Accelerating Computational Workloads: GPU Architectures, Programming Models, and Applications

VASANTH PUGALENTHI, California Polytechnic State University, United States
JOAQUIN ARREDONDO, California Polytechnic State University, United States
SABAWOON HAKIMI, California Polytechnic State University, United States
TINH-PHONG NGUYEN, California Polytechnic State University, United States

Parallel computing has been transformed with the introduction of Graphics Processing Units (GPUs), which provide previously unheard-of scalability and performance for demanding applications like real-time simulations, scientific workloads, and AI training. GPUs, which were once created for graphics rendering, are now capable of handling hundreds of threads at once, revolutionizing the way data-intensive tasks are carried out. With an emphasis on their function in speeding up workloads in edge computing, distributed systems, and network applications, this analysis investigates the relationship between GPU-based parallel computing and networking systems. In order to satisfy the growing expectations of scalable performance, high throughput, and ultra-low latency in contemporary applications, networking and GPU convergence is essential. The importance of GPUs in accomplishing these goals is demonstrated by important use cases including distributed machine learning, network function virtualization (NFV), and real-time packet processing. Furthermore, new opportunities for parallel and distributed task optimization have been made possible by the incorporation of GPUs into distributed systems such as blockchain, federated learning, and edge AI. This review offers a targeted but thorough summary of recent developments in the field by looking at GPU-based solutions for network problems such edge-based inference, scalable distributed computing frameworks, and high-speed packet processing. For researchers and practitioners looking to use GPU capabilities to handle the growing complexity of networking systems, this work attempts to provide a fundamental resource.

CCS Concepts: • **Do Not Use This Code → Generate the Correct Terms for Your Paper**; *Generate the Correct Terms for Your Paper*; Generate the Correct Terms for Your Paper; Generate the Correct Terms for Your Paper.

Additional Key Words and Phrases: Do, Not, Us, This, Code, Put, the, Correct, Terms, for, Your, Paper

ACM Reference Format:
Vasanth Pugalenthi, Joaquin Arredondo, Sabawoon Hakimi, and Tinh-Phong Nguyen. . Accelerating Computational Workloads: GPU Architectures, Programming Models, and Applications. , ( ), 12 pages. https://doi.org/XXXXXXX.XXXXXXX

Authors' Contact Information: Vasanth Pugalenthi, California Polytechnic State University, San Luis Obispo, United States, vasanth.pugalen@gmail.com; Joaquin Arredondo, California Polytechnic State University, San Luis Obispo, United States, joaquinanthonyarredondo@gmail.com; Sabawoon Hakimi, California Polytechnic State University, San Luis Obispo, United States, sabohakimi@gmail.com; Tinh-Phong Nguyen, California Polytechnic State University, San Luis Obispo, United States, tinhphong04@gmail.com.

ACM XXXX-XXXX//-ART
https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Brief History of GPUs

The concept of the GPU evolved from the early graphic controllers used by microcomputers during the 1980s, and pioneering companies in the graphics acceleration field included ATI, Matrox, and Nvidia [Peddie 2023]. The revolution of the GPU as a programmable, not fixed-function, hardware occurred in the late 1990s when Nvidia introduced the GeForce 256, which had transformation and lighting (TL) in the processor [Peddie 2023].

Early GPUs performed fixed-function tasks such as rendering simple shapes and textures. Early 2000s' programmable shaders permitted more flexible rendering techniques [Peddie 2023]. Unified shader architectures allowed greater GPU capabilities to perform general-purpose computations. In recent times, AI-accelerated processing and compute shaders enabled GPUs to support diverse applications other than graphics, including machine learning, simulations, and scientific computing [Peddie 2023].

Reasons for Conducting the Survey

GPUs have emerged as a component of numerous applications, from AI, data sciences, to high performance computing. Their integration into networked systems, distributed as well as edge computing, is becoming the norm. In this paper, we do just that: discuss how GPUs are accelerating computation across all domains, and provide a structured overview of their architecture, their programming models, and future applications.

2 GPU Architecture and Programming Models

Graphics Processing Units' programming interfaces and architectures form a significant foundation for describing their capability and application in modern computing architectures. GPUs are no longer composed of custom-designed hardware produced solely for graphical rendering but are also composed of adaptive parallel processors with multiple computational loads that they can address. This development has been preceded and accompanied by deep architectural innovations and the development of sophisticated programming models that expose the parallel processing capabilities of such devices. The intricate co-dependence of GPU hardware architecture and software environments determines not only performance characteristics but also the convenience and effectiveness with which programmers can access GPU computing power. Understandings of these elementary components provide crucial context for why GPUs have become the center of attention for accelerating computation in areas ranging from artificial intelligence to scientific simulations and beyond.

2.1 Evolution of GPU Architecture

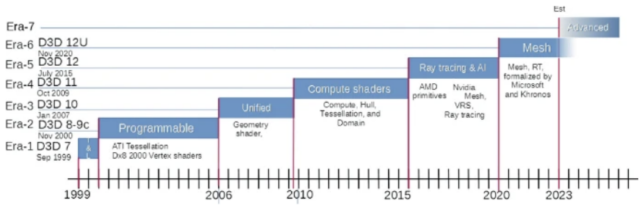

Fig. 1. The History of the GPU by Jon Peddie, via SpringerLink. (https://link-springer-com.calpoly.idm.oclc.org/chapter/10.1007/978-90-481-9971-6_14).

Six Phases of GPU Development:

- First-Generation GPUs (1999–2000): Fixed Function – GPUs during this generation were largely designed for accelerating fixed graphics workloads. Hardware implementation of TL processing, as first introduced by Nvidia's GeForce 256, was the start of offloading of complex rendering from the CPU. However, these GPUs were non-programmable and were limited to specific rendering tasks [Peddie 2023].

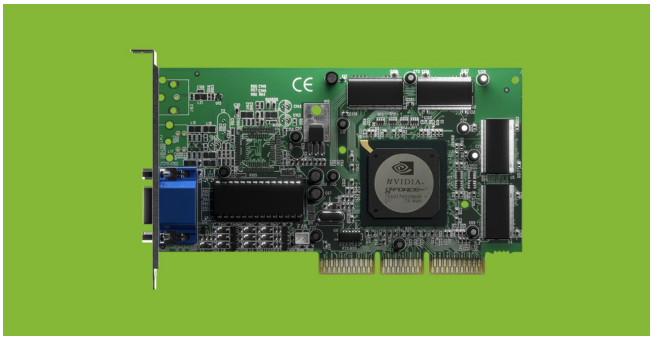

Fig. 2. Nvidia's GeForce 256, via Nvidia. (https://blogs.nvidia.com/wp-content/uploads/2024/10/geforce-256-nvidia-blog-image-1280x660-2-2.jpp).

- Second-Generation GPUs (2000–2006): Programmable Shaders – The arrival of DirectX 8 and OpenGL paved the path for the transition of GPUs from fixed-function pipelines to programmable shaders, where programmers could program proprietary lighting and texturing effects. The achievement helped in achieving more realistic rendering of graphics and opened the doors for general-purpose computing on GPUs (GPGPU), making the foundation for future achievements [Peddie 2023].
- Third-Generation GPUs (2006–2009): Unified Shaders – Unified shader architectures enabled GPUs to dynamically allocate resources across diverse shading tasks (vertex, pixel, and geometry) dynamically. This shift improved parallelism and efficiency significantly, significantly improving both gaming graphics and computational workloads aside from rendering, and making GPUs more general-purpose friendly [Peddie 2023].
- Fourth-Generation GPUs (2009–2015): Compute Shaders and GPGPU – The introduction of compute shaders marked the official entry of GPUs into high-performance computing. CUDA (2007) and OpenCL (2008) introduced the possibility of using GPUs as heavyweight tools for parallel processing, accelerating workloads for scientific simulation, cryptography, and financial modeling. In this period, GPUs evolved from being graphic-only hardware to being central components of large-scale computational infrastructure [Peddie 2023].
- Ray Tracing and AI Boost – The introduction of real-time ray tracing for games (using NVIDIA RTX technology) revolutionized rendering by simulating the behavior of light to unprecedented levels of accuracy. Meanwhile, AI acceleration also saw growth, with GPUs including specialized Tensor Cores to enable deep learning and training neural networks. This generation solidified GPUs as a pillar of AI research and machine learning applications [Peddie 2023].

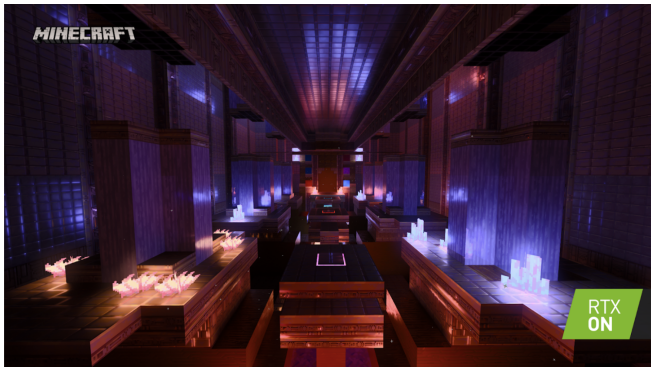

Fig. 3. RTX Ray Tracing being used for Minecraft Shaders, via Nvidia. (https://images.nvidia.com/aem-dam/Solutions/geforce/campaigns/minecraft-with-rtx/phase2/Portal_Pioneers_NEW_Crystal_room_4_ON_with_Logo.png).

- Sixth-Generation GPUs (2020–Present): AI-Boosted Compute and Mesh Shaders – Modern GPUs contain built-in AI accelerators, such as NVIDIA Tensor Cores and AMD Matrix Cores, which optimize deep learning computations. In addition, with the inclusion of Mesh Shaders, geometry processing is more optimized, and complex rendering operations are easier to implement. This period has also seen the integration of multi-GPU computing, where GPUs work together to solve exascale computing issues in cloud AI, supercomputing, and real-time simulation [Peddie 2023].

Components of Modern GPUs

Tensor Processors – AI workload acceleration

Tensor processors are hardware accelerators specifically designed to enhance AI and machine learning workloads, particularly deep learning workloads. The processors employ matrix operations that are the foundation of neural network computation. Thanks to the GPU's single instruction, multiple data (SIMD) architecture, tensor processors are capable of optimizing computation performance

Shader		Thread Mapping	Topology
Vertex Shader	No access to connectivity	1 Vertex	No influence
Geometry Shader	Variable output doesn't fit HW well	1 Primitive / 1 Output Strip	Triangle Strips
Tessellation Shader	Fixed-function topology	1 Patch / 1 Evaluated Vertex	Fast Patterns
Mesh Shader	Compute shader features	Flexible	Flexible within work group allocation

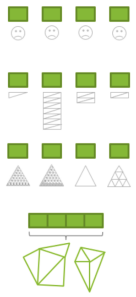

Fig. 4. Mesh shaders represent the next step in handling geometric complexity, via Nvidia. (https://developer.nvidia.com/blog/introduction-turing-mesh-shaders/).

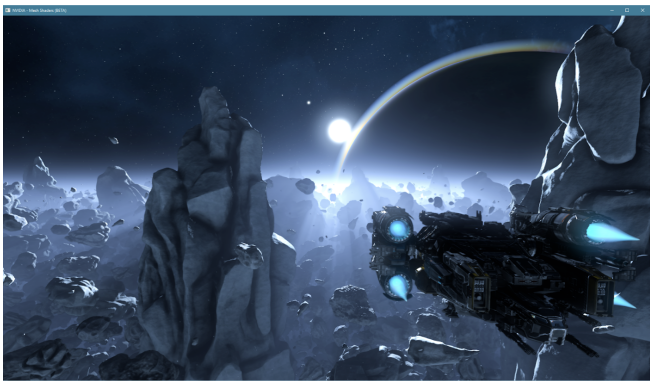

Fig. 5. NVIDIA Asteroids demo uses mesh shading. (https://developer.nvidia.com/blog/introduction-turing-mesh-shaders/).

greatly by performing extensive parallel computations on big data sets and are essential for applications such as image recognition, recommendation systems, and scientific simulation. Tensor processors have evolved to encompass dedicated deep learning cores, such as NVIDIA's Tensor Cores, which accelerate mixed-precision matrix multiplications required for training and inference in AI models [Peddie 2023].

Memory Managers – Efficient data transfer between GPU cores and memory

Efficient memory management takes center stage in maximizing the performance of GPUs, as modern GPUs execute on the basis of high-bandwidth memory architectures that are designed for parallel computation. Memory managers guarantee efficient data transport between memory and GPU cores, reducing latency and avoiding bottlenecks that could occur because of inefficient data access patterns. These managers can utilize dedicated architectures such as unified memory, high-speed caches, and memory compression algorithms to optimize the use of GPU resources. Additionally, memory management innovation such as integrating AI-driven prediction mechanisms boosts performance by prefetching data that will be employed for upcoming computations [Peddie 2023].

Streaming Multiprocessors (SMs) / Compute Units (CUs) – Primitive computing units for parallel execution

NVIDIA GPU Streaming Multiprocessors (SMs) and AMD GPU Compute Units (CUs) are the fundamental units of parallel execution. They consist of multiple arithmetic logic units (ALUs) or floating-point units (FPUs), which execute instructions on numerous threads simultaneously. Modern SMs and CUs hold a variety of shaders, including vertex, geometry, and compute shaders, to enable seamless workload partitioning. The multiprocessor-cluster architecture allows enormous scalability, which allows GPUs to accelerate non-graphic workloads such as AI training, physics simulations, and scientific computations [Peddie 2023].

Ray Tracing Cores – Dedicated real-time ray tracing hardware

Ray tracing cores are dedicated hardware blocks optimized to accelerate real-time ray tracing, a rendering technique that simulates how light acts when it encounters surfaces in order to produce highly realistic images. These cores perform elaborate calculations with regard to ray intersection, reflections, and refraction, lowering significantly the computation load for the general-purpose GPU cores. Assisted by AI technologies such as NVIDIA's Deep Learning Super Sampling (DLSS), ray tracing cores achieve improved performance at high image quality. The advent of ray tracing hardware in particular has revolutionized game design, movie visual effects, and real-time simulation, and with an unparalleled sense of realism [Peddie 2023].

2.2 Comparative GPU Architectures

The article A Comparative Study of GPU Programming Models and Architectures Using Neural Networks presents a comprehensive comparison between the Nvidia Fermi and AMD/Ati Radeon architectures and the CUDA and OpenCL programming models [Pallipuram et al. 2011]. This is relevant because it demonstrates how design decisions in architecture affect computation workloads, specifically on high-workload applications such as AI and neural networks. Understanding their strengths and weaknesses, we can better comprehend recent computing and optimisation strategies on GPUs. The contrast between AMD/ATi Radeon and NVIDIA Fermi architectures says a lot about the influence of varied design philosophies on computational efficiency. Understanding the differences is understanding how and why we ended up with the sophisticated AI and HPC workloads we enjoy today. Looking back at these formative architectures, we are best suited to put into perspective the innovations that afforded us productive AI and HPC workloads of today.

Architectural Differences and Performance Implications

NVIDIA Fermi is deterministic in performance because of its scalar processing paradigm [Pallipuram et al. 2011]. AMD Radeon's VLIW architecture provides greater theoretical performance but at the cost of engaging time-consuming instruction scheduling [Pallipuram et al. 2011]. Memory performance: Fermi cache hierarchy is optimized and minimizes latency, whereas Radeon suffers from

write unit stalls [Pallipuram et al. 2011]. Neural Network Performance: Fermi provides a 1095× speed-up for Hodgkin-Huxley models, which is better than Radeon's 588× speed-up [Pallipuram et al. 2011].

AMD/ATi Radeon and NVIDIA Fermi Architectures:

Feature	NVIDIA Fermi (Tesla C2050)	AMD Radeon (5870)
Core Organization	448 CUDA cores in 14 SMs	1600 ALUs in 320 stream cores
Memory Architecture	L1/L2 cache, 64KB shared memory per SM	8 KB L1 per compute unit, 512 KB L2
Processing Model	Scalar processor architecture	VLIW (5-way ALU)
Compute Throughput	1.105 TFlops/s	2.72 TFlops/s
Memory Bandwidth	144 GB/s	155 GB/s
Error Correction (ECC)	Supported	Not available

2.3 GPU Programming Models

The contrast between OpenCL and CUDA offers instructive lessons regarding how rival programming models take advantage of the capabilities of GPUs for various workloads. Being an NVIDIA product, CUDA is NVIDIA GPU-specific and gives high-granularity control of memory and execution parameters. Being an open standard backed by various vendors, OpenCL thus has broader compatibility but might need finer tuning to achieve peak performance. These programming models must be familiar with choosing the most appropriate framework for certain high-performance computing tasks.

CUDA vs. OpenCL: A Comparative Overview

NVIDIA CUDA specifies GPU execution in C-based kernels and groups the threads into thread blocks to be run on Streaming Multiprocessors (SMPs) [Pallipuram et al. 2011]. CUDA provides fine-grained memory control through registers, thread block shared memory, and global memory for each thread [Pallipuram et al. 2011]. OpenCL is an open heterogeneous computing platform for GPUs, CPUs, and FPGAs [Pallipuram et al. 2011]. It takes advantage of work-items and work-groups instead of CUDA's threads and blocks, and it is cross-platform but generally less performance-optimized [Pallipuram et al. 2011].

Optimization Methods for CUDA and OpenCL:

Memory Optimization: CUDA leverages shared memory and memory coalescing, while OpenCL leverages explicit memory management and prefetching to improve memory access [Pallipuram et al. 2011]. Execution Optimization: The CUDA execution model is optimized for NVIDIA GPUs to schedule threads more efficiently, whereas OpenCL is manually optimized to achieve optimal performance on various architectures [Pallipuram et al. 2011]. Instruction Optimization: Both models use loop unrolling and minimizing branch divergence techniques for computational optimization [Pallipuram et al. 2011].

Performance Comparison: CUDA vs. OpenCL: CUDA is also faster than OpenCL on NVIDIA hardware due to hardware-specific optimizations being better in CUDA. OpenCL is more portable and needs more optimization to run best on various architectures [Pallipuram et al. 2011]. CUDA achieves 976.2× speed-up in Hodgkin-Huxley models whereas OpenCL achieves 878.4× speed-up, showing CUDA performance on NVIDIA hardware. On Fermi GPUs with optimized memory management. For Hodgkin-Huxley models, CUDA achieves 976.2× speed-up and OpenCL achieves 878.4× speed-up [Pallipuram et al. 2011].

3 GPUs in Artificial Intelligence and Machine Learning

As advancements in artificial intelligence and machine learning accelerate, traditional graphics processing units (GPUs) may struggle to keep pace with increasing computational demands. Now, GPUs are being complemented by specialized accelerators like intelligence processing units (IPUs) and reconfigurable dataflow units (RDUs). While GPUs remain the most widely used for AI workloads, IPUs offer better efficiency for tasks involving fine-grained parallelism, while RDUs are optimized for dataflow processing. Understanding the trade-offs between each accelerator helps choose the right hardware for AI applications.

3.1 Transformative Role of GPUs in AI

Parallel Computing Capabilities for Matrix Operations

The emergence of Graphics Processing Units (GPUs) has played a key role in the evolution of artificial intelligence (AI), particularly in machine learning (ML) and deep learning (DL), where matrix operations play a central role. Unlike Central Processing Units (CPUs), which process sequentially, GPUs capitalize on parallelism to execute thousands of calculations simultaneously. This makes them very efficient in handling complex mathematical operations such as matrix multiplication and vector transformation which are fundamental neural network training, where high volumes and billions of parameters require parallel processing [Youvan 2023].

One major milestone in the development of GPU-accelerated AI was the launch of NVIDIA's Compute Unified Device Architecture (CUDA), which gave scientists direct access to the cores of GPUs to carry out general computing. CUDA made it possible to tap into the parallelism in GPUs to their full extent, significantly reducing the time required to train large AI systems. In addition with the help of deep learning frameworks like TensorFlow and PyTorch, scientists have been able to design more complex AI systems, expanding the boundaries of computational feasibility.

Performance Improvements Over Traditional CPU Processing

GPUs outperform traditional CPUs in AI tasks because they execute a high volume of parallel operations efficiently. CPUs contain a small number of high-power cores that are serially optimized, whereas modern GPUs consist of thousands of smaller cores to run large workloads in parallel. This architectural difference allows GPUs to achieve dramatic speedups in deep learning model training—usually 50 times quicker than CPU-based calculations [Youvan 2023].

The second key advantage of GPUs in AI is their increased memory bandwidth. AI applications tend to load and process massive data sets, and GPUs, with their high-bandwidth memory (HBM) and optimized access to memory, provide faster data transfer and computation. This reduces bottlenecks typically faced by CPU-based processing and allows AI models to scale efficiently to more complex architectures [Youvan 2023].

Furthermore, GPUs enable large-scale AI applications, from computer vision to natural language processing (NLP), by providing scalable computational resources that are not possible with traditional CPU-based systems. As deep learning models become increasingly complex, innovations in GPUs, such as tensor cores and mixed-precision computing, are optimizing AI training [Youvan 2023].

3.2 GPU Frameworks for AI/ML

NVIDIA's CUDA Ecosystem

One of the major drivers behind the usage of GPUs in AI and ML is the CUDA ecosystem by NVIDIA. CUDA is a parallel architecture and programming model that allows developers to access the processing powers of GPUs to make them efficiently work on AI workloads. CUDA includes low-level optimizations to allow deep learning models to perform high-rate matrix operations and take advantage of data parallelism. CUDA helped AI scientists to overcome the conventional CPU bottlenecks, making GPUs the hardware choice to train and infer AI [Peng et al. 2024].

One of the most important aspects of CUDA's contribution to deep learning is cuDNN (CUDA Deep Neural Network Library), which provides implementations of critical neural network operations such as convolutions, pooling, and activations, accelerated by the GPU. By accelerating these operations on NVIDIA GPUs, cuDNN significantly improves deep learning performance, enabling more rapid training and more accurate inference [Peng et al. 2024].

Integration with TensorFlow and PyTorch

The leading deep learning frameworks, TensorFlow and PyTorch, are completely optimized to be accelerated by CUDA and cuDNN. These frameworks have revolutionized AI development by simplifying the process of incorporating GPUs and allowing developers to focus on the architecture of the model rather than working with low-level hardware details [Peng et al. 2024].

TensorFlow has native GPU support through CUDA so that AI models can be trained in parallel across many GPUs. This significantly accelerates training operations so that deep learning experiments can be tested at a large scale.

PyTorch, known for its dynamic computation graph, integrates seamlessly with CUDA, allowing AI models to leverage automatic GPU utilization. PyTorch's dynamic computation feature makes it extremely well-suited for research and prototyping, where flexibility is vital [Peng et al. 2024]. These integrations have driven major advancements in AI research, making it possible to train and deploy large-scale AI models with unprecedented speed and accuracy.

3.3 GEmerging AI/ML Accelerators

Comparison of GPUs with IPUs and RDUs

The recent expansion in artificial intelligence (AI) and machine learning (ML) has given rise to a requirement for purpose-specific accelerators apart from conventional GPUs. Two instances are the Graphcore Intelligence Processing Unit (IPU) and the Sambanova Reconfigurable Dataflow Unit (RDU), both designed to overcome the restrictions imposed by conventional von Neumann architectures [Peng et al. 2024].

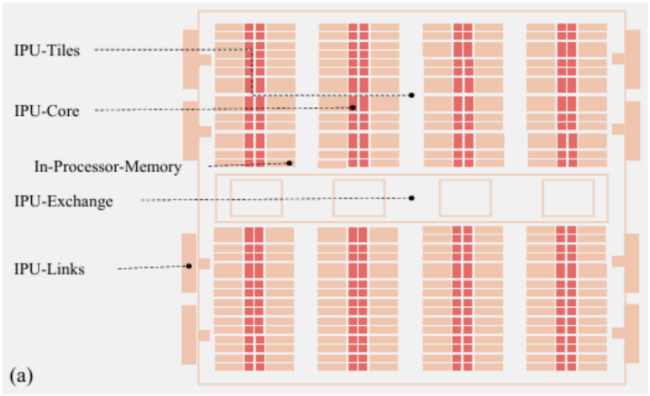

Fig. 6. Graphcore IPU architecture. arXiv, via Cornell University. (https://arxiv.org/abs/2311.04417?utm_source=chatgpt.com).

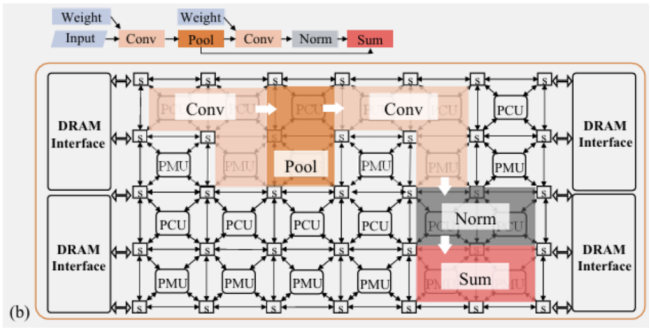

Fig. 7. Sambanova RDU architecture. arXiv, via Cornell University. (https://arxiv.org/abs/2311.04417?utm_source=chatgpt.com).

GPUs have been dominant in accelerating AI due to their Single Instruction Multiple Thread (SIMT) architecture, where thousands of threads can be executed in parallel. They still suffer from bottlenecks in non-uniform data access workloads, such as Graph Neural Networks (GNNs) [Peng et al. 2024].

On the other hand, the Graphcore IPU employs a Multiple Instruction Multiple Data (MIMD) strategy, with 1,472 independent tiles on each chip, each of which can execute distinct instructions. This enables the IPU to achieve fine-grained parallelism and be more effective for sparse calculations [Peng et al. 2024].

The Sambanova RDU, on the other hand, is based on a Coarse-Grained Reconfigurable Array (CGRA) architecture, and it focuses on dataflow execution. This means that instead of being instruction-based computations, RDUs are based on data movement, and they

are therefore highly effective with transformer-based models such as BERT and GPT [Peng et al. 2024].

Understanding TFLOPs and AI Accelerator Performance

A key performance metric when comparing AI accelerators is TFLOPs (Tera Floating-Point Operations Per Second). One TFLOP is one trillion (10^{12}) floating-point operations per second, and it is a standard unit to measure computational performance in AI and deep learning applications. The higher the TFLOPs, the more calculations an accelerator can perform per second, and the more it will influence the training and inference speeds of AI models [Peng et al. 2024].

Although, TFLOPs is an important metric of raw computing ability, AI performance is also determined by other factors, including architecture design, memory bandwidth, and software optimizations.

Specialized vs. General-Purpose Architectures

Key distinctions between GPUs, IPUs, and RDUs is their architectural focus:

- GPUs: General-purpose accelerators that are optimized for matrix-intensive calculations, taking advantage of their high memory bandwidth (1.6 TB/s in NVIDIA A100) and Tensor Cores for deep learning.
- IPUs: Prefer task-level parallelism, with the capability to run multiple independent streams of instructions. The Graphcore GC200 IPU achieves 62.5 TFLOPS (FP32) and 250 TFLOPS (FP16) and is particularly optimized for sparse workloads [Peng et al. 2024].
- RDUs: Designed to run dataflow in a way that is customized to reduce instruction latency and improve inference throughput. The Sambanova SN10 RDU delivers 325 TFLOPs (FP16) but is only specialized for AI workloads, unlike GPUs [Peng et al. 2024].

These distinctions highlight that while GPUs remain the most general-purpose, IPUs and RDUs are more scalable and efficient in specific AI workloads, such as graph processing and NLP inference [Peng et al. 2024].

Performance Characteristics for Different AI Workloads

General Matrix Multiplication (GEMM) Performance

- NVIDIA A100 GPU provides 312 TFLOPs (FP16) and 19.5 TFLOPs (FP32), making it the fastest for dense matrix operations.
- Graphcore GC200 IPU delivers 250 TFLOPs (FP16) and 62.5 TFLOPs (FP32), offering comparable performance but specifically optimized for sparse matrix operations.
- Sambanova SN10 RDU reaches 325 TFLOPs (FP16) but lacks higher precision (FP32) support, which limits its versatility [Peng et al. 2024].

Transformer-Based Model Performance (BERT, GPT)

- Sambanova RDU significantly outperforms IPUs and GPUs in transformer-based models due to its dataflow execution model.

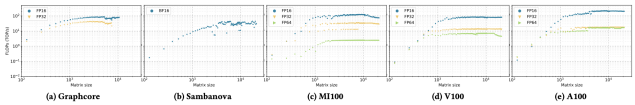

Fig. 8. Cross-platform evaluation on square GEMM operators. arXiv, via Cornell University. (https://arxiv.org/abs/2311.04417?utm_source=chatgpt.com).

- GPUs, such as the NVIDIA A100, remain the best option for training large models, but RDUs provide a 4× inference speedup for BERT [Peng et al. 2024].
- Graphcore IPUs, while efficient for certain NLP tasks, do not match RDUs' performance in sequential transformer-based workloads [Peng et al. 2024].

Graph Neural Networks (GNNs) Performance

- Graphcore IPU is the top performer for GNN tasks, achieving 2×–4× higher throughput than GPUs, thanks to independent tile execution that reduces memory bottlenecks [Peng et al. 2024].
- GPUs (NVIDIA A100 and V100) struggle with sparse and irregular workloads, performing 30–40% worse than IPUs in GNN training [Peng et al. 2024].
- RDUs are not well-suited for GNNs, as their dataflow architecture is optimized for sequential models like transformers [Peng et al. 2024].

2D Convolutional Neural Network (CNN) Performance

- Graphcore GC200 IPU delivers 8.18× higher FP16 convolution throughput than NVIDIA V100, making it the most efficient CNN accelerator [Peng et al. 2024].
- NVIDIA A100 remains the leading choice for large-scale CNN model training due to its high memory bandwidth and Tensor Cores.
- Sambanova RDU underperforms in convolution-based tasks as it is not optimized for CNN workloads [Peng et al. 2024].

Sparse Matrix Multiplication (SPMM) Performance (Used in AI and Scientific Computing)

- Graphcore IPU excels in sparse matrix workloads, achieving 1.5× speedup over GPUs.
- NVIDIA A100 GPU, with PyTorch CSR format, reaches 12.24× higher throughput than V100, making it the best choice for dense SPMM tasks [Peng et al. 2024].
- Sambanova SN10 RDU lacks compiler support for SPMM, making it unsuitable for such workloads [Peng et al. 2024].

4 Large-Scale GPU Computing Systems

The MIT Supercloud is a high-performance computing (HPC) system that is meant to optimize AI and machine learning-related workloads by utilizing 448 NVIDIA Volta V100 GPUs across 224 nodes. The system is meant to be used by AI scientists, government research centers, and scientific fields that require high parallel processing.

Unlike conventional HPC clusters, the Supercloud is optimized to support interactive AI development, enabling users to prototype, train, and deploy deep learning models with efficiency.

4.1 High-Performance Computing with GPUs

Supercloud's use of GPUs reveals significant inefficiencies and possible optimization targets:

GPU Usage Trends:

- GPU's have an average job time of 30 minutes, though durations can vary between less than one minute and more than 20 hours.
- 70% of jobs on the GPU have a queue time of less than one minute, demonstrating the efficiency with which Supercloud schedules AI workloads.
- The majority of jobs have low resource utilization, with a median streaming multiprocessor (SM) utilization of only 16% and memory bandwidth utilization of just 2% [Li et al. 2022].

Idle vs. Active Phases:

- Supercloud workloads follow unpredictable usage patterns, shifting between active and idle periods at random. On average, a typical job runs actively 84% of the time, but some jobs use GPUs for as little as 14% of their runtime. While most jobs maintain low average utilization, 22% spike to 100% SM utilization at some point—making peak demand a critical factor in scheduling [Li et al. 2022].

Energy Efficiency and Power Limitations:

- The average power consumption per job is 45W, with the highest recorded usage at only 87W—well below the V100 GPU's 300W TDP [Li et al. 2022].
- More than 60% of jobs would not be affected by a 150W power cap, suggesting that power limiting could increase job throughput without negatively impacting performance [Li et al. 2022].

These findings highlight potential optimization targets for dynamic GPU scheduling, resource-aware job placement, and power-efficient capping to significantly improve GPU utilization in HPC environments.

4.2 Challenges in GPU Resource Management

Inefficiencies in Current GPU Utilization

One of the most significant issues in modern GPU-based HPC systems is the inefficient utilization of GPU resources. The MIT Supercloud report reveals that approximately 60% of total GPU hours are consumed by immature workloads—i.e., exploratory, developmental, and IDE jobs-rather than production-quality applications [Li et al. 2022]. This is a tremendous resource commitment in early phases of the AI workflow that underutilized the expensive GPU hardware.

The detailed analysis of GPU utilization metrics shows that most work streams at surprisingly suboptimal rates of efficiency. The median streaming multiprocessor (SM) utilization was found to be a mere 16%, memory bandwidth use at a laughable 2%, and memory size utilization at a nominal 9% [Li et al. 2022]. Moreover, work uses in excess of 50% of the available SM resources for merely 20%, suggesting meaningful wasted compute capability in production deployments.

Most revealing perhaps is the categorization of tasks by the stage of development. While mature tasks constitute roughly 60% of all tasks on the system, they consume a mere 39% of all GPU hours. Conversely, development tasks targeted for hyperparameter tuning constitute 18% of tasks but consume 34% of GPU hours [Li et al. 2022]. Such significant investment of resources in development stages is a paradigm shift in HPC resource usage patterns compared to conventional scientific computing workloads.

The research also discovers significant temporal variation in resource utilization. GPU tasks exhibit irregular activity patterns, alternating between active and idle states, with the median task utilizing GPU resources for only 84% of its duration [Li et al. 2022]. Such temporal inconsistency offers yet another inefficiency in resource scheduling and allocation.

Node Specifications			
Number of Nodes	224	Node RAM	384 GB
Number of CPU Cores	8960 cores (two CPUs per node, each with 20 cores; 2-way hyperthreading per-core)	Processor	Intel Xeon Gold 6248
Interconnect	100 Gb/s Omnipath two-layer partial fat-tree	Network	25 Gb/s Ethernet CX-4
GPU Specifications			
Number of GPUs	448	Type	Nvidia Volta V100
GPUs per Node	2	RAM	32 GB
Storage Specifications			
Local	1 TB SSD & 3.8 TB HDD	Shared	873 TB SSD

Fig. 9. Specifications of the Supercloud system. MIT Supercloud, via IEEE Explore. (https://ieeexplore-ieee-org.calpoly.idm.oclc.org/document/9773216).

Job Scheduling Challenges

GPU job scheduling poses several unique challenges in modern AI-focused HPC environments. Older HPC job scheduling methods, which were written for long-running, CPU-intensive scientific calculations, are not adapted to the requirements of deep learning workloads.

First, the high degree of variation in job properties, both between users and across a user's submissions, complicates scheduling. An MIT Supercloud study found that an average user submits jobs with highly diverse runtime and resource usage properties, with the job runtime coefficient of variation often exceeding 155% [Li et al. 2022]. The variability complicates forecasting resource usage and making scheduling choices.

Second, multi-GPU tasks introduce additional complexity into the scheduling. While 16% of Supercloud tasks utilized more than one GPU, these tasks alone contributed about 50% of total GPU hours [Li et al. 2022]. The complexity is further increased by the finding that about 40% of multi-GPU tasks utilize at least one or more idle GPUs, i.e., inefficient resource utilization within even explicitly demanding multi-GPU tasks.

Finally, the increasing dominance of exploratory and development workloads requires scheduling systems to be able to manage different job priorities and quality-of-service requirements. The Supercloud study shows that different types of jobs (mature, exploratory, development, and IDE) have extremely different patterns of resource utilization and runtime behavior [Li et al. 2022]. Current scheduling techniques lack the complexity to differentiate between these types and allocate resources accordingly.

4.3 Future Directions for GPU HPC

Dynamic Resource-Sharing Techniques

GPU usage patterns in HPC environments indicate the great potential of dynamic resource-sharing techniques. The Supercloud study highlights that most GPU-accelerated workloads exhibit low utilization of various resources (SM, memory, PCIe bandwidth), with the utilization of resources varying extensively during job execution, oscillating between idle and active periods [Li et al. 2022]. Such a pattern offers an opportunity for sophisticated resource-sharing techniques.

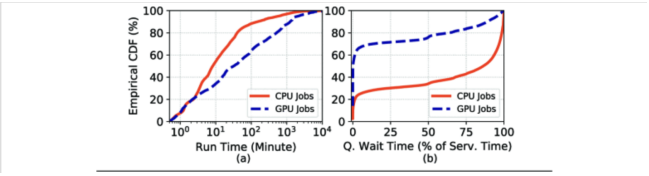

Fig. 10. MIT Supercloud, via IEEE Explore. (https://ieeexplore-ieee-org.calpoly.idm.oclc.org/document/9773216).

Future HPC GPU systems will likely use more advanced space-sharing mechanisms that allow multiple jobs to simultaneously share non-contentious GPU resources. The reality that different jobs use each type of resource in different ways suggests that workloads with complementary attributes can be co-located at low performance cost [Li et al. 2022]. For instance, compute-bound jobs can be co-located with memory-bound jobs and can share GPU resources to attain higher overall hardware usage.

Time-sharing techniques also have potential, particularly in light of the observation that GPUs are idle for significant portions of job runtimes, with such idleness occurring at unpredictable times [Li et al. 2022]. Preemptive scheduling might be employed in future systems to reclaim these idle resources, which would have significant efficiency benefits. However, as the authors note, co-location interference remains difficult to predict and solve, indicating more work must be done on online architectural tools for predicting idle times and resource contention patterns.

Multi-Tier GPU Setups

The Supercloud experiment makes a compelling case for multi-level GPU infrastructure that provides GPUs of varying capabilities and costs to satisfy workload demand. As most GPU hours are consumed by exploratory, development, and IDE jobs with obviously lower resource utilization, there is certainly scope to reallocate these workloads to lower-capacity or lower-cost GPUs [Li et al. 2022].

Future GPU HPC systems may employ tiered hardware architectures where high-end, high-performance GPUs are reserved for production workloads that can fully leverage their capabilities, and lower-end GPUs are employed for development and exploratory workloads. This would significantly improve cost-effectiveness without compromising performance for production workloads.

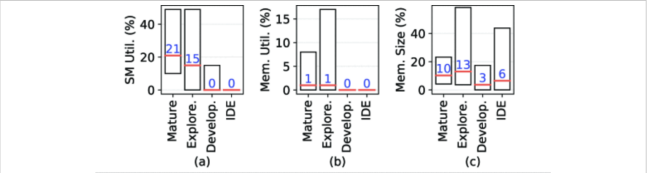

Fig. 11. Supercloud System, via IEEE Explore. (https://ieeexplore-ieee-org.calpoly.idm.oclc.org/document/9773216).

The work also identifies opportunities for power management strategies in systems in the future. With the discovery that the median average GPU workload power draw is only 45W (compared with the 300W peak V100 GPU usage utilized in this study), a great opportunity lies with power-capping and over-provisioning strategies [Li et al. 2022]. Power allocation dynamically can be utilized in future systems for re-assignment of power between idle GPUs to those handling priority or high-workload tasks.

Advanced Scheduling Strategies

Future GPU HPC systems for the next generation will also require smarter scheduling policies that are cognizant of the unique character of AI and machine learning workloads. Research on the Supercloud shows that users exhibit various job demands and expectations, meaning that schedulers should identify and serve these varying requirements [Li et al. 2022].

One possible direction is the development of job classification systems that can automatically categorize workloads based on their development stage and the resources they require. The new job classification of the Supercloud project into mature, exploratory, development, and IDE jobs provides a foundation for such directions [Li et al. 2022]. Future schedulers can then use different policies for each class, optimizing resource allocation respectively.

Also, the finding that even work submitted by the same user has extremely varied characteristics breaks the common HPC practice of user-consistent workloads [Li et al. 2022]. This would suggest that next-generation scheduling systems will need to move beyond predictive resource management approaches based on user-specific policies and embrace more dynamic, workload-oriented approaches.

Finally, the study foresees potential for development and exploration workload-specific checkpoint/restart mechanisms. As these types of jobs normally run to failure or timeout, low-overhead checkpointing mechanisms that store state and enable resumption efficiently are becoming ever more important [Li et al. 2022]. Next-generation systems can incorporate architectural and system-level support for these kinds of mechanisms, perhaps combined with low-latency persistent storage options to minimize job interruption overhead.

5 Specialized Applications of GPU Parallel Computing

GPU parallel computing has also extended beyond general-purpose applications to specialized applications with unique computational requirements. This section discusses three distinct specialized applications that reflect the versatility and revolutionary effect of GPU acceleration for challenging computational applications.

5.1 Physics and Multibody Simulations

Macroscopic simulation of multibody dynamics, particularly with granular materials, is a primary computational challenge in the past and was traditionally tackled by continuum approximations, not discrete elements. Tasora et al. present a revolutionary approach taking advantage of parallel computation on a GPU to offer the capability to simulate hundreds of millions of rigid bodies interacting with frictional contact [Tasora et al. 1970].

The authors then go on to identify three significant roadblocks that have limited sequential computation in scientific applications: the memory block (imbalance between CPU processing power and memory access rate), the instruction level parallelism block (limitations of speculative execution), and the power dissipation block (thermal limitations limiting clock frequency increases). These roadblocks have forced the transition to parallel computing architectures, particularly GPUs, capable of providing much higher computational throughputs for appropriate problems.

Their approach is founded on the derivation of multibody dynamics as a differential variational inequality (DVI) that can simulate advanced interactions between rigid bodies, including bilateral constraints, frictional contact, and externally applied loads. The formulation accommodates both smooth and non-smooth effects in the form of impacts and jumps in velocity. The authors transform this DVI to a time-stepping formulation with the CCP to be solved at each step, which is now the compute bottleneck as well as collision detection.

More innovative about their approach is reformulating the CCP solver to exploit the parallel nature of GPU architecture. They understand that their solution's iterative algorithm has some components that are easy to parallelize (contact and constraint equations) and other components that are more difficult to parallelize to avoid race conditions (velocity updates). With a parallel segmented scan algorithm and data structures of low memory transfer and high arithmetic intensity, they achieve high-performance execution on NVIDIA's CUDA platform.

Benchmark runs demonstrate the effectiveness of this method. For pebble bed nuclear reactor simulation—a problem with significant practical use in nuclear engineering—the GPU version is linearly scalable with number of bodies but at a much diminished slope compared to the CPU version. At 128,000 particles, it is approximately a 10× speedup with growing difference in performance for greater problem size [Tasora et al. 1970].

The largest test case, with 1.1 million bodies that collide in a three-dimensional box containing spheres of different densities, shows the empirical viability of this approach for intractable problems. The simulation well simulates complex granular physics phenomena, such as the "floating" behavior of lower-density objects—phenomena impossible to simulate faithfully with continuum approaches.

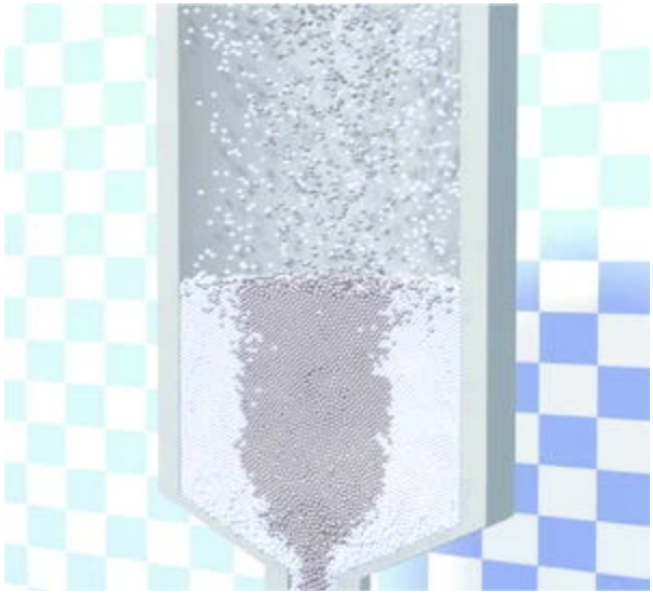

Fig. 12. Pebble bed nuclear reactor simulation, via SpringerLink. (https://link-springer-com.calpoly.idm.oclc.org/chapter/10.1007/978-90-481-9971-6_14).

The applicability spans many other areas. In construction equipment simulation, it enables improved prediction of machine-terrain interaction. In military applications, it enables detailed analysis of off-road mobility on granular terrain like sand. In space exploration, it helps design rovers to move on extraterrestrial ground. Nuclear engineering is assisted by precise modeling of fuel pebble flow dynamics in reactor designs. These share in common the need for discrete rather than continuum techniques in order to include the most important physics.

The authors also outline future research directions, including domain decomposition approaches to overcome memory limitations, algebraic multi-grid methods to improve convergence rates, hardware infrastructure to support even larger simulations, and experimental validation at both macro and micro scales. These developments promise to further expand the applicability of GPU-accelerated multibody dynamics simulation to increasingly complex systems [Tasora et al. 1970].

5.2 Blockchain and Database Operations

The integration of GPU acceleration with blockchain systems represents a significant advancement in addressing performance bottlenecks that traditionally limit transaction throughput in distributed ledger technologies. Iliakis et al. provide a comprehensive analysis of how GPU-accelerated key-value stores can dramatically enhance the performance of blockchain database operations [Iliakis et al. 2022].

At the heart of most blockchain implementations, particularly those that host cryptocurrencies such as Bitcoin, Litecoin, and Ethereum, are key-value databases like LevelDB that store transaction history, block indices, and other metadata. The databases

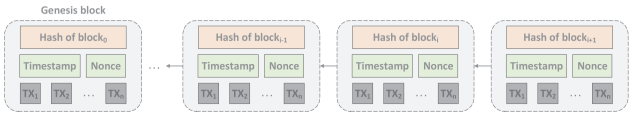

Fig. 13. Example of a blockchain, via The Institution of Engineering and Technology. (https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/blc2.12011?utm_source=chatgpt.com).

become performance bottlenecks in full nodes that authenticate transactions and maintain blockchain consistency. The authors acknowledge that full nodes carry out operations that are primarily database search queries, thereby being ideal for parallel processing acceleration.

The MegaKV system proposed in the paper utilizes parallel computation of the GPU to accelerate key-value store performance in-memory. The system schedules key-value operations onto hundreds of thousands of lightweight GPU threads that can run in parallel, supporting concurrent execution. This fully utilizes the GPU's high memory bandwidth as well as its massive multi-threading feature for memory access latency masking.

Experimental benchmarking demonstrates performance gains of unprecedented order, as operations accelerated by a GPU are returned at two- to three-orders-of-magnitude higher throughput rate than non-accelerated CPU-based strategies. Under practical blockchain loads characterized by varying key-value sizes and read-write patterns of access, MegaKV can process 2-8 billion transactions in the course of 60 seconds in comparison to 20-30 million LevelDB transactions achieved under the same timescale. The tremendous acceleration achieved in processing stands directly in counterpoint to concerns of scalability inherent in blockchain schemes while ensuring integrity of security and decentralization.

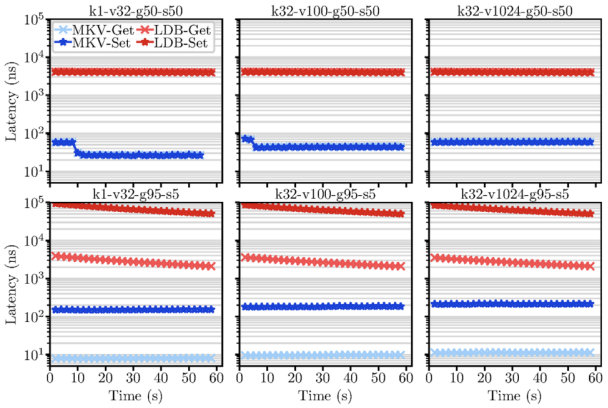

Fig. 14. Query response time in nanoseconds for MegaKV and LevelDB, via The Institution of Engineering and Technology. (https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/blc2.12011?utm_source=chatgpt.com).

Apart from performance optimization, the method of GPU acceleration also demonstrates higher energy efficiency. Although it makes use of additional hardware, the energy required to run a certain number of database queries with the help of GPU acceleration is much lower compared to conventional CPU-based implementations. Such energy efficiency carries significant implications regarding the sustainability of blockchain networks and can reduce operation costs and environmental impact.

The combination of key-value stores empowered by GPUs and blockchain technology represents a promising research direction for raising the throughput, scalability, and energy efficiency of distributed ledger technologies. These developments would play a key role in broadening the adoption and use in real-world applications of blockchain systems in non-cryptocurrency industries [Iliakis et al. 2022].

5.3 Network Traffic Processing

GASPP Framework for Stateful Packet Processing

The GASPP (GPU-Accelerated Stateful Packet Processing) system is designed to leverage current GPUs for the processing of high-speed network traffic. Unlike typical CPU-centric techniques, GASPP transfers packet processing completely to the GPU and unifies operations such as flow monitoring, TCP stream reassembly, stateful traffic classification, and packet encryption. This facilitates more efficiency and scalability with fewer CPU bottlenecks. Another critical innovation in GASPP is its zero-copy capability, which eliminates unnecessary copies of memory between the network interface card (NIC) and the GPU, which significantly maximizes data throughput [Vasiliadis et al. 2014].

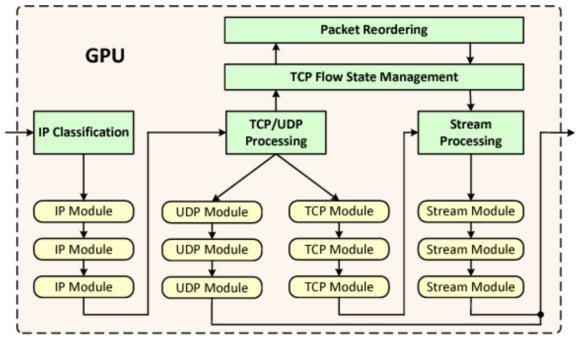

Fig. 15. GPU Packet Processing Pipeline, via Semantic Scholar. (https://www.semanticscholar.org).

Addressing Flow Irregularities Across SIMT Threads

Among the most significant problems of packet processing with GPUs is control flow divergence in Single Instruction, Multiple Threads (SIMT) architectures. Traditional GPU programming is not effective when several flows in the network should take alternative processing paths. GASPP mitigates this issue by scheduling packets dynamically based on their characteristics (e.g., protocol type, packet size, and processing requirement), such that SIMT units execute similar work concurrently. This reduces thread divergence and maximizes parallel execution efficiency [Vasiliadis et al. 2014]. Also, parallelized TCP stream reassembly ensures efficient management of out-of-order packets without too much buffering.

Multi-Gigabit Forwarding Rates for Complex Networking Operations

GASPP can provide multi-gigabit packet-forwarding rates, including during the performance of computationally expensive operations such as intrusion detection, encryption, and deep packet inspection. Using packet transfer batching and GPU memory access pattern optimization, GASPP provides good throughput rates of processing, up to 47.8 Gbit/s for packets of full size. In addition, its modular design enables multiple network applications to operate concurrently, saturating the GPU and making network performance and security monitoring activities convergent on one hardware platform [Vasiliadis et al. 2014].

6 Comparative Analysis and Future Trends

Cross-Domain Performance Comparison

Comparative evaluation of GPU-accelerated software across many fields of activity highlights their breakthrough impact on scientific computing, artificial intelligence, and networking. Sequential CPU-oriented computers strangle on intensive computation, while GPUs employ parallelism to achieve radical accelerations. Pallipuram et al. (2011) demonstrates that CUDA and OpenCL impact processing of neural networks, while GASPP (Vasiliadis et al., 2014) presents instances of stateful packet inspection on a GPU. Similarly, blockchain (Iliakis et al., 2022) and high-performance computing (Li et al., 2022) use cases reveal the general scalability of GPUs over a broad set of workloads. The point here is that while GPUs provide amazingly fantastic performance, their efficiency is highly workload- and optimization-sensitive. *Common Optimization Techniques Across*

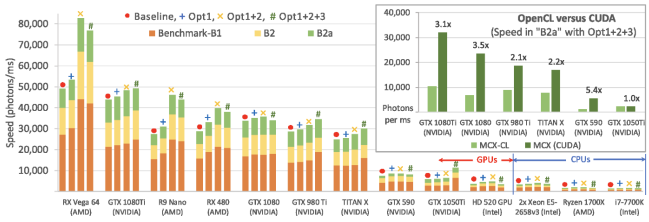

Fig. 16. CUDA vs. OPENCL, via Nvidia. (https://forums.developer.nvidia.com/t/significant-speed-gap-between-cuda-and-opencl-how-to-debug/57530).

Applications

Optimization strategies vary but share common principles across domains. Memory coalescing, kernel fusion, and workload balancing are critical strategies to obtain peak GPU performance. Stream-based execution enables asynchronous task handling in CUDA and OpenCL implementations. Zero-copy memory transfers and SIMT (Single Instruction, Multiple Threads) optimizations are used by GASPP in networking to handle irregular flow processing, whereas AI workloads take advantage of mixed-precision arithmetic and tensor cores for deep learning acceleration. These mechanisms collectively boost throughput, reduce latency, and enhance scalability in GPU-accelerated applications.

Emerging Trends in GPU Architecture and Programming Models

The evolution of GPU architectures continues to push computational boundaries. Emerging hardware accelerators like IPUs (Intelligence Processing Units) and RDUs (Reconfigurable Data Units) provide specialized processing beyond general-purpose GPUs. Peng et al. (2024) discuss the accelerating trend of domain-specific accelerators and their integration with traditional GPU workloads. In addition, advancements in unified memory models and cross-platform frameworks like SYCL are bridging the gap between vendor-specific implementations, providing greater accessibility and portability in GPU computing.

Energy Efficiency Considerations

Energy efficiency remains a critical concern, particularly for large GPU deployments. Youvan (2023) observes that while GPUs deliver high performance, their power consumption necessitates workload scheduling and adaptive power management. Dynamic voltage scaling, memory compression, and multi-GPU workload balancing have been proposed to mitigate power inefficiencies. As AI and real-time processing demands grow, energy-to-performance ratio optimization will be key to enabling sustainable GPU deployment in data centers and edge computing. *Future Research Directions and Chal-*

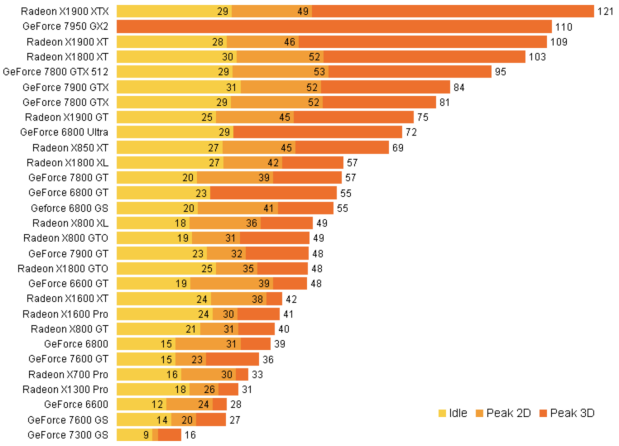

Fig. 17. Graphic Cards by Power Consumption, via MSCodes. (https://ms.codes/blogs/computer-hardware/graphics-card-by-power-consumption).

lenges

Despite significant progress, several challenges remain in GPU-accelerated computing. Key research directions include:

- Hybrid Architectures: Research into CPU-GPU hybrids balancing dynamic workloads between the two processors by utilizing the strengths of both.
- Real-Time Processing: Enhancing the real-time processing capability of GPU systems for time-sensitive workloads such as stock market transactions and autonomous vehicle technologies.
- Security in GPU Computing: Overcoming vulnerability exposures in GPU-powered systems, particularly with the risk of data leakage being executed through multi-tenancy architectures prevalent in clouds.

- Scalability in Distributed GPU Systems: Discussing how to seamlessly integrate GPUs into huge-scale distributed platforms for AI, blockchain, and HPC.

In resolving these, subsequent studies will be able to tap into greater potential from GPU computing and open up even better parallel processing solutions in the fields of scalable efficiency and flexibility.

7 Conclusion

Summary of Key Insights

GPU computing in computational workloads has revolutionized parallel computing, scalability at scale, and acceleration of AI, scientific computing, and networking workloads. Even though GPUs were initially designed to be used for rendering graphics, they have now started accelerating state-of-the-art workloads such as distributed machine learning, real-time packet processing, and HPC. Such key optimizations as zero-copy memory copies, SIMT execution optimizations, and dynamic resource-sharing mechanisms have tuned GPU performance in many areas. The comparison brings to the fore the reality that although GPUs outperform traditional CPUs on most parallelizable workloads, their efficiency relies on workload-dependent optimizations and sound programming models like CUDA and OpenCL.

Implications for Researchers and Practitioners

For researchers, the findings imply the need for advanced scheduling algorithms, CPU-GPU hybrid architectures, and power management mechanisms to mitigate GPU resource underutilization. Practitioners can utilize GPU acceleration to make large-scale applications in AI, blockchain, and real-time data analytics more scalable with performance vs. energy efficiency trade-offs. The result from HPC systems such as MIT Supercloud implies the need for dynamic resource allocation, multi-level GPU infrastructures, and workload-aware job scheduling. Such methods will be invaluable to institutions that will desire to attain optimum computational throughput at minimum operational cost.

The Evolving Face of GPUs in Computational Acceleration

GPUs will continue to improve as well, blending with domain-specific accelerators such as Intelligence Processing Units (IPUs) and Reconfigurable Dataflow Units (RDUs) as computation demands continue to grow.

Future directions are in the areas of domain-specific architecture, heterogeneous computing platforms, and increasingly sophisticated AI-based optimization algorithms for workload scheduling. Energy efficiency remains an issue, but emerging power-capping and load-balancing techniques offer green solutions for large-scale GPU deployments. Having overcome these challenges, the future of computing on GPUs will bring unprecedented capability, and with it, innovation in artificial intelligence, high-performance networking, and distributed computing architecture.

References

Konstantinos Iliakis, Konstantina Koliogeorgi, Antonios Litke, Theodora Varvarigou, and Dimitrios Soudris. 2022. GPU accelerated blockchain over key-value database transactions. *IET Blockchain* 2 (2022), 1–12. doi:10.1049/blc2.12011

Baolin Li, Rohin Arora, Siddharth Samsi, Tirthak Patel, William Arcand, David Bestor, Chansup Byun, Rohan Basu Roy, Bill Bergeron, John Holodnak, Michael Houle, Matthew Hubbell, Michael Jones, Jeremy Kepner, Anna Klein, Peter Michaleas, Joseph McDonald, Lauren Milechin, Julie Mullen, Andrew Prout, Benjamin Price, Albert Reuther, Antonio Rosa, Matthew Weiss, Charles Yee, Daniel Edelman, Allan Vanterpool, Anson Cheng, Vijay Gadepally, and Devesh Tiwari. 2022. AI-Enabling Workloads on Large-Scale GPU-Accelerated System: Characterization, Opportunities, and Implications. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. 1224–1237. doi:10.1109/HPCA53966.2022.00093

Vivek K. Pallipuram, Mohammad Bhuiyan, and Melissa C. Smith. 2011. A comparative study of GPU programming models and architectures using neural networks - The Journal of Supercomputing. https://link.springer.com/article/10.1007/s11227-011-0631-3

Jon Peddie. 2023. What is a GPU? In *The History of the GPU-Steps to Invention*. Springer, 333–345.

Hongwu Peng, Caiwen Ding, Tong Geng, Sutanay Choudhury, Kevin Barker, and Ang Li. 2024. Evaluating Emerging AI/ML Accelerators: IPU, RDU, and NVIDIA/AMD GPUs. arXiv:2311.04417 [cs.AR] https://arxiv.org/abs/2311.04417

Alessandro Tasora, Dan Negrut, and Mihai Anitescu. 1970. GPU-based parallel computing for the simulation of complex multibody systems with unilateral and bilateral constraints: An overview. https://link.springer.com/chapter/10.1007/978-90-481-9971-6_14#citeas

Giorgos Vasiliadis, Lazaros Koromilas, Michalis Polychronakis, and Sotiris Ioannidis. 2014. GASPP: A GPU-accelerated stateful packet processing framework. https://www.usenix.org/conference/atc14/technical-sessions/presentation/vasiliadis

Douglas Youvan. 2023. Parallel Precision: The Role of GPUs in the Acceleration of Artificial Intelligence. doi:10.13140/RG.2.2.21937.76641

Received 24 January 2025; revised 06 March 2025