPGPU) has led to numerous studies comparing their performance with that of traditional CPUs. GPUs can outperform CPUs in various applications, such as machine learning, simulations, and data analysis, due to their ability to process multiple threads simultaneously. However, it is essential to note that GPUs may not always be the optimal choice for all applications, as some tasks may not be well-suited for parallel execution or may require specific instruction sets that are better supported by CPUs. The performance of GPUs is calculated in terms of FLOPS (Floating Point Operations Per Second).

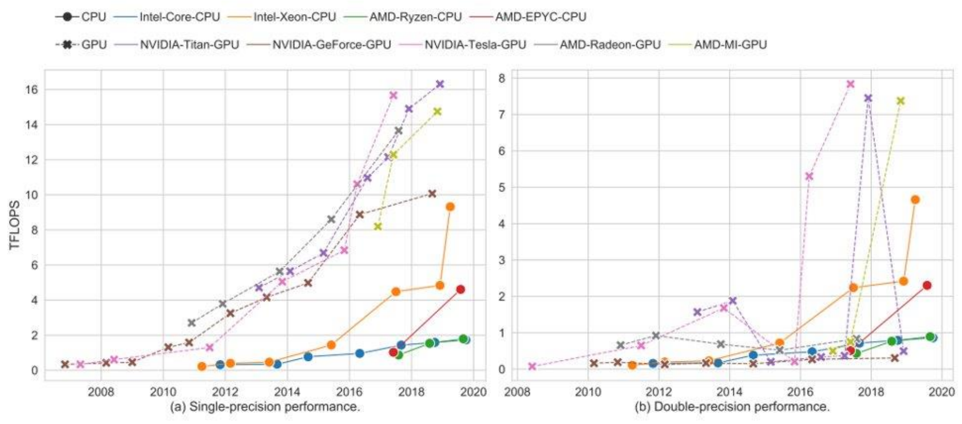

Fig.4 Performance of CPU vs GPU

According to study by Yifan Sun, Nicolas Bohm Agostini, Shi Dong, and David Kaeli, the performance of GPUs is exponentially increasing and the FLOPS per watt doubles around every three to four years.

iii. Energy efficiency considerations

CPUs generally consume more power to achieve their goals, especially when handling diverse tasks. However, for parallelizable tasks, CPUs may not be the most energy-efficient choice due to their limited number of cores and sequential processing approach. On the other hand, GPUs can execute these tasks with lower energy consumption per operation than CPUs, thanks to their architecture, which is specifically designed for the concurrent execution of thousands of threads. Nevertheless, when dealing with workloads that are not parallelizable, GPUs may use more energy without delivering a corresponding increase in performance.

A study by Yifan Sun, Nicolas Bohm Agostini, Shi Dong, and David Kaeli demonstrated that GPUs often provide better performance-per-watt ratios compared to CPUs for various parallel computing tasks, making them a more energy-efficient option in such situations. However, the energy efficiency of GPUs can be affected by factors including programming models, memory access patterns, and the degree of parallelism present in the application.

GPU ARCHITECTURE

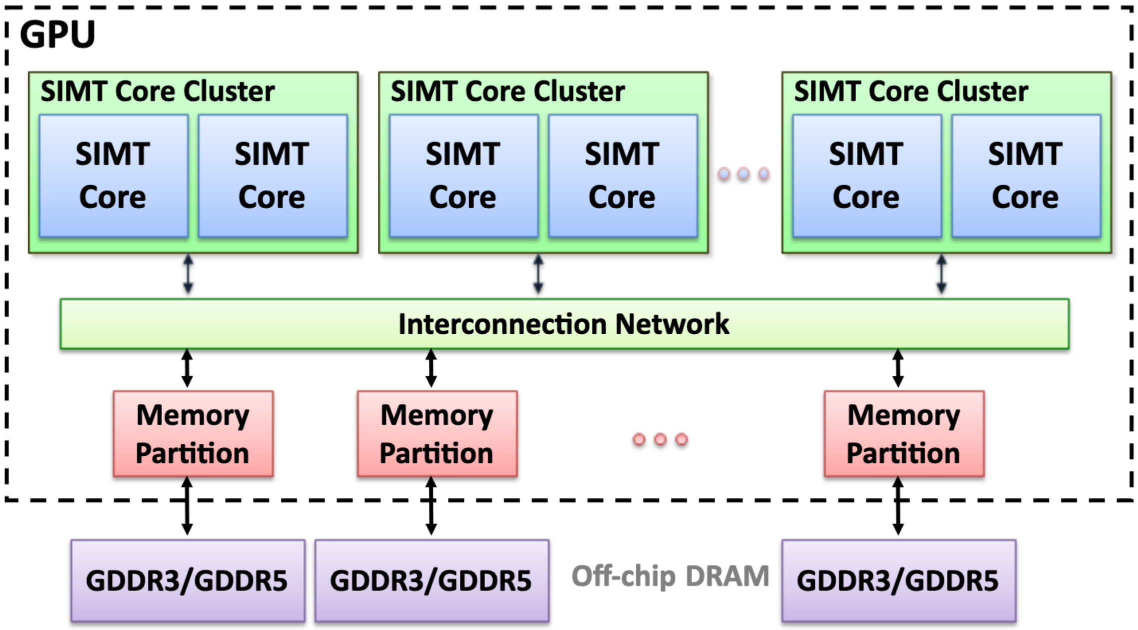

Fig.5 GPU Architecture

The architecture of a GPU includes various components designed to enable efficient processing of computationally intensive tasks such as graphics rendering and scientific simulations. The GPU's processing power is harnessed through the use of multiple cores, which can each execute a single-instruction multiple-thread (SIMT) program. These cores are capable of running thousands of threads simultaneously and can communicate with each other through scratchpad memory and barrier operations. Additionally, each core has its own first-level instruction and data caches, which serve to reduce memory system traffic. The large number of threads running on each core helps to compensate for the latency involved in accessing memory when data is not found in the first-level caches. (Aamodt el at. 2018).

In general, in a generic GPU architecture, a GPU has multiple SIMT (single instruction multiple threads) clusters, each cluster is mapped to a particular region in the memory and each cluster has numerous SMMs. some common components that are typically included

Streaming Multiprocessors (SMs)

SMs are the main processing units within a GPU. Each SM typically contains multiple CUDA cores (or equivalent) that can perform calculations in parallel. The number of SMs can vary depending on the GPU model.

Memory

GPUs typically have their own dedicated memory (VRAM) that is used to store data and instructions for processing. The amount of memory can vary depending on the GPU model.

Memory Controller

The memory controller is responsible for managing the flow of data between the GPU's memory and the processing units.

Texture Mapping Units (TMUs)

TMUs are responsible for mapping textures onto 3D objects in a scene.

Raster Operations Pipeline (ROP)

The ROP is responsible for finalizing the rendering process by processing pixel data and generating output images.

Compute Capability

Modern GPUs are capable of performing more than just graphics rendering and are often used for general-purpose computing tasks such as machine learning and scientific simulations. The computing capability of a GPU refers to its ability to perform these types of calculations.

Bus Interface

The bus interface is responsible for communicating with the CPU and other components in the system, allowing data to be transferred to and from the GPU.

2.1 Microarchitecture of SIMT CORE

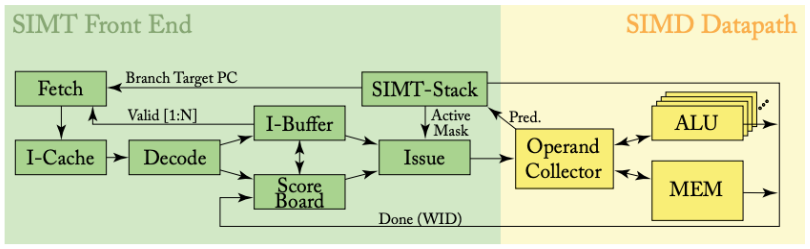

Fig.6 SIMT Core

In order to optimize performance, GPUs organize threads into groups known as warps or wave fronts by NVIDIA and AMD, respectively. These warps represent the unit of scheduling, with the hardware selecting a warp for scheduling during each cycle. Using the one-loop approximation, the program counter of the selected warp is used to access instruction memory and find the next instruction to execute. Once the instruction is fetched, it is decoded and source operand registers are fetched from the register file. The SIMT execution mask values are determined in parallel with fetching source operands from the register file. Execution proceeds in a single-instruction, multiple-data manner with each thread executing on the functional unit associated with a lane provided the SIMT execution mask is set. GPUs may contain several function units, each supporting only a subset of instructions. To achieve higher performance per unit area, some GPUs execute a single warp or wave front over several clock cycles by clocking the function units at a higher frequency. When scheduling a large number of warps, it can be difficult to keep track of each state and requires a significant number of registers. To address this issue, the number of warps can be decreased, and the number of instructions scheduled from the same warp can be increased. A scoreboard is used to keep track of dependencies, which helps remove any potential conflicts that may arise from reading operands from the same register bank. To avoid stalls and optimize throughput, instructions may be replayed and kept in the instruction buffer until at least some part of the instruction is completed. Read barriers are also utilized to prevent hazards, and overlap between register banks can be used to avoid conflicts when reading operands from the same bank (Aamodt el at. 2018).

2.2 Architecture of Nvidia GeForce GTX 980

The GTX 980 is based on Nvidia's Maxwell architecture, which is designed to be power-efficient while still delivering high performance. It has 2048 CUDA cores, which are used for the parallel processing of graphics and other computations. In total it has 4 memory controllers each attached to a graphic processing cluster. Each cluster has 4 SMMs, in total it has 16 SMSs. In SMMs, each SMM has its own I-buffer and warp scheduler, register file, and dispatch units. One SMM consists of 128 cores which are divided into 4 blocks. Along with this, it has a Load/Store unit and SFUs. The GTX 980 has a base clock speed of 1126 MHz, which can be boosted to 1216 MHz under load. It also has 4 GB of GDDR5 memory with a 256-bit memory interface, providing a maximum bandwidth of 224 GB/s.

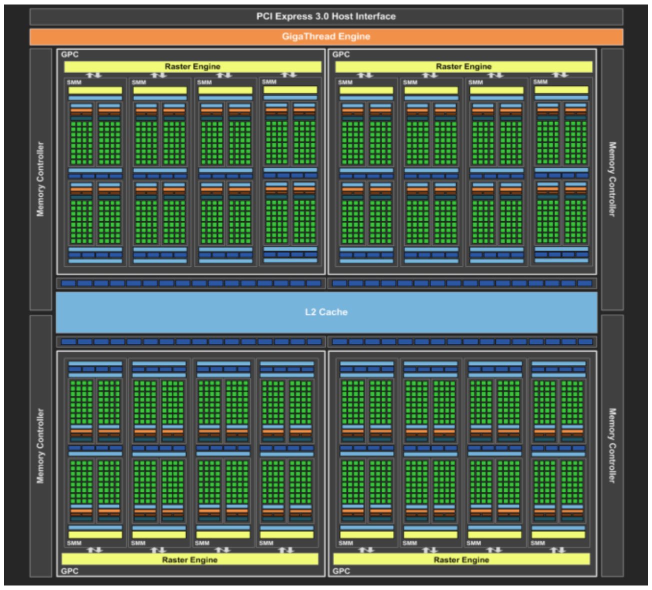

Fig.7 Nvidia GeForce GTX 980

One of the key features of the Maxwell architecture is the use of dynamic parallelism, which allows the GPU to create and launch new threads on the fly, without having to return to the CPU. This allows for more efficient processing of complex tasks, such as physics simulations or artificial intelligence algorithms.

The GTX 980 also includes Nvidia's VXGI (Voxel Global Illumination) technology, which enables real-time global illumination and dynamic reflections in games. It also supports Nvidia's G-Sync technology, which synchronizes the refresh rate of the monitor with the output of the GPU, eliminating screen tearing and reducing input lag. In terms of power consumption, the GTX 980 has a TDP (thermal design power) of 165 watts, which is relatively low compared to previous high-end graphics cards. This is partly due to the use of Nvidia's MFAA (Multi-Frame Anti-Aliasing) technology, which provides improved image quality with less processing power than traditional anti-aliasing techniques (NVIDIA., 2014).

LATEST GPU NVIDIA H100

3.1 NVIDIA

NVIDIA has been a pioneer in the field of GPU accelerators and has contributed significantly to the development and adoption of GPU computing. They offer a wide range of GPU accelerators tailored for various applications, including high-performance computing, artificial intelligence, and gaming.

3.1.1 CUDA Architecture

The Compute Unified Device Architecture (CUDA) is a parallel computing platform and programming model developed by NVIDIA for general-purpose GPU computing. CUDA allows developers to harness the parallel processing capabilities of NVIDIA GPUs using a C-like programming language. The architecture consists of an array of streaming multiprocessors (SMs) that contain a set of scalar processors (SPs), each capable of executing multiple threads concurrently (Gupta et al., 2012).

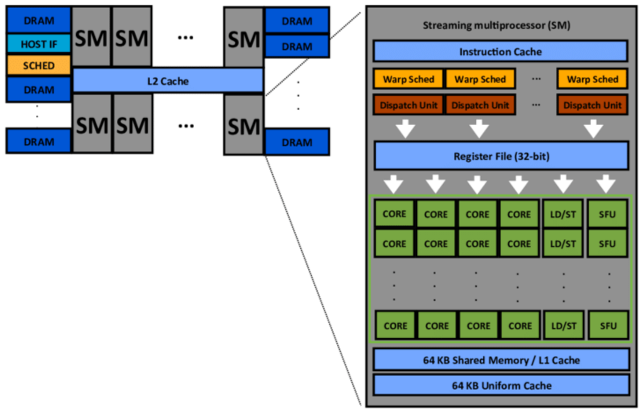

Fig.8 CUDA Architecture

In a CUDA-enabled GPU, threads are organized into thread blocks, which are then distributed among SMs. Each SM manages the scheduling and execution of its threads, as well as the local memory and registers for its assigned thread blocks. This hierarchical structure allows for efficient parallel processing, leveraging the immense computational capabilities of modern GPUs for a wide range of applications, including high-performance computing, artificial intelligence, and scientific simulations.

3.1.2. A100 GPU Accelerator

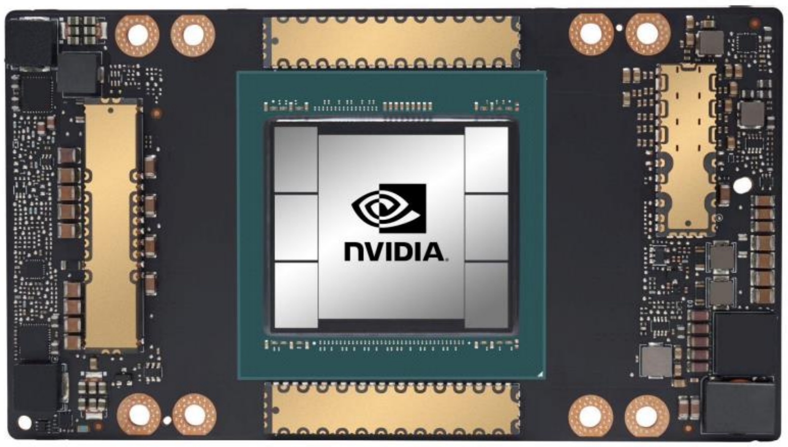

Fig.9 NVIDIA A100 GPU on the new SXM4 module.

The A100 GPU accelerator is based on NVIDIA's Ampere architecture, which is designed to deliver significant improvements in performance, energy efficiency, and scalability for a wide range of workloads, including artificial intelligence, deep learning, and high-performance computing. The A100 GPU features several technical innovations that contribute to its impressive performance

Third-generation Tensor Cores

These specialized hardware components provide high-performance AI and deep learning computations, enabling faster training and inference times for large neural networks. They support mixed-precision arithmetic, allowing developers to use lower-precision data types without compromising model accuracy (NVIDIA, 2020).

Multi-Instance GPU (MIG)

MIG technology enables the A100 GPU to be partitioned into multiple independent instances, each with dedicated resources, to optimize resource utilization and performance. This feature is particularly beneficial for data center and cloud environments, where multiple users and workloads share the same GPU resources (NVIDIA, 2020).

Enhanced memory architecture

The A100 GPU incorporates a high-bandwidth memory (HBM2) architecture, providing a substantial increase in memory bandwidth and capacity compared to previous generations. This improvement enables the A100 to handle larger datasets and more complex workloads with ease.

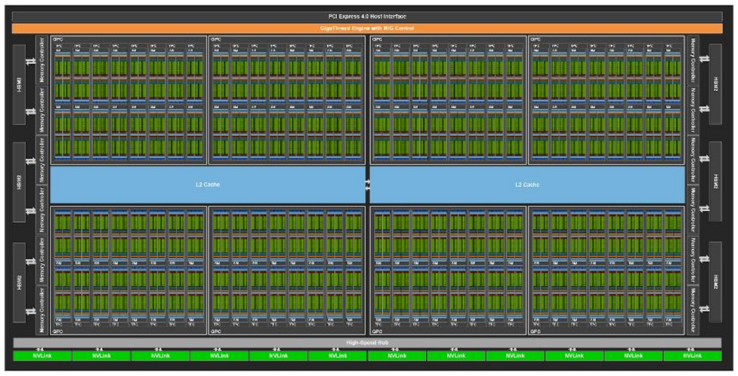
PCI Express 4.0 Host Interface
GigaThread Engine with MIG Control
GPC
TPC
SM
Memory Controller
HBM2
L2 Cache
High-Speed Hub
NVLink

Fig.10 GA100 Full GPU with 128 SMs. The A100 Tensor Core GPU has 108 SMs.

A100 GPU Accelerator Applications

The A100 GPU accelerator's exceptional performance and scalability make it well-suited for various applications across different industries. Some notable use cases include:

Deep learning

The A100's third-generation Tensor Cores enable faster training and inference of deep learning models, which are widely used in image and speech recognition, natural language processing, and recommendation systems (NVIDIA, 2020).

High-performance computing (HPC)

The A100 GPU's high computational power and memory bandwidth make it an ideal choice for HPC workloads, such as fluid dynamics simulations, molecular modeling, and climate research (NVIDIA, 2020).

Data analytics

The A100's advanced memory architecture allows it to handle large datasets and perform complex calculations, making it suitable for big data processing and analytics tasks in areas like finance, healthcare, and marketing.

3.1.3. H100 GPU Accelerator

The H100 GPU accelerator is based on NVIDIA's Hopper architecture, which further enhances performance and efficiency for high-performance computing and AI workloads.

Fig.11 NVIDIA H100 GPU on new SXM5 Module

Some key technical features of the H100 GPU include:

Fourth-generation Tensor Cores

The H100 GPU incorporates the latest generation of Tensor Cores, which deliver higher throughput for AI and deep learning tasks compared to the third-generation Tensor Cores found in the A100 GPU. This improvement results in faster training and inference times for a variety of AI workloads.

Sparsity Engine

The H100 GPU introduces a new Sparsity engine, which is designed to accelerate inference workloads with sparse neural networks. By exploiting the inherent sparsity in many AI models, the Sparsity engine can deliver significant performance improvements for inference tasks (NVIDIA, 2022).

Advanced memory architecture

The H100 GPU features HBM3 memory technology, providing even higher memory bandwidth and capacity than the A100 GPU. This improvement allows the H100 to handle even larger datasets and more complex workloads with ease.

H100 GPU Accelerator Applications

The H100 GPU accelerator builds upon the capabilities of the A100 GPU and offers further performance improvements, making it suitable for even more demanding workloads.

Some applications for the H100 GPU include:

AI and deep learning

The H100's fourth-generation Tensor Cores and Sparsity engine provide even greater throughput for AI and deep learning tasks, allowing faster training and inference times for various AI workloads.

Advanced HPC

The H100 GPU's increased computational power and memory bandwidth make it an ideal choice for more complex HPC workloads, such as large-scale simulations, quantum chemistry, and astrophysics research (NVIDIA, 2022).

Real-time analytics

The H100's advanced memory architecture and high throughput capabilities enable it to process large datasets in real-time, making it suitable for applications like fraud detection, cybersecurity, and smart city management.

In summary, both the A100 and H100 GPU accelerators offer impressive performance and scalability for a wide range of applications in artificial intelligence, high-performance computing, and data analytics. Their advanced architectures, featuring Tensor Cores, Sparsity engines, and cutting-edge memory technologies, provide the necessary computational power and flexibility to meet the demands of modern workloads across various industries.

3.1.4. Performance of A100 and H100 GPU Accelerators

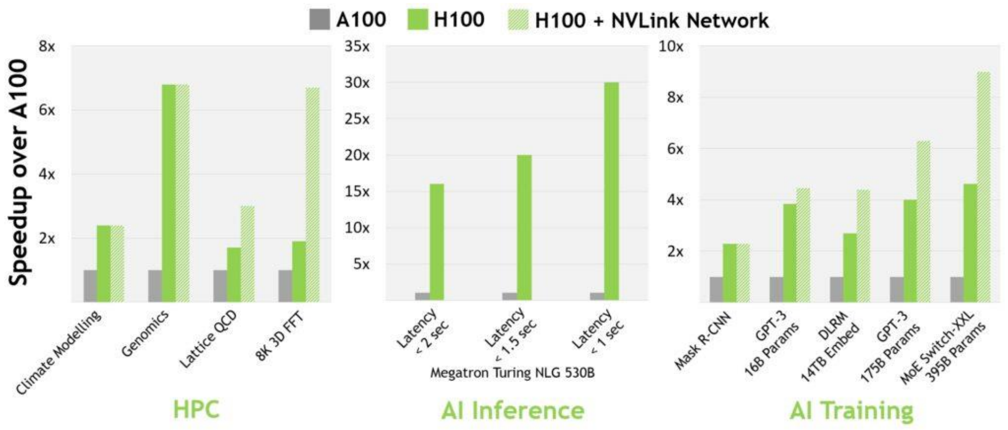

Fig.12 Performance of A100 and H100

The A100 cluster utilizes HDR InfiniBand network, while the H100 cluster employs NDR InfiniBand network with the NVLink Switch System where indicated. The number of GPUs used varies for different workloads (NVIDIA Developer, 2022).

During GTC Spring 2022, NVIDIA introduced the Grace Hopper Superchip product. The Hopper H100 Tensor Core GPU will power the Grace Hopper Superchip CPU+GPU architecture, specifically designed for terabyte-scale accelerated computing, delivering 10x greater performance on large-model AI and HPC applications.

The Grace Hopper Superchip takes advantage of the versatility of the Arm architecture to create a CPU and server architecture purpose-built for accelerated computing. The H100 GPU is connected to the NVIDIA Grace CPU through a high-speed NVIDIA chip-to-chip interconnect, providing 900 GB/s of total bandwidth, which is seven times faster than PCIe Gen5. This cutting-edge design offers up to 30x higher aggregate bandwidth compared to the fastest servers available today and up to 10x greater performance for applications that process terabytes of data (NVIDIA Developer, 2022).

CASE STUDY

Apache spark 3.0

Apache Spark is an open-source distributed computing system that enables the processing of large-scale data sets. With the introduction of Spark 3.0, the system now supports GPU acceleration, which significantly boosts the performance of machine learning tasks. Spark's new GPU support leverages the power of GPUs to speed up the training and inference of machine learning models. GPUs excel at performing matrix operations, which are critical for many machine-learning algorithms.

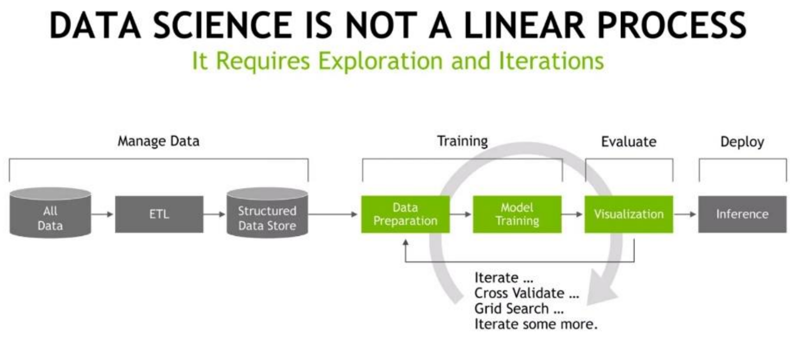

Fig .13 Data Science Process

To provide GPU support, Spark 3.0 includes the CUDA-based Spark Accelerator for Apache Arrow or cuDF. cuDF is a GPU DataFrame library that provides a Pandas-like API for working with data on the GPU. It enables data scientists and engineers to take advantage of the parallel processing capabilities of the GPU to speed up data manipulation and preprocessing tasks, such as joining, filtering, and aggregating. In addition to cuDF, Spark 3.0 also includes GPU-accelerated MLlib algorithms for common machine learning tasks, such as linear regression, logistic regression, and k-means clustering.

A typical data science problem follows these steps in its data pipeline path. Collecting the data, extracting the data, and loading the data into the database. Storing the data in the structured data store. Then it follows these steps in data training, data preparation, model training, and evaluating the model and inference from the data. Between each step, there is a time delay because this pipeline is divided into multiple stages and it has very high latency due to stalls between each step.

To eliminate these stalls, in apache spark 3.0, all the steps in the data pipeline are integrated into a single data pipeline. From collecting data to deploying data can be automated in a single pipeline which reduces latencies.

The main advantage of apache spark 3.0 is it has added support for GPU acceleration through RAPIDS, a bundle of open-source software libraries developed by NVIDIA to enable GPU acceleration for data processing and machine learning workloads. With RAPIDS, Spark can take advantage of GPUs to accelerate data processing and machine learning tasks. This is achieved by offloading compute-intensive tasks to the GPU, which can perform computations in parallel, leading to significant performance gains.

RAPIDS integrate with Spark through the use of Spark plugins, which enable users to take advantage of GPU acceleration without modifying their Spark code. With the RAPIDS plugin, Spark can leverage the power of GPUs to accelerate data processing and machine learning workloads, including ETL, SQL queries, and machine learning algorithms. One of the key benefits of using RAPIDS with Spark is the ability to process large volumes of data much faster than traditional CPU-based systems. This is particularly important for use cases that require processing large amounts of data in near real-time, such as fraud detection, recommendation engines, and financial analytics.

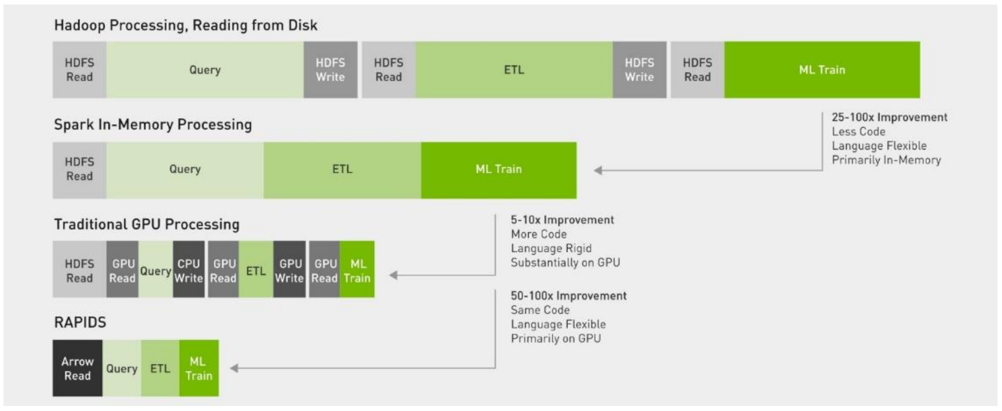

Fig.14 Rapids

Through rapids, it is possible to achieve 50 to 100 times faster speed when compared to a data pipeline that involves HDFS read and disk operations and with a normal spark in-memory data pipeline it is possible to achieve 25 to 100 times faster speed for the same pipeline (NVIDIA., 2023).

REFERENCES

Smith, R. (2015). 20 Years of PlayStation: A Retrospective. Retrieved from https://www.anandtech.com/show/9795/20-years-of-playstation-a-retrospective

NVIDIA. (n.d.). GeForce 256: The First GPU. Retrieved from https://www.nvidia.com/en-us/geforce/20-series/

Hruska, J. (2006). NVIDIA Ships 500 Millionth GPU. Retrieved from https://www.extremetech.com/extreme/76208-nvidia-ships-500-millionth-gpu

Owens, J. D., Luebke, D., Govindaraju, N., Harris, M., Krüger, J., Lefohn, A. E., & Purcell, T. J. (2008). A Survey of General-Purpose Computation on Graphics Hardware. Computer Graphics Forum, 26(1), 80-113. doi:10.1111/j.1467-8659.2007.01103.x

Khronos Group. (2009). OpenCL - The Open Standard for Parallel Programming of Heterogeneous Systems. Retrieved from https://www.khronos.org/opencl/

TOP500. (2020). TOP500 List - June 2020. Retrieved from https://www.top500.org/lists/top500/2020/06/

Mittal, S., & Vetter, J. S. (2015). A Survey of CPU-GPU Heterogeneous Computing Techniques. ACM Computing Surveys (CSUR), 47(4), 1-35. doi:10.1145/2788397

Yifan Sun, Nicolas Bohm Agostini, Shi Dong, and David Kaeli (2020). Summarizing CPU and GPU Design Trends with Product Data. https://doi.org/10.48550/arXiv.1911.11313

Rogers, Aamodt., W. W. L., && T. M., Fung, T. G. (2018). General-Purpose Graphics Processor Architectures. Book.

NVIDIA Corporation. (2014). NVIDIA Maxwell GM204 Architecture Whitepaper.Microway, Inc.https://www.microway.com/download/whitepaper/NVIDIA_Maxwell_GM204_Architecture_Whitepaper.pdf

Gupta, S., Agrawal, A., Gopalakrishnan, K., & Narayanan, P. (2012). Deep Learning with Limited Numerical Precision. In International Conference on Machine Learning (pp.

1737-1746). Retrieved from https://www.researchgate.net/figure/High-level-overview-of-a-CUDA-GPU-architecture_fig1_274653719

NVIDIA. (2020). NVIDIA A100 Tensor Core GPU Architecture. Retrieved from https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/nvidia-ampere-architecture-whitepaper.pdf

NVIDIA. (2022). NVIDIA H100 Tensor Core GPU: Hopper Architecture. Retrieved from https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/nvidia-hopper-architecture-whitepaper.pdf

NVIDIA. (2020). NVIDIA A100 Tensor Core GPU Architecture. Retrieved from https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/nvidia-ampere-architecture-whitepaper.pdf

NVIDIA. (2022). NVIDIA H100 Tensor Core GPU: Hopper Architecture. Retrieved from https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/nvidia-hopper-architecture-whitepaper.pdf

NVIDIA. (2023). NVIDIA Apache Spark™ 3.0.0 GPU Acceleration for Data Science. NVIDIA. https://www.nvidia.com/en-us/deep-learning-ai/solutions/data-science/apache-spark-3/