NVIDIA A100 Tensor Core GPU Architecture

UNPRECEDENTED ACCELERATION AT EVERY SCALE

V1.0

Table of Contents

List of Figures

List of Tables

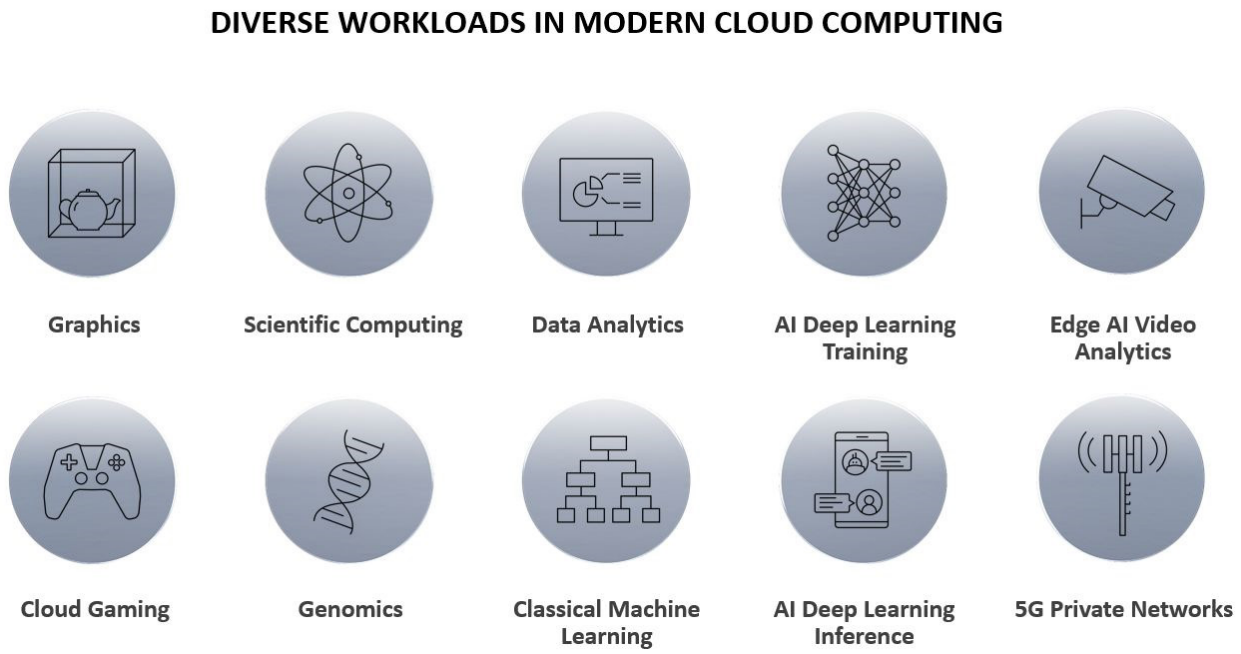
Introduction

The diversity of compute-intensive applications running in modern cloud data centers has driven the explosion of NVIDIA GPU-accelerated cloud computing. Such intensive applications include AI deep learning training and inference, data analytics, scientific computing, genomics, edge video analytics and 5G services, graphics rendering, cloud gaming, and many more. From scaling-up AI training and scientific computing, to scaling-out inference applications, to enabling real-time conversational AI, NVIDIA GPUs provide the necessary horsepower to accelerate numerous complex and unpredictable workloads running in today's cloud data centers.

NVIDIA® GPUs are the leading computational engines powering the AI revolution, providing tremendous speedups for AI training and inference workloads. In addition, NVIDIA GPUs accelerate many types of HPC and data analytics applications and systems, allowing customers to effectively analyze, visualize, and turn data into insights. NVIDIA's accelerated computing platforms are central to many of the world's most important and fastest-growing industries.

HPC has grown beyond supercomputers running computationally-intensive applications such as weather forecasting, oil & gas exploration, and financial modeling. Today, millions of NVIDIA GPUs are accelerating many types of HPC applications running in cloud data centers, servers, systems at the edge, and even deskside workstations, servicing hundreds of industries and scientific domains.

AI networks continue to grow in size, complexity, and diversity, and the usage of AI-based applications and services is rapidly expanding. NVIDIA GPUs accelerate numerous AI systems and applications including: deep learning recommendation systems, autonomous machines (self-driving cars, factory robots, etc.), natural language processing (conversational AI, real-time language translation, etc.), smart city video analytics, software-defined 5G networks (that can deliver AI-based services at the Edge), molecular simulations, drone control, medical image analysis, and more.



Diverse and computationally-intensive workloads in modern cloud data centers require NVIDIA GPU acceleration

Figure 1. Modern cloud datacenter workloads require NVIDIA GPU acceleration

In 2017, the NVIDIA Tesla® V100 GPU introduced powerful new "Tensor Cores" that provided tremendous speedups for the matrix computations at the heart of deep learning neural network training and inferencing operations. In 2018, the NVIDIA Tesla® T4 GPU using NVIDIA Turing™ Tensor Cores and the Tensor RT™ inference optimizer and runtime brought significant speedups to data center inferencing with energy-efficient performance. Turing Tensor Cores also enabled amazing new AI capabilities in Turing GPU-based GeForce® gaming PCs and Quadro® workstations.

On the industry-standard MLPerf AI benchmark, NVIDIA Volta™ GPUs delivered winning results in the training categories, while Turing GPUs won the data center and edge categories in the recently introduced MLPerf inferencing benchmarks. NVIDIA Jetson AGX Xavier™ also delivered the best inferencing performance of all commercially available SoC devices.

For over a decade, the NVIDIA CUDA® development platform has unleashed the power of GPUs to accelerate a wide variety of application areas. Innovations and improvements in APIs, software stacks, libraries, and code optimizers are just as important as advancements in GPU hardware. The NVIDIA CUDA Toolkit, provides numerous software tools for developers, including the NVIDIA CUDA-X™ GPU-accelerated libraries for AI, HPC, and data analytics. Also many containers for AI frameworks and HPC applications, including models and scripts, are available for free in the NVIDIA GPU Cloud™ (NGC) to simplify programming and speed up

development and deployment of GPU-accelerated applications. Kubernetes on NVIDIA GPUs is also available for free to enable enterprises to seamlessly scale up and scale out training and inference deployments across multi-cloud GPU clusters.

Introducing NVIDIA A100 Tensor Core GPU - our 8th Generation Data Center GPU for the Age of Elastic Computing

The new **NVIDIA® A100 Tensor Core GPU** builds upon the capabilities of the prior NVIDIA Tesla V100 GPU, adding many new features while delivering significantly faster performance for HPC, AI, and data analytics workloads. Powered by the NVIDIA Ampere architecture-based GA100 GPU, the A100 provides very strong scaling for GPU compute and deep learning applications running in single- and multi-GPU workstations, servers, clusters, cloud data centers, systems at the edge, and supercomputers. The A100 GPU enables building elastic, versatile, and high throughput data centers.

The A100 GPU includes a revolutionary new **"Multi-Instance GPU" (or MIG)** virtualization and GPU partitioning capability that is particularly beneficial to Cloud Service Providers (CSPs). When configured for MIG operation, the A100 permits CSPs to improve utilization rates of their GPU servers, delivering up to 7x more GPU Instances for no additional cost. Robust fault isolation allows customers to partition a single A100 GPU safely and securely.

A100 adds a powerful new Third-Generation Tensor Core that boosts throughput over V100 while adding comprehensive support for DL and HPC data types, together with a new Sparsity feature to deliver a further doubling of throughput.

New TensorFloat-32 (TF32) Tensor Core operations in A100 provide an easy path to accelerate FP32 input/output data in DL frameworks and HPC, running 10x faster than V100 FP32 FMA operations or 20x faster with sparsity. For FP16/FP32 mixed-precision DL, the A100 Tensor Core delivers 2.5x the performance of V100, increasing to 5x with sparsity.

New Bfloat16 (BF16)/FP32 mixed-precision Tensor Core operations run at the same rate as FP16/FP32 mixed-precision. Tensor Core acceleration of INT8, INT4, and binary round out support for DL inferencing, with A100 sparse INT8 running 20x faster than V100 INT8. For HPC, the A100 Tensor Core includes new IEEE-compliant FP64 processing that delivers 2.5x the FP64 performance of V100.

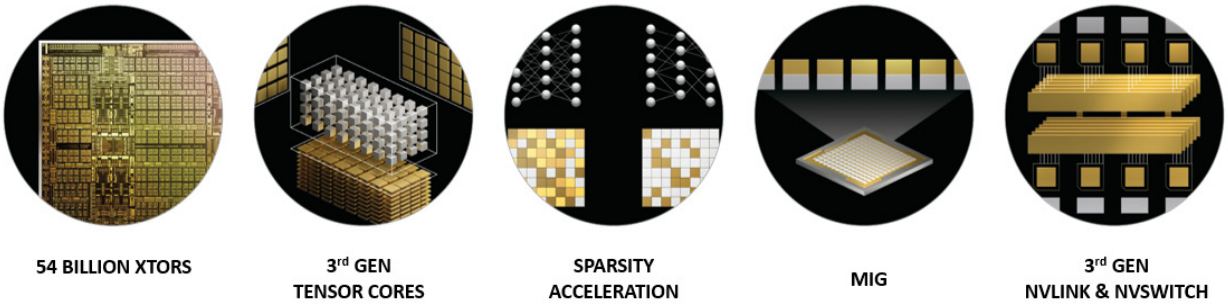


Figure 2. New Technologies in NVIDIA A100

The A100 GPU is designed for broad performance scalability. Customers can share a single A100 using MIG GPU partitioning technology, or use multiple A100 GPUs connected by the new Third-generation NVIDIA NVLink® high-speed interconnect in powerful new NVIDIA DGX™, NVIDIA HGX™, and NVIDIA EGX™ systems. A100-based systems connected by the new NVIDIA NVSwitch™ and Mellanox® state-of-the-art InfiniBand™ and Ethernet solutions can be scaled out to tens, hundreds, or thousands of A100s in compute clusters, cloud instances, or immense supercomputers to accelerate many types of applications and workloads. Additionally, the A100 GPU's revolutionary new hardware capabilities are enhanced by new CUDA 11 features that improve programmability and reduce AI and HPC software complexity.

The NVIDIA A100 GPU is the **first elastic GPU architecture** with the ability to scale-up to giant GPUs using NVLink, NVSwitch, and InfiniBand, or scale-out to support multiple independent users with MIG, simultaneously achieving great performance and lowest cost per-GPU instance.

The NVIDIA A100 Tensor Core GPU delivers the greatest generational leap in NVIDIA GPU accelerated computing ever.

NVIDIA A100 Tensor Core GPU Overview

Next-generation Data Center and Cloud GPU

Increasingly complex and varied AI, HPC, and data analytics workloads require additional GPU computing power, multi-GPU connectivity enhancements, and a comprehensive suite of supporting software stacks. NVIDIA meets these growing GPU computing challenges with the new NVIDIA A100 Tensor Core GPU based on the NVIDIA Ampere GPU architecture, combined with new CUDA software advances.

The A100 GPU includes many core architecture enhancements that deliver significant speed-ups for AI, HPC, and data analytics workloads compared to V100, as explained throughout this paper. The new **Sparsity** feature further accelerates math operations by up to 2x. High-bandwidth HBM2 memory and larger, faster caches feed data to the increased numbers of CUDA Cores and Tensor Cores.

The new Third-generation NVLink and PCIe Gen 4 speed up multi-GPU system configurations. Many other enhancements enable strong scaling for hyperscale data centers, and robust Multi-Instance GPU (MIG) virtualization for Cloud Service Provider (CSP) systems and their customers. NVIDIA Ampere architecture also improves ease of programming, while lowering latencies, and reducing AI and HPC software complexity. NVIDIA Ampere architecture GPUs deliver all these new features with greater performance per watt than the prior generation NVIDIA Volta GPUs.

The NVIDIA A100 GPU is architected to not only accelerate large complex workloads, but also efficiently accelerate many smaller workloads. A100 enables building data centers that can accommodate unpredictable workload demand, while providing fine-grained workload provisioning, higher GPU utilization, and improved TCO.

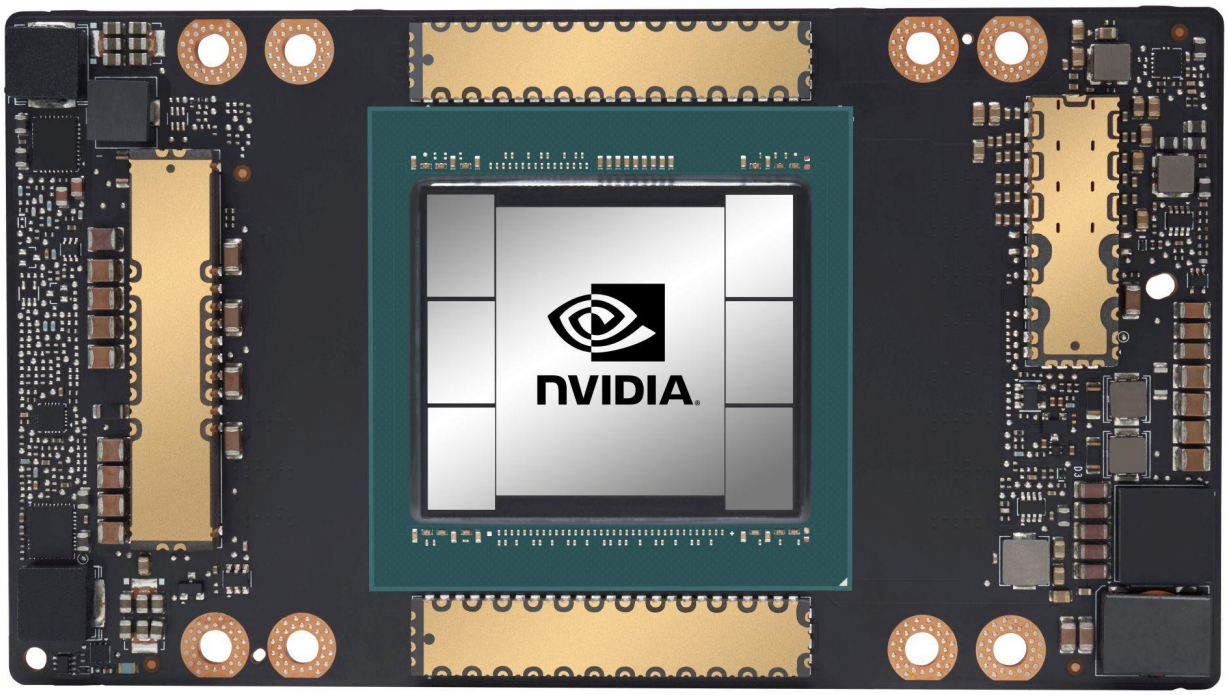


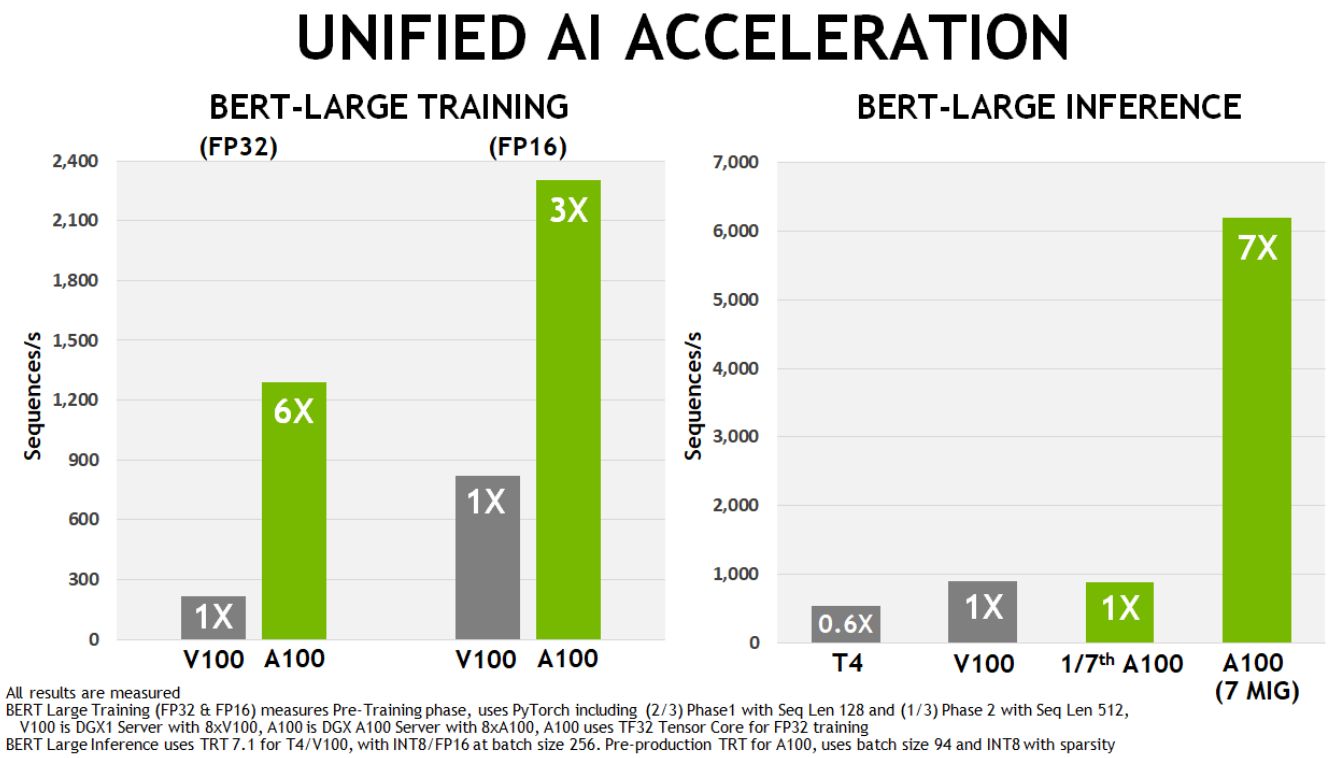
Figure 3. NVIDIA A100 GPU on new SXM4 Module

A100's versatility helps infrastructure managers maximize the utility of every GPU in their data center to meet different-sized performance needs, from the smallest job to the biggest multi-node workload. A100 powers the NVIDIA data center platform that includes Mellanox HDR InfiniBand (IB), NVSwitch, HGX A100, and the Magnum IO SDK for scaling up. This integrated team of technologies efficiently scales to tens of thousands of GPUs to train the most complex AI networks at unprecedented speed.

Diffusing accelerated computing within enterprise and cloud environments demands high utilization on small workloads. With the new Multi-Instance GPU technology, each A100 can be divided into as many as seven GPU Instances for optimal utilization and to expand access to every user and application.

Industry-leading Performance for AI, HPC, and Data Analytics

The NVIDIA A100 GPU delivers exceptional speedups over V100 for AI training and inference workloads as shown in Figure 4. Similarly, Figure 5 shows substantial performance improvements across different HPC applications.



A100 GPU performance in BERT deep learning training and inference scenarios compared to NVIDIA Tesla V100 and NVIDIA Tesla T4.

Figure 4. Unified AI Acceleration for BERT-LARGE Training and Inference

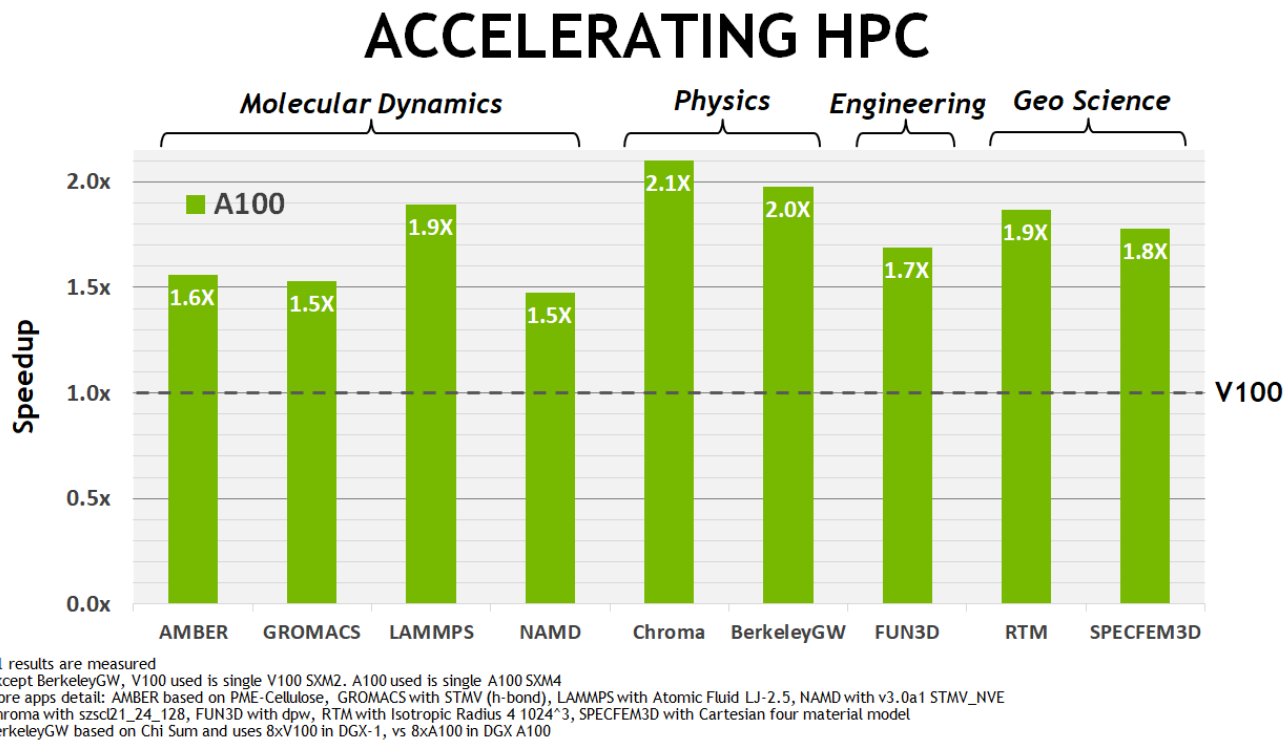


Figure 5. A100 GPU HPC application speedups compared to NVIDIA Tesla V100

A100 GPU Key Features Summary

The NVIDIA A100 Tensor Core GPU is the world's fastest cloud and data center GPU accelerator designed to power computationally-intensive AI, HPC, and data analytics applications.

Fabricated on TSMC's 7nm N7 manufacturing process, the NVIDIA Ampere architecture-based GA100 GPU that powers A100 includes **54.2** billion transistors with a die size of **826 mm2**.

A high-level summary of key A100 features is provided below for a quick understanding of the important new A100 technologies and performance levels. In-depth architecture information is presented in subsequent sections.

A100 GPU Streaming Multiprocessor (SM)

The new SM in the NVIDIA Ampere architecture-based A100 Tensor Core GPU significantly increases performance, builds upon features introduced in both the Volta and Turing SM architectures, and adds many new capabilities.

The A100 **Third-Generation Tensor Cores** enhance operand sharing and improve efficiency, and add powerful new data types including:

- TF32 Tensor Core instructions which accelerate processing of FP32 data
- IEEE-compliant FP64 Tensor Core instructions for HPC
- BF16 Tensor Core instructions at the same throughput as FP16

Table 1. NVIDIA A100 Tensor Core GPU Performance Specs

Peak FP64[1]	9.7 TFLOPS
Peak FP64 Tensor Core[1]	19.5 TFLOPS
Peak FP32[1]	19.5 TFLOPS
Peak FP16[1]	78 TFLOPS
Peak BF16[1]	39 TFLOPS
Peak TF32 Tensor Core[1]	156 TFLOPS | 312 TFLOPS[2]
Peak FP16 Tensor Core[1]	312 TFLOPS | 624 TFLOPS[2]
Peak BF16 Tensor Core[1]	312 TFLOPS | 624 TFLOPS[2]
Peak INT8 Tensor Core[1]	624 TOPS | 1,248 TOPS[2]
Peak INT4 Tensor Core[1]	1,248 TOPS | 2,496 TOPS[2]

1 - Peak rates are based on GPU Boost Clock.
2 - Effective TFLOPS / TOPS using the new Sparsity feature

New **Sparsity** support in A100 Tensor Cores can exploit fine-grained structured sparsity in deep learning networks to double the throughput of Tensor Core operations. Sparsity features are described in detail in the "**A100 Introduces Fine-Grained Structured Sparsity**" section below.

The larger and faster L1 cache and shared memory unit in A100 provides 1.5x the aggregate capacity per SM compared to V100 (192 KB vs 128 KB per SM) to deliver additional acceleration for many HPC and AI workloads.

A number of other new SM features improve programmability and reduce software complexity.

40 GB HBM2 and 40 MB L2 Cache

To feed its massive computational throughput, the NVIDIA A100 GPU has 40 GB of high-speed HBM2 memory with a class-leading 1555 GB/sec of memory bandwidth - a 73% increase compared to Tesla V100. In addition, the A100 GPU has significantly more on-chip memory including a 40 MB Level 2 (L2) cache - nearly 7x larger than V100 - to maximize compute performance. With a new partitioned crossbar structure, the A100 L2 cache provides 2.3x the L2 cache read bandwidth of V100.

To optimize capacity utilization, the NVIDIA Ampere architecture provides L2 cache residency controls for you to manage data to keep or evict from the cache. A100 also adds Compute Data Compression to deliver up to an additional 4x improvement in DRAM bandwidth and L2 bandwidth, and up to 2x improvement in L2 capacity.

Multi-Instance GPU (MIG)

The new Multi-Instance GPU (MIG) feature allows the A100 Tensor Core GPU to be securely partitioned into as many as seven separate GPU Instances for CUDA applications, providing multiple users with separate GPU resources to accelerate their applications and development projects.

With MIG, each instance's processors have separate and isolated paths through the entire memory system - the on-chip crossbar ports, L2 cache banks, memory controllers, and DRAM address busses are all assigned uniquely to an individual instance. This ensures that an individual user's workload can run with predictable throughput and latency, with the same L2 cache allocation and DRAM bandwidth, even if other tasks are thrashing their own caches or saturating their DRAM interfaces.

MIG increases GPU hardware utilization while providing a defined QoS and isolation between different clients (such as VMs, containers, and processes). MIG is especially beneficial for Cloud Service Providers who have multi-tenant use cases, and it ensures one client cannot impact the work or scheduling of other clients, in addition to providing enhanced security and allowing GPU utilization guarantees for customers.

Third-Generation NVLink

The third-generation of NVIDIA's high-speed NVLink interconnect implemented in A100 GPUs and the new NVSwitch significantly enhances multi-GPU scalability, performance, and reliability. With more links per GPU and switch, the new NVLink provides much higher GPU-GPU communication bandwidth, and improved error-detection and recovery features.

Third-generation NVLink has a data rate of 50 Gbit/sec per signal pair, nearly doubling the 25.78 Gbits/sec rate in V100. A single A100 NVLink provides 25 GB/second bandwidth in each direction similar to V100, but using only half the number of signal pairs per link compared to

V100. The total number of links is increased to twelve in A100, versus 6 in V100, yielding 600 GB/sec total bandwidth versus 300 GB/sec for V100.

Support for NVIDIA Magnum IO™ and Mellanox Interconnect Solutions

The NVIDIA A100 Tensor Core GPU is fully compatible with NVIDIA Magnum IO and Mellanox state-of-the-art InfiniBand and Ethernet interconnect solutions to accelerate multi-node connectivity. The NVIDIA Magnum IO APIs integrate computing, networking, file systems, and storage to maximize IO performance for multi-GPU, multi-node accelerated systems. It interfaces with CUDA-X™ libraries to accelerate IO across a broad range of workloads, from AI to data analytics to visualization.

PCIe Gen 4 with SR-IOV

The A100 GPU supports PCI Express Gen 4 (PCIe Gen 4) which doubles the bandwidth of PCIe 3.0/3.1 by providing 31.5 GB/sec versus 15.75 GB/sec for x16 connections. The faster speed is especially beneficial for A100 GPUs connecting to PCIe 4.0-capable CPUs, and to support fast network interfaces, such as 200 Gbit/sec InfiniBand. A100 also supports Single Root Input/Output Virtualization (SR-IOV), which allows sharing and virtualizing a single PCIe connection for multiple processes or Virtual Machines (VMs).

Improved Error and Fault Detection, Isolation, and Containment

It is critically important to maximize GPU uptime and availability by detecting, containing, and often correcting errors and faults, rather than forcing GPU resets, especially in large multi-GPU clusters and single-GPU, multi-tenant environments such as MIG configurations. The NVIDIA A100 Tensor Core GPU includes new technology to improve error/fault attribution, isolation, and containment as described in the in-depth architecture sections below.

Asynchronous Copy

The A100 GPU includes a new asynchronous copy instruction that loads data directly from global memory into SM shared memory, eliminating the need for intermediate register file (RF) usage. Async-copy reduces register file bandwidth, uses memory bandwidth more efficiently, and reduces power consumption. As the name implies, asynchronous copy can be done in the background while the SM is performing other computations.

Asynchronous Barrier

The A100 GPU provides hardware-accelerated barriers in shared memory. These barriers are available using CUDA 11 in the form of ISO C++-conforming barrier objects. Asynchronous barriers split apart the barrier arrive and wait operations, and can be used to overlap asynchronous copies from global memory into shared memory with computations in the SM. They can be used to implement producer-consumer models using CUDA threads. Barriers also

provide mechanisms to synchronize CUDA threads at different granularities, not just warp or block level.

Task Graph Acceleration

CUDA Task Graphs provide a more efficient model for submitting work to the GPU. A task graph consists of a series of operations, such as memory copies and kernel launches, connected by dependencies. Task graphs enable a define-once/run-repeatedly execution flow. A predefined task graph allows the launch of any number of kernels in a single operation, greatly improving application efficiency and performance. A100 adds new hardware features to make the paths between grids in a task graph significantly faster.

NVIDIA A100 Tensor Core GPU Architecture In-Depth

The NVIDIA A100 GPU based on NVIDIA Ampere architecture is engineered to provide as much AI and HPC computing power as possible from its many new architectural features and optimizations. A100 is built on the TSMC 7nm N7 FinFET fabrication process that provides higher transistor density, improved performance, and better power efficiency than the 12nm FFN process used in Tesla V100. A new Multi-Instance GPU (MIG) capability provides enhanced client/application fault isolation and QoS for multi-tenant and virtualized GPU environments which is especially beneficial to Cloud Service Providers. A faster and more error-resilient third-generation of NVIDIA's NVLink interconnect delivers improved multi-GPU performance scaling for hyperscale data centers.

The NVIDIA GA100 GPU is composed of multiple GPU Processing Clusters (GPCs), Texture Processing Clusters (TPCs), Streaming Multiprocessors (SMs), and HBM2 memory controllers.

The **full implementation** of the GA100 GPU includes the following units:

- 8 GPCs, 8 TPCs/GPC, 2 SMs/TPC, 16 SMs/GPC, 128 SMs per full GPU
- 64 FP32 CUDA Cores/SM, 8192 FP32 CUDA Cores per full GPU
- 4 Third-generation Tensor Cores/SM, 512 Third-generation Tensor Cores per full GPU
- 6 HBM2 stacks, 12 512-bit Memory Controllers

The **NVIDIA A100 Tensor Core GPU implementation** of the GA100 GPU includes the following units:

- 7 GPCs, 7 or 8 TPCs/GPC, 2 SMs/TPC, up to 16 SMs/GPC, 108 SMs
- 64 FP32 CUDA Cores/SM, 6912 FP32 CUDA Cores per GPU
- 4 Third-generation Tensor Cores/SM, 432 Third-generation Tensor Cores per GPU
- 5 HBM2 stacks, 10 512-bit Memory Controllers

The TSMC 7nm N7 process used to build the GA100 GPU allows many more GPCs, TPCs, and SM units, along with many other new hardware features in a die size similar to the Volta GV100 GPU (which was fabricated on TSMC's 12nm FFN process).

Figure 6 shows a full GA100 GPU with 128 SMs. The A100 is based on GA100 and has 108 SMs.

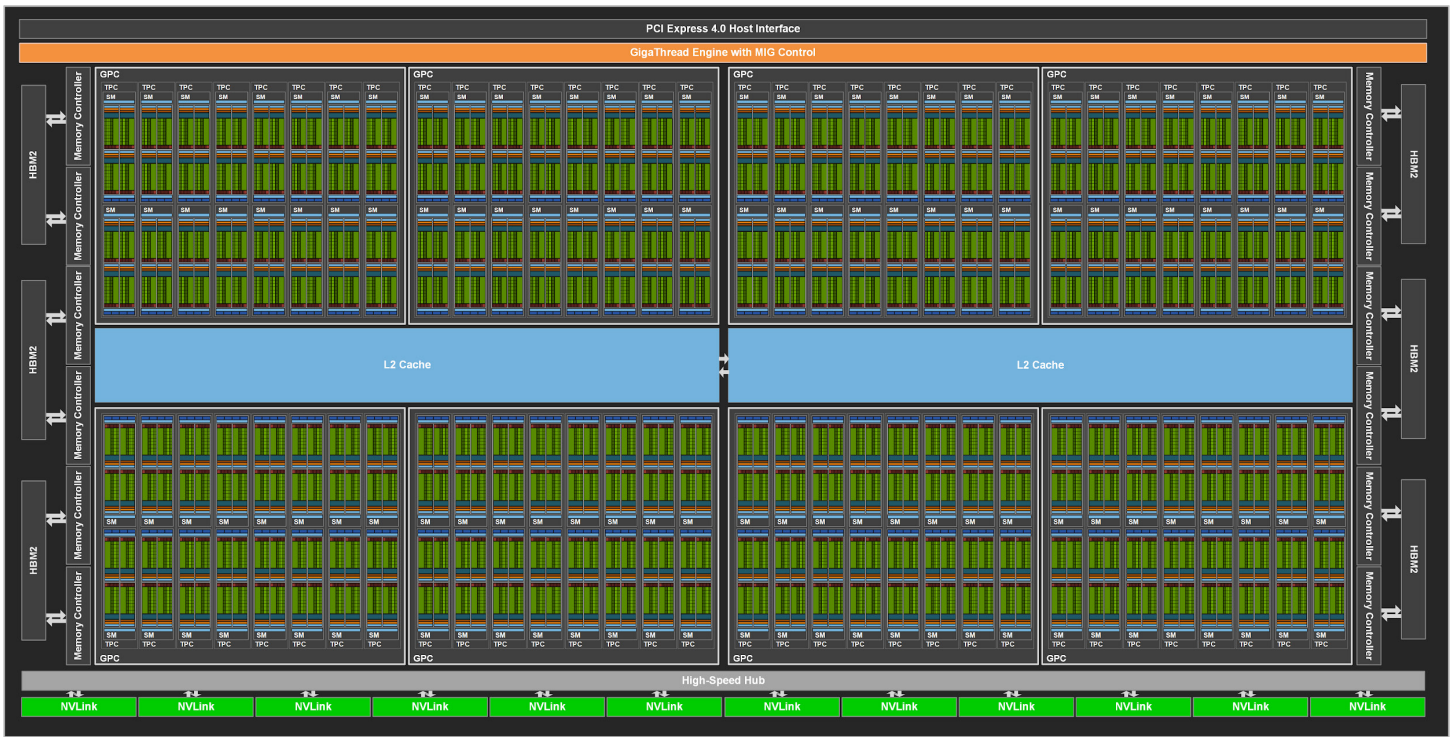


Figure 6. GA100 Full GPU with 128 SMs (A100 Tensor Core GPU has 108 SMs)

A100 SM Architecture

The new A100 SM significantly increases performance, builds upon features introduced in both the Volta and Turing SM architectures, and adds many new capabilities and enhancements.

The A100 SM diagram is shown in Figure 7. Volta and Turing have eight Tensor Cores per SM, with each Tensor Core performing 64 FP16/FP32 mixed-precision fused multiply-add (FMA) operations per clock. The A100 SM includes new third-generation Tensor Cores that each perform 256 FP16/FP32 FMA operations per clock. A100 has four Tensor Cores per SM, which together deliver 1024 dense FP16/FP32 FMA operations per clock, a 2x increase in computation horsepower per SM compared to Volta and Turing.

Key SM features are briefly highlighted below (and described in detail in subsequent sections):

- Third-generation Tensor Cores:
  - Acceleration for all data types including FP16, BF16, TF32, FP64, INT8, INT4, and Binary.
  - New Tensor Core sparsity feature exploits fine-grained structured sparsity in deep learning networks, doubling the performance of standard Tensor Core operations.

    - TF32 Tensor Core operations in A100 provide an easy path to accelerate FP32 input/output data in DL frameworks and HPC, running 10x faster than V100 FP32 FMA operations, or 20x faster with sparsity.
    - FP16/FP32 mixed-precision Tensor Core operations deliver unprecedented processing power for DL, running 2.5x faster than Tesla V100 Tensor Core operations, increasing to 5x with sparsity.
    - BF16/FP32 mixed-precision Tensor Core operations run at the same rate as FP16/FP32 mixed-precision.
    - FP64 Tensor Core operations deliver unprecedented double precision processing power for HPC, running 2.5x faster than V100 FP64 DFMA operations.
    - INT8 Tensor Core operations with sparsity deliver unprecedented processing power for DL Inference, running up to 20x faster than V100 INT8 operations.
- 192 KB of combined shared memory and L1 data cache, 1.5x larger than V100 SM
- New asynchronous copy instruction loads data directly from global memory into shared memory, optionally bypassing L1 cache, and eliminating the need for intermediate register file (RF) usage
- New shared-memory-based barrier unit (asynchronous barriers) for use with the new asynchronous copy instruction
- New instructions for L2 cache management and residency controls
- New warp-level reduction instructions supported by CUDA Cooperative Groups
- Many programmability improvements which reduce software complexity

Figure 7. GA100 Streaming Multiprocessor (SM)

Third-Generation NVIDIA Tensor Core

Tensor Cores are specialized high-performance compute cores for matrix math operations that provide groundbreaking performance for AI and HPC applications. Tensor Cores perform matrix multiply and accumulate (MMA) calculations. Hundreds of Tensor Cores operating in parallel in one NVIDIA GPU enable massive increases in throughput and efficiency. Tensor Cores were first introduced in the NVIDIA Tesla V100 GPU, and further enhanced in NVIDIA's more recent Turing GPUs. (Refer to the NVIDIA Tesla V100 GPU Architecture for background information on Tensor Core operation.)

Table 2. A100 speedup over V100 (TC=Tensor Core, GPUs at respective clock speeds)

	V100	A100	A100 Sparsity[1]	A100 Speedup	A100 Speedup with Sparsity
A100 FP16 vs V100 FP16	31.4 TFLOPS	78 TFLOPS	NA	2.5x	NA
A100 FP16 TC vs V100 FP16 TC	125 TFLOPS	312 TFLOPS	624 TFLOPS	2.5x	5x
A100 BF16 TC vs V100 FP16 TC	125 TFLOPS	312 TFLOPS	624 TFLOPS	2.5x	5x
A100 FP32 vs V100 FP32	15.7 TFLOPS	19.5 TFLOPS	NA	1.25x	NA
A100 TF32 TC vs V100 FP32	15.7 TFLOPS	156 TFLOPS	312 TFLOPS	10x	20x
A100 FP64 vs V100 FP64	7.8 TFLOPS	9.7 TFLOPS	NA	1.25x	NA
A100 FP64 TC vs V100 FP64	7.8 TFLOPS	19.5 TFLOPS	NA	2.5x	NA
A100 INT8 TC vs V100 INT8	62 TOPS	624 TOPS	1248 TOPS	10x	20x
A100 INT4 TC	NA	1248 TOPS	2496 TOPS	NA	NA
A100 Binary TC	NA	4992 TOPS	NA	NA	NA

1 - Effective TOPS / TFLOPS using the new Sparsity Feature

A100 Tensor Cores Boost Throughput

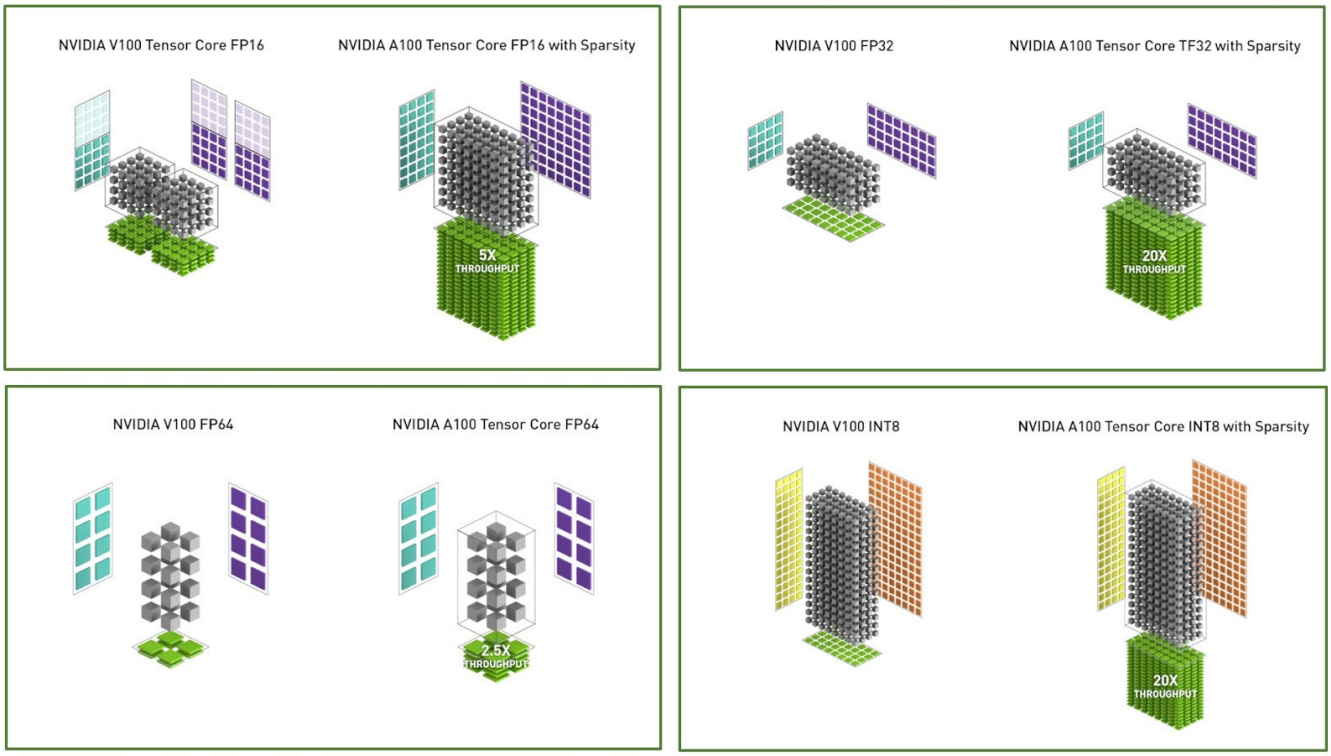
The new third-generation Tensor Core architecture in A100 delivers double the raw dense Tensor throughput per SM compared to V100, accelerates more data types, and delivers tremendous additional 2x speedups for sparse matrix computations.

General Matrix-Matrix Multiplication (GEMM) operations are at the core of neural network training and inference, and are used to multiply large matrices of input data and weights in various layers. The GEMM operation computes the matrix product $D = A * B + C$, where C and D are m-by-n matrices, A is an m-by-k matrix, and B is a k-by-n matrix. The problem size of such GEMM operations running on Tensor Cores is defined by the matrix sizes, and typically denoted as m-by-n-by-k.

Using FP16/FP32 mixed-precision Tensor Core operations as an example, at the hardware level, each Tensor Core in the Volta architecture can execute 64 FP16 fused multiply-add operations (FMAs) with FP32 accumulation per clock, allowing it to compute a mixed-precision 4x4x4 matrix multiplication per clock. Since each Volta SM includes eight Tensor Cores, a single SM delivers 512 FP16 FMA operations per clock or 1024 individual FP16 floating point operations per clock. Each of the A100 Tensor Cores can execute 256 FP16 FMA operations per clock, allowing it to compute the results for an 8x4x8 mixed-precision matrix multiplication per clock. Each SM in the A100 GPU includes four of the new redesigned Tensor Cores and therefore each SM in A100 delivers 1024 FP16 FMA operations per clock (or 2048 individual FP16 floating point operations per clock).

Comparing total GPU performance, not just SM-level performance, the NVIDIA A100 Tensor Core GPU with its 108 SMs includes a total of 432 Tensor Cores that deliver up to 312 TFLOPS of dense mixed-precision FP16/FP32 performance. That equates to 2.5x the mixed-precision Tensor Core performance of the entire Tesla V100 GPU, and 20x V100's standard FP32 (FMA operations running on traditional FP32 CUDA cores) throughput.

Figure 8 compares V100 and A100 FP16 Tensor Core operations, and also compares V100 FP32, FP64, and INT8 standard operations to respective A100 TF32, FP64, and INT8 Tensor Core operations. Throughputs are aggregate per GPU, with A100 using sparse Tensor Core operations for FP16, TF32, and INT8. Note the upper left diagram shows two V100 FP16 Tensor Cores, since a V100 SM has two Tensor Cores per SM partition, while an A100 SM one.



A100 Tensor Core operations compared to V100 Tensor Core and standard operations for different data types.

Figure 8. A100 vs V100 Tensor Core Operations

A100 Tensor Cores Support All DL Data Types

In addition to FP16 precision introduced on the Volta Tensor Core, and the INT8, INT4 and binary 1-bit precisions added in the Turing Tensor Core, the A100 Tensor Core adds support for TF32, BF16 and FP64 formats. (FP64 double-precision MMA is discussed in the next section).

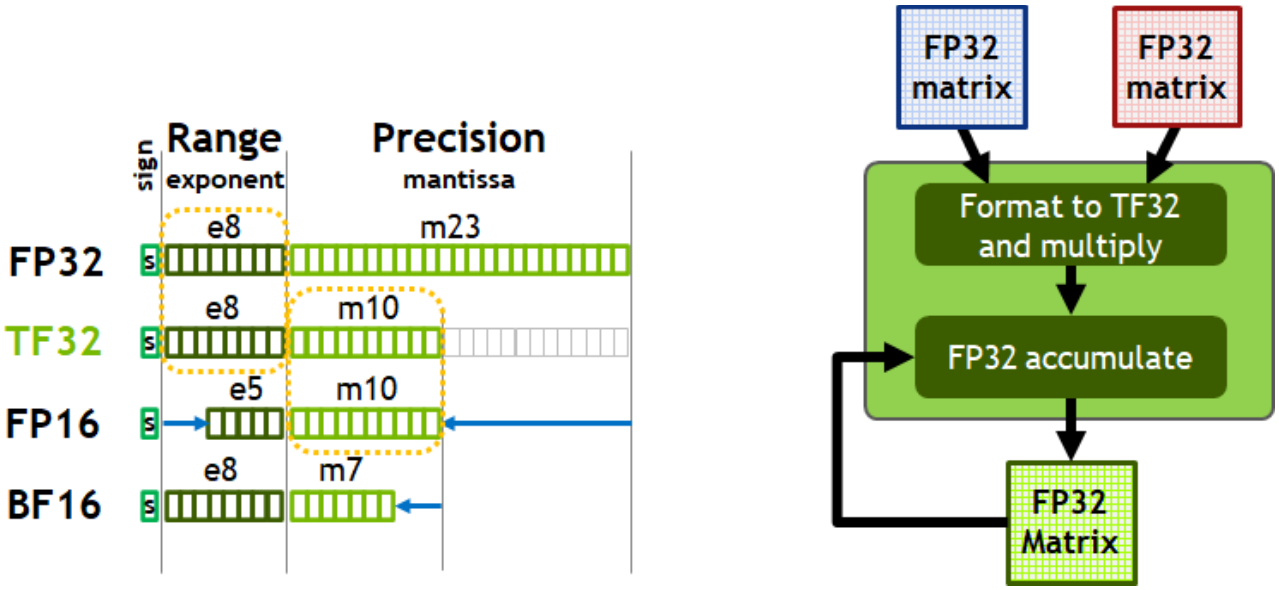
Volta GPU architecture introduced Tensor Cores that operate on IEEE FP16 data types, providing 8x more math throughput compared to V100 FP32. Results are accumulated into FP32 for mixed precision training or FP16 for inference. From a raw architectural performance perspective, if both A100 and V100 were operating at the same clock speed, a single A100 SM delivers 2x FP16 Tensor Core performance compared to the V100 SM, and 16x compared to standard V100 (and A100) FP32 FFMA operations.

Turing architecture extended Tensor Cores to handle more inference use cases by adding INT8, INT4, and Binary support. On Turing these provided 16x, 32x, and 128x more math throughput when compared to FP32. The A100 SM delivers 2x INT8, INT4, and Binary Tensor Core performance compared to the Turing SM, respectively, and 32x, 64x, and 256x compared to A100 FP32 FFMA.

NVIDIA Ampere architecture adds three additional formats to Tensor Cores – BF16, TF32 and FP64. BF16 is an alternative to IEEE FP16, and includes an 8-bit exponent, 7-bit mantissa, and 1 sign-bit. Both FP16 and BF16 have been shown to successfully train neural networks in mixed-precision mode, matching FP32 training results without hyper-parameter adjustment. Both FP16 and BF16 modes of Tensor Cores provide 16x more math throughput than FP32 in A100 GPUs.

Today, the default math for AI training is FP32, without tensor core acceleration. The NVIDIA Ampere architecture introduces new support for TF32, enabling AI training to use tensor cores by default with no effort on the user's part. Non-tensor operations continue to use the FP32 datapath, while TF32 tensor cores read FP32 data and use the same range as FP32 with reduced internal precision, before producing a standard IEEE FP32 output. TF32 includes an 8-bit exponent (same as FP32), 10-bit mantissa (same precision as FP16) and 1 sign-bit.

As with Volta, Automatic Mixed Precision (AMP) enables users to use mixed precision with FP16 for AI training with just a few lines of code changes. Using AMP, A100 delivers a further 2X faster Tensor Core performance over TF32.



TensorFloat-32 (TF32) provides the range of FP32 with the precision of FP16, 8x precision vs. BF16 (left). A100 accelerates tensor math with TF32 while supporting FP32 input and output data (right), enabling easy integration into DL and HPC programs and automatic acceleration of DL frameworks.

Figure 9. TensorFloat-32 (TF32)

Table 3. A100 Tensor Core Input / Output Formats and Performance vs FP32 FFMA.

	INPUT OPERANDS	ACCUMULATOR	TOPS	X-factor vs. FFMA	SPARSE TOPS	SPARSE X-factor vs. FFMA
V100	FP32	FP32	15.7	1x	-	-
	FP16	FP32	125	8x	-	-
A100	FP32	FP32	19.5	1x	-	-
	TF32	FP32	156	8x	312	16x
	FP16	FP32	312	16x	624	32x
	BF16	FP32	312	16x	624	32x
	FP16	FP16	312	16x	624	32x
	INT8	INT32	624	32x	1248	64x
	INT4	INT32	1248	64x	2496	128x
	BINARY	INT32	4992	256x	-	-
	IEEE FP64		19.5	1x	-	-

Note: TOPS column indicates TFLOPS for floating-point ops and TOPS for integer ops. X-factors compare MMA ops with and without sparsity to standard FP32 FFMA ops. (Sparse TOPS represents effective TOPS / TFLOPS using the new Sparsity feature.)

To summarize the user choices for NVIDIA Ampere architecture math for Deep Learning training:

- By default, TF32 tensor cores are used, with no adjustment to user scripts. Up to 8x more throughput compared to FP32 on A100 and up to 10x compared to FP32 on V100.
- FP16 or BF16 mixed-precision training should be used for maximum training speed. Up to 2x more throughput compared to TF32, and up to 16x compared to FP32 on A100 and up to 20x compared to FP32 on V100.

A100 Tensor Cores Accelerate HPC

The performance needs of High-Performance Computing (HPC) applications are growing rapidly. Many applications from a wide range of scientific and research disciplines rely on double precision (FP64) computations. To meet the rapidly growing compute needs of HPC computing, A100 Tensor Cores support acceleration of IEEE-compliant FP64 computations, delivering up to 2.5x the FP64 performance of the NVIDIA Tesla V100 GPU. The new Double Precision Matrix Multiply Add instruction on A100 replaces 8 DFMA instructions on V100, reducing instruction fetches, scheduling overhead, register reads, datapath power, and shared memory read bandwidth. Using Tensor Cores, each SM in A100 computes a total of 64 FP64 FMA operations/clock (or 128 FP64 operations/clock), which is twice the throughput of Tesla V100. The A100 Tensor Core GPU with 108 SMs delivers a peak FP64 throughput of 19.5 TFLOPS, which is 2.5x that of Tesla V100.

With support for these new formats, the A100 Tensor Cores can be used to accelerate HPC workloads, iterative solvers, and various new AI algorithms.

Mixed Precision Tensor Cores for HPC

One of the most promising applications for mixed-precision Tensor Cores in HPC is the field of iterative refinement methods. Iterative refinement methods are commonly used for solving linear systems of equations, which occur ubiquitously in HPC applications in a wide range of fields such as earth science, fluid dynamics, healthcare, material science, and nuclear energy, as well as oil and gas exploration.

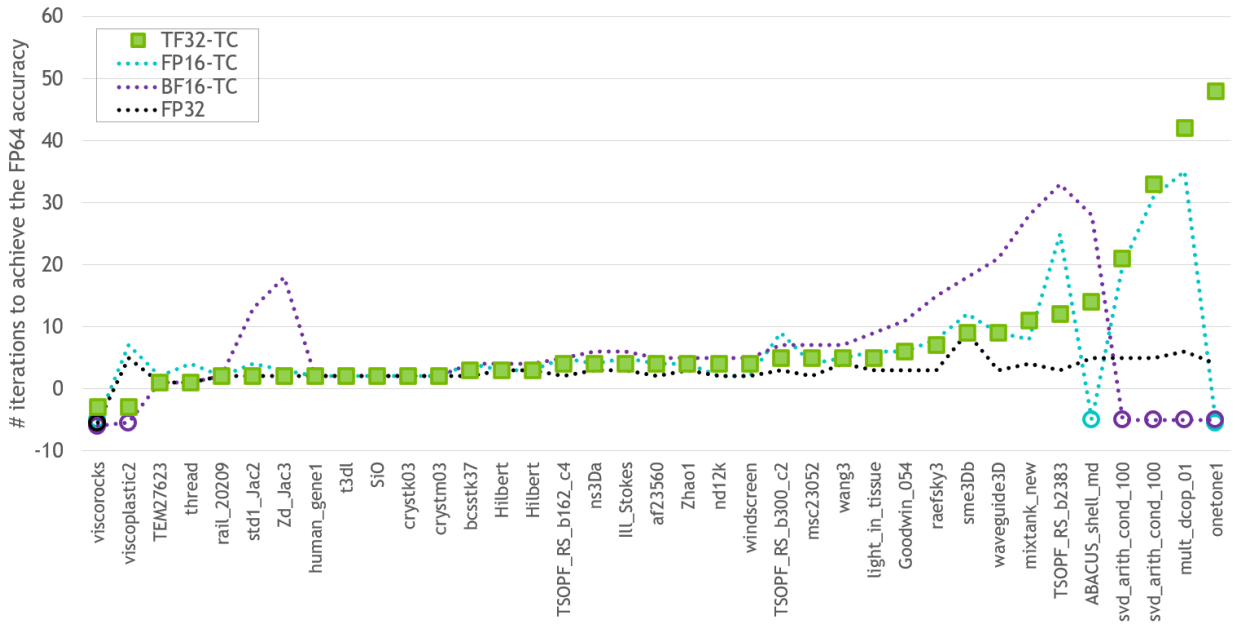
The Tensor Core Accelerated Iterative Refinement Solver (TCAIRS) in cuSOLVER automates usage of mixed precision for this application. Last year, a fusion reaction study for the International Thermonuclear Experimental Reactor demonstrated that mixed-precision techniques delivered a speedup of 3.5x on V100 for such solvers using V100's FP16 Tensor Cores. The same technology used in that study tripled the Summit supercomputer's performance on the HPL-AI benchmark.

cuSOLVER in CUDA 11.0 adds support for A100's new tensor core formats including TF32. Figure 10 and Figure 11 below show results of the TCAIRS solver on 37 tests from the SuiteSparse Matrix collection, comparing convergence rate and performance for FP32, FP16 with input scaling, BF16, and TF32. These were compared to the performance of the reference FP64 solver which leverages the FP64 Tensor Cores on the A100. In cases where the mixed-

precision solver automatically falls back to the FP64 solver due to slow or no convergence, the number of iterations were recorded as negative, and speedup is less than one, as it included the cost of the failed attempt.

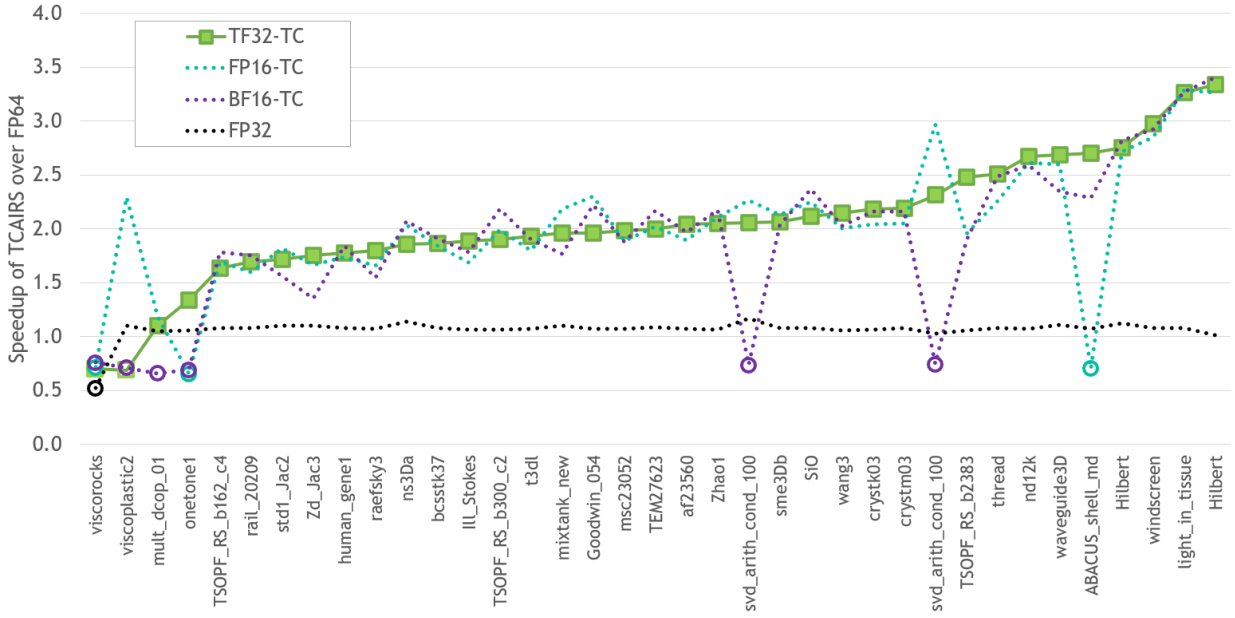
As shown in Figure 10 and Figure 11, TF32 delivered the fastest and most robust results compared to other Tensor Core modes. The number of iterations to converge was the lowest for TF32 amongst the Tensor Core modes. While FP32 had one fallback case, TF32 had only two, compared to three for FP16 with input scaling, and six for BF16 Tensor Core modes. The geomean speedup over the FP64 solver was 2.0X for TF32 Tensor Cores compared to 1.9X for FP16 and 1.8X for BF16. For large matrices with a size of around 40K with complex numbers, the TCAIRS solver delivers speedups of up to 4X with TF32 on A100.

The applications of mixed-precision Tensor Core acceleration are not limited to dense linear solvers, and can be extended to sparse problems, in addition to other numerical methods where matrix multiplies represent a significant portion of the algorithmic complexity.



Number of iterations taken by the TCAIRS solver compared for 37 different problems to converge to FP64 accuracy for different precisions. The results are sorted by increasing number of iterations for TF32. Negative figures indicate the solver did not converge with reduced precision and fell back to a full FP64 solution.

Figure 10. Iterations of TCAIRS Solver to Converge to FP64 Accuracy



Speedup of the TCAIRS solver over the baseline FP64 direct solver compared for 37 different problems. Cases where speedups are less than one indicate that the TCAIRS solver did not converge with reduced precision and fell back to a full FP64 solution. The results are sorted by increasing speedup for TF32.

Figure 11. TCAIRS solver speedup over the baseline FP64 direct solver

A100 Introduces Fine-Grained Structured Sparsity

With the A100 GPU, NVIDIA introduces Fine-Grained Structured Sparsity, a novel approach which doubles compute throughput for deep neural networks.

Sparsity is possible in deep learning because the importance of individual weights evolves during the learning process, and by the end of network training, only a subset of weights have acquired a meaningful purpose in determining the learned output. The remaining weights are no longer needed.

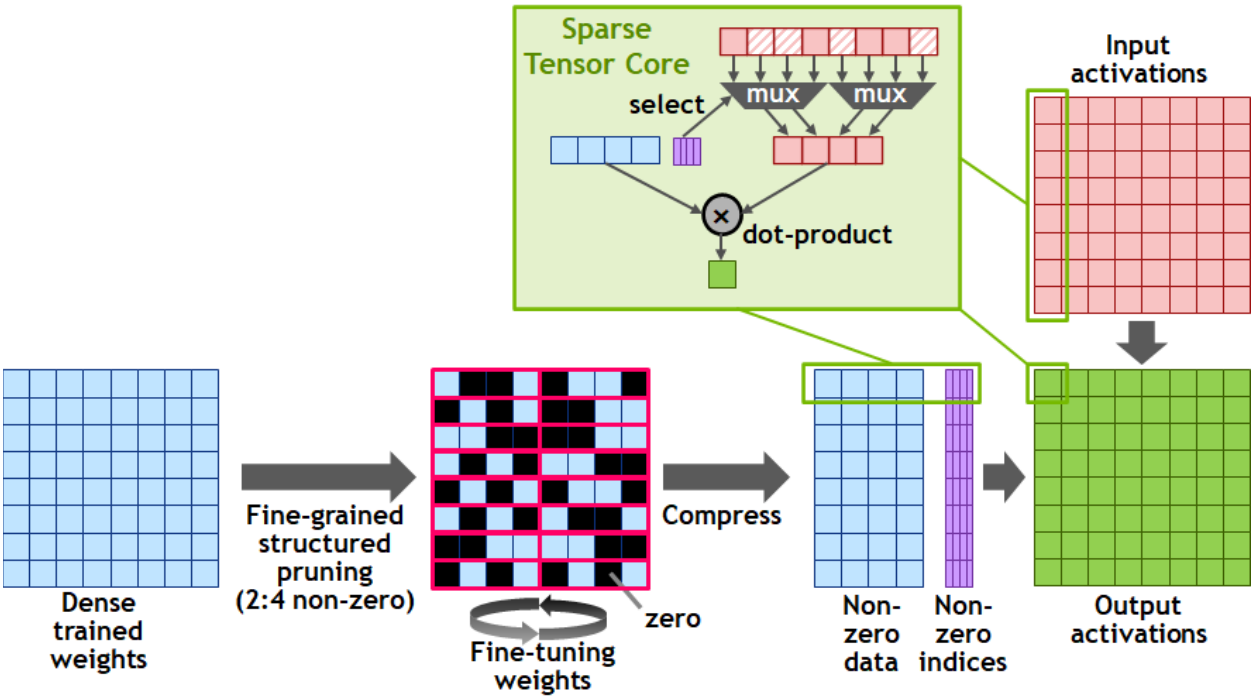
Fine grained structured sparsity imposes a constraint on the allowed sparsity pattern, making it more efficient for hardware to do the necessary alignment of input operands. NVIDIA engineers have found that because deep learning networks are able to adapt weights during the training process based on training feedback, in general the structure constraint does not impact the accuracy of the trained network for inferencing. This enables inferencing acceleration with sparsity. For training acceleration, sparsity needs to be introduced early in the process to offer a performance benefit, and methodologies for training acceleration without accuracy loss are an active research area.

Refer to **Appendix B - Sparse Neural Network Primer** for additional background information on sparsity.

Sparse Matrix Definition

Structure is enforced through a new 2:4 sparse matrix definition that allows two non-zero values in every four-entry vector.

A100 supports 2:4 structured sparsity on rows, as shown in Figure 12 below. Due to the well-defined structure of the matrix, it can be compressed efficiently and reduce memory storage and bandwidth by almost 2x.



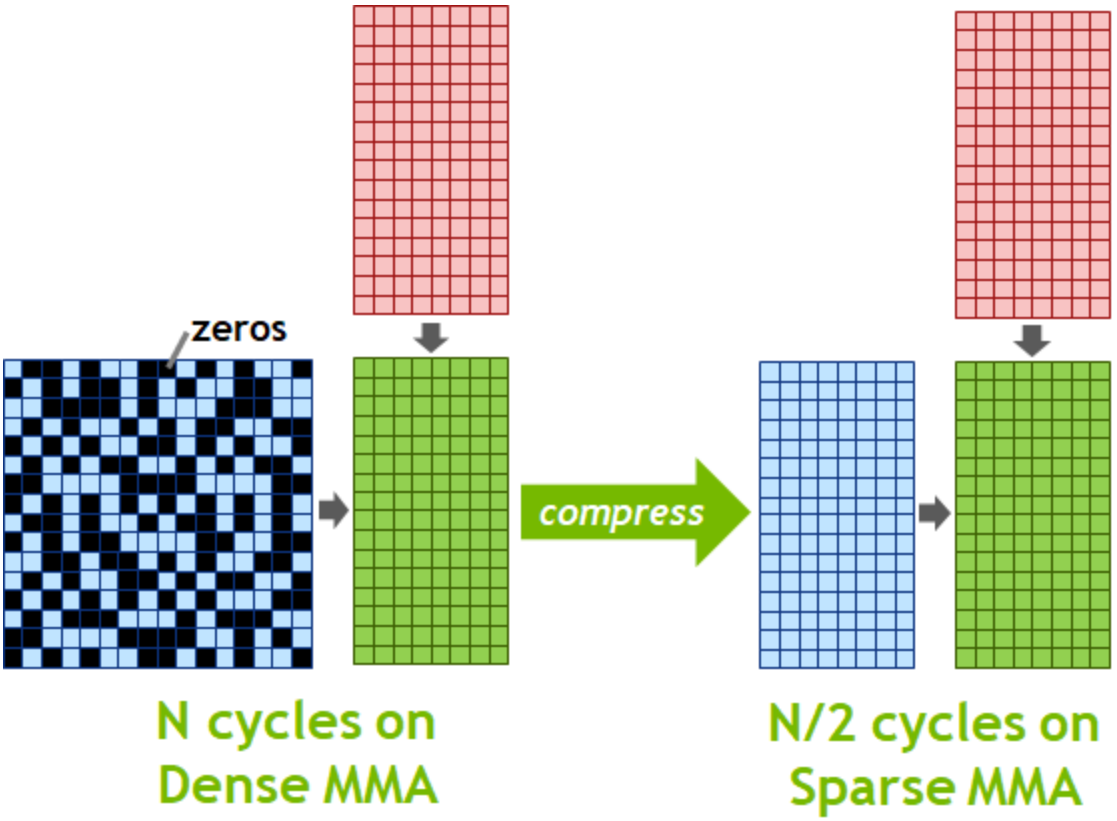
A100 Fine-Grained Structured Sparsity prunes trained weights with a 2-out-of-4 non-zero pattern, followed by a simple and universal recipe for fine-tuning the non-zero weights. The weights are compressed for a 2x reduction in data footprint and bandwidth, and the A100 Sparse Tensor Core doubles math throughput by skipping the zeros.

Figure 12. A100 Fine-Grained Structured Sparsity

NVIDIA has developed a simple and universal recipe for sparsifying deep neural networks for inference using this 2:4 structured sparsity pattern. The network is first trained using dense weights, then fine-grained structured pruning is applied, and finally the remaining non-zero weights are fine-tuned with additional training steps. This method results in virtually no loss in inferencing accuracy based on evaluation across dozens of networks spanning vision, object detection, segmentation, natural language modeling, and translation.

Sparse Matrix Multiply-Accumulate (MMA) Operations

A100's new Sparse MMA instructions skip the compute on entries that have zero values, resulting in a doubling of the Tensor Core compute throughput. For example, in Figure 13 Below, Matrix A is a Sparse matrix with 50% sparsity following the required 2:4 structured pattern and Matrix B is a dense matrix of half the size. A standard MMA operation would not skip the zero values and would compute the result for the entire 16x8x16 matrix multiply in N cycles. Using a Sparse MMA instruction, only the elements in each row of Matrix A that have a non-zero value are matched with the corresponding elements from Matrix B. This transforms the computation into a smaller matrix multiply that takes just N/2 cycles, a 2x speedup.



Example Dense MMA and Sparse MMA operations using 16x16 sparse matrix (Matrix A), multiplied by a dense 16x8 matrix (Matrix B). Sparse MMA operation on right doubles throughput by skipping compute of zero values

Figure 13. Example Dense MMA and Sparse MMA operations

Combined L1 Data Cache and Shared Memory

First introduced in Volta V100, the NVIDIA combined L1 data cache and shared memory subsystem architecture significantly improves performance, while also simplifying programming and reducing the tuning required to attain at or near-peak application performance. Combining data cache and shared memory functionality into a single memory block provides the best overall performance for both types of memory accesses. The combined capacity of the L1 data cache and shared memory is 192 KB/SM in A100 versus 128 KB/SM in V100.

L1 cache integration within the shared memory block ensures the L1 cache has low latency and high bandwidth. The L1 functions as a high-throughput conduit for streaming data while simultaneously providing high-bandwidth and low-latency access to frequently reused data—the best of both worlds. The A100's larger L1/shared memory subsystem further amplifies performance of applications that use the L1 data cache when accessing device memory, allowing performance levels approaching that of using and explicitly managing fast shared memory. (See the NVIDIA Tesla V100 Whitepaper for examples on how a combined L1 data

cache and shared memory subsystem allows L1 cache operations to attain the benefits of shared memory performance.)

Simultaneous Execution of FP32 and INT32 Operations

Similar to Tesla V100 and Turing GPUs, the A100 SM also includes separate FP32 and INT32 cores, allowing simultaneous execution of FP32 and INT32 operations at full throughput, while also increasing instruction issue throughput. Many applications have inner loops that perform pointer arithmetic (integer memory address calculations) combined with floating-point computations that will benefit from simultaneous execution of FP32 and INT32 instructions. Each iteration of a pipelined loop can update addresses (INT32 pointer arithmetic) and load data for the next iteration while simultaneously processing the current iteration in FP32.

A100 HBM2 and L2 Cache Memory Architectures

The design of a GPU's memory architecture and hierarchy is critical to application performance, and impacts GPU size, cost, power usage, and programmability. Many different memory subsystems exist in a GPU, from the large complement of off-chip DRAM (frame buffer) device memory, to varying levels and types of on-chip memories, to the register files used in computations in the SM. High-performance HBM2 is the DRAM technology used in the A100 GPU.

The global and local memory areas accessed by CUDA programs reside in HBM2 memory space, and is referred to as device memory in CUDA parlance. Constant memory space resides in device memory and is cached in the constant cache. Texture and surface memory spaces reside in device memory and are cached in texture cache. The L2 cache caches reads from and writes to HBM2 (device) memory. HBM2 and L2 memory spaces are accessible to all SMs and all applications running on the GPU.

A100 HBM2 DRAM Subsystem

As HPC, AI, and analytics datasets continue to grow and problems looking for solutions get increasingly complex, more GPU memory capacity and higher memory bandwidth is a necessity. Tesla P100 was the world's first GPU architecture to support the high-bandwidth HBM2 memory technology, while Tesla V100 provided a faster, more efficient, and higher capacity HBM2 implementation. A100 raises the bar yet again on HBM2 performance and capacity.

HBM2 memory is composed of memory stacks located on the same physical package as the GPU, providing substantial power and area savings compared to traditional GDDR5/6 memory designs, allowing more GPUs to be installed in systems. Fundamental details of HBM2 technology are included in our Pascal Architecture Whitepaper.

The A100 GPU includes **40 GB of fast HBM2 DRAM memory** on its SXM4-style circuit board. The memory is organized as five active HBM2 stacks with eight memory dies per stack. With a 1215 MHz (DDR) data rate the A100 HBM2 delivers **1555 GB/sec memory bandwidth**, which is more than 1.7x higher than Tesla V100 memory bandwidth.

ECC Memory Resiliency

The A100 HBM2 memory subsystem supports Single-Error Correcting Double-Error Detecting (SECDED) Error Correction Code (ECC) to protect data. ECC provides higher reliability for compute applications that are sensitive to data corruption. It is especially important in large-scale cluster computing environments where GPUs process very large datasets and/or run applications for extended periods. Other key memory structures in A100 are also protected by SECDED ECC including the L2 cache and the L1 caches and register files inside all the SMs.

A100 L2 Cache

The A100 GPU in the A100 Tensor Core GPU includes 40 MB of L2 cache, which is 6.7x larger than Tesla V100 L2 cache. The substantial increase in L2 cache size significantly improves performance of many HPC and AI workloads because larger portions of datasets and models can now be cached and repeatedly accessed at much higher speed than reading from and writing to HBM2 memory. Some workloads that are limited by DRAM bandwidth will benefit from the larger L2 cache, such as deep neural networks using small batch sizes.

The A100 L2 cache is divided into two partitions to enable higher bandwidth and lower latency memory access. Each L2 partition localizes and caches data for memory accesses from SMs in the GPCs directly connected to the partition. This structure enables A100 to deliver a 2.3x L2 bandwidth increase over V100. Hardware cache-coherence maintains the CUDA programming model across the full GPU, and applications will automatically leverage the bandwidth and latency benefits of A100's new L2 cache.

L2 cache is a shared resource for the GPCs and SMs and lies outside of the GPCs. The L2 cache is divided into two partitions to enable higher bandwidth and lower latency memory access. Each L2 partition localizes and caches data for memory accesses from SMs in the GPCs directly connected to the partition. This structure enables A100 to deliver a 2.3x L2 bandwidth increase over V100. Hardware cache-coherence maintains the CUDA programming model across the full GPU, and applications will automatically leverage the bandwidth and latency benefits of A100's new L2.

Each L2 cache partition is divided into 40 L2 cache slices. Eight 512 KB L2 slices are associated with each memory controller. As mentioned in the MIG section below, an L2 slice group composed of 10 L2 cache slices is included in each GPU slice in a GPU Instance of a MIG configuration. The A100 L2 read bandwidth is 5120 Bytes/clk, compared to V100 L2 cache read bandwidth of 2048 Bytes/clk.

The NVIDIA Ampere architecture provides L2 cache residency controls for the programmer to manage data to keep or evict from the cache (see the **CUDA Advances for NVIDIA Ampere Architecture GPUs** section below for more details).

The NVIDIA Ampere architecture adds Compute Data Compression to accelerate unstructured sparsity and other compressible data patterns. Compression in L2 provides up to 4x improvement to DRAM read/write bandwidth, up to 4x improvement in L2 read bandwidth, and up to 2x improvement in L2 capacity.

Table 4. Comparison of NVIDIA Data Center GPUs

GPU Features	NVIDIA Tesla P100	NVIDIA Tesla V100	NVIDIA A100
GPU Codename	GP100	GV100	GA100
GPU Architecture	NVIDIA Pascal	NVIDIA Volta	NVIDIA Ampere
GPU Board Form Factor	SXM	SXM2	SXM4
SMs	56	80	108
TPCs	28	40	54
FP32 Cores / SM	64	64	64
FP32 Cores / GPU	3584	5120	6912
FP64 Cores / SM (excl. Tensor)	32	32	32
FP64 Cores / GPU (excl. Tensor)	1792	2560	3456
INT32 Cores / SM	NA	64	64
INT32 Cores / GPU	NA	5120	6912
Tensor Cores / SM	NA	8	4[2]
Tensor Cores / GPU	NA	640	432
GPU Boost Clock	1480 MHz	1530 MHz	1410 MHz
Peak FP16 Tensor TFLOPS with FP16 Accumulate[1]	NA	125	312/624[3]
Peak FP16 Tensor TFLOPS with FP32 Accumulate[1]	NA	125	312/624[3]
Peak BF16 Tensor TFLOPS with FP32 Accumulate[1]	NA	NA	312/624[3]
Peak TF32 Tensor TFLOPS[1]	NA	NA	156/312[3]
Peak FP64 Tensor TFLOPS[1]	NA	NA	19.5
Peak INT8 Tensor TOPS[1]	NA	NA	624/1248[3]
Peak INT4 Tensor TOPS[1]	NA	NA	1248/2496[3]
Peak FP16 TFLOPS[1] (non-Tensor)	21.2	31.4	78
Peak BF16 TFLOPS[1] (non-Tensor)	NA	NA	39
Peak FP32 TFLOPS[1] (non-Tensor)	10.6	15.7	19.5
Peak FP64 TFLOPS[1] (non-Tensor)	5.3	7.8	9.7
Peak INT32 TOPS[1]	NA	15.7	19.5
Texture Units	224	320	432
Memory Interface	4096-bit HBM2	4096-bit HBM2	5120-bit HBM2
Memory Size	16 GB	32 GB / 16 GB	40 GB
Memory Data Rate	703 MHz DDR	877.5 MHz DDR	1215 MHz DDR

Memory Bandwidth	720 GB/sec	900 GB/sec	1555 GB/sec
L2 Cache Size	4096 KB	6144 KB	40960 KB
Shared Memory Size / SM	64 KB	Configurable up to 96 KB	Configurable up to 164 KB
Register File Size / SM	256 KB	256 KB	256 KB
Register File Size / GPU	14336 KB	20480 KB	27648 KB
TDP	300 Watts	300 Watts	400 Watts
Transistors	15.3 billion	21.1 billion	54.2 billion
GPU Die Size	610 mm²	815 mm²	826 mm2
TSMC Manufacturing Process	16 nm FinFET+	12 nm FFN	7 nm N7
1. Peak rates are based on GPU Boost Clock. **2. Four Tensor Cores in an A100 SM have 2x the raw FMA computational power of eight Tensor Cores in a GV100 SM.** **3. Effective TOPS / TFLOPS using the new Sparsity Feature**			

Note: Because the A100 Tensor Core GPU is designed to be installed in high-performance servers and data center racks to power AI and HPC compute workloads, it does not include display connectors, NVIDIA RT Cores for ray tracing acceleration, or an NVENC encoder.

Maximizing Tensor Core Performance and Efficiency for Deep Learning Applications

As discussed above, the NVIDIA Tensor Core was first introduced in the NVIDIA Volta GPU architecture to deliver a significant performance speedup for matrix multiplication operations that are common in neural network training and inferencing. Tensor Cores on Volta supported an 8x peak speedup for mixed-precision matrix multiplication compared to standard FP32 precision operations.

Compared to Tesla V100, the NVIDIA Ampere architecture-based A100 GPU has more SMs (108 vs 80) with third-generation Tensor Cores capable of larger Tensor operations. The Tensor Cores in the A100 GPU support peak mixed-precision compute performance that is 16x higher than standard FP32 FMA operations.

Several new features and optimizations were introduced in the NVIDIA Ampere architecture that enhance utilization of the Tensor Cores, improve programmability, reduce software complexity, reduce memory bandwidth usage, and reduce latency and other overheads.

Strong Scaling Deep Learning Performance

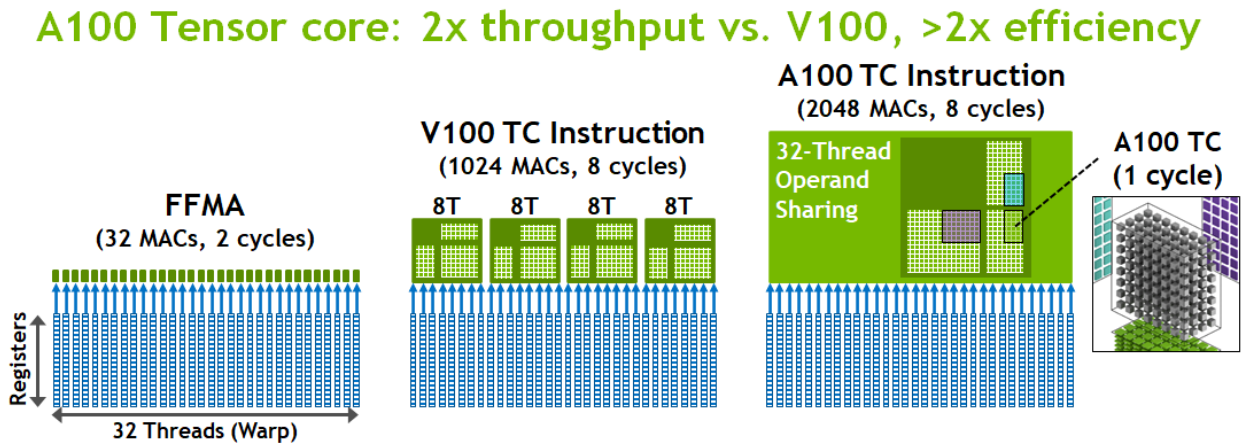
Deep learning requires massive compute resources, but the parallelism is broken up into small sequentially dependent chunks of work. A typical deep neural network consists of long chains of

interconnected layers. Each layer performs an operation similar to General Matrix Multiplication (GEMM) by taking a matrix of input values and multiplying it with a matrix of weights to create an output matrix. The output matrix then typically goes through some activation math before being sent to the next layer of the network. The output matrix of each GEMM is broken down into smaller tiles which map across the multiple SMs in the GPU.

The NVIDIA Ampere architecture targets strong scaling to deliver speedups on existing deep neural networks. Weak scaling is an easier target where the workload parallelism must grow to leverage the faster architecture. With strong scaling the workload per GPU is fixed from one architecture to the next. In the context of deep learning, this means that the GEMM tile size per SM must not grow even while the A100 Tensor Cores consume data 2.5x faster than V100. The NVIDIA Ampere architecture implemented several features and optimizations, described below, to deliver data to the Tensor Cores at a faster rate and more efficiently.

New NVIDIA Ampere Architecture Features Improved Tensor Core Performance

Data sharing improvements - The NVIDIA Ampere architecture third-generation Tensor Core allows data to be shared across all 32 threads in a warp, compared to 8 threads on Volta's Tensor Core. Sharing data across more threads reduces the register file bandwidth for feeding data to the Tensor Cores. It also reduces the amount of redundant data loaded into the register files from shared memory (SMEM), which saves both bandwidth and register file storage. To further improve efficiency, A100 Tensor Core instructions increase the k dimension of the matrix multiply per instruction by up to 4x relative to V100. Overall, when computing matrix multiply operations A100 issues 8x fewer instructions and performs 2.9x fewer register file accesses than V100.

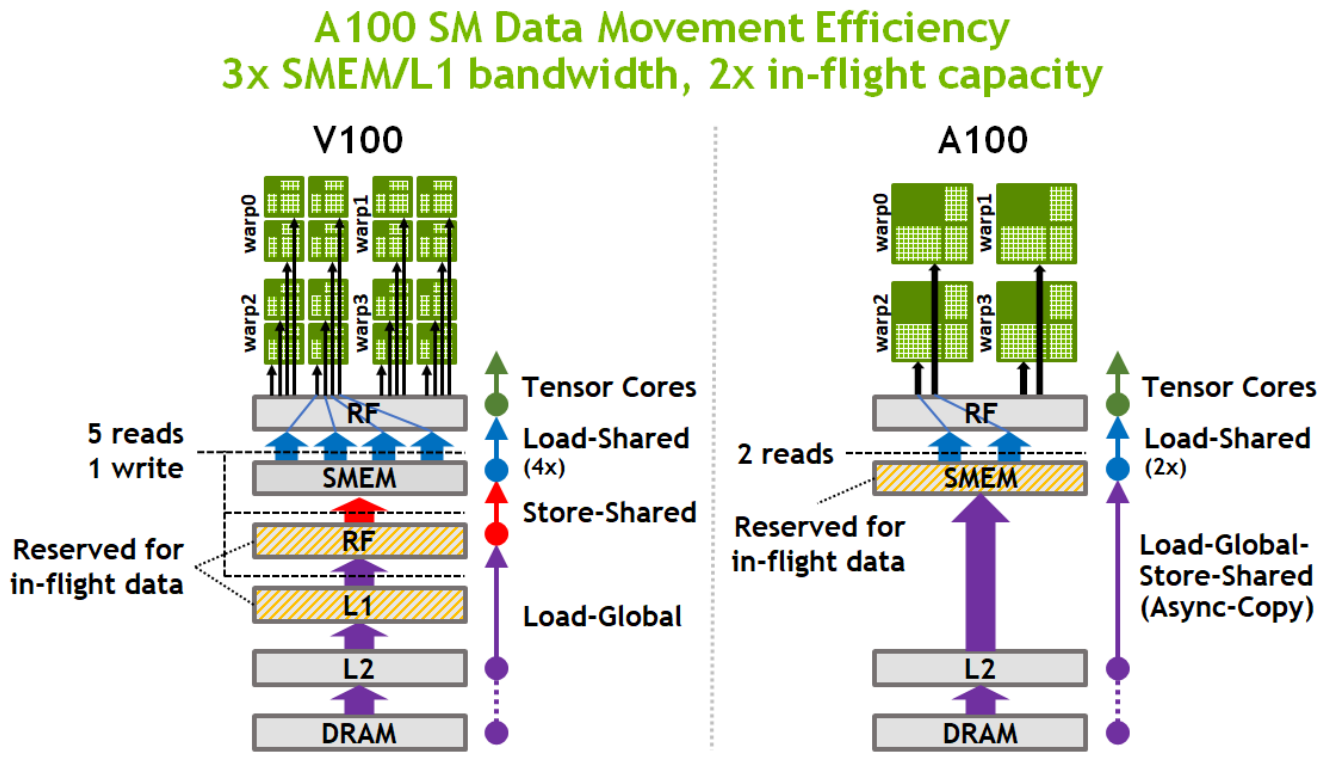


16x16x16 matrix multiply	FFMA	V100 TC	A100 TC	A100 vs. V100 (improvement)	A100 vs. FFMA (improvement)
Thread sharing	1	8	32	4x	32x
Hardware instructions	128	16	2	8x	64x
Register reads+writes (warp)	512	80	28	2.9x	18x
Cycles	256	32	16	2x	16x

A100's Tensor Core increases thread sharing by 4x over V100. For a 16x16x16 matrix multiply, A100's enhanced 16x8x16 Tensor Core (TC) instructions improve on V100 by reducing register accesses from 80 to 28, and hardware instructions issued from 16 to 2. Cycle counts are per SM partition. Note: Each V100 8x8x4 TC instruction (CUDA warp-level instruction) is translated into four lower-level MMA hardware instructions.

Figure 14. A100 Tensor Core Throughput and Efficiency

Data Fetch Improvements - NVIDIA Ampere architecture includes a new asynchronous copy instruction that loads data directly from global memory (typically from L2 cache and DRAM) into SM shared memory. On Volta, data was first loaded through L1 cache into the register file with load-global instructions, then transferred from the register file to shared memory with store-shared instructions, and finally loaded from shared memory into registers of multiple threads and warps with load-shared instructions. The new load-global-store-shared asynchronous copy instruction in NVIDIA Ampere architecture GPUs saves SM internal bandwidth by avoiding a roundtrip through the register file, and also eliminates the need to allocate register file storage for the in-flight data transfers. More details on the asynchronous copy instruction are provided later in this paper.



A100 improves SM bandwidth efficiency with a new load-global-store-shared asynchronous copy instruction that bypasses L1 cache and register file (RF). Additionally, A100's more efficient Tensor Cores reduce shared memory (SMEM) loads.

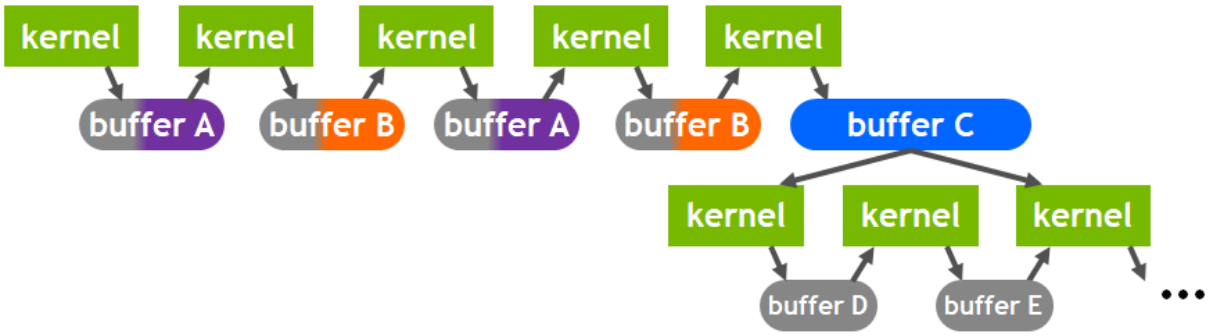
Figure 15. A100 SM Data Movement Efficiency

New asynchronous barriers work together with the asynchronous copy instruction to enable efficient data fetch pipelines, and A100 increases maximum SMEM allocation per SM 1.7x to 164 KB (vs 96 KB on V100). With these improvements A100 SMs continuously data stream data to keep the L2 cache constantly utilized.

L2 Cache and DRAM Bandwidth improvements - The NVIDIA A100 GPU's increased number of SMs and more powerful Tensor Cores in turn increase the required data fetch rates from DRAM and L2 cache. To feed the Tensor Cores, A100 implements a 5-site HBM2 memory subsystem with bandwidth of 1555 GB/sec, over 1.7x faster than V100. A100 further provides 2.3x the L2 cache read bandwidth of V100.

Alongside the raw data bandwidth improvements, A100 improves data fetch efficiency and reduces DRAM bandwidth demand with a 40 MB L2 cache that is almost 7x larger than that of Tesla V100. To fully exploit the L2 capacity A100 includes improved cache management controls. Optimized for neural network training and inferencing as well as general compute workloads, the new controls ensure that data in the cache is used more efficiently by minimizing writebacks to memory and keeping reused data in L2 to reduce redundant DRAM traffic.

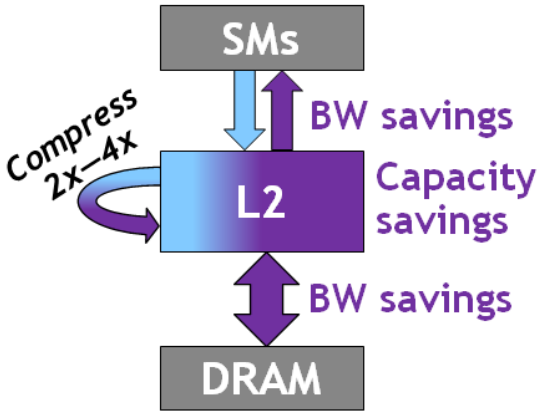
For example, for DL inferencing workloads, ping-pong buffers can be persistently cached in the L2 for faster data access, while also avoiding writebacks to DRAM. For producer-consumer chains, such as those found in DL training, L2 cache controls can optimize caching across the write-to-read data dependencies. In LSTM networks, recurrent weights that are shared across multiple GEMM operations can be preferentially cached and reused in L2.



A100 L2 cache residency controls help applications reduce DRAM bandwidth. This example shows different data buffers highlighted with colors to indicate data that has been marked for persistent caching in L2.

Figure 16. A100 L2 cache residency controls

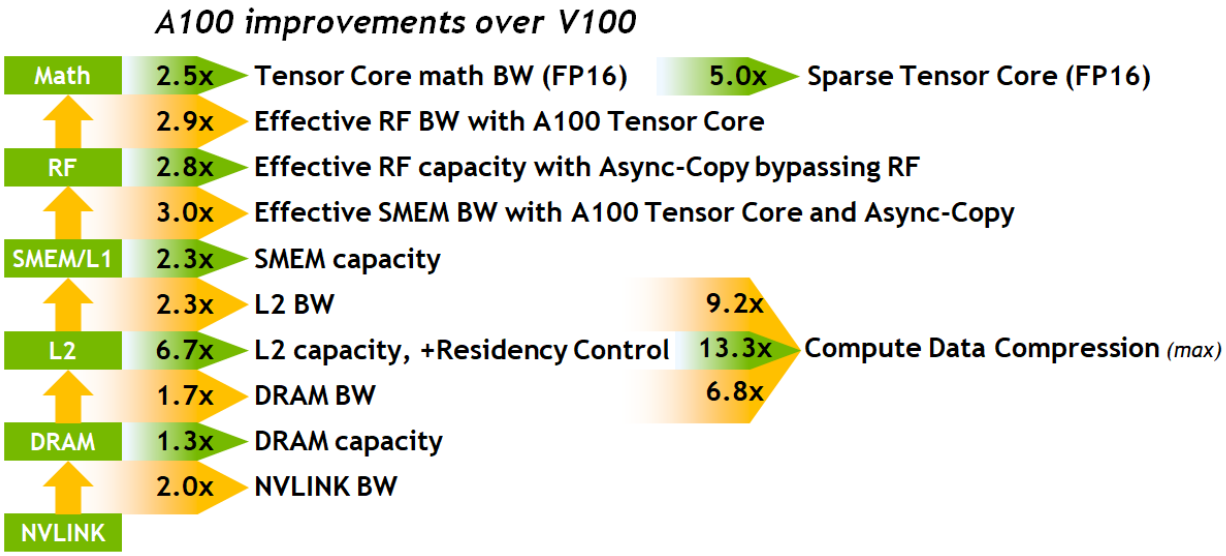
Compression - To boost efficiency and enhance strong scaling, A100 adds Compute Data Compression. Compression saves up to 4x DRAM read/write bandwidth, up to 4x L2 read bandwidth, and up to 2x L2 capacity.



A100 Compute Data Compression improves DRAM bandwidth, L2 bandwidth, and L2 capacity.

Figure 17. A100 Compute Data Compression

Summary - Figure 18 below summarizes the improvements that A100 delivers across all levels of the compute and memory hierarchy. These innovations enable A100 to strong scale deep learning to unprecedented levels of performance.



A100 strong-scaling innovations and improvements over V100 across the compute and memory hierarchy.

Figure 18. A100 strong-scaling innovations

Compute Capability

The A100 GPU supports the new Compute Capability 8.0. Table 5 compares the parameters of different Compute Capabilities for NVIDIA GPU architectures.

Table 5. Compute Capability: GP100 vs GV100 vs GA100

GPU Features	NVIDIA Tesla P100	NVIDIA Tesla V100	NVIDIA A100
GPU Codename	GP100	GV100	GA100
GPU Architecture	NVIDIA Pascal	NVIDIA Volta	NVIDIA Ampere
Compute Capability	6.0	7.0	8.0
Threads / Warp	32	32	32
Max Warps / SM	64	64	64
Max Threads / SM	2048	2048	2048
Max Thread Blocks / SM	32	32	32
Max 32-bit Registers / SM	65536	65536	65536
Max Registers / Block	65536	65536	65536
Max Registers / Thread	255	255	255
Max Thread Block Size	1024	1024	1024
FP32 Cores / SM	64	64	64
Ratio of SM Registers to FP32 Cores	1024	1024	1024
Shared Memory Size / SM	64 KB	Configurable up to 96 KB	Configurable up to 164 KB

MIG (Multi-Instance GPU) Architecture

While many data center workloads continue to scale, both in size and complexity, some acceleration tasks aren't as demanding, such as early-stage development or inference on simple models at low batch sizes. Data center managers aim to keep resource utilization high, so an ideal data center accelerator doesn't just go big- it also efficiently accelerates many smaller workloads.

Background

In 2017, the NVIDIA Tesla V100 GPU introduced hardware accelerated Multi Process Server (MPS) support, which allowed multiple applications to simultaneously execute on separate GPU execution resources (SMs).

Using Volta MPS for deep learning inference applications, versus traditional GPU work submission methods, delivered much higher throughput and lower latency, permitting many individual inference jobs to be submitted concurrently to the GPU with improved overall GPU utilization. (See the NVIDIA Tesla V100 GPU Architecture Whitepaper for more details on Volta MPS.)

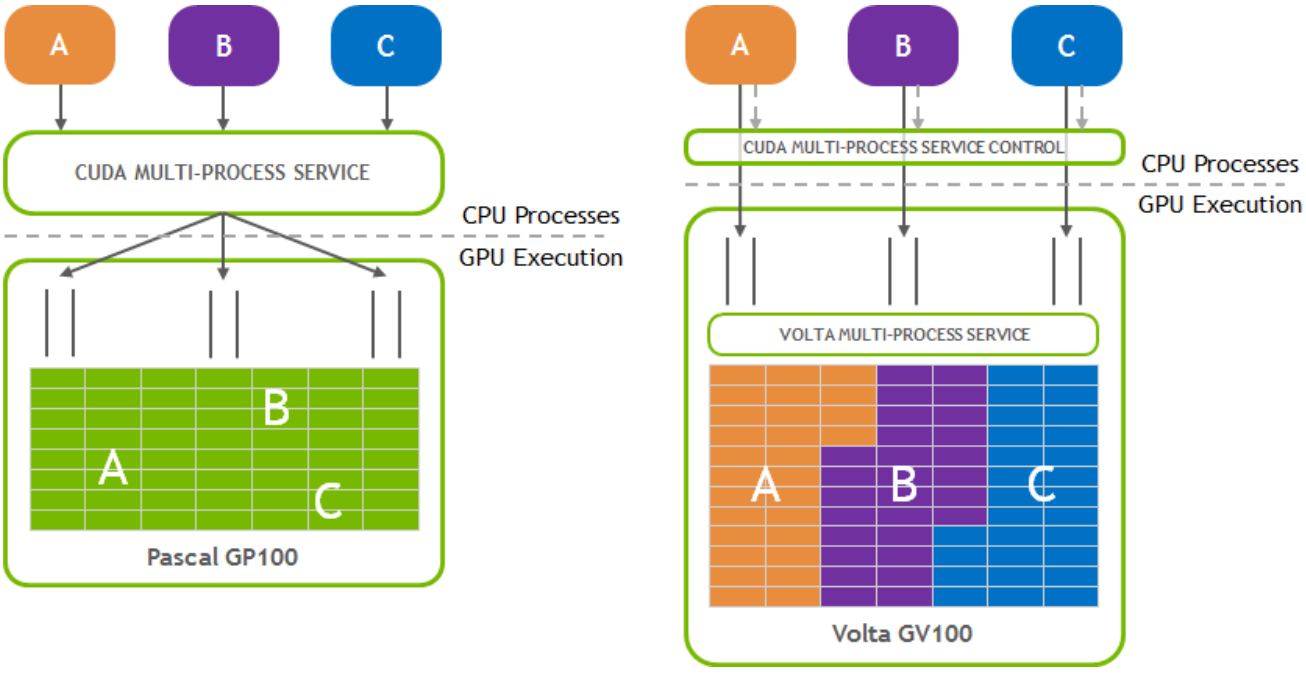


Figure 19. Software-based MPS in Pascal vs Hardware-Accelerated MPS in Volta

However, because memory system resources were shared across all the applications, one application could interfere with the others if it had high demands for DRAM bandwidth or its requests oversubscribed the L2 cache. Volta MPS, which remains fully supported on Ampere,

was designed for sharing the GPU among applications from a single user, but not for multi-user or multi-tenant use cases.

MIG Capability of NVIDIA Ampere GPU Architecture

The new MIG feature can partition each A100 into as many as seven GPU Instances for optimal utilization, effectively expanding access to every user and application.

The A100 GPU new MIG capability can divide a single GPU into multiple GPU partitions called GPU Instances. Each instance's SMs have separate and isolated paths through the entire memory system — the on-chip crossbar ports, L2 cache banks, memory controllers and DRAM address busses are all assigned uniquely to an individual instance. This ensures that an individual user's workload can run with predictable throughput and latency, with the same L2 cache allocation and DRAM bandwidth even if other tasks are thrashing their own caches or saturating their DRAM interface.

Using this capability, MIG can partition available GPU compute resources to provide a defined quality of service (QoS) with fault isolation for different clients (such as VMs, containers, processes, and so on). It enables multiple GPU Instances to run in parallel on a single, physical A100 GPU. MIG also keeps the CUDA programming model unchanged to minimize programming effort.

CSPs can use MIG to raise utilization rates on their GPU servers, delivering up to 7x more GPU Instances at no additional cost. MIG supports the necessary QoS and isolation guarantees needed by CSPs to ensure that one client (VM, container, process) cannot impact the work or scheduling from another client.

CSPs often partition their hardware based on customer usage patterns. Effective partitioning only works if hardware resources are providing consistent bandwidth, proper isolation, and good performance during runtime.

With NVIDIA Ampere architecture-based GPU, users will be able to see and schedule jobs on their new virtual GPU Instances as if they were physical GPUs. MIG works with Linux operating systems and their hypervisors. Users can run containers with MIG using runtimes such as Docker Engine, with support for container orchestration using Kubernetes coming soon.

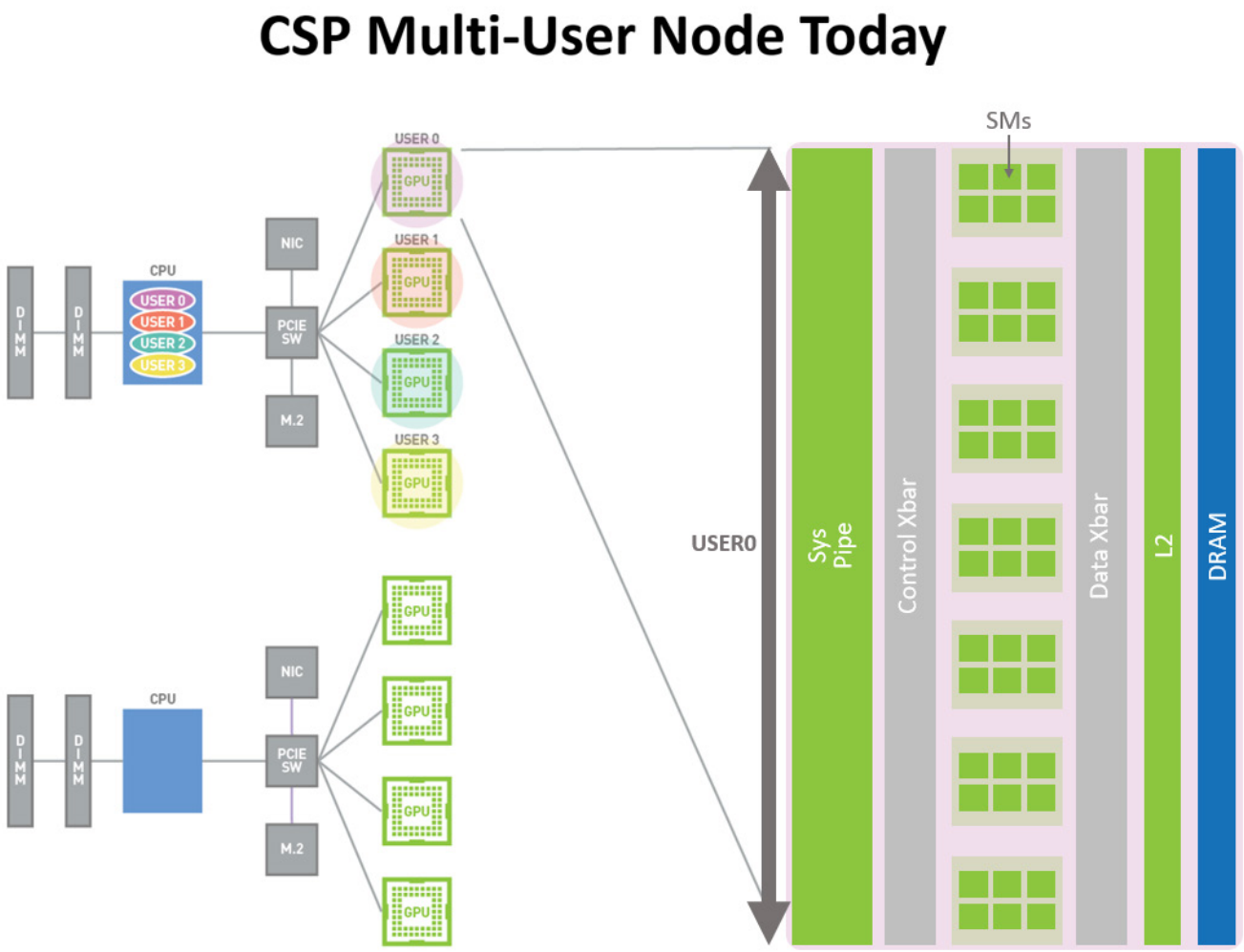
Important Use Cases for MIG

An important use case for MIG, called "Multi-Tenant" (leveraged using NVIDIA's vGPU technology), can be used by CSPs to rent out separate GPU Instances to different customers. Applications running in each GPU Instance are isolated and protected from faults occurring in applications running simultaneously in other GPU Instances. Data protection, fault isolation, and QoS is mandatory in such use cases.

Another MIG use case called "Single Tenant, Single User" can support a single user using a single workstation to run multiple GPU-based applications, where fault isolation between applications is critical. Additionally, a "Single-Tenant, Multi-User" scenario can be useful for a

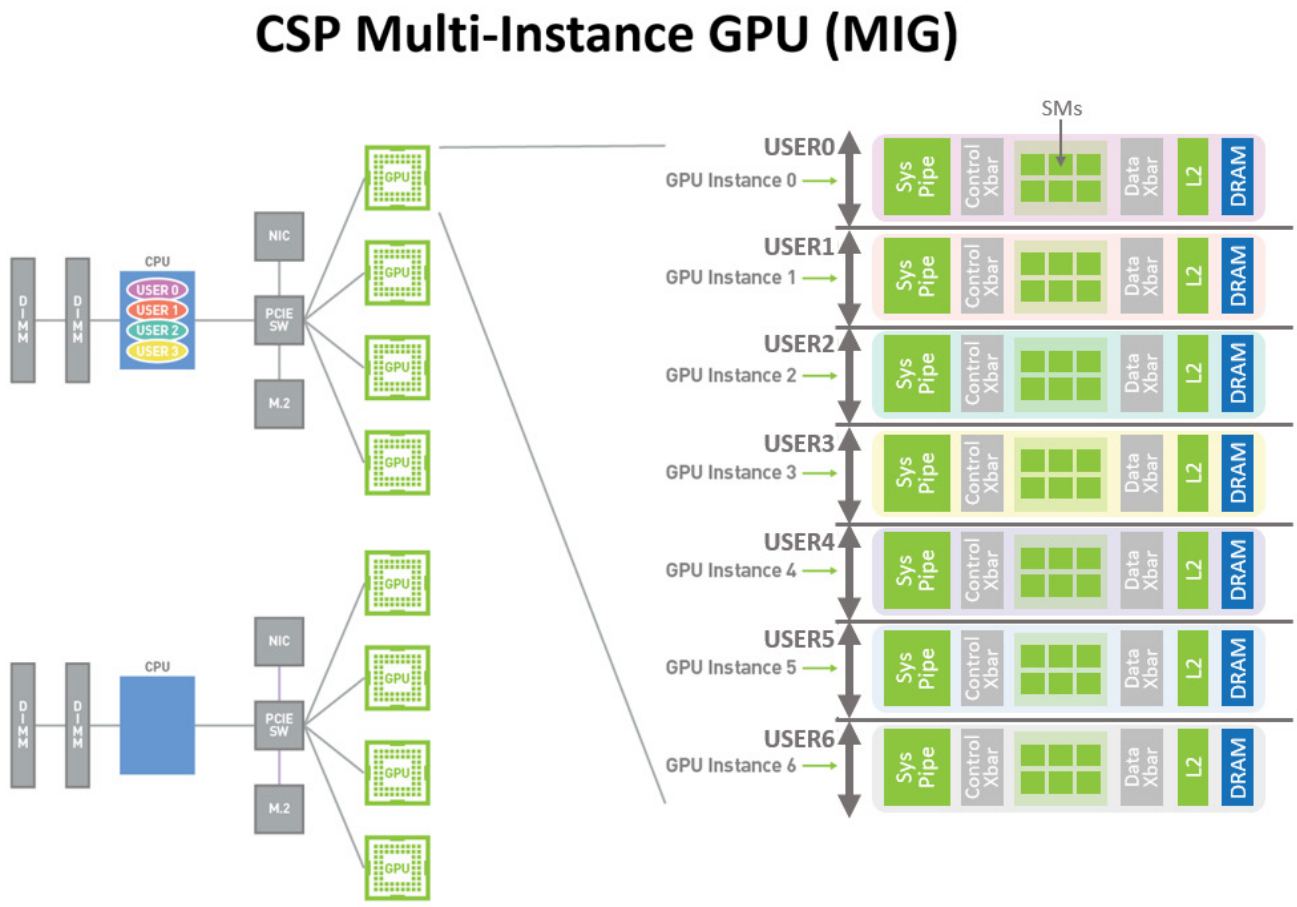
company to support internal workgroups, or to provide services to multiple external users, such as AI Inference services, or various other types of GPU-accelerated services.

MIG allows compute resources to be partitioned across different Virtual Machines (VMs), and allows multiple VMs to execute simultaneously while maintaining fault isolation. Consistent performance can be maintained even if a VM is migrated to another GPU. In addition, better utilization of GPUs can be obtained by packing multiple VMs on the same GPU.



CSP Multi-user node today (pre-A100) shows how accelerated GPU instances are available for usage only at full physical GPU granularity for users in different organizations, even if the user applications don't require a full GPU.

Figure 20. CSP Multi-user node Today



This CSP MIG diagram shows how multiple independent users from the same or different organizations can be assigned their own dedicated, protected, and isolated GPU Instance within a single physical GPU. (See MIG configuration and GPU partitioning details below).

Figure 21. Example CSP MIG Configuration

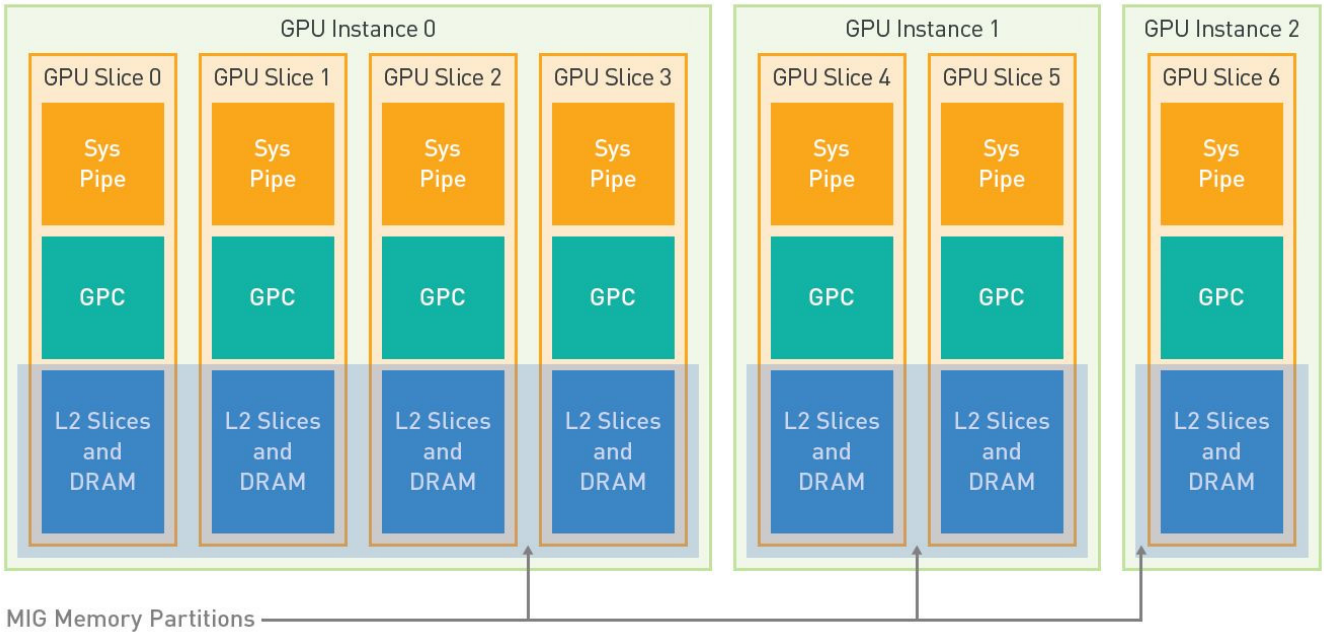
MIG Architecture and GPU Instances in Detail

Creating GPU Instances can be thought of as splitting one big GPU into multiple smaller GPUs, with each GPU Instance having dedicated compute and memory resources. Each GPU Instance behaves like a smaller, fully capable independent GPU that includes a predefined number of GPCs, SMs, L2 cache slices, memory controllers, and frame buffer memory.

A GPU Instance is constructed from multiple “GPU slices”, where each GPU slice includes a “Sys Pipe” (defined below), one GPC, one L2 slice group (an L2 slice group includes 10 L2 cache slices), and access to a portion of frame buffer memory. The A100 GPU supports a total of 7 GPU slices. Note: In MIG operating mode, the single GPC in each GPU slice has seven TPCs (14 SMs) enabled, which allows all GPU slices to have the same consistent compute performance.

The Sys Pipe, which is part of the new A100 GigaThread™ Engine, is a unit that communicates with the host CPU and schedules work to a GPC (and its SMs) in the GPU slice. The A100 Tensor Core GPU includes seven total Sys Pipes to support MIG. One A100 Sys Pipe is similar to prior GPU architectures, supporting both Graphics and Compute work, and the six additional new Sys Pipes support Compute-only workloads. When operating in Graphics Mode, A100 behaves similar to past GPUs using the single graphics-capable Sys Pipe with a single graphics context. When in Compute Mode, all seven Sys Pipes can run multiple Compute contexts simultaneously. Note that graphics pipeline operations are not supported when the A100 GPU is in MIG mode. MIG is a Compute Mode-only feature.

A GPU Memory slice is another MIG structure that includes all the L2 slice groups (blocks of L2 cache slices) and associated frame buffer memory contained across all the GPU slices in a GPU Instance. Application contexts running in one GPU Instance will not use the L2 slices of another GPU Instance, which effectively isolates and proportions memory bandwidth used across the different GPU Instances.



(Note that unit sizes depicted do not correspond to actual physical area on the GPU die.)

Figure 22. Example MIG compute configuration with three GPU Instances.

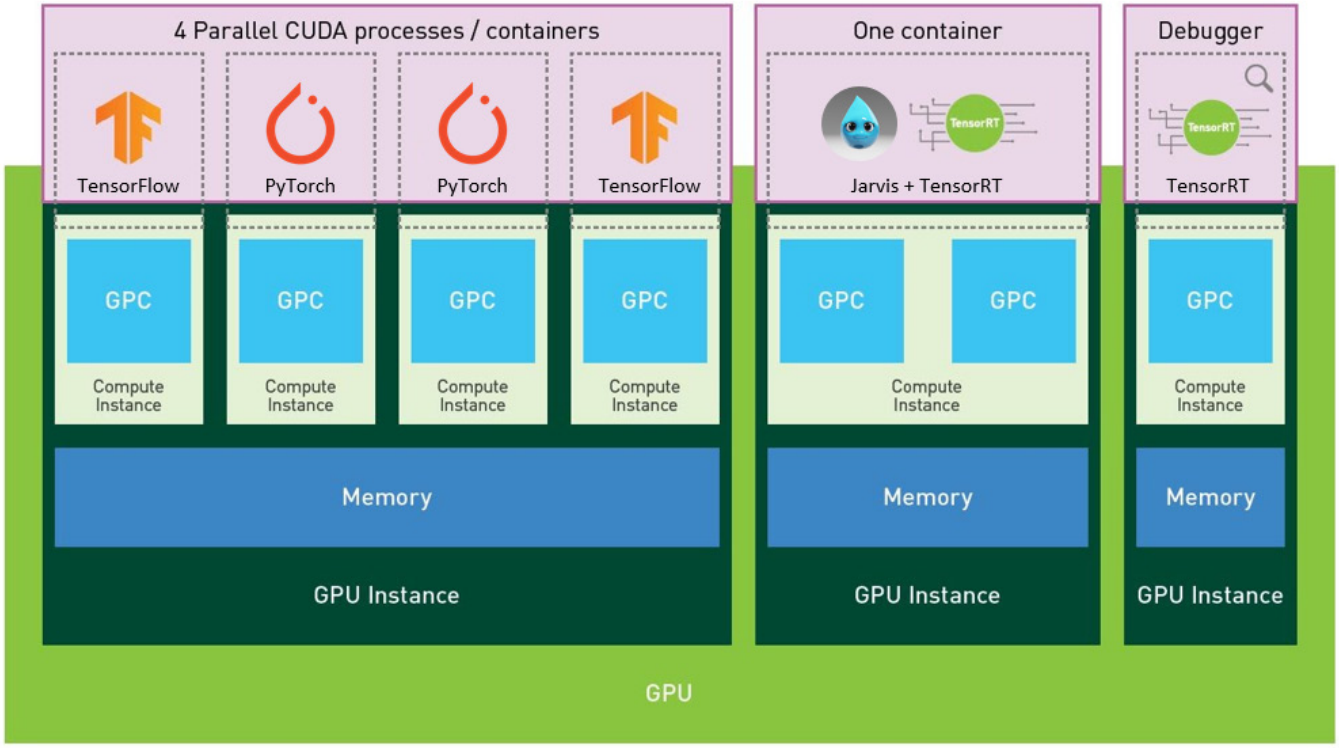
A single GPU Instance provides memory QoS to all client applications running on the Instance. Multiple Instances proportionately share GPU frame buffer memory.

Variable numbers of GPU slices can be statically assigned to each GPU Instance to support the desired partitioning of computational and memory bandwidth resources, in addition to improving QoS, fault isolation, error containment, and error recovery.

Compute Instances

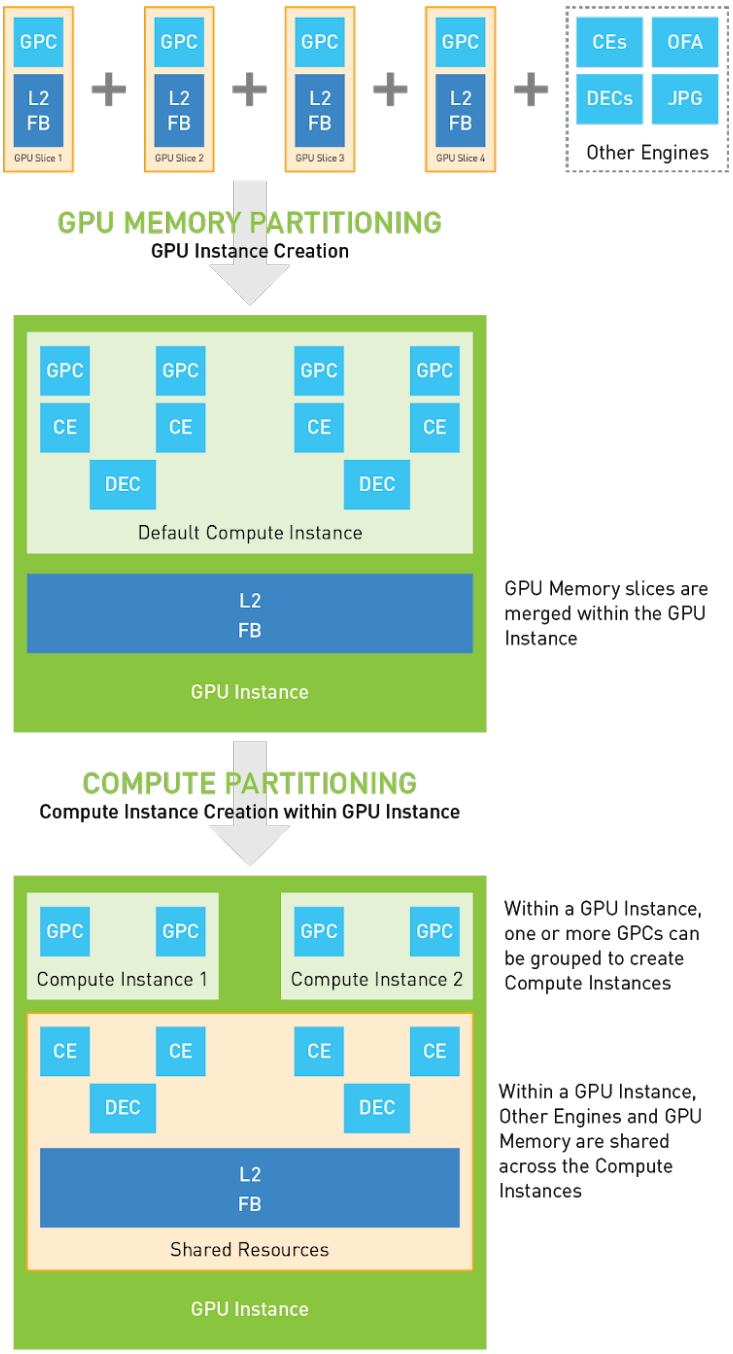
A “Compute Instance” is another grouping that can configure different levels of Compute power created within a GPU Instance, encapsulating all the compute resources (number of GPCs, Copy Engines, NVDEC units, etc.) that can execute work in the GPU Instance. By default, a single Compute Instance is created under each GPU Instance, exposing all the GPU compute resources available within the GPU Instance. A GPU Instance can be subdivided into multiple smaller Compute Instances to further split its compute resources.

Compute Instances each support Volta-style MPS functionality, where multiple different CPU processes (host application contexts) can be combined into a single CUDA context and run on the GPU. The maximum number of MPS clients is proportional to Compute Instance size. MPS is fully supported in A100 and is particularly important for HPC use cases that demand the throughput of MPS for MPI.



Example of multiple independent GPU Compute workloads running in parallel using a MIG configuration on an A100 GPU with three GPU Instances and variable numbers of Compute Instances within each GPU Instance.

Figure 23. MIG Configuration with multiple independent GPU Compute workloads

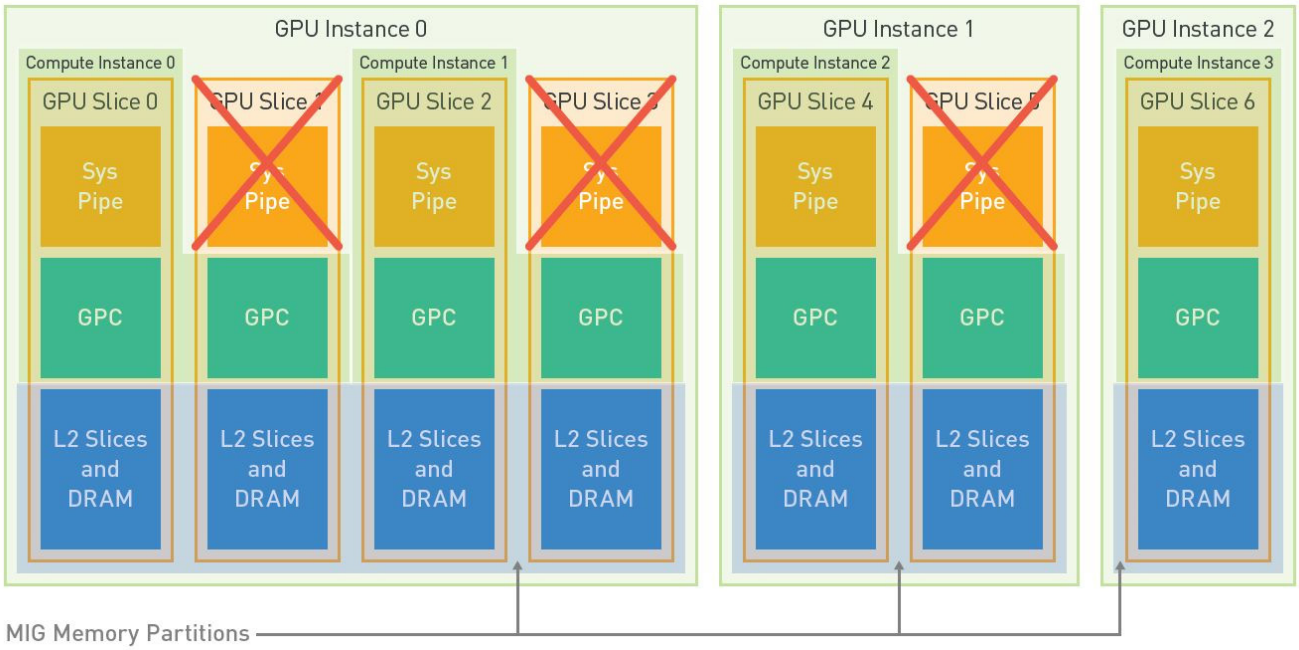


Shows the creation of a four GPU slice GPU Instance, followed by creating two Compute Instances each with two GPCs. The Copy Engines (CE) and NVDEC decoders (DEC) are also depicted.

Figure 24. Example MIG partitioning process

Compute Instances Enable Simultaneous Context Execution

Compute Instances are what enable multiple contexts to run simultaneously on the GPU. A single Compute Instance can encompass one or more GPU slices, but a Compute Instance can also be configured to allow a single Sys Pipe to connect to multiple GPCs, L2 slices, and memory from other GPU slices. In such cases, Sys Pipes from other GPU slices in the GPU Instance are disabled.



GPU Instance 0 has two Compute Instances, each with one Sys Pipe. (Note that unit sizes depicted do not correspond to actual physical area on the GPU die.)

Figure 25. Example MIG config with three GPU Instances and four Compute Instances.

Technically, a Compute Instance is defined as including one Sys Pipe with up to 7 GPCs within a GPU Instance. All applications sharing a Compute Instance share a single Sys Pipe, and each Compute Instance can context switch separately from other Compute Instances. Prior to A100, all GPCs context-switched together, but in A100, GPCs in different Compute Instances are context-switched separately, allowing each Sys Pipe to context switch with a subset of the GPCs, thereby allowing multiple Compute Instances to operate independently.

Note that each Compute Instance also permits Volta-style MPS functionality, where multiple different CPU processes (application contexts) can be combined into a single application context and run on the GPU.

In general, many MIG configurations are possible, with details being described in developer and system administrator documentation.

MIG Migration

An important MIG feature to manage, tune, service, and load-balance vGPU (virtual GPU) virtual machine (VM) configurations is the ability to migrate vGPUs between GPU Instances on a single GPU, and more frequently between different GPUs in a cluster. The migration process is conceptually straightforward. State information of the GPU slices of a vGPU within a GPU Instance are saved and then restored onto another GPU Instance with the same number of GPU slices.

When various GPUs in a cluster are only partially utilized, MIG migration allows moving and packing jobs onto fewer GPUs, reducing fragmentation, and often reducing the number of physical GPUs necessary to support a given number of vGPUs. This can free up certain GPUs to run larger jobs, or have the unused GPUs placed in power-saving mode to reduce data center costs. MIG migration also allows GPUs to be deloaded for servicing, without killing the jobs.

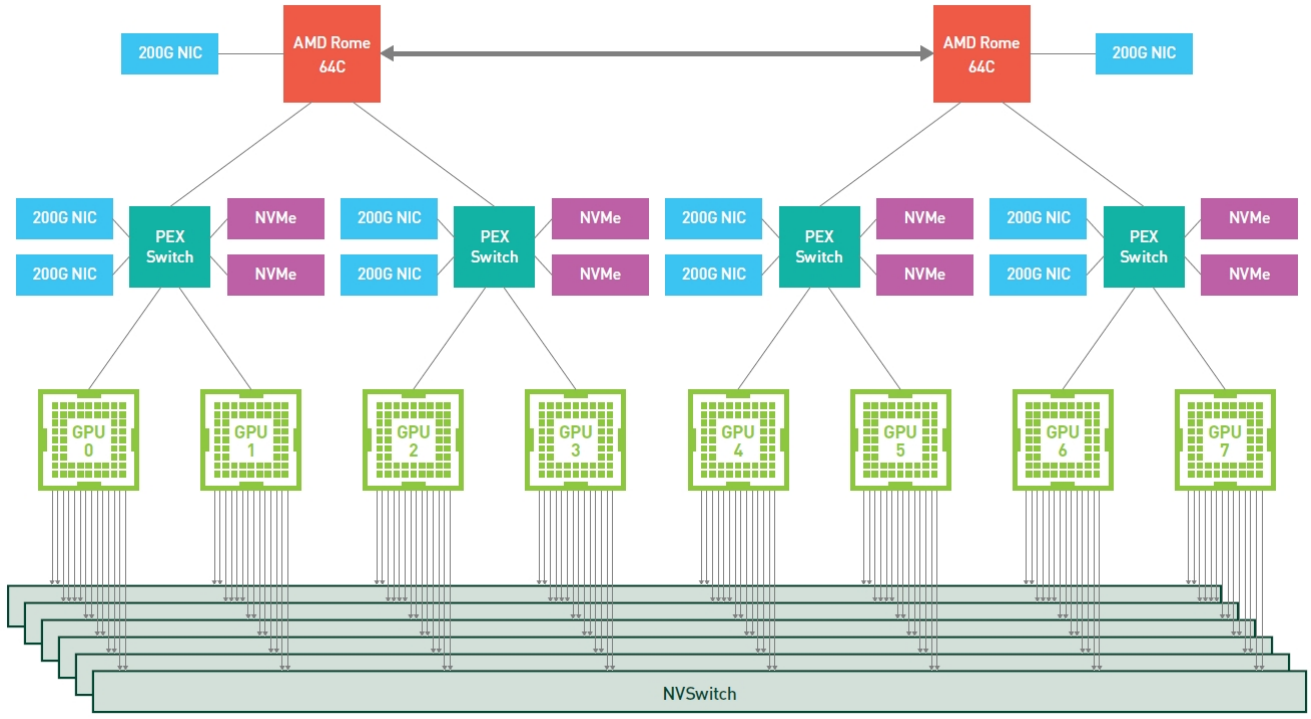
Third-Generation NVLink

The third-generation of NVIDIA's high-speed NVLink interconnect is implemented in the NVIDIA Ampere architecture-based A100 GPU and the new NVSwitch. NVLink is a lossless, high-bandwidth, low-latency shared memory interconnect, and includes resiliency features such as link-level error detection and packet replay mechanisms to guarantee successful transmission of data.

The new NVLink significantly enhances multi-GPU scalability, performance, and reliability with more links per GPU, much faster GPU-GPU communication bandwidth, and improved error-detection and recovery features. A100 GPUs can use NVLink links to access peer GPU memory at bandwidths much higher than achievable with PCI Express.

The new NVLink has a data rate of 50 Gbit/sec per signal pair, nearly doubling the 25.78 Gbits/sec rate in Tesla V100. Each link uses 4 differential signal pairs (4 lanes) in each direction compared to 8 signal pairs (8 lanes) in Volta. A single link provides 25 GB/second bandwidth in each direction similar to Volta GPUs, but uses only half the signals compared to Volta. The total number of NVLink links is increased to twelve in A100, versus six in Tesla V100, yielding a whopping 600 GB/sec total bandwidth for an entire A100 versus 300 GB/sec for Tesla V100.

The twelve NVLink links in each A100 allow a variety of configurations with high-speed connections to other GPUs and switches. To meet the growing computational demands of larger and more complex DNNs and HPC simulations, the new DGX A100 system (see Appendix A) includes eight A100 GPUs connected by the new NVLink-enabled NVSwitch. Multiple DGX A100 systems can be connected via a networking fabric like Mellanox InfiniBand and Mellanox Ethernet to scale out data centers, creating very powerful, even supercomputer-class systems. More powerful NVIDIA DGX POD™ and NVIDIA DGX SuperPOD™ systems will include multiple DGX A100 systems to provide much greater compute power with strong scaling.



Note Third-Generation NVLink connectivity through NVSwitches.

Figure 26. NVIDIA DGX A100 with Eight A100 GPUs

All writes in the third-generation NVLink are now non-posted, allowing synchronization to be performed at the requester, and error attribution to be returned to a specific execution context. New features to improve the efficiency of small payload writes and dataless responses were also added.

PCIe Gen 4 with SR-IOV

The A100 GPU supports PCI Express Gen 4 (PCIe Gen 4) which provides 31.5 GB/sec of bandwidth per direction for x16 connections, double the bandwidth of PCIe 3.0/3. The faster speed is especially beneficial for A100 GPUs connecting to PCIe 4.0-capable CPUs, and for faster network interfaces, such as supporting 200 Gbit/sec InfiniBand for improved GPU cluster performance. A100 also supports Single Root Input/Output Virtualization (SR-IOV) that allows sharing and virtualizing of a single PCIe-connected GPU for multiple processes or Virtual Machines (VMs). A100 also allows a Virtual Function (VF) or Physical Function (PF) from a single SR-IOV PCIe-connected GPU to access a peer GPU over NVLink.

Error and Fault Detection, Isolation, and Containment

Improving GPU uptime and availability by detecting, containing, and often correcting errors and faults, rather than forcing GPU resets is critically important, especially in large multi-GPU

clusters and single-GPU, multi-tenant environments such as MIG configurations. The NVIDIA A100 Ampere architecture GPU includes much new technology to improve error/fault attribution (attribute which applications are causing errors), isolation (isolate faulty applications so they do not affect other applications running on the same GPU or in a GPU cluster), and containment (ensuring errors in one application do not leak and affect other applications).

The new NVIDIA Ampere architecture fault handling technologies are particularly important for MIG environments to ensure proper isolation and security between clients sharing the single GPU. NVLink-connected GPUs now also have more robust error-detection and recovery features as described in the NVLink section above. Page faults at the remote GPU are sent back to the source GPU through NVLink. Remote access fault communication is a critical resiliency feature for large GPU computing clusters to help ensure faults in one process or VM do not bring down other processes or VMs.

Additional A100 Architecture Features

The NVIDIA A100 GPU includes a number of other new and improved features that enhance application performance and improve programmability. We list a few of those new features below. Also be sure to check out the NVIDIA Developer site for additional information.

NVJPG Decode for DL Training

The A100 GPU adds a new hardware-based JPEG decode feature. One of the fundamental issues in achieving high throughput for DL training / inference for images is the input bottleneck of JPEG decode. CPUs and GPUs are not very efficient for JPEG decode due to the serial operations used for processing image bits. Also, if JPEG decode is done in the CPU, PCIe becomes another bottleneck. A100 addresses these issues by adding a hardware JPEG decode engine.

A100 includes a 5-core hardware JPEG decode engine called NVJPG. Applications can batch images into chunks of up to five images and pass onto NVJPG for processing. These images can be of heterogeneous sizes, though for best performance, images of similar sizes should be batched together wherever possible.

JPEG decode formats supported:

- YUV420
- YUV422
- YUV444
- YUV400
- RGBA

Performance @ GPU boost clock (1410 MHz) for large image such as 1 Mpixel or above in Mpixels/sec is as follows:

Table 6. NVJPG Decode Rate at different video formats

	Mpixels/sec
4:2:0 (10:1 compression ratio)	7000
4:4:4 (10:1 compression ratio)	3335

Note: For smaller images such as 224 x 224, Mpixels/sec can be 30-40% lower than above.

Optical Flow Accelerator

Optical flow and stereo disparity are two fundamental and related ways of analyzing images in computer vision. Optical flow measures the apparent motion of points between two images, while stereo disparity measures the (inverse) depth of objects from a system of two parallel calibrated cameras.

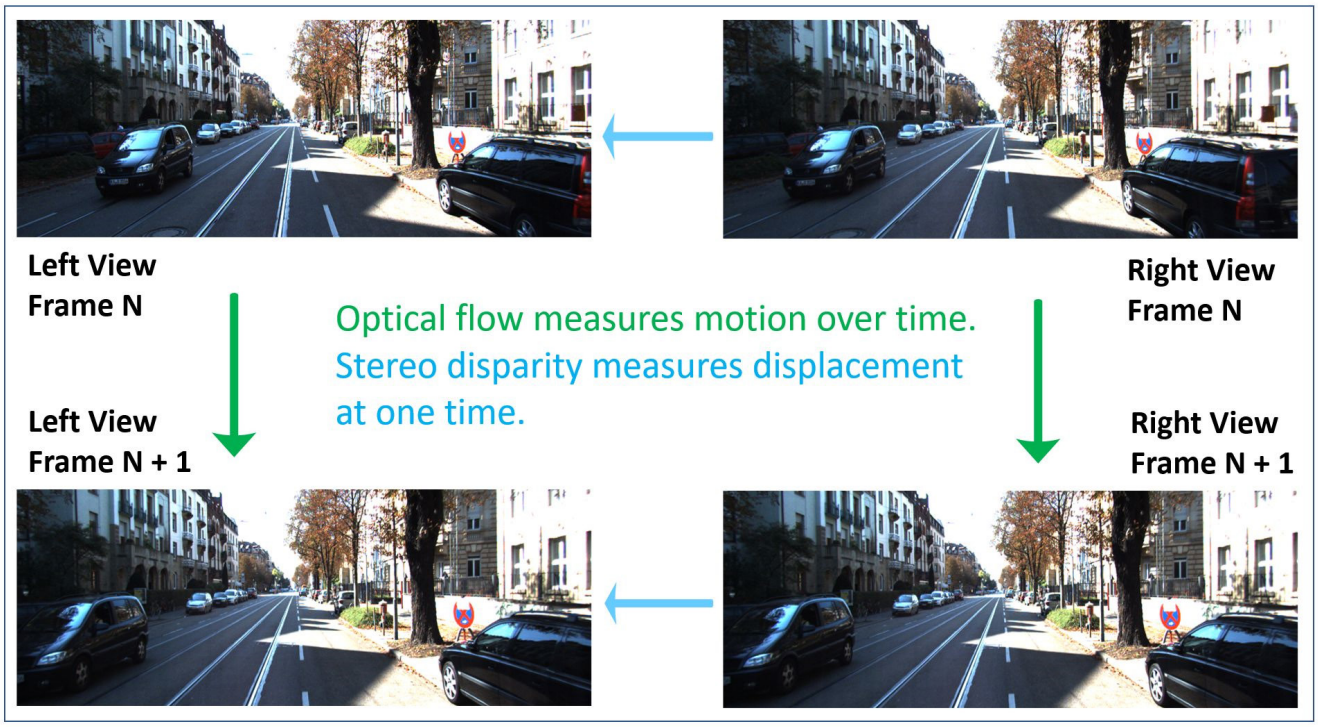


Figure 27. Illustration of optical flow and stereo disparity

Optical flow and stereo disparity are used in computer vision tasks across a broad range of applications including automotive and robotic navigation, movie production, video analysis and understanding, and augmented and virtual reality. The measurement of optical flow and stereo disparity have been studied for decades, but despite great improvements in the state of the art,

they remain challenging problems, especially to obtain real time dense data at the pixel rates of modern cameras, which routinely exceed 50 Mpixels/second and can easily reach 10 times that.

The GA100 Optical Flow Accelerator is a hardware module that supports both optical flow and stereo disparity estimation at high pixel rates. Quality and performance can be tuned through parameter selection.

Atomics Improvements

The A100 GPU builds on V100 atomic operation advances by improving throughput of atomic operations in global memory, which is especially beneficial for DL workloads. Many DL workloads use FP16 atomic operations during training and inference operations. A100 improves throughput of FP16 atomics by 11x and FP32 atomics by 2.7x over V100.

NVDEC for DL

A100 improves video decode capability significantly compared to V100. In a DL platform, input video is compressed in any of the industry standards, such as H264 / HEVC / VP9, etc. One of the significant challenges in achieving high end-to-end throughput in a DL platform is to be able to keep the input video decode performance matching the training / inference performance. Otherwise, the full DL performance of the GPU cannot be utilized. A100 makes a big leap in this area by adding five NVDEC (NVidia DECode) units.

Comparison to V100:

- 5 NVDECs in A100 compared to 1 NVDEC in V100
- HEVC decode performance improvement per NVDEC
- HEVC 4:4:4 support in A100

Table 7. GA100 HW decode support

	Bit depth	**Chroma format**
H264	8-bit	4:2:0
HEVC	8/10/12 bit	4:2:0 / 4:4:4
VP9	8/10/12 bit	4:2:0

Table 8. Decode performance @ GPU boost clock (1410 MHz)

(Measured in number of concurrent streams supported)

A100 # of Streams	HEVC decode	H264 decode	VP9 decode
4K30	44	22	31
1080p30	157	75	108
720p30	304	167	192

Table 9. A100 vs V100 Decode Comparison @ 1080p30

(Measured in number of concurrent streams supported)

# of Streams	HEVC decode	H264 decode	VP9 decode
A100 1080p30	157	75	108
V100 1080p30	22	16	22

Video use cases:

- Video classification / understanding
- Intelligent video analytics on edge platform
- Autonomous driving DL training

CUDA Advances for NVIDIA Ampere Architecture GPUs

NVIDIA® CUDA® is a parallel computing platform and programming model created by NVIDIA to give application developers access to the massive parallel processing capability of NVIDIA GPUs. CUDA is the foundation for GPU acceleration of deep learning as well as a wide range of other computation- and memory-intensive applications ranging from astronomy, to molecular dynamics simulation, to computational finance. Thousands of GPU-accelerated applications are built on the NVIDIA CUDA parallel computing platform. The flexibility and programmability of CUDA have made it the platform of choice for researching and deploying new deep learning and parallel computing algorithms.

NVIDIA Ampere architecture GPUs are designed to improve GPU programmability and performance, while also reducing software complexity. NVIDIA Ampere architecture GPUs and CUDA programming model advances accelerate program execution and lower the latency and overhead of many operations. CUDA 11 provides programming and API support for Third-Generation Tensor Cores, Sparsity features, CUDA Graphs, Multi-Instance GPU, L2 cache residency controls, and a number of other new capabilities of the NVIDIA Ampere architecture.

The sections below cover some of the key NVIDIA Ampere architecture-related CUDA advances.

CUDA Task Graph Acceleration

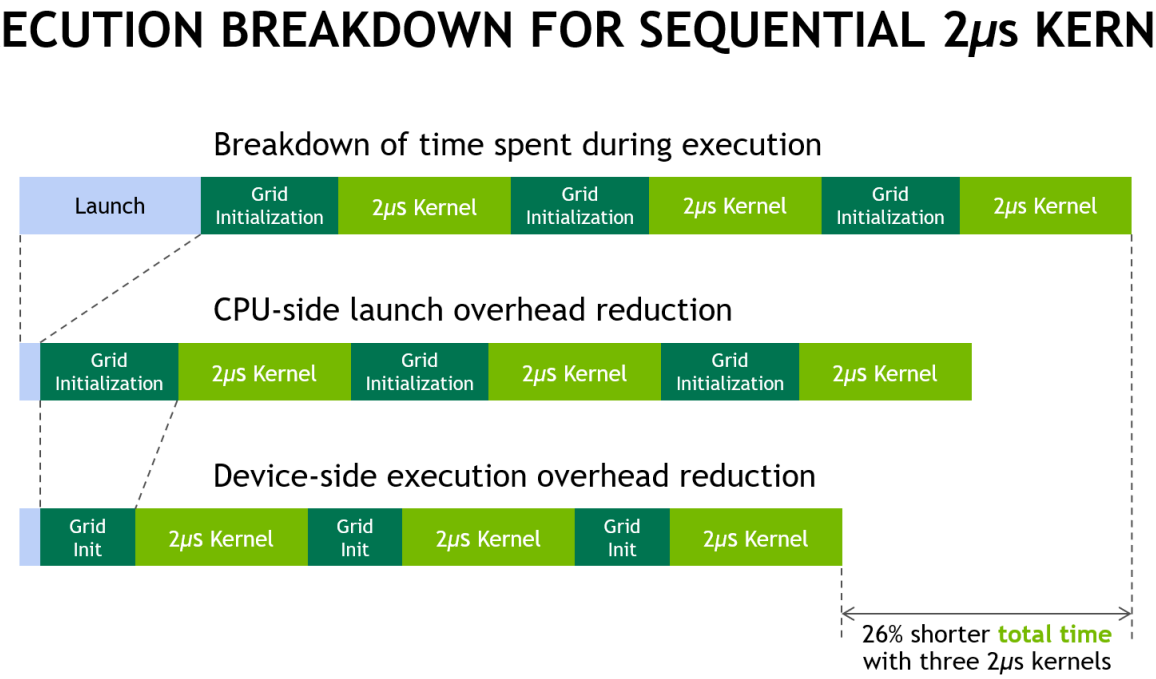
CUDA Task Graph Basics

Many GPU-intensive applications such as deep neural network training and scientific simulations have an iterative structure where the same workflow is executed repeatedly. Using CUDA Streams for such workflows requires that the work be resubmitted to the GPU by the CPU with every iteration, which consumes both time and CPU resources. CUDA Task Graphs were introduced as part of the CUDA 10 release in 2018, and provide a more efficient model for submitting work to the GPU. A task graph consists of a series of operations, such as memory copies and kernel launches, connected by dependencies, and is defined separately from its execution. Task graphs enable a define-once/run-repeatedly execution flow. A predefined task graph allows launch of any number of kernels in one single operation, greatly improving application efficiency and performance.

Execution of work on the GPU breaks down into three stages: launch, grid initialization, and kernel execution. For GPU kernels with short runtimes in particular, these overheads can be a significant fraction of the overall end-to-end execution time.

Separating out the definition of a task graph from its execution (where the task graph is executed repeatedly) reduces CPU kernel launch costs significantly. Task graphs also enable the CUDA driver to perform a number of optimizations because the whole workflow is visible to

the driver, including execution, data movement, and synchronization interactions, which can improve execution performance in a variety of cases.



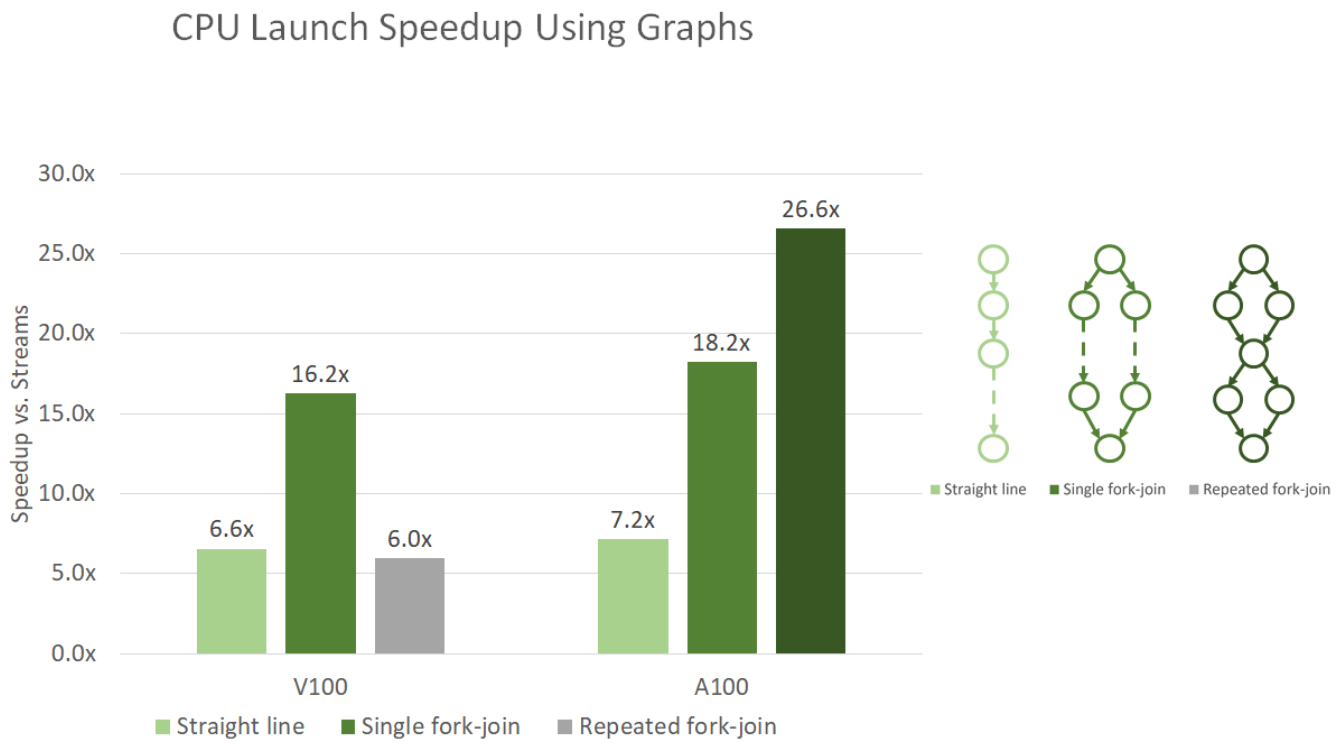
One aspect to notice is that efficiency gains benefit different processors: CPU execution time benefits for its single launch operation, while GPU execution time benefits from a per-kernel savings, which can add up significantly as the number of nodes in the graph increases.

Figure 28. Execution Breakdown for Sequential 2us Kernels.

Task Graph Acceleration on NVIDIA Ampere Architecture GPUs

The A100 GPU accelerates a number of optimizations enabled by task graphs. These fall into two categories: launch optimizations and execution dependency optimizations. These optimizations are aimed at inter-kernel latency and overhead reduction, and are all possible because in a task graph the whole workflow is known in advance. They are particularly effective for strong-scaled workloads with very short duration kernels, for which overheads comprise a significant fraction of runtime.

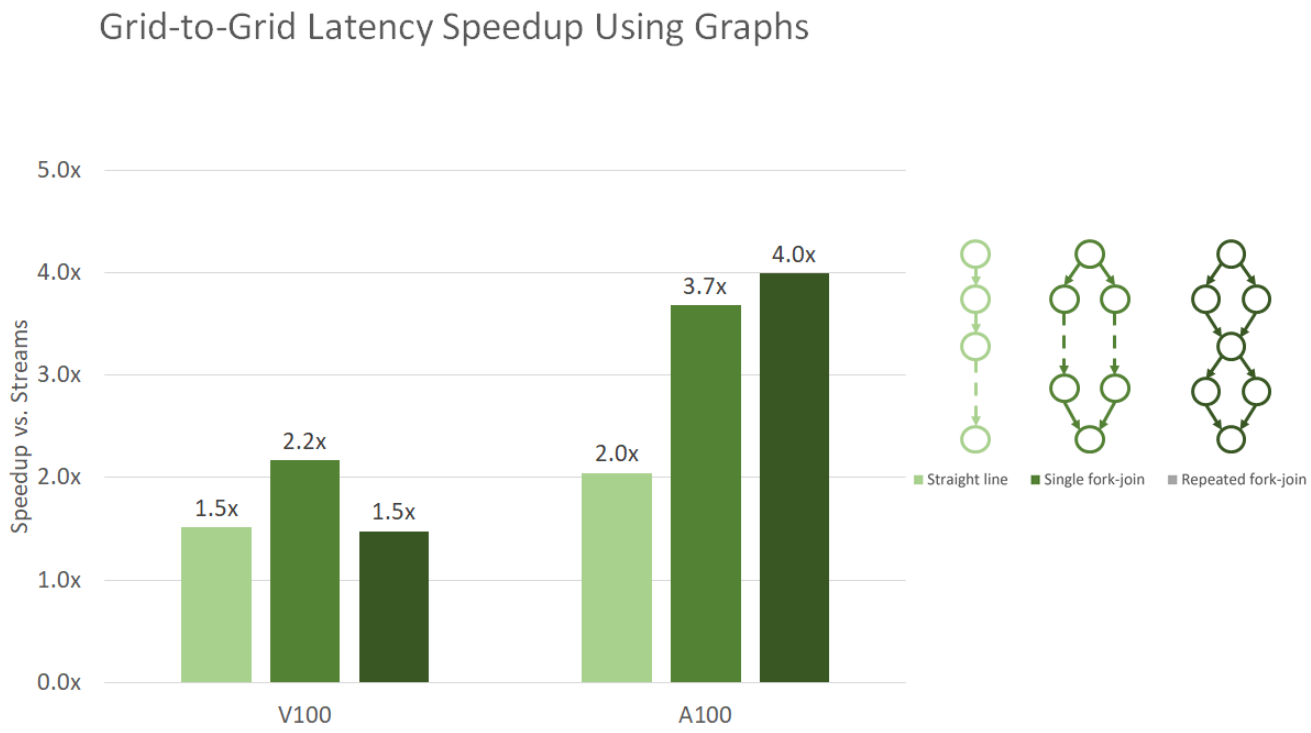
Launch optimizations rely on the graph topology identifying the whole workflow, enabling efficient upload of kernel data that will be needed to both launch and run the work. First the initial launch of a graph is able to submit multiple work items to the GPU in a single operation. This relates directly to a large reduction in launch overhead as seen by the CPU. Next, the A100 GPU can use dependency information built into the graph to upload kernel information more efficiently into the SMs for execution, significantly reducing the latency before the first instruction in a kernel starts to run.



Comparing CPU cost of whole-graph launch vs. the equivalent time to launch without graphs. Graph launch provides a significant boost on any hardware, but A100's launch optimizations provide a significant gain, in particular for graphs with more complex topologies.

Figure 29. Impact of Task Graph acceleration on CPU launch latency

Execution dependency optimizations address more complex graphs where the workflow forks and re-joins. The A100 GPU architecture includes the capability to follow multiple dependencies in a fork, automatically executing dependent kernels with the shortest possible latency. This translates directly into significant gains for both launch and grid-to-grid execution latency for topologically complex graphs.



Comparing execution latencies for work launched as a graph vs. as independent kernels in CUDA streams. The NVIDIA Ampere microarchitecture has improved dependency tracking, able to execute even complex workflows very efficiently.

Figure 30. Grid-to-Grid Latency Speedup using CUDA graphs

For more details on the use CUDA Task Graphs refer to the CUDA Programming Guide, and Getting Started with CUDA Graphs.

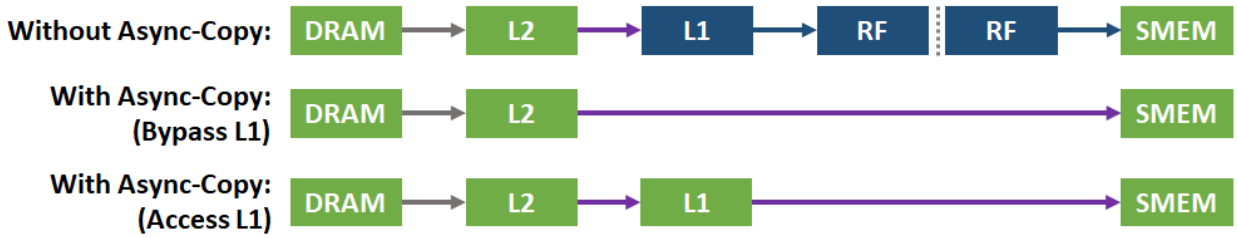
CUDA Asynchronous Copy Operation

CUDA 11 includes a new asynchronous copy (async copy) API to take advantage of the A100 GPU's hardware-accelerated direct-copy-to-shared functionality. Async copy performs an asynchronous (non-blocking) direct memory transfer from global memory to shared memory, bypassing the SM threads and combining the functions of separate "load from global memory into a register", and "write to shared memory from a register" operations into a single, efficient operation.

Async copy eliminates the need for intermediate staging of data through the register file (RF), reducing register file bandwidth. It also efficiently uses memory bandwidth and reduces power consumption. As the name implies, async copy works asynchronously, allowing other computations to occur during global-to-shared memory copies. Async copy is able to notify the program of copy completion via the GPU's new barrier feature (see next section).

Bypassing L1 and the register file can significantly accelerate memory copy performance, especially for multiple successive async-copy operations that copy large amounts of data from global to shared memory.

Two variants of the async copy instruction are available for different usage scenarios. BYPASS, which bypasses L1 cache and the register file as described above, and ACCESS which saves data to L1 for subsequent accesses and reuse.



The upper pipeline depicts the case of not using async copy, loading from DRAM or L2, through L1, then into the register file (RF), and finally storing from RF to shared memory (SMEM). The lower two pipelines show async copy fetching from DRAM or L2 and directly storing into shared memory, with optional L1 cache access.

Figure 31. A100 Asynchronous Copy vs No Asynchronous Copy

From a user perspective, async copy behaves similarly to executing separate load-global and store-shared instructions, but without consuming thread resources for temporary storage. The instruction allows independent global and shared memory addresses per thread. Programs must use barriers to ensure proper ordering of writes and visibility of loads and stores among threads in a thread block. New asynchronous per-thread barriers described in the following section accomplish that task.

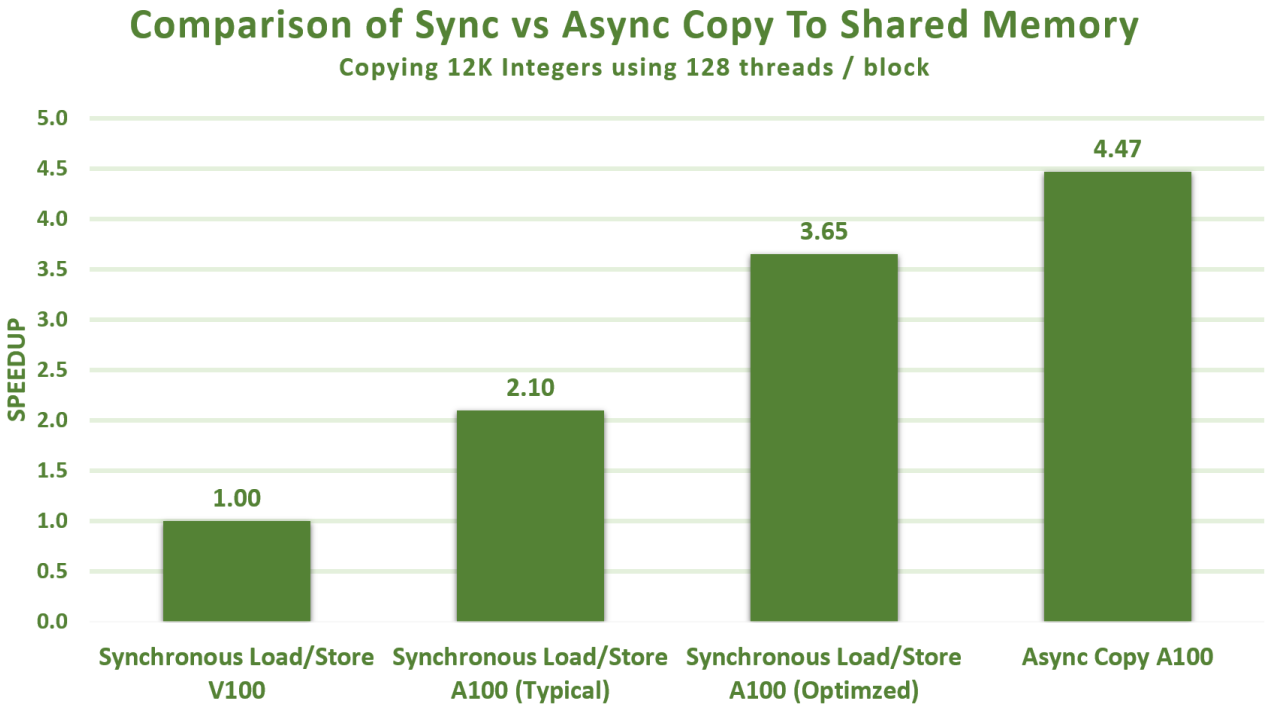
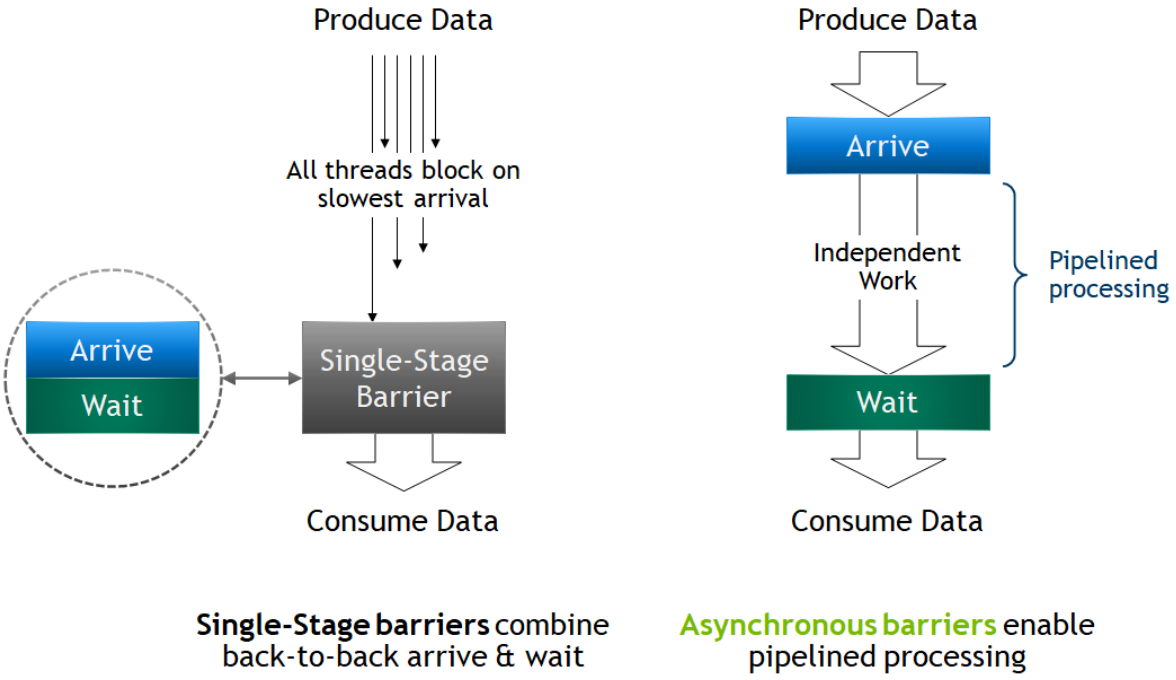


Figure 32. Synchronous vs Asynchronous Copy to Shared Memory

Asynchronous Barriers

NVIDIA A100 GPU provides hardware-accelerated barriers in shared memory. These barriers are made available in CUDA 11 in the form of ISO C++-conforming barrier objects. An asynchronous barrier differs from a normal single-stage barrier in that the notification by a thread that it has reached the barrier (the “arrival”) is separated from the operation to wait for others to arrive at the barrier (the “wait”). This increases execution efficiency by allowing a thread to perform additional operations unrelated to the barrier, making more efficient use of the wait time. Asynchronous barriers can be used to implement producer-consumer models using CUDA threads, or can be used simply as a single-stage barrier if desired.



An asynchronous barrier allows a thread to indicate that its data is ready and then continue to work on independent operations, postponing the wait so that idle time is reduced. This is a form of asynchronous processing known as *pipelining*, and is commonly used to hide high latency operations such as memory loads (see "async copy", above).

Figure 33. A100 Asynchronous Barriers

The new Asynchronous Barriers also provide a significant advancement in synchronization granularity compared with barriers on previous architectures, by allowing hardware-accelerated synchronization of any subset of CUDA threads within the block. Previous architectures only accelerate synchronization at a whole-warp or whole-block level. Barriers can be used to overlap asynchronous copies from global memory into shared memory (described in the previous section) by having the copy operation signal ("arrive on") the barrier when it is complete. This allows overlap of the copy with other execution in the SM, hiding the latency of the copy and increasing efficiency.

L2 Cache Residency Control

When a CUDA kernel accesses a data region in the global memory repeatedly, such data can be considered to be "persisting". On the other hand, if the data is only accessed once, such data can be considered to be "streaming". DL workloads in particular have a dependence upon persisting data accesses.

Starting with CUDA 11.0, devices of compute capability 8.0 like A100 have the capability to influence persistence of data in the L2 cache and set aside a portion of L2 cache for persistent data accesses, allowing higher bandwidth and lower latency accesses to the global memory.

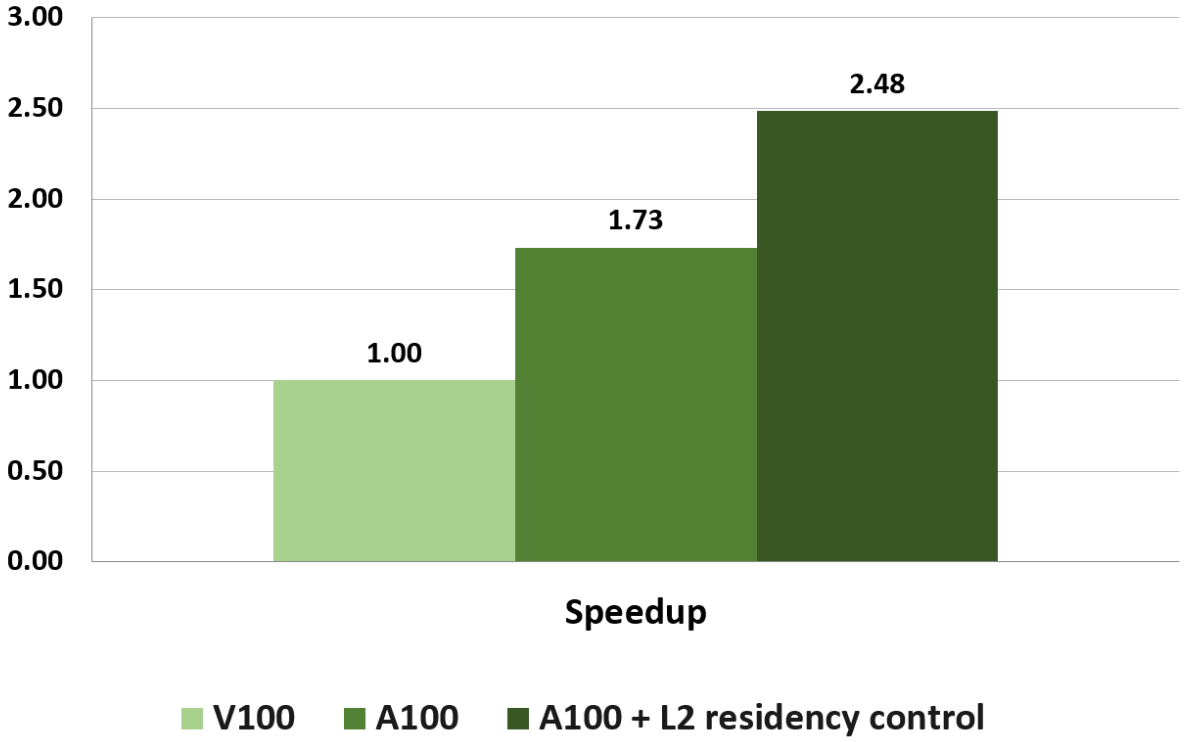
The capability to influence the persistence of data in the L2 cache allows A100 GPUs to use the large 40MB L2 cache more efficiently. For example, recurrent weights in many LSTM networks can be made persistent in L2 and re-used between GEMM operations. A100 allows L2 cache to be set-aside for persistent accesses in $1/16^{th}$ increments (2.5 MB).

Persistent accesses have prioritized use of this set-aside portion of L2 cache. Normal or streaming accesses to global memory can only utilize this portion of L2 when it is unused by persistent accesses. L2 persistence can be set up using CUDA Streams or CUDA Graphs. However, note that when the GPU is configured in Multi-Instance GPU (MIG) mode, the L2 cache set-aside functionality is disabled.

Details including how to set up and use L2 cache set-aside areas for persistent accesses, how multiple CUDA kernels that execute concurrently can share the L2 set-aside cache, and how to clear and reset the set-aside area for non-persistent future accesses are provided in the CUDA Programming Guide.

Residency of data in the L2 cache can be managed via an address-range-based window which designates an address range for which all read and write accesses will be cached persistently in L2. The memory operations themselves require no annotation.

A100 also supports finer-grained per-memory-operation controls where L2 residency is specified on a per-access basis. The access-based controls include fractional allocation based on address hash, and cover use-cases like producer-consumer buffers. Please see the CUDA Programming Guide for specifics on how these features are supported.



Example of performing histograms in global memory. Data set of 256 million integer elements. Histogram size five Million integer bins.

Figure 34. A100 L2 residency control example

When performing histograms using a large number of bins that do not fit in shared memory, one must compute histograms by directly performing atomic operations in GPU global memory. In Figure 34 above, we compute a histogram of five million integer bins using a dataset of 256 million integers. Five million integer bins have a footprint of 20 MB, hence they do not fit in shared memory, but can fit in the GPU's L2 Cache. By marking the histogram bin region persistent, we can achieve 2.5x speedup over V100, and a speedup of 43% over an A100 scenario that does not use residency control.

Cooperative Groups

Cooperative Groups extends its programming model (originally introduced in CUDA 9) to encapsulate the asynchronous memory copy in a group-wide collective. This utilizes A100's hardware acceleration for the non-blocking memory copy from global to shared memory as well as providing (blocking) software fallbacks in the other direction and on earlier architectures.

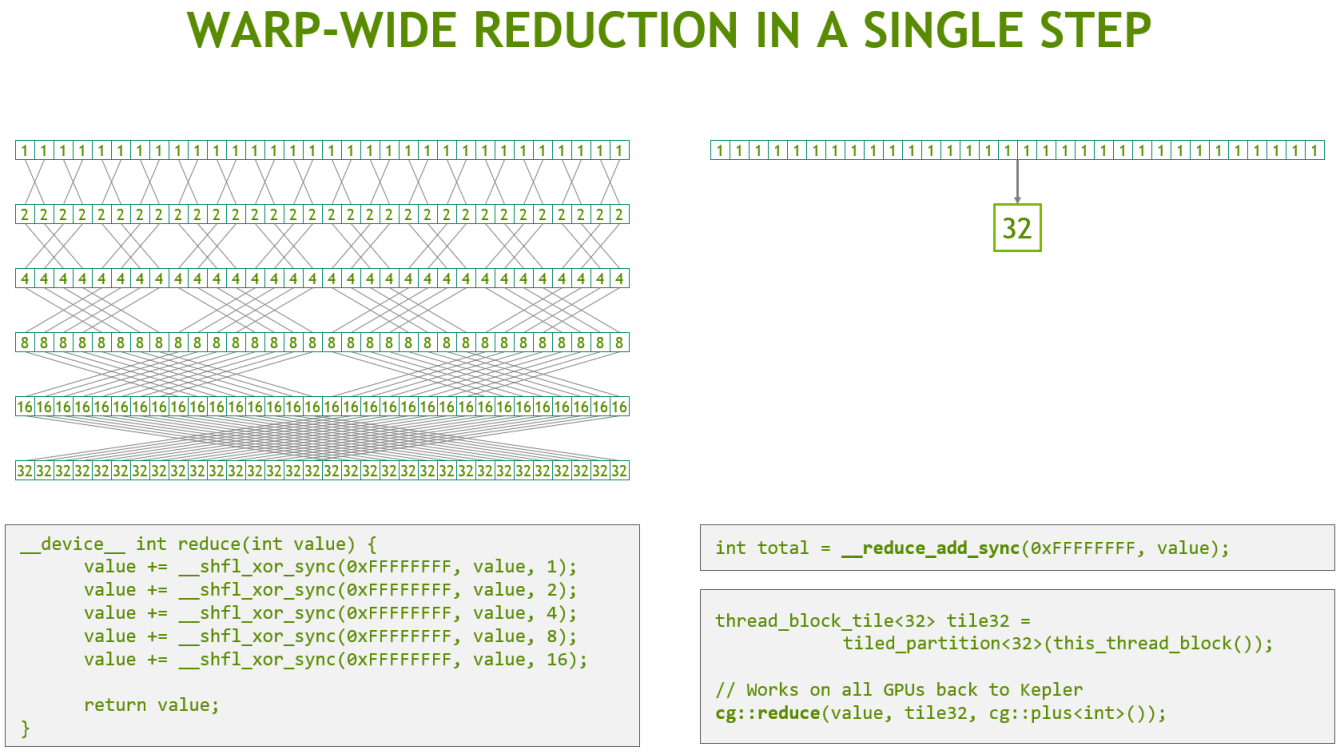
Cooperative Groups uses the threads named in the group to distribute the workload automatically and as efficiently as possible, deducing the correct alignment and data transfer size per thread. While the default operation behaves as a single-stage pipeline, overloads are

provided to extend this to multi-stage pipelines in cooperation with the new memory pipeline object provided by CUDA.

Once the transfers are kicked off, the data can be read following a call to wait, which signals that the pipeline is empty or that the corresponding stage has completed moving data to shared memory.

Using A100's powerful new warp reduce instruction, Cooperative Groups expands its set of collectives with a reduce API. This performs a reduction operation on the data provided by each thread named in the group passed in. The hardware can accelerate arithmetic ADD, MIN, or MAX operations and the logical AND, OR, or XOR. Additional types and operations are implemented in software as well as fallbacks on older generation hardware.

Cooperative launches continue to provide interesting benefits to CUDA developers, and we've been able to reduce grid synchronization overhead by up to 30%, and remove the need for separate compilation when writing cooperative kernels and taking advantage of grid groups.



Pre-A100 warp-scope reductions are based on the SHFL operation and require 5 steps to complete, following the data exchange pattern shown on the left. The A100 GPU has hardware-accelerated reductions which produce a result in a single step.

Figure 35. Warp-Wide Reduction

Conclusion

NVIDIA's mission is to accelerate the work of the da Vincis and Einsteins of our time. Scientists, researchers, and engineers are focused on solving some of the world's most important scientific, industrial, and big data challenges using high-performance computing (HPC) and artificial intelligence (AI). The NVIDIA® A100 Tensor Core GPU delivers the next giant leap in our accelerated data center platform, providing unmatched acceleration at every scale and enabling these innovators to do their life's work in their lifetime. A100 powers numerous application areas including HPC, Genomics, 5G, Rendering, Deep Learning, Data Analytics, Data Science, and Robotics.

Advancing the most important HPC and AI applications today—personalized medicine, conversational AI, and deep recommender systems—requires researchers to go big. A100 powers the NVIDIA data center platform that includes Mellanox HDR InfiniBand (IB), NVIDIA NVSwitch, NVIDIA HGX-A100, and the Magnum IO SDK for scaling up. This integrated team of technologies efficiently scales to tens of thousands of GPUs to train the most complex AI networks at unprecedented speed.

The new Multi-Instance GPU (MIG) of the A100 GPU can partition each A100 into as many as seven GPU accelerators for optimal utilization, effectively improving GPU resource utilization and GPU access to more users and GPU-accelerated applications. With A100's versatility, infrastructure managers can maximize the utility of every GPU in their data center to meet different-sized performance needs, from the smallest job to the biggest multi-node workload.

Appendix A - NVIDIA DGX A100

Today's enterprise needs to scale AI to transform their business and not only survive, but thrive in challenging times. However, most enterprises lack the infrastructure and know-how to operationalize AI at scale. Their dependency on legacy systems and architecture has resulted in data centers that are over-spent on servers, inefficient, and unable to meet the unique demands of training, inference, and analytics.

NVIDIA DGX A100 - The Universal System for AI Infrastructure

DGX A100 is the 3rd generation of the world's most advanced, purpose-built AI system. It delivers an unprecedented 5 PFLOPS (petaFLOPS) of performance in a single system. DGX A100 revolutionizes the enterprise data center with a new construct for infrastructure that's designed to unify all AI workloads on a new, universal platform and architecture. DGX A100, powered by A100 and MIG, transforms the enterprise data center. It enables architects to plan, deploy and scale their data center, now optimized for heterogeneous workloads using homogeneous infrastructure. DGX A100 gives AI innovators the power they need to do their most important work, from development to deployment at scale.

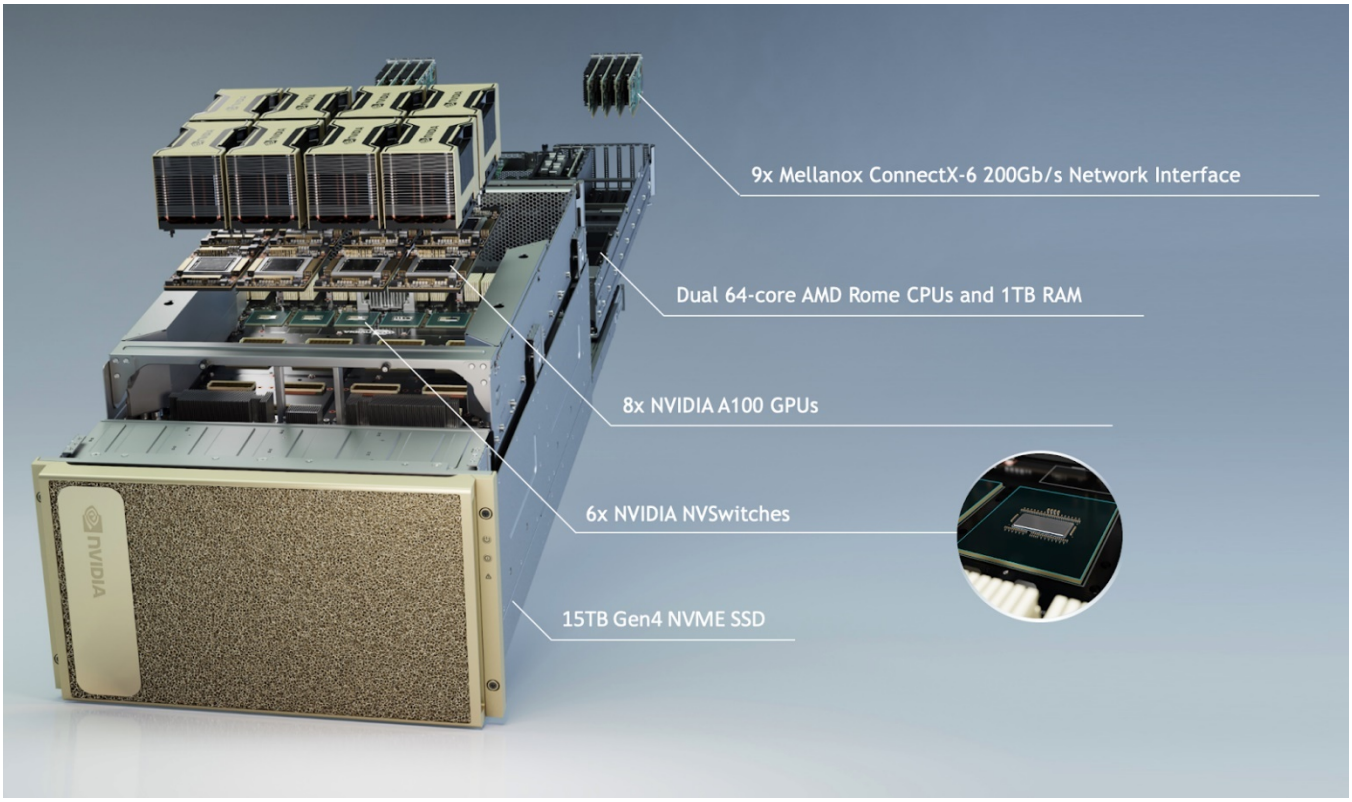


Figure 36. NVIDIA DGX 100 System

The performance of DGX A100 can be easily scaled up by connecting several thousands of DGX A100 systems, or scaled down for unparalleled efficiency by slicing each A100 GPU in the system into seven separate GPU Instances. Multi-Instance GPU (MIG) innovation is a breakthrough technology from NVIDIA, providing up to 56 discrete accelerators in a single DGX A100, each fully isolated and secured at the hardware level with their own high bandwidth memory, cache, and compute cores. MIG allows users to mix and match multiple training and inference jobs in parallel on the same system with dedicated resources for optimal utilization.

Running optimized software from NVIDIA GPU Cloud (NGC), the combination of dense compute power and complete workload flexibility makes DGX A100 an ideal choice for both single node deployments and large scale Slurm and Kubernetes clusters deployed with NVIDIA DeepOps.

Game-changing Performance

The eight NVIDIA A100 GPUs in the DGX A100 use the new high-performance third-generation NVLink to interconnect through six new NVSwitches with 4.8 TB/s total bidirectional bandwidth (2.4 TB/s full-duplex). Each NVIDIA A100 GPU features third-generation Tensor Cores with TF32 precision and sparsity that provides up to 20x the performance of standard FP32 FMA operations on V100 with zero code changes. DGX A100 delivers up to 6x the performance of V100 based DGX-1 for AI training. DGX A100 also sets a new bar for compute density by packing up to 5 PFLOPS of AI performance into a 6U form factor.

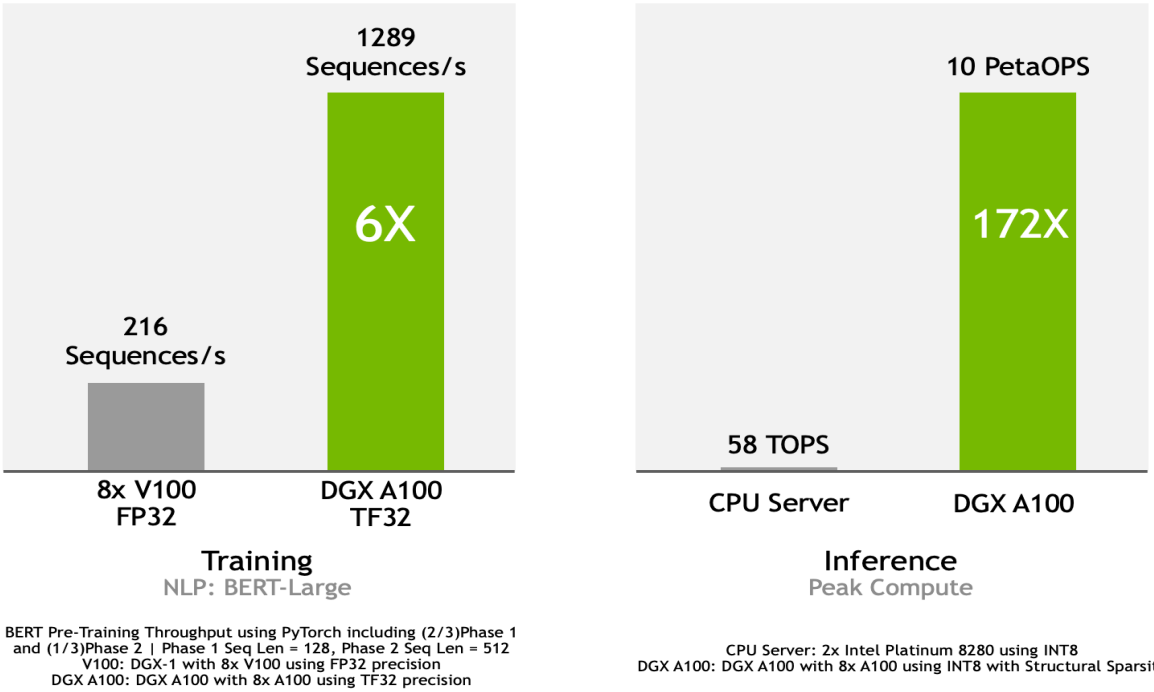


Figure 37. DGX A100 Delivers unprecedented AI performance for training and inference.

Unmatched Data Center Scalability

With the fastest IO architecture of any DGX system, NVIDIA DGX A100 is the foundational building block for large AI clusters such as NVIDIA DGX SuperPOD, the enterprise blueprint for scalable AI infrastructure. DGX A100 debuts next generation NVLink that is 10x faster than PCIe Gen 4, a new NVSwitch, and eight Mellanox ConnectX-6 HDR InfiniBand adapters each running at 200 Gb/s providing a high-speed fabric for large scale AI workloads. DGX A100 also supports the Magnum IO software SDK for efficient scaling of applications to tens of thousands of GPUs. The combination of massive GPU-accelerated compute, state-of-the-art networking hardware, and software optimizations means NVIDIA DGX A100 can scale to hundreds or thousands of nodes to meet the biggest challenges, such as conversational AI and large-scale image classification.

Fully Optimized DGX Software Stack

The DGX A100 software has been built to run AI workloads at scale. A key goal is to enable practitioners to deploy deep learning frameworks, data analytics, and HPC applications on the DGX A100 with minimal setup effort. The design of the platform software is centered around a minimal OS and driver install on the server, and provisioning of all application and SDK software available through the NGC Private Registry.

The NGC Private Registry provides GPU-optimized containers for deep learning (DL), machine learning (ML), and high-performance computing (HPC) applications, along with pretrained models, model scripts, Helm charts, and software development kits (SDKs). This software has been developed, tested, and tuned on DGX systems, and is compatible with all DGX products: DGX-1, DGX-2, DGX Station, and DGX A100. The NGC Private Registry also provides a secure space for storing custom containers, models, model scripts, and Helm charts that can be shared with others within the enterprise. Learn more about the NGC Private Registry in this blog post.

Figure 38 shows how all these pieces fit together as part of the DGX software stack.

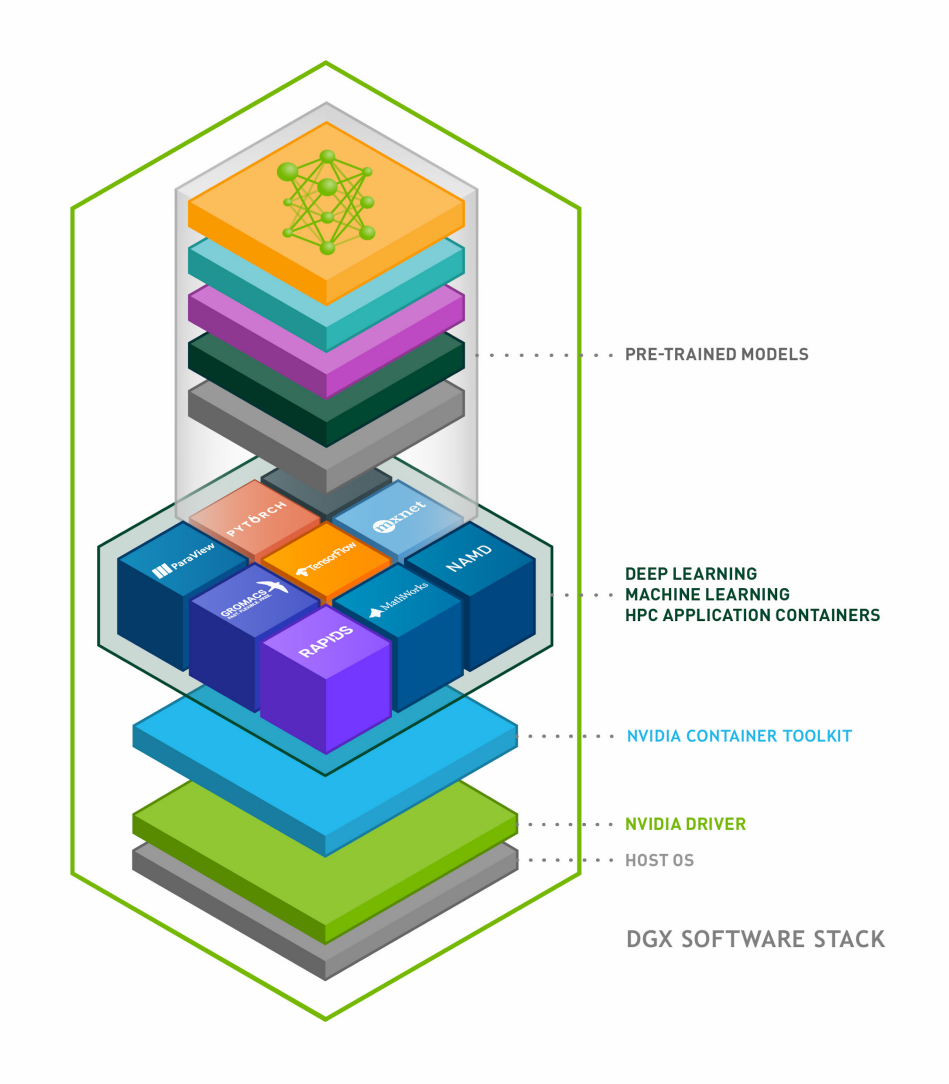


Figure 38. NVIDIA DGX Software Stack

The DGX software stack includes the following major components.

- **The NVIDIA** CUDA Toolkit is the development environment for creating high performance GPU-accelerated applications. CUDA 11 enables software developers and devops engineers to reap the benefits of the major innovations in the new NVIDIA A100 GPU, including the following:
    - Support for new input data type formats and performance optimizations in CUDA libraries for linear algebra
    - configuration and management of MIG instances on Linux operating systems, part of the DGX Software stack

Read more about what's new in the CUDA 11 Features Revealed Devblog.

- **The NVIDIA Container Toolkit** allows users to build and run GPU accelerated Docker containers. The toolkit includes a container runtime library and utilities to automatically configure containers to leverage NVIDIA GPUs.
- **GPU-accelerated containers** feature software to support:
    - Deep learning frameworks for training, such as PyTorch, MXNet, and TensorFlow
    - Inference platforms, such as TensorRT
    - Data analytics, such as RAPIDS, the suite of software libraries for executing end-to-end data science and analytics pipelines entirely on GPUs.
    - High-Performance Computing (HPC), such as CUDA-X HPC, OpenACC, and CUDA®.

For more information on DGX A100 please see our blog Defining AI Innovation with DGX A100.

NVIDIA DGX A100 System Specifications

Table 10. NVIDIA DGX A100 System Specifications

Specification	DGX-1	DGX A100
GPUs	8x Tesla V100 GPUs	8x NVIDIA A100 GPUs
TFLOPS	960 (GPU FP16) + 3 (CPU FP32)	5 (GPU Tensor PFLOP) + 3 (CPU FP32)
GPU Memory	32 GB per GPU / 256 GB per DGX-1 Node	40GB per GPU/320 GB per DGX A100 Node
CPU	Dual 20-core Intel® Xeon® E5-2698 v4 2.2 GHz	2-socket, 128 core AMD Rome 7742, 2.25 GHz (base), 3.4GHz (Max boost)
System Memory	512 GB 2133 MHz DDR4 LRDIMM	1 TB 3200 MHz DDR4 base config, additional 1TB can be order to get to 2TB max
Storage	Data cache drives: 7TB (4x 1.92TB SSD) OS drive: 480 GB SAS SSDs	Data cache drives: 15TB (4x 3.84TB gen4 NVME. Can add 15TB optional to get 30TB max OS drives: 2x 1.92TB NVME SSDs
Network	Dual 10 GbE 4 Mellanox 100 Gb/sec InfiniBand/100GigE	8 single port Mellanox ConnectX-6 HDR InfiniBand 200Gb/s 1 dual-port Mellanox ConnectX-6 10/25/40/50/100/200Gb/s Ethernet Optional 10th dual-port Mellanox ConnectX-6

System Weight	134 lbs	271 lbs
System Dimensions	866 L x 444 W x 131 H (mm)	Height: 10.4 in (264.0 mm) Width: 19.0 in (482.3 mm) Max Length: 35.3 in (897.1 mm) Max
Rack Units	3 RUs	6 RUs
Power	3200 W (Max). Four 1600 W load-balancing power supplies (3+1 redundant), 200-240 V(ac), 10 A	6500 W (Max). Six 3kW power supplies. 3+3 redundant, 200-240 V(ac), 16 A
Operating	10 - 35°C	5oC - 35oC
Cooling	Air	Air

Appendix B - Sparse Neural Network Primer

Even though GPU compute performance has rapidly increased to keep up with the growing complexity of DNNs, scientists are researching new techniques to reduce the compute, memory, and energy used in training these dense networks, and also to reduce the size of the trained networks such that they can fit in edge devices that have small memories. Techniques to prune a dense network into a sparse network that delivers the same level of accuracy is being widely and actively researched by industry and academia.

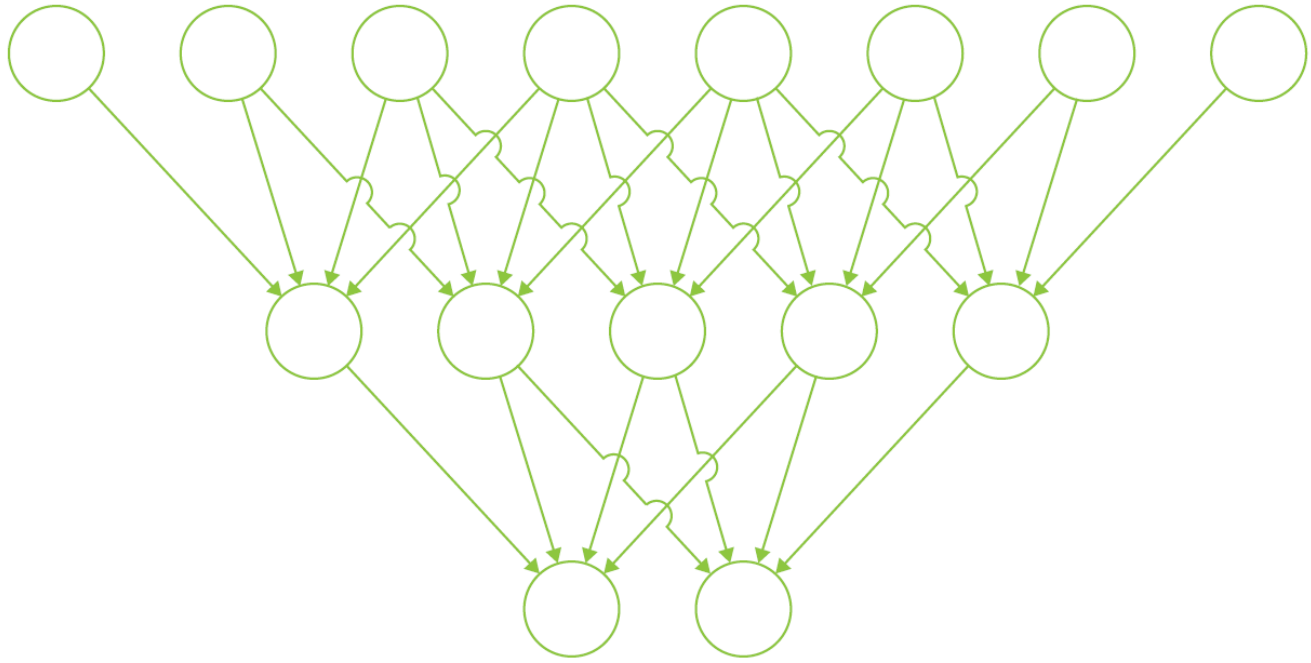


Figure 39. Dense Neural Network

Deep Neural Networks (DNNs) consist of several layers of neurons or nodes that are interconnected with each other. Typically, each neuron or node in a fully connected DNN is connected with every neuron in the next layer of the network. This means that if a network has n nodes in a layer and is connected to n nodes in the next layer, the number of interconnections between the two layers would be n^2. The complexity of new neural networks has rapidly increased over the last few years resulting in DNNs that contain tens to hundreds of layers, and thousands of neurons with millions of interconnections.

Pruning and Sparsity

Pruning, in a nutshell, is the technique used to zero out or remove nodes and interconnections that have little to no contribution to the final accuracy of the network. For AI training, this could mean zeroing out many of the weight and activation matrices that have near-zero values, and retraining to optimize the remaining weights. For inferencing it could be done by rounding down to zero the values of weights that have near zero values, or removing from the network the interconnections and nodes that have near zero values. In other words, after pruning, the network is now sparse due to fewer nodes and interconnections. Numerous research papers have explored various techniques of pruning for sparse networks and these can be found online for further reading on this topic.

Exploiting sparsity in a neural network delivers several performance benefits. First, compute throughput can be increased by skipping computation on zero-value matrix elements. Second, memory bandwidth usage can be reduced by fetching only the non-zero elements. Third, for latency-bound inferencing applications, latency can be reduced by fetching more of the non-zero values from memory and storing them locally on-chip.

Fine-Grained and Coarse-Grained Sparsity

Research on sparsity can be generally split into fine-grained sparsity, which explores zeroing out specific weights distributed across the neural network, and coarse-grained sparsity, which explores zeroing out entire sub-networks of a neural network.

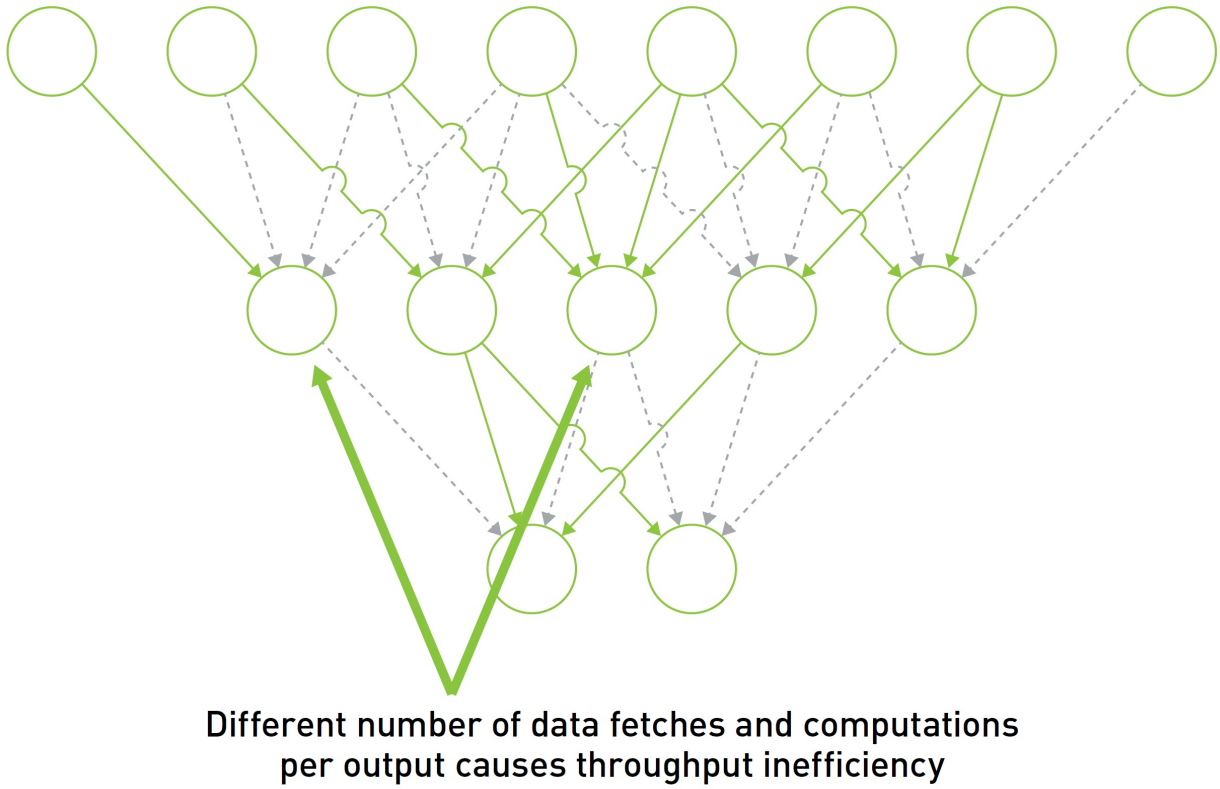


Figure 40. Fine-Grained Sparsity

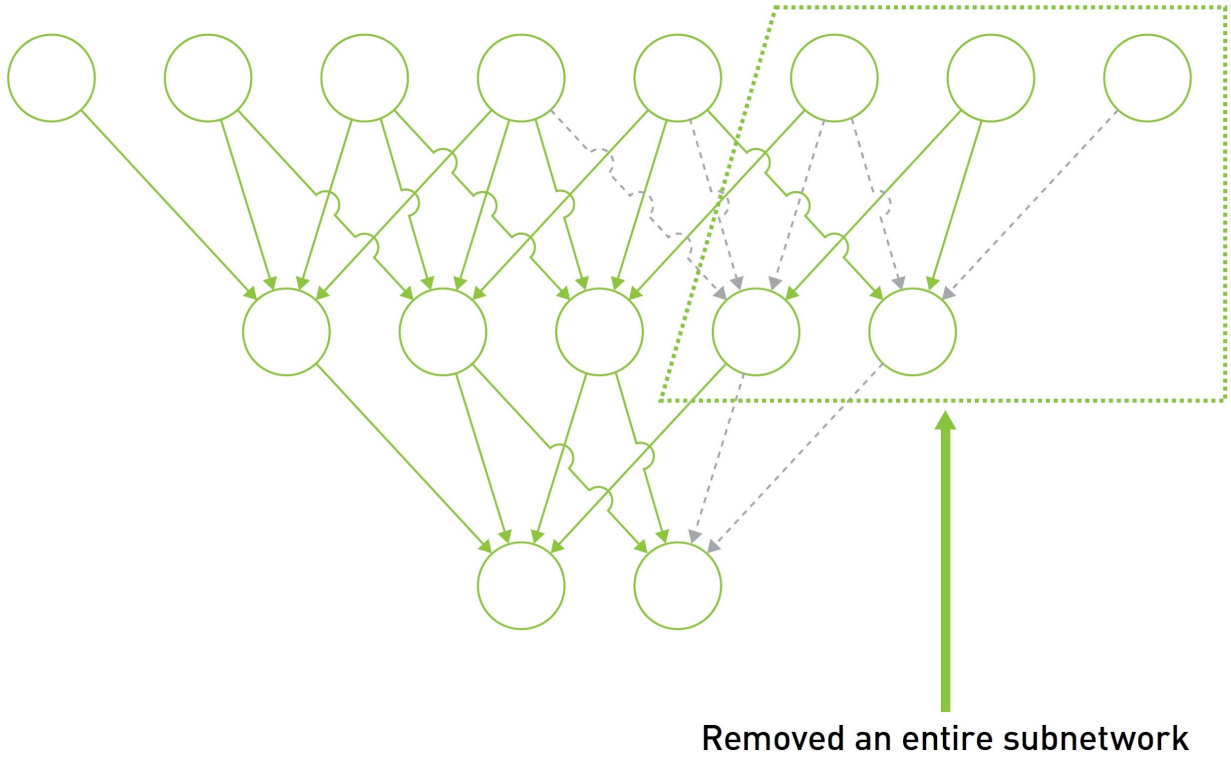


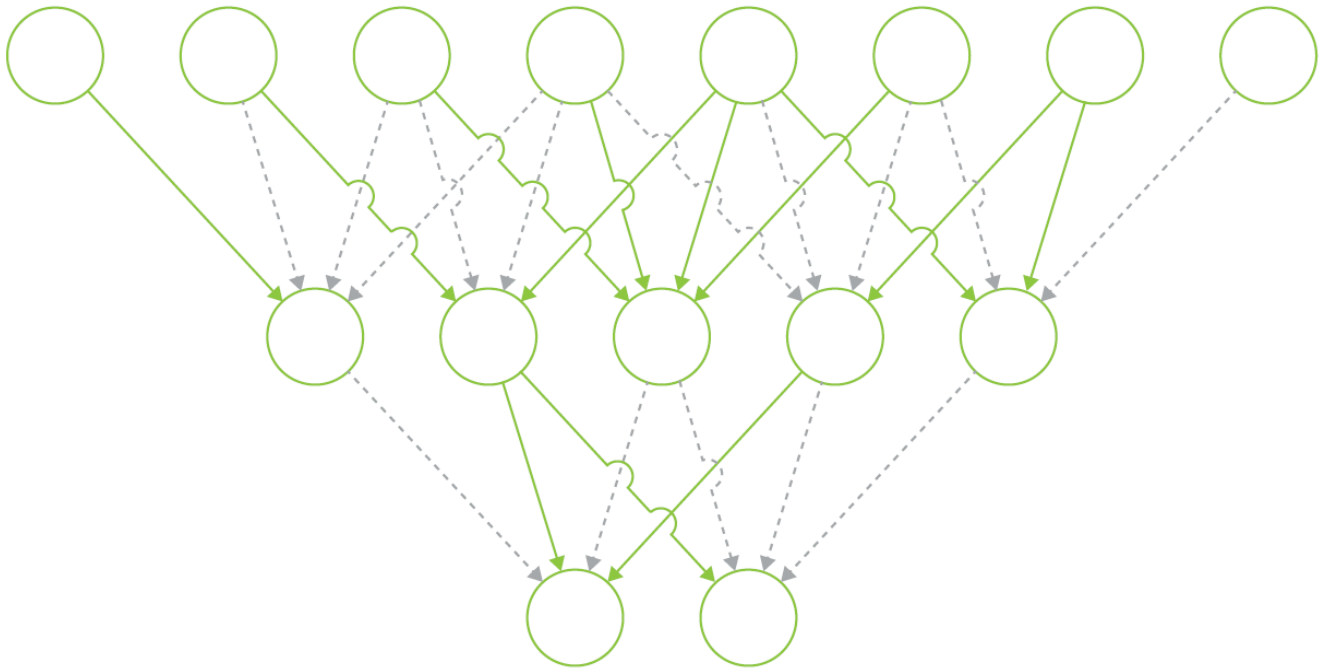
Figure 41. Coarse Grained Sparsity

A network with fine-grained sparsity would have the same number of nodes, but fewer edges irregularly distributed across the network. As seen in Figure 40, the amount of data fetched from memory and computations required to compute the output of each node would vary from node to node. This leads to irregular memory accesses and load balancing issues that reduce the parallel nature of compute workload and thus reducing GPU compute throughput.

A network pruned for coarse-grained sparsity (Figure 41) will have entire sub-sections of the network removed. While this helps maintain the parallel nature of the workload and improves throughput, a larger loss in accuracy may be undesirable.

Fine-grained structured sparsity, as supported in the NVIDIA A100 GPU, allows the network to be sparse, but requires each node to perform the same amount of data fetches and computation. The below Fine Grained Structured Sparsity results in balanced workload distribution and better utilization of compute nodes. The pruned edges represented by the lighter colored connections would be represented by zero values in the weight matrices.

Figure 42 represents a network with fine-grained structured sparsity where every node in the second and third layer have an equal number of sparse connections.



Fine Grained Structured Sparsity results in balanced workload distribution and better utilization of compute nodes. The pruned edges represented by the lighter colored connections would be represented by zero values in the weight matrices.

Figure 42. Fine Grained Structured Sparsity

Large amounts of research has been published by both academia and the AI industry on sparsity. But no standard practice of using sparsity has been established to optimize compute throughput without compromising accuracy. Fine-grained structured sparsity as implemented on NVIDIA A100, along with the simple and universal recipes provided by NVIDIA to sparsify deep neural networks, results in virtually no loss in accuracy based on evaluation across various popular neural networks. Table 11 below compares the accuracy achieved with fine tuning using 2:4 sparsity to training using dense matrices.

Table 11. Accuracy achieved on various networks with 2:4 fine grained structured sparsity

Neural Network	Accuracy achieved with Dense FP16 Matrices	Accuracy achieved with 2:4 sparse FP16 Matrices
Image Classification. Training Dataset - Imagenet. Accuracy metric = Top-1		
ResNet-50	76.6	76.8
Inception v3	77.1	77.1
Wide ResNet-50	78.5	78.4
VGG19	75.0	75.0
ResNeXt-101-32x8d	79.3	79.5
Image Segmentation and Detection. Training Dataset - COCO 2017. Accuracy metric = bbox AP		
MaskRCNN-ResNet-50	37.9	37.9
SSD-R50	24.8	24.8
Natural Language Processing. Training Dataset - En-De WMT'14. Accuracy metric = BLEU score		
GNMT	24.6	24.9
FairSeq Transformer	28.2	28.5
Natural Language Modeling. Accuracy metric = BPC for Transformer XL on enwik8 and F1 score for BERT on SQuAD v1.1		
Transformer XL	1.06	1.06
BERT Base	87.6	88.1
BERT Large	91.1	91.5

Notice

Trademarks

Copyright