UNITED STATES DISTRICT COURT
FOR THE WESTERN DISTRICT OF TEXAS
MIDLAND-ODESSA DIVISION

NEURAL AI, LLC	)	
	)	
	)	
Plaintiff,	)	
v.	)	Civil Action No. 7:24-cv-00221
	)	
NVIDIA CORPORATION	)	
	)	JURY TRIAL DEMANDED
	)	
Defendant.	)	
	)	

FIRST AMENDED COMPLAINT FOR PATENT INFRINGEMENT

Neural AI, LLC ("Neural AI" or "Plaintiff") alleges against Defendant Nvidia Corporation ("Nvidia" or "Defendant") the following:

1. This case involves patented technologies that revolutionized, and have become widely adopted in, the field of graphical processor unit ("GPU")-accelerated computing for artificial intelligence, machine learning, and complex numerical simulations. GPU-accelerated computing powers many of the most advanced and powerful forms of artificial intelligence that have exploded over the past decade.

2. Highly complex numerical simulations, such as the prediction of protein chains, genetic sequences and cryptographic sequences, and advanced machine learning techniques such as deep learning neural networks, require hardware capable of a high degree of parallel processing for efficient computation. GPUs, which generally have hundreds to thousands more computational processors or "cores" than central processing units ("CPUs"), are the preferred hardware for executing such simulations and machine learning techniques. Indeed, the parallel operation of thousands of high-performance GPUs have become a basic necessity for the execution and training

Neural AI, LLC, Exhibit 2006
Page 2006 - 1
IPR2025-00608, NVIDIA Corporation v. Neural AI, LLC

of complex natural language and image-generation models, such as ChatGPT's GPT-4 and Sora.AI. (*See* https://www.fierceelectronics.com/sensors/chatgpt-runs-10k-nvidia-training-gpus-potential-thousands-more.)

3. Before Plaintiff's innovations, the conventional wisdom in the field of GPU-accelerated computing was that the exchange of intermediate outputs between a GPU and a CPU was too computationally expensive. This was so because the GPU, adapted for highly parallel processing tasks (*e.g.*, graphically modeling a physics engine or rendering complex moving images), was ill-suited for handling operations better left to the CPU, like interacting with a user's mouse and keyboard or sending and receiving simple datasets. Plaintiff's foundational technology changed this by inventing techniques that leveraged the unique advantages of *both* the CPU and the GPU to enable their efficient interplay in hardware-accelerated computing.

4. Plaintiff's patented technologies are enshrined in U.S. Patent Nos. 8,648,867 ("the '867 Patent"), RE49,461 ("the '461 Patent"), and RE48,438 ("the '438 Patent") (collectively, "the Asserted Patents" or "The GPU-Based Acceleration Patents").

NATURE OF THE CASE

5. Plaintiff brings claims under the patent laws of the United States, 35 U.S.C. § 1, et seq., for infringement of the Asserted Patents. Defendant has infringed and continues to infringe each of the Asserted Patents under at least 35 U.S.C. §§271(a), 271(b) and 271(c).

THE PARTIES

6. Plaintiff Neural AI, LLC, is the owner by assignment of each of the Asserted Patents.

7. The technology of the Asserted Patents underpins multiple artificial intelligence and accelerated computing products that incorporate the patented technology, such as Neurala,

Inc.'s Vision Inspection Automation (VIA), Vision AI software, and Brain Builder platform.

8. Neural AI is a Texas limited liability company and is a registered business in Texas. Neural AI maintains its principal office in this District, at 510 Austin Avenue, Suite 2554, Waco, TX 76701.

9. Defendant Nvidia Corporation is a Delaware corporation with its headquarters and principal place of business in Santa Clara, California. (*See* https://investor.nvidia.com/financial-info/sec-filings/sec-filings-details/default.aspx?FilingId=17293267, U.S. Securities and Exchange Commission Form 10-K for Fiscal Year Ended January 28, 2024; https://nvidianews.nvidia.com/multimedia/santa-clara-headquarters.) Defendant Nvidia Corporation is registered with the Secretary of State to conduct business in Texas. Nvidia has an office in this District located in Austin, Texas. (*See* https://www.nvidia.com/en-us/contac.)

JURISDICTION & VENUE

10. This action arises under the Patent Laws of the United States, 35 U.S.C. § 1, *et seq*. The Court has subject matter jurisdiction pursuant to 28 U.S.C. §§ 1331 and 1338(a).

11. This Court has personal jurisdiction over Defendant because it regularly conducts business in the State of Texas and in this District. This business includes operating systems, using and/or providing computer hardware, software, firmware, and platforms, and/or providing services and/or engaging in activities in Texas and in this District that infringe one or more claims of the Asserted Patents, as well as inducing and contributing to the direct infringement of others through acts in this District.

12. Nvidia has also, directly and through its extensive network of partnerships, including with local IT service providers, purposefully and voluntarily placed products and/or provided services that practice and/or implement the methods, systems, and apparatuses claimed

in the Asserted Patents into the stream of commerce with the intention and expectation that they will be purchased and used by customers in this District, as detailed below. (*See* https://www.nvidia.com/en-us/about-nvidia/partners/.)

13. Defendant has also acknowledged that this Court has personal jurisdiction over it in cases filed against it in this District. (*See, e.g.*, *Vantage Micro LLC v. NVIDIA Corporation*, Case No. 6:19-cv-00582-RP, ECF 22 (W.D. Tex., Jan. 4, 2020) (admitting to personal jurisdiction); *Ocean Semiconductor LLC v. NVIDIA Corporation*, Case No. 6:20-cv-01211-ADA, ECF 14 (W.D. Tex., Mar. 12, 2021) (same).) Defendant has admitted "it is subject to this Court's general personal jurisdiction." (*Id.*)

14. Venue is proper in this District pursuant to 28 U.S.C. §§ 1391(b) and (c) and 28 U.S.C. § 1400(b) because Defendant Nvidia Corporation has regular and systematic contacts within this District and has committed acts of infringement within this District.

15. Defendant Nvidia Corporation is a registered business in Texas and has regular and established places of business in this District. Nvidia has an office in this District located at 11001 Lakeline Blvd, Suite 100 Bldg. 2, Austin, Texas 78717. (*See* https://craft.co/nvidia.) Nvidia's Austin office has "54,000 SF of new shell office and DVS labs" and "35,000 SF of offices, testing and software labs." (*See* https://kiddgrp.com/project/nvidia-corporation/.)

16. Defendant Nvidia Corporation has hundreds of employees in this District—including positions in engineering, sales, marketing, and finance. LinkedIn lists approximately 792 persons associated with Nvidia and identified as being located in the Austin or Austin metropolitan area. (*See* https://www.linkedin.com/company/nvidia/people/?facetGeoRegion=104472865%2C90000064.) LinkedIn also lists approximately 1,158 persons associated with Nvidia and identified as being

located in the State of Texas. (*See id.*)

17. In addition, Defendant Nvidia Corporation has over 100 jobs posted for the State of Texas on its affiliated Workday page with approximately 93 of those jobs—the vast majority of which are engineering jobs—listed for Austin, Texas. (*See* https://nvidia.wd5.myworkdayjobs.com/NVIDIAExternalCareerSite?locations=91336993fab910af6d702939a7fcc2d9&locations=91336993fab910af6d702b631b94c2de (approximately 111 Nvidia job postings for Texas).) These jobs are particularly relevant to the Asserted Patents and Accused Products, as defined below, because they pertain to artificial intelligence, machine learning, deep learning, data centers, accelerated computing, high performance computing ("HPC"), and related hardware, software, and/or firmware—including Nvidia's GPUs, CPUs, systems-on-a-chip ("SoCs"), platforms, and application programming interfaces.

18. Nvidia's operations in this District include client outreach and sales for each of the Accused Products and related or supporting services. As detailed above, Nvidia has customer-facing personnel and operations in this District. Nvidia also provides technical support to partners and customers for its products in the District.

19. Nvidia has committed acts of infringement within this District. Nvidia uses the Accused Products in this District in manners that practice the Asserted Patents, including by testing the Accused Products and by using the Accused Products at its offices and premises in this District.

20. Defendant makes, uses, advertises, offers for sale, and/or sells hardware for accelerated computing, including GPUs, CPUs, and SoCs; computers for accelerated computing (*e.g.*, supercomputers, servers, and data centers for high performance computing); and computer platform software-as-a-service ("SaaS") that implements accelerated computing (including the Accused Products) in the State of Texas and in this District directly and/or through its partnerships

with businesses in the State of Texas and in this District. Defendant also provides data center and HPC services that practice the Asserted Patents in the State of Texas and in this District directly and/or through its partnerships with businesses in the State of Texas and in this District.

21. Nvidia sells, offers for sale, advertises, makes, installs, and/or otherwise provides hardware, software, firmware, and/or computer platforms for accelerated computing and data center and HPC services, including the Accused Products, the use of which infringes the Asserted Patents in this District and the State of Texas. (*See* https://www.nvidia.com/en-us/data-center/solutions/accelerated-computing/.) Nvidia performs these acts directly and/or through its partnerships with other entities. (*See id.* ("NVIDIA has defined a range of accelerated platforms that each consist of hardware systems designed according to the needs of the use case as well as the software stack that enables the operation and management of the business applications. These hardware systems and software are available from NVIDIA and our partners.").)

22. Nvidia also uses a network of partners, which comprise re-sellers, managed service providers, and product and solution experts, to provide the Accused Products and implementation services for the Accused Products to customers in this District. Each of these partners sells, offers for sale, installs, and/or implements Nvidia's accelerated computing hardware, software, and/or computer platform services. (*See* https://www.nvidia.com/en-us/about-nvidia/partners/.)

23. Nvidia's partners include "Data Center Provider[s]." (*See* https://www.nvidia.com/en-us/about-nvidia/partners/.) Nvidia's Data Center Provider partners "offer colocation services such as high-density data center facilities, interconnected infrastructure, and state-of-art cooling technologies for hosting NVIDIA DGX™ servers globally." (*See id.*) Nvidia's Data Center Provider partners in the "NVIDIA DGX-Ready Data Center program, built on the NVIDIA DGX™ platform and delivered by NVIDIA partners," help "accelerate the scaling

of AI across [a customer's] organization." (*See* https://www.nvidia.com/en-us/data-center/colocation-partners/#aligned-energy.)

24. As further detailed below, Nvidia engages in activities that directly infringe the Asserted Patents within this District. For example, Nvidia's operation and use of its accelerated computing hardware, software, and/or computer platform services, including its data center-scale accelerated computing platforms, within this District infringe the Asserted Patents.

25. Nvidia also infringes (directly or indirectly) the Asserted Patents by providing services in connection with the Accused Products including installing, maintaining, supporting, operating, providing instructions, and/or advertising Nvidia's computer platform, data center, and HPC services within this District. For example, under Nvidia's cloud and data center line of products and services, the Nvidia DGX platform is a "a fully integrated hardware and software AI platform" and "combines the best of NVIDIA software, infrastructure, and expertise in a modern, unified AI development solution." (*See* https://www.nvidia.com/en-us/data-center/dgx-platform/.) Indeed, "DGX infrastructure is a complete AI solution, and includes NVIDIA AI Enterprise software to accelerate data science pipelines and streamline development and deployment of production-grade AI applications." (*See id.*) Nvidia platform user and partner customers infringe the Asserted Patents by installing and operating Nvidia's computer platform software, which performs the claimed methods in the Asserted Patents within this District. (*See also, e.g.,* https://www.nvidia.com/en-us/data-center/products/ai-enterprise/ (Nvidia AI Enterprise); https://developer.nvidia.com/cuda-zone (Nvidia CUDA Toolkit); https://www.nvidia.com/en-us/data-center/gpu-cloud-computing/ (GPU Cloud Computing).)

26. Defendant encourages and induces its customers of the Accused Products to perform the methods claimed in the Asserted Patents. For example, Nvidia makes its accelerated

computing platforms and services available on its website, widely advertises those platforms and services, provides applications that allow partners and users to access those platforms and services, provides instructions for installing, and maintaining those platforms and services and supporting software and/or firmware, and provides technical support to users. (*See* https://www.nvidia.com/en-us/data-center/dgx-support/.)

27. Nvidia further encourages and induces its customers to operate Nvidia's hardware and software in an infringing manner, and to use Nvidia's infringing computer platforms, by providing directions for and encouraging customers to install software, such as software for NVIDIA AI Enterprise and CUDA, (*see* https://docs.nvidia.com/ai-enterprise/deployment-guide-vmware/0.1.0/software.html; https://developer.nvidia.com/cuda-downloads), which offers evaluation, installation, configuration, customization, and development of Nvidia's infringing software products and services.

28. Defendant also contributes to the infringement of its customers and end users of the Accused Products by offering within the United States or importing into the United States the Accused Products, which are for use in practicing, and under normal operation practice, one or more of the methods claimed in the Asserted Patents, constituting a material part of the inventions claimed, and not a staple article or commodity of commerce suitable for substantial non-infringing uses. Indeed, as shown herein, the Accused Products and the example functionality described below have no substantial non-infringing uses and are specifically designed to practice the methods claimed in the Asserted Patents.

29. On information and belief, Defendant has not disputed that venue is proper in this District in cases filed against it in this District. (*See, e.g.*, *Vantage Micro LLC v. NVIDIA Corp.*, No. 6:19-cv-00582, ECF 22; *Polaris Innovations Ltd. v. Dell Inc. et al.*, No. 5:16-cv-00451, ECF

19; *Cirrus Logic, Inc. v. ATI Techs., et al.*, No. 1:03-cv-00302, ECF 6.)

30. Defendant's infringement adversely impacts Plaintiff in this District.

PLAINTIFF'S PATENTED INNOVATIONS

31. The Asserted Patents pioneered the adaptation of GPU-acceleration technology to the supervised execution of complex artificial intelligence algorithms and numerical simulations, such that it became possible for the first time to dynamically supervise, review, and correct intermediate "solutions" that were produced by these accelerated algorithms and simulations without performance loss.

The GPU-Based Acceleration Patents
U.S. Patent Nos. 8,648,867, RE49,461, and RE48,438

32. The '867, '461, and '438 Patents are part of the same patent family and generally disclose and claim systems and methods related to the accelerated execution of numerical simulations and neural networks such that the intermediate outputs of a given execution "step" can be dynamically transferred from the GPU to the CPU, reviewed, and corrected within the same computational cycle before being fed as inputs to the next execution step.

33. The '867 Patent is entitled "Graphic Processor Based Accelerator System and Method," was filed on September 24, 2007, and was duly and legally issued by the United States Patent and Trademark Office ("USPTO") on February 11, 2014. The '867 Patent claims priority to Provisional Application No. 60/826,892, filed on September 25, 2006. A true and correct copy of the '867 Patent is attached as Exhibit 1.

34. The '438 Patent is entitled "Graphic Processor Based Accelerator System and Method," was filed on November 9, 2017, and was duly and legally issued by the USPTO on February 16, 2021. The '438 Patent is a re-issue of the '867 Patent and claims priority to Provisional Application No. 60/826,892, filed on September 25, 2006. A true and correct copy of

the ’438 Patent is attached as Exhibit 2.

35. The ’461 Patent is also entitled “Graphic Processor Based Accelerator System and Method,” was filed on December 29, 2020, and was duly and legally issued by the USPTO on March 14, 2023. The ’461 Patent is a re-issue of the ’867 Patent and claims priority to Provisional Application No. 60/826,892, filed on September 25, 2006. A true and correct copy of the ’461 Patent is attached as Exhibit 3.

36. The ’867 Patent improves upon prior GPU acceleration technology by disclosing and claiming a novel hardware and firmware system for performing a numerical simulation that permits dynamic editing of the outputs that flow from intermediate “steps” of that simulation, before they become inputs to the next “step.” In particular, the ’867 patent discloses a CPU tethered to a GPU-based accelerator, each with their own corresponding memories, and an accelerator “controller” that coordinates transfers of data between the CPU and the GPU-based accelerator, such that the intermediate results from one step can be transferred from the GPU-based accelerator to the CPU, reviewed and corrected by the CPU, and transferred back to the GPU-based accelerator before the next computational cycle begins.

37. The ’867 Patent explains that performing the numerical computation in this stepwise fashion enables the system to eliminate “race conditions,” *i.e.*, conflicts that occur when two programmatic “threads” attempt to change the same shared data at the same time, which would otherwise occur when other system elements attempt to access intermediate outputs of the numerical computation. (*See* ’867 Patent, 5:60-6:31.) This avoids the computational overhead prevalent in conventional GPU-based accelerator architectures when transferring data from the accelerator to the CPU.

38. By enabling such "controller-driven data exchange" between the GPU-based accelerator and the CPU, the system described in the '867 Patent allows for an "input parser" executing on a CPU core to "change input…on the fly during the simulation," thus enabling automatic review and dynamic error correction of the numerical simulations or neural networks that are being executed on the claimed system. (*See id.*, 9:8-17.) Such dynamic, in-execution review and error correction of whether each intermediate "step" of a simulation or neural network is generating correct results is essential to the performance and reliability of large language models, image classification, and image generation models that have become prevalent today. Because of the scale to which such simulations and models have grown, it is no longer feasible to "restart" them from scratch, only to correct them as they execute.

39. The '461 and '438 Patents disclose hardware and firmware configurations similar to those of the '867 Patent, but are directed to using those configurations to process the layers of an artificial neural network ("ANN"). The '461 Patent is directed to further interplay between the CPU and the GPU-based accelerator: separating the CPU and GPU-based accelerator into separate "streams," whereby the CPU executes a "user interaction stream" (*e.g.*, enabling the parsing and dynamic editing of intermediate outputs, or for the ANN to be paused and resumed), while the accelerator executes a "computational stream" that executes the layers of the artificial neural network. When the ANN is initialized, control over the generation of outputs shifts to the computational stream. However, once a pre-defined layer of the ANN has completed execution, or is interrupted, control over the generation of outputs and feeding of inputs is shifted back to the CPU's user interaction stream.

40. The Asserted Patents describe this "shift of priorities" as "[t]he crucial feature of the interaction between the User Interaction Stream and the Computational Stream." ('867 Patent,

7:45-47.) Even though the computational stream is in control during the ANN computation, priority shifting enables "[t]he user [to] retain[] the ability to interrupt the simulation, change the input, or to change the display properties of the framework" because the user's "interactions are queued to be performed at times determined by the controller-driven data exchange to avoid corruption of the data." (*Id*., 8:47-57.)

ACCUSED PRODUCTS

41. Nvidia offers, sells, and uses several products that provide and implement GPU-acceleration hardware, software, platforms, and services for individuals and enterprises and incorporate Plaintiff's patented technologies. (*See* https://www.nvidia.com/en-us/solutions/ai/inference/; https://marketplace.nvidia.com/en-us/data-center/?page=4; https://marketplace.nvidia.com/en-us/laptops-workstations/?page=9; https://marketplace.nvidia.com/en-us/software/?page=3.)

42. The Accused Products include Nvidia's GPU accelerators and superchips. (*See* https://resources.nvidia.com/l/en-us-gpu.) Nvidia's GPU accelerators include Nvidia's GPUs with Nvidia's "Hopper," "Ada Lovelace," "Ampere," "Turing," "Volta," "Pascal," and "Maxwell" GPU architectures. (*See* https://docs.nvidia.com/deeplearning/cudnn/archives/cudnn-896/support-matrix/index.html.) These GPUs are specifically designed to run and implement GPU-based hardware acceleration using Nvidia's proprietary CUDA (Compute Unified Device Architecture) platform and CUDA libraries for GPU acceleration. (*See id.* (Nvidia GPU architectures implementing Nvidia's cuDNN (CUDA Deep Neural Network) library for GPU acceleration.); https://developer.nvidia.com/cuda-gpus.)

43. Nvidia's Hopper GPUs include the H100 and H200 GPUs. (*See* https://www.nvidia.com/en-us/data-center/technologies/hopper-architecture/ (Hopper

architecture); https://www.nvidia.com/en-us/data-center/h100/ (H100); https://www.nvidia.com/en-us/data-center/h200/ (H200).) In addition, Nvidia's superchips that implement GPU accelerators include the GH200, or Grace Hopper Superchip, which implements the Hopper-GPU architecture. (*See* https://www.nvidia.com/en-us/data-center/grace-hopper-superchip/ (GH200).)

44. Nvidia's Ada Lovelace (or Lovelace) GPUs include Nvidia Data Center GPUs, including L40, L40S, and L4 GPUs; Nvidia Workstation and Professional Laptop GPUs, including RTX Ada Generations series GPUs and Laptop GPUs; and GeForce RTX 40 series GPUs and Laptop GPUs. (*See* https://www.nvidia.com/en-us/technologies/ada-architecture/ (Ada Lovelace architecture). *See* https://www.nvidia.com/en-us/data-center/l40/ (L40); https://www.nvidia.com/en-us/data-center/l40s/ (L40S); https://www.nvidia.com/en-us/data-center/l4/ (L4). *See* https://resources.nvidia.com/en-us-design-viz-stories-ep/l40-linecard (Nvidia Professional GPUs); https://www.nvidia.com/en-us/ai-on-rtx/ (RTX GPUs featuring "Accelerated Development"); https://www.nvidia.com/en-us/design-visualization/desktop-graphics/ (RTX Ada Generation GPUs); https://www.nvidia.com/en-us/design-visualization/rtx-professional-laptops/compare-table/ (RTX Ada Generation Laptop GPUs). *See* https://www.nvidia.com/en-us/geforce/graphics-cards/40-series/ (GeForce RTX 40 GPUs); https://www.nvidia.com/en-us/geforce/graphics-cards/compare/ (GeForce RTX 40 GPUs); https://www.nvidia.com/en-us/geforce/laptops/compare/ (GeForce RTX 40 Laptop GPUs).)

45. Nvidia's Ampere GPUs include Nvidia Data Center GPUs, including A100, A40, A30, A16, A10, and A2 GPUs; Nvidia Workstation and Professional Laptop GPUs, including RTX A series GPUs and Laptop GPUs; GeForce RTX 30 series GPUs and Laptop GPUs; and GeForce MX570 Laptop GPU. (*See* https://www.nvidia.com/en-us/data-center/ampere-

architecture/ (Ampere architecture). *See* https://www.nvidia.com/en-us/data-center/a100/ (A100); https://www.nvidia.com/en-us/data-center/a40/ (A40); https://www.nvidia.com/en-us/data-center/a30/ (A30); https://www.nvidia.com/en-us/data-center/a16/ (A16); https://www.nvidia.com/en-us/data-center/a10/ (A10); https://www.nvidia.com/en-us/data-center/a2/ (A2). *See* https://www.nvidia.com/en-us/design-visualization/desktop-graphics/ (RTX A GPUs); https://www.nvidia.com/en-us/design-visualization/rtx-professional-laptops/compare-table/ (RTX A Laptop GPUs). *See* https://www.nvidia.com/en-us/geforce/graphics-cards/30-series/ (GeForce RTX 30 GPUs); https://www.nvidia.com/en-us/geforce/graphics-cards/compare/ (GeForce RTX 30 GPUs); https://www.nvidia.com/en-us/geforce/laptops/compare/30-series/ (GeForce RTX 30 Laptop GPUs); https://www.nvidia.com/en-us/geforce/gaming-laptops/mx-570/ (GeForce MX570 Laptop GPU).)

46. Nvidia's Turing GPUs include Nvidia Data Center GPUs, including Tesla T4 GPUs and Quadro RTX 8000 (passive) and Quadro RTX 6000 (passive) GPUs; Nvidia Workstation and Professional Laptop GPUs, including T series GPUs and Laptop GPUs, Quadro T series Laptop GPUs, and Quadro RTX series GPUs and Laptop GPUs; Titan series Titan RTX GPU; GeForce RTX 20 series and GeForce GTX 16 series GPUs and Laptop GPUs; and GeForce MX550, MX450, and MX430 Laptop GPUs. (*See* https://www.nvidia.com/en-us/geforce/turing/ (Turing architecture). *See* https://www.nvidia.com/en-us/data-center/tesla-t4/ (Tesla T4); https://www.nvidia.com/en-gb/design-visualization/quadro-data-center/ (Quadro RTX 8000 (passive) and Quadro RTX 6000 (passive). *See* https://www.nvidia.com/en-us/design-visualization/quadro/ (T series GPUs/Laptop GPUs, Quadro T series Laptop GPUs, and Quadro RTX GPUs/Laptop GPUs); https://www.nvidia.com/en-us/design-visualization/desktop-graphics (T series GPUs/Laptop GPUs); https://www.nvidia.com/content/dam/en-

zz/Solutions/titan/documents/titan-rtx-for-creators-us-nvidia-1011126-r6-web.pdf (Titan RTX); https://www.nvidia.com/en-us/geforce/20-series/ (GeForce RTX 20 GPUs); https://www.nvidia.com/en-us/geforce/graphics-cards/compare/ (GeForce RTX 20 GPUs and GeForce GTX 16 GPUs); https://www.nvidia.com/en-us/geforce/gaming-laptops/compare-20-series/ (GeForce RTX 20 Laptop GPUs); https://www.nvidia.com/en-us/geforce/gaming-laptops/compare-16-series/ (GeForce GTX 16 Laptop GPUs); https://www.nvidia.com/en-us/geforce/gaming-laptops/mx-550/ (GeForce MX550 Laptop GPU); https://www.nvidia.com/en-us/geforce/gaming-laptops/mx-450/ (GeForce MX450 Laptop GPU); https://wccftech.com/nvidia-geforce-mx450-turing-discrete-notebook-gpu-gddr6-pcie-4/ (GeForce M Laptop GPUs).)

47. Nvidia's Volta GPUs include Nvidia Data Center GPUs, including the Tesla V100 GPU; Nvidia Workstation GPUs, including Quadro GV100; and Titan series Titan V GPU. (*See* https://www.nvidia.com/en-us/data-center/volta-gpu-architecture/ (Volta architecture); https://www.nvidia.com/en-us/data-center/v100/ (Tesla V100); https://www.nvidia.com/content/dam/en-zz/Solutions/design-visualization/productspage/quadro/quadro-desktop/quadro-volta-gv100-data-sheet-us-nvidia-704619-r3-web.pdf (Quadro GV100); https://nvidianews.nvidia.com/news/nvidia-titan-v-transforms-the-pc-into-ai-supercomputer (Titan V).)

48. Nvidia's Pascal GPUs include Nvidia Data Center GPUs, including Tesla P100, P40, and P4 GPUs; Nvidia Workstation and Professional Laptop GPUs, including the Quadro GP100 GPU and Quadro P series GPUs and Laptop GPUs; Titan series Titan Xp and Titan X GPUs; GeForce GTX 10 series GPUs and Laptop GPUs; and GeForce MX300 series, MX200 series, and MX150 Laptop GPUs. (*See* https://developer.nvidia.com/pascal;

https://www.nvidia.com/en-us/data-center/pascal-gpu-architecture/ (Pascal architecture). *See* https://www.nvidia.com/en-us/data-center/tesla-p100 (Tesla P100); https://developer.nvidia.com/cuda-gpus (Tesla P40 and P4); https://www.nvidia.com/content/dam/en-zz/Solutions/design-visualization/productspage/quadro/quadro-desktop/quadro-pascal-gp100-data-sheet-us-nv-704562-r1.pdf (Quadro GP100); https://www.nvidia.com/en-us/design-visualization/quadro/ (Quadro P series GPUs/Laptop GPUs). *See* https://www.nvidia.com/content/geforce-gtx/NVIDIA_TITAN_X_USER_GUIDE_v02.pdf (Titan X); https://www.nvidia.com/content/geforce-gtx/NVIDIA_TITAN_Xp_USER_GUIDE_v02.pdf (Titan Xp); https://www.nvidia.com/en-us/geforce/10-series/ (GeForce GTX 10); https://www.nvidia.com/en-us/geforce/graphics-cards/compare/ (GeForce GTX 10 GPUs); https://www.nvidia.com/en-us/geforce/news/gfecnt/nvidia-geforce-gtx-10-series-laptops/ (GeForce GTX 10 Laptop GPUs); https://www.nvidia.com/en-us/geforce/gaming-laptops/mx-350/ (GeForce MX350 Laptop GPU); https://www.nvidia.com/en-us/geforce/gaming-laptops/mx-330/ (GeForce MX330 Laptop GPU); https://wccftech.com/nvidia-geforce-mx450-turing-discrete-notebook-gpu-gddr6-pcie-4/ (GeForce M Laptop GPUs).)

49. Nvidia's Maxwell GPUs include Nvidia Data Center GPUs, including Tesla M60, M40, and M10 GPUs; Nvidia Workstation and Professional Laptop GPUs, including Quadro M series GPUs and Laptop GPUs, the NVS 810 GPU, and Tesla M6 series Laptop GPUs; Titan series GTX Titan X GPU; GeForce GTX 900 series and GeForce GTX 700 series GPUs and Laptop GPUs; and GeForce MX130 series and MX110 Laptop GPUs. (*See* https://developer.nvidia.com/blog/maxwell-most-advanced-cuda-gpu-ever-made/ (Maxwell architecture); https://www.nvidia.com/content/dam/en-zz/Solutions/design-

visualization/solutions/resources/documents1/nvidia-m60-datasheet.pdf (M60); https://images.nvidia.com/content/tesla/pdf/78071_Tesla_M40_24GB_Print_Datasheet_LR.PDF (M40); https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/tesla-m10/pdf/188359-Tesla-M10-DS-NV-Aug19-A4-fnl-Web.pdf (M10); https://www.nvidia.com/en-us/design-visualization/quadro/ (Quadro M GPUs/Laptop GPUs); https://www.nvidia.com/docs/IO/146527/nvs-810-datasheet.pdf (NVS 810); https://images.nvidia.com/content/tesla/pdf/188300-Tesla-M6-DS-Aug19-A4-fnl-Web.pdf (Tesla M6); https://www.nvidia.com/content/geforce-gtx/GTX_TITAN_X_User_Guide.pdf (GTX Titan X); https://developer.nvidia.com/maxwell-compute-architecture (GeForce GTX 900 and 700 GPUs/Laptop GPUs); https://wccftech.com/nvidia-geforce-mx450-turing-discrete-notebook-gpu-gddr6-pcie-4/ (GeForce M Laptop GPUs).)

50. These GPUs and superchips implement, and are specifically designed for, GPU-acceleration for artificial intelligence and neural networks. Nvidia's proprietary CUDA platform for parallel computing, which includes GPU-acceleration libraries such as cuDNN (CUDA Deep Neural Network), is implemented in the Nvidia Hopper, Ada Lovelace, Ampere, Turing, Volta, Pascal, and Maxwell GPU architectures.

1. GPU, CUDA Toolkit, and CUDA Driver Requirements

The following sections highlight the compatibility of NVIDIA® cuDNN versions with the various supported NVIDIA CUDA® Toolkit, CUDA driver, and NVIDIA hardware versions.

Table 1. GPU, CUDA Toolkit, and CUDA Driver Requirements

cuDNN Package[1]	CUDA Toolkit Version	Supports static linking?[2]	NVIDIA Driver Version		CUDA Compute Capability	Supported NVIDIA Hardware
			Linux	Windows		
cuDNN 8.9.6 for CUDA 12.x	12.2	Yes	>=525.60.13	>=527.41	9.0[3]	
	12.1	No			8.9[4]	NVIDIA Hopper™[5]
	12.0				8.6	NVIDIA Ada Lovelace architecture[5]
cuDNN 8.9.6 for CUDA 11.x	11.8	Yes	>= 450.80.02	>=452.39	8.0	NVIDIA Ampere architecture
	11.7	No			7.5	NVIDIA Turing™
	11.6				7.0	NVIDIA Volta™
	11.5				6.1	NVIDIA Pascal™
	11.4				6.0	NVIDIA Maxwell[6]
	11.3				5.0	
	11.2[7]					
	11.1[8]					
	11.0[9]					

(*See* https://docs.nvidia.com/deeplearning/cudnn/archives/cudnn-896/support-matrix/index.html (emphasis added).)

51. The Accused Products further include Nvidia's supercomputers and servers that implement its GPU accelerators and superchips. These supercomputers and servers include: the EGX line of servers for data centers and edge devices, the HGX line of supercomputers, the DGX line of supercomputers, and the OVX line of supercomputers. (*See* https://www.nvidia.com/en-us/data-center/solutions/accelerated-computing/.)

52. Nvidia's "EGX hardware portfolio" includes "accelerators [that] combine the performance of NVIDIA Ampere GPUs." (*See* https://www.nvidia.com/en-us/data-center/products/egx/; *see* https://www.nvidia.com/en-us/design-visualization/egx-graphics/.) Nvidia's HGX "AI supercomputing platform brings together the full power of NVIDIA GPUs, NVIDIA NVLink™, NVIDIA networking, and fully optimized AI and high-performance computing (HPC) software stacks." (*See* https://www.nvidia.com/en-us/data-center/hgx/; https://nvdam.widen.net/s/5kgbjq2v2t/hpc-hgx-h100-datasheet-nvidia-web.) One example

configuration includes "four or eight H200 or H100 GPUs." (*See id.*; *see* https://nvdam.widen.net/s/5kgbjq2v2t/hpc-hgx-h100-datasheet-nvidia-web.) Nvidia's DGX supercomputers include the DGX H200, DGX BasePOD, and DGX SuperPOD with DGX GB200. (*See* https://www.nvidia.com/en-us/data-center/dgx-platform/; *see also* https://www.nvidia.com/en-us/data-center/base-command/; https://resources.nvidia.com/en-us-dgx-software/nvidia-base-command (DGX Base Command operating system for DGX data centers.) And Nvidia's OVX supercomputers implement "L40S GPUs . . . for both complex AI and graphics-intensive workloads." (*See* https://www.nvidia.com/en-us/data-center/products/ovx/; *see* https://resources.nvidia.com/en-us-ovx/ovx-datasheet.)

53. The Accused Products further include Nvidia's software, platforms, and services for accelerated computing. These include CUDA, Nvidia AI Enterprise, the DGX Platform, Nvidia Omniverse, Nvidia Drive, Nvidia Isaac Sim, and Nvidia NGC.

54. CUDA is Nvidia's proprietary "parallel computing platform and programming model." (*See* https://developer.nvidia.com/cuda-zone.) CUDA is designed to support Nvidia's GPU accelerators and superchips and includes software specifically for GPU-acceleration such as the cuDNN "GPU-accelerated library." (*See id.*; https://developer.nvidia.com/cudnn.) In addition, Nvidia's CUDA-X, built on top of CUDA, is a collection of "GPU-accelerated microservices and libraries for AI." (*See* https://www.nvidia.com/en-us/technologies/cuda-x/.) Nvidia also offers the CUDA Toolkit and SDK Manager for developing GPU-accelerated applications. (*See* https://developer.nvidia.com/cuda-toolkit; https://developer.nvidia.com/sdk-manager.)

55. In addition, Nvidia AI Enterprise is Nvidia's "end-to-end, cloud-native software platform" for "accelerat[ing] data science pipelines . . . and other generative AI applications." (*See* https://www.nvidia.com/en-us/data-center/products/ai-enterprise/.) It is Nvidia's "'operating

system' for enterprise AI." (*See id.*)

56. In addition, Nvidia's DGX platform is "is a complete AI solution, and includes NVIDIA AI Enterprise software." (*See* https://www.nvidia.com/en-us/data-center/dgx-platform/.) Nvidia DGX Cloud is "an AI-training-as-a-service platform which includes cloud-based infrastructure and software for AI, customizable pretrained AI models, and access to NVIDIA experts." (*See* https://d18rn0p25nwr6d.cloudfront.net/CIK-0001045810/1cbe8fe7-e08a-46e3-8dcc-b429fc06c1a4.pdf, Nvidia U.S. Securities and Exchange Commission Form 10-K for Fiscal Year Ended January 28, 2024 at 6.)

57. In addition, Nvidia Omniverse is "a development platform and operating system for building virtual world simulation applications, available as a software subscription." (*See* https://d18rn0p25nwr6d.cloudfront.net/CIK-0001045810/1cbe8fe7-e08a-46e3-8dcc-b429fc06c1a4.pdf, Nvidia U.S. Securities and Exchange Commission Form 10-K for Fiscal Year Ended January 28, 2024 at 6.) Nvidia Omniverse implements software and services "into existing software tools and simulation workflows for building AI systems." (*See* https://www.nvidia.com/en-us/omniverse/.)

58. In addition, Nvidia Drive is a platform that "consists of both the AI infrastructure and in-vehicle hardware and software" for autonomous vehicles. (*See* https://www.nvidia.com/en-us/self-driving-cars/.) "NVIDIA DRIVE Infrastructure encompasses data center hardware, software, and workflows—both on premises and in NVIDIA DGX Cloud & Omniverse." (*See id.*)

59. In addition, Nvidia Isaac Sim is a platform that enables "developers to design, simulate, test, and train AI-based robots and autonomous machines in a physically-based virtual environment." (*See* https://developer.nvidia.com/isaac/sim.) It is built on Nvidia Omniverse. (*See id.*)

60. In addition, Nvidia NGC is a collection of software services and tools that support "end-to-end AI and digital twin workflows" that runs on "NVIDIA GPU-accelerated platforms." (*See* https://www.nvidia.com/en-us/gpu-cloud/.) NGC "offers a collection of cloud services . . . for generative AI, drug discovery, and speech AI solutions, and the NGC Private Registry for securely sharing proprietary AI software." (*See id.*)

FIRST CAUSE OF ACTION
(INFRINGEMENT OF THE '867 PATENT)

61. Plaintiff realleges and incorporates by reference the allegations of the preceding paragraphs of this Complaint.

62. Defendant has infringed and continues to infringe one or more claims of the '867 Patent in violation of 35 U.S.C. § 271 in this District and elsewhere in the United States and will continue to do so. The Accused Products, including features of, *e.g.*, the Grace Hopper Superchip (GH200), at least when used for their ordinary and customary purposes, practice each element of at least claim 16 of the '867 Patent as demonstrated below.

63. For example, claim 16 of the '867 Patent recites:

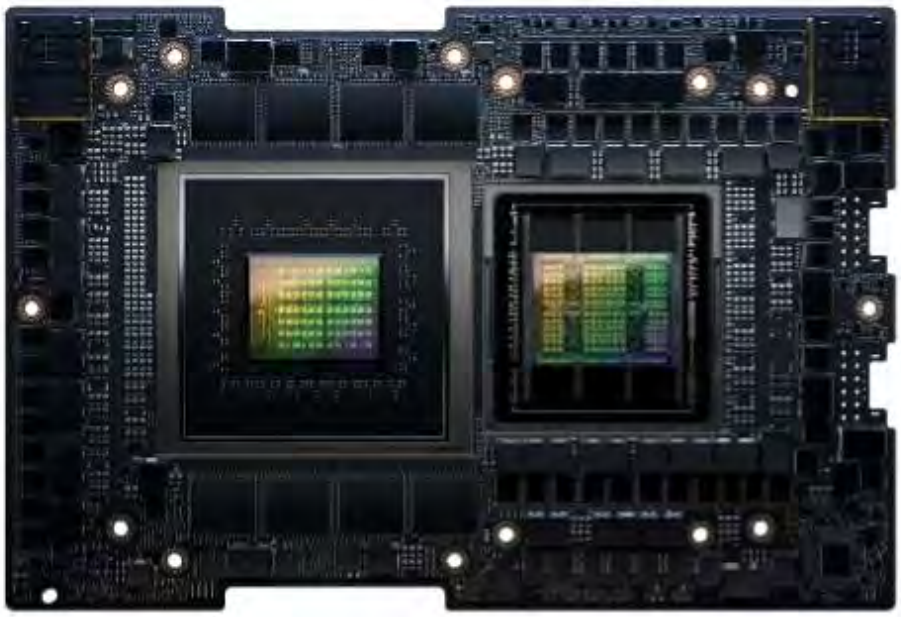
> 16. A method for performing a numerical simulation on input data in a computer system including a central processing unit and an accelerator, the method comprising:
>
> receiving, by an accelerator, first input data from the central processing unit;
>
> transferring, by an accelerator controller, the first input data into a first partition, referenced by first pointer, of an accelerator memory before a first computational cycle of the numerical simulation;
>
> performing, by at least one graphics processing unit during the first computational cycle, at least one calculation on the first portion of the input data as to generate first output data;

storing, by the accelerator controller, the first output data into a second partition, referenced by a second pointer, of the accelerator memory; and

swapping the first pointer with the second pointer at the end of the first computational cycle, such that the first output data becomes an input for a second computational cycle of the numerical simulation.

64. The Accused Products perform each step of the method of claim 16 of the ’867 Patent. To the extent the preamble is construed to be limiting, the Accused Products perform *a method for performing a numerical simulation on input data in a computer system including a central processing unit and an accelerator*, as further explained below. For instance, the Grace Hopper Superchip (GH200) “brings together the groundbreaking performance of the NVIDIA Hopper GPU with the versatility of the NVIDIA Grace™ CPU . . . in a single Superchip.”

Inside NVIDIA’s First GPU-CPU Superchip



The NVIDIA® GH200 Grace Hopper architecture brings together the groundbreaking performance of the NVIDIA Hopper GPU with the versatility of the NVIDIA Grace™ CPU, connected with a high bandwidth and memory coherent NVIDIA NVLink Chip-2-Chip (C2C)® interconnect in a single Superchip, and support for the new NVIDIA NVLink Switch System.

(*See* https://resources.nvidia.com/en-us-grace-cpu/nvidia-grace-hopper (emphasis added).)

65. The "Grace Hopper Superchip is the first true heterogeneous accelerated platform for high-performance computing (HPC) and AI workloads. It ***accelerates*** applications with the strengths of both GPUs and CPUs while providing the simplest and most productive heterogeneous programming model to date."

The NVIDIA GH200 Grace Hopper Superchip is the first true heterogeneous accelerated platform for high-performance computing (HPC) and AI workloads. It accelerates applications with the strengths of both GPUs and CPUs while providing the simplest and most productive heterogeneous programming model to date, enabling scientists and engineers to focus on solving the world's most important problems. Together with NVIDIA networking technologies, NVIDIA GH200 provides the recipe for the next generation of HPC supercomputers and AI factories, enabling customers to take on larger datasets, more complex models, and new workloads, solving them more quickly than before.

(*See* https://resources.nvidia.com/en-us-grace-cpu/nvidia-grace-hopper (emphasis added).)

66. In addition, the Accused Products, including the Grace Hopper Superchip, implement CUDA, Nvidia's proprietary "parallel computing platform and programming model." CUDA enables NVIDIA GPUs to be used for general purpose computing tasks. CUDA further includes the CUDA Toolkit, which "includes GPU-accelerated libraries, a compiler, development tools and the CUDA runtime." As an example, the "CUDA® Deep Neural Network library (cuDNN) is a GPU-acceleration library of primitives for deep neural networks." It "provides highly tuned implementations for standard routines" for GPU-based acceleration.

CUDA Zone

CUDA® is a parallel computing platform and programming model developed by NVIDIA for general computing on graphical processing units (GPUs). With CUDA, developers are able to dramatically speed up computing applications by harnessing the power of GPUs.

In GPU-accelerated applications, the sequential part of the workload runs on the CPU – which is optimized for single-threaded performance – while the compute intensive portion of the application runs on thousands of GPU cores in parallel. When using CUDA, developers program in popular languages such as C, C++, Fortran, Python and MATLAB and express parallelism through extensions in the form of a few basic keywords.

The CUDA Toolkit from NVIDIA provides everything you need to develop GPU-accelerated applications. The CUDA Toolkit includes GPU-accelerated libraries, a compiler, development tools and the CUDA runtime.

(*See* https://developer.nvidia.com/cuda-zone (emphasis added).)

NVIDIA cuDNN

The NVIDIA CUDA® Deep Neural Network library (cuDNN) is a GPU-accelerated library of primitives for deep neural networks. cuDNN provides highly tuned implementations for standard routines such as forward and backward convolution, attention, matmul, pooling, and normalization.

(*See* https://developer.nvidia.com/cudnn (emphasis added).)

67. Nvidia GPU architectures that implement CUDA and cuDNN include the Hopper (*e.g.*, Grace Hopper Superchip (GH200), H100), Ada Lovelace, Ampere, Turing, Volta, Pascal, and Maxwell GPU architectures of the Accused Products.

1. GPU, CUDA Toolkit, and CUDA Driver Requirements

The following sections highlight the compatibility of NVIDIA® cuDNN versions with the various supported NVIDIA CUDA® Toolkit, CUDA driver, and NVIDIA hardware versions.

Table 1. GPU, CUDA Toolkit, and CUDA Driver Requirements

cuDNN Package[1]	CUDA Toolkit Version	Supports static linking?[2]	NVIDIA Driver Version		CUDA Compute Capability	Supported NVIDIA Hardware
			Linux	Windows		
cuDNN 8.9.6 for CUDA 12.x	12.2	Yes	>=525.60.13	>=527.41	9.0[3]	
	12.1	No			8.9[4]	NVIDIA Hopper™[5]
	12.0				8.6	NVIDIA Ada Lovelace architecture[6]
cuDNN 8.9.6 for CUDA 11.x	11.8	Yes	>= 450.80.02	>=452.39	8.0	NVIDIA Ampere architecture
	11.7	No			7.5	NVIDIA Turing™
	11.6				7.0	NVIDIA Volta™
	11.5				6.1	NVIDIA Pascal™
	11.4				6.0	NVIDIA Maxwell[6]
	11.3				5.0	
	11.2[7]					
	11.1[8]					
	11.0[9]					

(*See* https://docs.nvidia.com/deeplearning/cudnn/archives/cudnn-896/support-matrix/index.html (emphasis added).)

68. The Accused Products perform a method that includes *receiving, by an accelerator, first input data from the central processing unit*. For instance, the "CUDA programming model" implements programming functions and instructions for CPUs and GPUs. "The host is the CPU available in the system" and "system memory associated with the CPU is called host memory." "The GPU is called a device and GPU memory likewise called device memory." As an example, the first main CUDA program execution step is "[c]opy[ing] the ***input data from host [CPU] memory to device [GPU] memory***, also known as host-to-device transfer."

Let me introduce two keywords widely used in CUDA programming model: *host* and *device*.

The host is the CPU available in the system. The system memory associated with the CPU is called host memory. The GPU is called a device and GPU memory likewise called device memory.

To execute any CUDA program, there are three main steps:

- Copy the input data from host memory to device memory, also known as host-to-device transfer.
- Load the GPU program and execute, caching data on-chip for performance.
- Copy the results from device memory to host memory, also called device-to-host transfer.

(*See* https://developer.nvidia.com/blog/cuda-refresher-cuda-programming-model/ (emphasis added).)

69. The Accused Products practice a method that includes *transferring, by an accelerator controller, the first input data into a first partition, referenced by first pointer, of an accelerator memory before a first computational cycle of the numerical simulation*. For instance, the GPU architecture of the Accused Products implements a ***controller***. As an example, the Hopper-GPU architecture implements "HBM3 memory ***controllers***" including "12 512-bit memory ***controllers***" coupled GPU memory including "6 HBM3 or HBM2e stacks," "80 GB HBM3, 5 HBM3 stacks," and "80 GB HBM2e, 5 HBM2e stacks."

The NVIDIA GH100 GPU is composed of multiple GPU processing clusters (GPCs), texture processing clusters (TPCs), streaming multiprocessors (SMs), L2 cache, and HBM3 memory controllers.

The full implementation of the GH100 GPU includes the following units:

- 8 GPCs, 72 TPCs (9 TPCs/GPC), 2 SMs/TPC, 144 SMs per full GPU
- 128 FP32 CUDA Cores per SM, 18432 FP32 CUDA Cores per full GPU
- 4 fourth-generation Tensor Cores per SM, 576 per full GPU
- 6 HBM3 or HBM2e stacks, 12 512-bit memory controllers
- 60 MB L2 cache
- Fourth-generation NVLink and PCIe Gen 5

The NVIDIA H100 GPU with SXM5 board form-factor includes the following units:

- 8 GPCs, 66 TPCs, 2 SMs/TPC, 132 SMs per GPU
- 128 FP32 CUDA Cores per SM, 16896 FP32 CUDA Cores per GPU
- 4 fourth-generation Tensor Cores per SM, 528 per GPU
- 80 GB HBM3, 5 HBM3 stacks, 10 512-bit memory controllers
- 50 MB L2 cache
- Fourth-generation NVLink and PCIe Gen 5

The NVIDIA H100 GPU with a PCIe Gen 5 board form-factor includes the following units:

- 7 or 8 GPCs, 57 TPCs, 2 SMs/TPC, 114 SMs per GPU
- 128 FP32 CUDA Cores/SM, 14592 FP32 CUDA Cores per GPU
- 4 fourth-generation Tensor Cores per SM, 456 per GPU
- 80 GB HBM2e, 5 HBM2e stacks, 10 512-bit memory controllers
- 50 MB L2 cache
- Fourth-generation NVLink and PCIe Gen 5

(*See* https://developer.nvidia.com/blog/nvidia-hopper-architecture-in-depth/ (emphasis added).)

70. The Accused Products implement CUDA, Nvidia's parallel computing platform. CUDA enables NVIDIA GPUs to be used for general purpose computing tasks and includes specialized GPU-acceleration libraries such as cuDNN. Examples of parameters used in CUDA include pointers "dst" ("Destination memory address") and "src" ("Source memory address"). For instance, exemplary CUDA function "cudaMemcpy" copies "bytes [data] from the memory area pointed to by src [source memory address pointer] to the memory area pointed to by dst

[destination memory address pointer], where kind [type of transfer] specifies the direction of the copy." One of the destinations is "cudaMemcpyHostToDevice," or host (CPU) to device (GPU).

__host__ cudaError_t cudaMemcpy (void* dst , const void* src , size_t count , cudaMemcpyKind kind)

Copies data between host and device.

Parameters

`dst`
- Destination memory address

`src`
- Source memory address

`count`
- Size in bytes to copy

`kind`
- Type of transfer

Returns

cudaSuccess, cudaErrorInvalidValue, cudaErrorInvalidMemcpyDirection

Description

Copies `count` bytes from the memory area pointed to by `src` to the memory area pointed to by `dst`, where `kind` specifies the direction of the copy, and must be one of cudaMemcpyHostToHost, cudaMemcpyHostToDevice, cudaMemcpyDeviceToHost, cudaMemcpyDeviceToDevice, or cudaMemcpyDefault. Passing cudaMemcpyDefault is recommended, in which case the type of transfer is inferred from the pointer values. However, cudaMemcpyDefault is only allowed on systems that support unified virtual addressing. Calling cudaMemcpy() with dst and src pointers that do not match the direction of the copy results in an undefined behavior.

(*See* https://docs.nvidia.com/cuda/cuda-runtime-api/group__CUDART__MEMORY.html (emphasis added).)

71. The Accused Products practice a method that includes *performing, by at least one graphics processing unit during the first computational cycle, at least one calculation on the first portion of the input data as to generate first output data*. For instance, CUDA uses "streams" to execute a sequence of commands in order. As shown below, an exemplary CUDA function "cudaMemcpyAsync" is used to copy data between a host (CPU) and a device (GPU). "Each stream copies its portion of input array hostPtr [pointer for CPU] to array inputDevPtr in device [GPU] memory." The stream then "processes inputDevPtr on the device [GPU] by calling MyKernel(), and copies the result outputDevPtr back to the same portion of hostPtr."

3.2.8.5.1. Creation and Destruction of Streams

A stream is defined by creating a stream object and specifying it as the stream parameter to a sequence of kernel launches and host `<->` device memory copies. The following code sample creates two streams and allocates an array `hostPtr` of `float` in page-locked memory.

```
cudaStream_t stream[2];
for (int i = 0; i < 2; ++i)
    cudaStreamCreate(&stream[i]);
float* hostPtr;
cudaMallocHost(&hostPtr, 2 * size);
```

Each of these streams is defined by the following code sample as a sequence of one memory copy from host to device, one kernel launch, and one memory copy from device to host:

```
for (int i = 0; i < 2; ++i) {
    cudaMemcpyAsync(inputDevPtr + i * size, hostPtr + i * size,
                    size, cudaMemcpyHostToDevice, stream[i]);
    MyKernel <<<100, 512, 0, stream[i]>>>
          (outputDevPtr + i * size, inputDevPtr + i * size, size);
    cudaMemcpyAsync(hostPtr + i * size, outputDevPtr + i * size,
                    size, cudaMemcpyDeviceToHost, stream[i]);
}
```

Each stream copies its portion of input array `hostPtr` to array `inputDevPtr` in device memory, processes `inputDevPtr` on the device by calling `MyKernel()`, and copies the result `outputDevPtr` back to the same portion of `hostPtr`. Overlapping Behavior describes how the streams overlap in this example depending on the capability of the device. Note that `hostPtr` must point to page-locked host memory for any overlap to occur.

(*See* https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#creation-and-destruction-of-streams (emphasis added).)

72. The Accused Products practice a method that includes *storing, by the accelerator controller, the first output data into a second partition, referenced by a second pointer, of the accelerator memory*. For instance, exemplary excerpts of CUDA code shown below demonstrate CUDA being used to calculate a square sub-matrix Csub of matrix C using the function MatMul. An exemplary CUDA stream "allocate[s] [matrix] C in device memory." After Matrices A and B are synchronized and multiplied, the exemplary CUDA stream "[w]rite[s] Csub to device [GPU] memory."

```
    // Load A and B to device memory
    Matrix d_A;
    d_A.width = d_A.stride = A.width; d_A.height = A.height;
    size_t size = A.width * A.height * sizeof(float);
    cudaMalloc(&d_A.elements, size);
    cudaMemcpy(d_A.elements, A.elements, size,
               cudaMemcpyHostToDevice);
    Matrix d_B;
    d_B.width = d_B.stride = B.width; d_B.height = B.height;
    size = B.width * B.height * sizeof(float);
    cudaMalloc(&d_B.elements, size);
    cudaMemcpy(d_B.elements, B.elements, size,
    cudaMemcpyHostToDevice);
    // Allocate C in device memory
    Matrix d_C;
    d_C.width = d_C.stride = C.width; d_C.height = C.height;
    size = C.width * C.height * sizeof(float);
    cudaMalloc(&d_C.elements, size);
```

* * * * *

```
        // Synchronize to make sure the sub-matrices are loaded
        // before starting the computation
        __syncthreads();
        // Multiply Asub and Bsub together
        for (int e = 0; e < BLOCK_SIZE; ++e)
            Cvalue += As[row][e] * Bs[e][col];
        // Synchronize to make sure that the preceding
        // computation is done before loading two new
        // sub-matrices of A and B in the next iteration
        __syncthreads();
    }
    // Write Csub to device memory
    // Each thread writes one element
    SetElement(Csub, row, col, Cvalue);
}
```

(*See* https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#shared-memory (emphasis added).)

73. In addition, exemplary CUDA function "cudaMalloc" is used to "allocate weight, work, and reserve space buffer sizes in the GPU memory." "The work-space buffer is used for temporary storage" and the " content can be discarded or modified after all GPU kernels launched by the corresponding API complete." The "reserve-space buffer" used to transfer intermediate results is used for transferring "intermediate results" as used in the cuDNN GPU-acceleration library for CUDA.

7.2.15. cudnnGetMultiHeadAttnBuffers()

This function computes weight, work, and reserve space buffer sizes used by the following functions:

- cudnnMultiHeadAttnForward()
- cudnnMultiHeadAttnBackwardData()
- cudnnMultiHeadAttnBackwardWeights()

```
cudnnStatus_t cudnnGetMultiHeadAttnBuffers(
        cudnnHandle_t handle,
        const cudnnAttnDescriptor_t attnDesc,
        size_t *weightSizeInBytes,
        size_t *workSpaceSizeInBytes,
        size_t *reserveSpaceSizeInBytes);
```

Assigning NULL to the reserveSpaceSizeInBytes argument indicates that the user does not plan to invoke multi-head attention gradient functions: cudnnMultiHeadAttnBackwardData() and cudnnMultiHeadAttnBackwardWeights(). This situation occurs in the inference mode.

Note: NULL cannot be assigned to weightSizeInBytes and workSpaceSizeInBytes pointers.

The user must allocate weight, work, and reserve space buffer sizes in the GPU memory using cudaMalloc() with the reported buffer sizes. The buffers can be also carved out from a larger chunk of allocated memory but the buffer addresses must be at least 16B aligned.

The work-space buffer is used for temporary storage. Its content can be discarded or modified after all GPU kernels launched by the corresponding API complete. The reserve-space buffer is used to transfer intermediate results from cudnnMultiHeadAttnForward() to cudnnMultiHeadAttnBackwardData(), and from cudnnMultiHeadAttnBackwardData() to cudnnMultiHeadAttnBackwardWeights(). The content of the reserve-space buffer cannot be modified until all GPU kernels launched by the above three multi-head attention API functions finish.

All multi-head attention weight and bias tensors are stored in a single weight buffer. For speed optimizations, the cuDNN API may change tensor layouts and their relative locations in the weight buffer based on the provided attention parameters. Use the cudnnGetMultiHeadAttnWeights() function to obtain the start address and the shape of each weight or bias tensor.

(*See* https://docs.nvidia.com/deeplearning/cudnn/archives/cudnn-893/api/index.html#cudnnGetMultiHeadAttnBuffers (emphasis added).)

74. The Accused Products practice a method that includes *swapping the first pointer with the second pointer at the end of the first computational cycle, such that the first output data becomes an input for a second computational cycle of the numerical simulation*. For example, the cuDNN GPU-acceleration library of the Accused Products implement operations that "take tensors as input and produce tensors as output."

2.2. Tensors and Layouts

Whether using the graph API or the legacy API, cuDNN operations take tensors as input and produce tensors as output.

(*See* https://docs.nvidia.com/deeplearning/cudnn/archives/cudnn-881/developer-guide/index.html#tensors-layouts (emphasis added).)

75. Nvidia confirms that CUDA implements pointer swapping for device (GPU) pointers.

Swap device Pointers

Home > Accelerated Computing ■ CUDA ■ CUDA Programming and Performance

SamWhite Aug 01 '15

Hi guys :)

I'm really sorry for the very basic question but I cannot find a straight answer on any of the online resources I have checked.

I am writing some code in cuda that 'ping pongs' on two sets of data, reading from one and writing to the other.

What I need to know is does the following piece of code actually work on **DEVICE** pointers in cuda. This is memory that has already been allocated on the device. I call the below in a method at the start of each frame.

```
cData* temp = d_data;
d_data= d_dataOld;
d_dataOld= temp;
```

If not can you suggest alternatives to simply changing which pointers I send to the kernels - I prefer to avoid that method in order to keep the code easy to read!

Thanks :)

* * * * *

3.7k views

Robert_Crovella Moderator Aug 01 '15

It works. A pointer in C is just a number.

A DEVICE pointer in cuda is just a C pointer, after all. There's no magic here.

(*See* https://forums.developer.nvidia.com/t/swap-device-pointers/38964 (emphasis added).)

3.2.8.5. Streams

Applications manage the concurrent operations described above through *streams*. A stream is a sequence of commands (possibly issued by different host threads) that execute in order. Different streams, on the other hand, may execute their commands out of order with respect to one another or concurrently; this behavior is not guaranteed and should therefore not be relied upon for correctness (for example, inter-kernel communication is undefined). The commands issued on a stream may execute when all the dependencies of the command are met. The dependencies could be previously launched commands on same stream or dependencies from other streams. The successful completion of synchronize call guarantees that all the commands launched are completed.

3.2.8.5.1. Creation and Destruction of Streams

A stream is defined by creating a stream object and specifying it as the stream parameter to a sequence of kernel launches and host `<->` device memory copies. The following code sample creates two streams and allocates an array `hostPtr` of `float` in page-locked memory.

```
cudaStream_t stream[2];
for (int i = 0; i < 2; ++i)
    cudaStreamCreate(&stream[i]);
float* hostPtr;
cudaMallocHost(&hostPtr, 2 * size);
```

Each of these streams is defined by the following code sample as a sequence of one memory copy from host to device, one kernel launch, and one memory copy from device to host:

```
for (int i = 0; i < 2; ++i) {
    cudaMemcpyAsync(inputDevPtr + i * size, hostPtr + i * size,
                    size, cudaMemcpyHostToDevice, stream[i]);
    MyKernel <<<100, 512, 0, stream[i]>>>
          (outputDevPtr + i * size, inputDevPtr + i * size, size);
    cudaMemcpyAsync(hostPtr + i * size, outputDevPtr + i * size,
                    size, cudaMemcpyDeviceToHost, stream[i]);
}
```

Each stream copies its portion of input array `hostPtr` to array `inputDevPtr` in device memory, processes `inputDevPtr` on the device by calling `MyKernel()`, and copies the result `outputDevPtr` back to the same portion of `hostPtr`. Overlapping Behavior describes how the streams overlap in this example depending on the capability of the device. Note that `hostPtr` must point to page-locked host memory for any overlap to occur.

Streams are released by calling `cudaStreamDestroy()`.

(*See* https://docs.nvidia.com/cuda/cuda-c-programming-guide/#creation-and-destruction-of-streams (emphasis added).)

76. Each claim in the ’867 Patent recites an independent invention. Neither claim 16, described above, nor any other individual claim is representative of all claims in the ’867 Patent.

77. Defendant has been aware of the technology patented by the ’867 Patent since at least 2007, when the inventors of the Asserted Patents first discussed their patented technologies

with Mr. Sanford Russell, then the CTO of Nvidia. At the time, the inventors asked Defendant to collaborate with them on training neural networks using Nvidia's GPUs. Defendant informed the inventors, through Mr. Russell, that it was not interested in the collaboration. Defendant has also cited the application for the '867 Patent in its own patent portfolio since at least June 28, 2010.

Cited By (42)

Publication number	Priority date	Publication date	Assignee	Title
US20130163195A1 *	2011-12-22	2013-06-27	Nvidia Corporation	System, method, and computer program product for performing operations on data utilizing a computation module

* * * * *

US8922566B2 *	2010-06-28	2014-12-30	Nvidia Corporation	Rechargeable universal serial bus external graphics device and method

(*See* https://patents.google.com/patent/US8648867B2/en?oq=8648867#citedBy (emphasis added).)

(12) **United States Patent**
Soma

(10) **Patent No.: US 8,922,566 B2**
(45) **Date of Patent: Dec. 30, 2014**

(54) **RECHARGEABLE UNIVERSAL SERIAL BUS EXTERNAL GRAPHICS DEVICE AND METHOD**

(75) Inventor: **Srinivas Soma**, Hyderabad (IN)

(73) Assignee: **NVIDIA Corporation**, Santa Clara, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 548 days.

(21) Appl. No.: **12/824,423**

(22) Filed: **Jun. 28, 2010**

2009/0141894 A1* 6/2009 Sahdra et al. 380/239
2009/0144456 A1* 6/2009 Gelf et al. 710/8
2010/0091025 A1* 4/2010 Nugent et al. 345/502

OTHER PUBLICATIONS

Reimer, Coming Soon: An External Video Card Near You?, ARS Technica (http://arstechnica.com/uncategorized/2006/08/7409/) (Aug. 2, 2006).*
White, NVIDIA Announces Quadro Plex, Monster Graphics for Pros, http://gizmodo.com/191187/nvidia-announces-quadro-plex-monster-graphics-for-pros, Gawker Media (Aug. 1, 2006).*
NVIDIA Quadro Plex 1000 Installation Guide, DI-02500-001_V03, NVIDIA Corporation (2007).*

* cited by examiner

Primary Examiner — James A Thompson
(74) *Attorney, Agent, or Firm* — Zilka-Kotab, PC

* * * * *

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,949,438	A *	9/1999	Cyman et al.	.....	345/502
6,507,172	B2 *	1/2003	Sherman	.....	320/134
2006/0171689	A1 *	8/2006	Smith	.....	386/125
2008/0117220	A1 *	5/2008	Gorchetchnikov et al.	...	345/503

(*See* https://patentimages.storage.googleapis.com/ee/13/e9/61df149c3fddc7/US8922566.pdf (Nvidia U.S. Patent No. 8,922,566) (emphasis added).)

Notice of References Cited	Application/Control No. 13/335,850	Applicant(s)/Patent Under Reexamination PELLETIER, SEAN MICHAEL	
	Examiner PAUL R. MYERS	Art Unit 2111	Page 2 of 2

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
*	A	US-8,612,652 B2	12-2013	Tsuei et al.	710/62
*	B	US-8,648,867 B2	02-2014	Gorchetchnikov et al.	345/501
*	C	US-8,687,007 B2	04-2014	Nugent et al.	345/502
*	D	US-8,730,247 B2	05-2014	Hiroi et al.	345/503

(*See* https://patentcenter.uspto.gov/applications/13335850/displayReferences/referenceForms?application= (Nvidia U.S. Appl. No. 13/335,850 August 12, 2014, List of References Cited by Examiner) (emphasis added).)

78. Starting in or around 2016, the inventors of the Asserted Patents held multiple discussions with Nvidia to invest in or purchase their AI company, Neurala, Inc., and all its assets, including the ’867 Patent and its related patents and applications. These discussions included at least Mr. Alvin Lin, an Nvidia Senior Director of Business Development, and Mr. Jeff Herbst, then an Nvidia Vice President of Business Development and head of Nvidia’s Inception GPU Ventures, in or around September 6, 2016. In or around October 2016, Nvidia, through its representatives,

initiated discussions with the inventors to invest in Neurala, Inc. for approximately $10 million.

79. The inventors also discussed their patented technology, in addition to the ’867 Patent and its family, with Defendant’s representatives at Nvidia’s artificial intelligence conference in or around June 2017. On or about June 26, 2017, Defendant received materials from the inventors, in lieu of a meeting on or about June 29, that identified the ’867 Patent and its family and described the technology in detail. Defendant had previously stated it was interested in the inventors’ solutions. Defendant also featured the inventors on its website as members of Defendant’s start-up incubator on or about September 25, 2019.

Computer Vision / Video Analytics

Inception Spotlight: AI Startup Neurala Sees 7X Speedup with NGC

Sep 25, 2019 0 Like Discuss (0)

By Nefi Alarcon

* * * * *

To help businesses develop custom computer vision solutions quickly, Neurala, a member of NVIDIA’s start-up incubator Inception, has developed Brain Builder, a cloud platform that provides data scientists and developers that are new to deep learning with the ability to quickly and easily train neural networks.

(*See* https://developer.nvidia.com/blog/inception-spotlight-ai-startup-neurala-sees-7x-speedup-with-ngc/ (September 25, 2019); *see also* https://www.youtube.com/watch?v=-WBtxGLoQNs (“Neurala Accelerating AI Video Annotation with NGC Containers” posted by Defendant’s YouTube account).)

80. Neural AI and/or its predecessors-in-interest have satisfied all statutory obligations required to collect pre-filing damages for the full period allowed by law for infringement of the

’867 Patent.

81. Defendant directly infringes at least claim 16 of the ’867 Patent, either literally or under the doctrine of equivalents, by performing the steps described above. For example, Defendant performs the claimed method in an infringing manner as described above by implementing the Accused Products as part of its accelerated computing operations and running corresponding software that implements the infringing performance. Defendant also performs the claimed method in an infringing manner when testing the operation of the Accused Products and corresponding systems. As another example, Defendant performs the claimed method when providing or administering services to third parties, customers, and partners using the Accused Products.

82. Defendant’s partners, customers, and users of its Accused Products and corresponding systems and services directly infringe at least claim 16 of the ’867 Patent, literally or under the doctrine of equivalents, at least by using the Accused Products and corresponding systems and services, as described above.

83. Defendant has actively induced and is actively inducing infringement of at least claim 16 of the ’867 Patent with specific intent to induce infringement, and/or willful blindness to the possibility that its acts induce infringement, in violation of 35 U.S.C. § 271(b). For example, Defendant encourages and induces customers to use Nvidia’s CUDA platform in a manner that infringes claim 16 of the ’867 Patent at least by offering and providing software that performs a method that infringes claim 16 when installed and operated by the customer using the Accused Products, and by engaging in activities relating to selling, marketing, advertising, promotion, installation, support, and distribution of the Accused Products.

84. Defendant encourages, instructs, directs, and/or requires third parties—including

its certified partners and/or customers—to perform the claimed method using the software, platform, services, and systems in infringing ways, as described above.

85. Defendant further encourages and induces its customers to infringe claim 16 of the ’867 Patent: 1) by making its accelerated computing and data center services available on its website, providing applications that allow users to access those services, widely advertising those services, and providing technical support and instructions to users (*see* https://www.nvidia.com/en-us/data-center/data-center-gpus/gpu-test-drive/); and 2) through activities relating to marketing, advertising, promotion, installation, support, and distribution of the Accused Products, including its CUDA platform, and services in the United States. (*See* https://www.nvidia.com/en-us/; *see* https://www.nvidia.com/en-us/about-nvidia/partners/; https://www.nvidia.com/en-us/data-center/where-to-buy/; https://www.nvidia.com/en-us/data-center/where-to-buy-tesla/.)

86. For example, Defendant shares instructions, guides, and manuals, which advertise and instruct third parties on how to use its hardware and platform as described above, including at least customers and partners. (*See* https://docs.nvidia.com/cuda/cuda-c-programming-guide/.) Defendant also provides customer service and technical support to purchasers of the Accused Products and corresponding systems and services, which directs and encourages customers to perform certain actions that use the Accused Products in an infringing manner. (*See* https://www.nvidia.com/en-us/support/; https://www.nvidia.com/en-us/support/enterprise/services/.)

87. Defendant and/or Defendant’s partners recommend and sell the Accused Products and provide technical support for the installation, implementation, integration, and ongoing operation of the Accused Products for each individual customer. On information and belief, each

customer enters into a contractual relationship with Defendant and/or one of Defendant's partners, which obligates each customer to perform certain actions in order to use the Accused Products. (*See* https://www.nvidia.com/en-us/agreements/; https://www.nvidia.com/en-us/agreements/cloud-services/nvidia-cloud-agreement/; https://www.nvidia.com/en-us/agreements/cloud-services/service-specific-terms-for-nvidia-dgx-cloud/.) Further, in order to receive the benefit of Defendant's and/or its partner's continued technical support and their specialized knowledge and guidance of the operability of the Accused Products, each customer must continue to use the Accused Products in a way that infringes the '867 Patent. (*See* https://www.nvidia.com/en-us/support/.)

88. Further, as the entity that provides installation, implementation, and integration of the Accused Products in addition to ensuring the Accused Product remains operational for each customer through ongoing technical support, on information and belief, Defendant and/or Defendant's partners affirmatively aid and abet each customer's use of the Accused Products in a manner that performs the claimed method of, and infringes, the '867 Patent.

89. Defendant also contributes to the infringement of its partners, customers, and users of the Accused Products by providing within the United States or importing into the United States the Accused Products, which are for use in practicing, and under normal operation practice, the methods, systems, and devices claimed in the Asserted Patents, constituting a material part of the inventions claimed, and not a staple article or commodity of commerce suitable for substantial non-infringing uses. Indeed, as shown above, the Accused Products and the example functionality have no substantial non-infringing uses but are specifically designed to practice the '867 Patent.

90. On information and belief, the infringing actions of each partner, customer, and/or user of the Accused Products are attributable to Defendant. For example, on information and belief,

Defendant directs and controls the activities or actions of its partners or others in connection with the Accused Products by contractual agreement or otherwise requiring partners or others to provide information and instructions to customers who acquire the Accused Products which, when followed, results in infringement. Defendant further directs and controls the operation of devices executing the Accused Products by programming the software which, when executed by a customer or user, performs the claimed method of at least claim 16 of the ’867 Patent.

91. Plaintiff has suffered and continue to suffer damages as a result of Defendant’s infringement of the ’867 Patent. Defendant is therefore liable to Plaintiff under 35 U.S.C. § 284 for damages in an amount that adequately compensates Plaintiff for Defendant’s infringement, but no less than a reasonable royalty.

92. Defendant’s infringement of the ’867 Patent is knowing and willful. Defendant had actual knowledge of the ’867 Patent application since at least 2010 and actual knowledge of the ’867 Patent, and its family, since at least 2017.

93. On information and belief, despite Defendant’s knowledge of the Asserted Patents and Plaintiff’s patented technology, Defendant made the deliberate decision to sell products and services that it knew infringe these patents. Defendant’s continued infringement of the ’867 Patent with knowledge of the ’867 Patent constitutes willful infringement.

SECOND CAUSE OF ACTION
(INFRINGEMENT OF THE ’438 PATENT)

94. Plaintiff realleges and incorporates by reference the allegations of the preceding paragraphs of this Complaint.

95. Defendant has infringed and continues to infringe one or more claims of the ’438 Patent in violation of 35 U.S.C. § 271 in this District and elsewhere in the United States and will continue to do so. The Accused Products, including features of, *e.g.*, the Grace Hopper Superchip (GH200), at least

when used for their ordinary and customary purposes, practice each element of at least claim 21 of the ’438 Patent as demonstrated below.

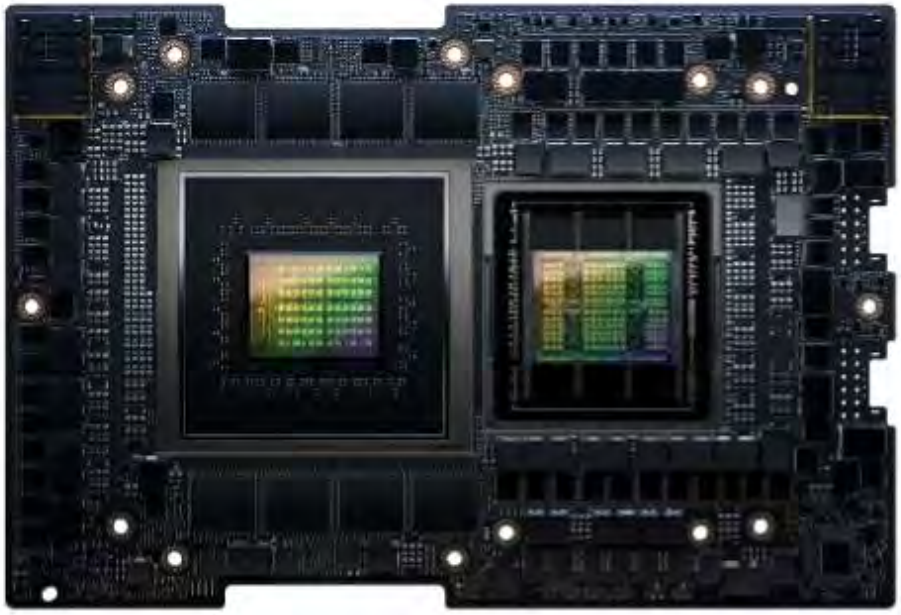
96. For example, claim 21 of the ’438 Patent recites:

> 21. A method of performing a sequence of computations representing an artificial neural network, the method comprising:
>
> receiving, at a central processing unit (CPU), first input data acquired from an external system in real time;
>
> initializing, by a controller operably coupled to a graphics processing unit (GPU), textures and shaders in a memory operably coupled to the GPU;
>
> transferring the first input data received by the CPU to the memory operably coupled to the GPU;
>
> performing, by the graphics processing unit (GPU), a first computation in the sequence of computations on the first input data based on the textures and shaders to generate first output data, computations in the sequence of computations representing respective layers of neurons in the artificial neural network, an output of the first computation in the sequence of computations representing an output of a first neuron in a first layer in the artificial neural network;
>
> storing, in the memory operably coupled to the GPU, the first input data and the first output data; and
>
> transferring second input data acquired from the external system in real time into the memory operably coupled to the GPU after the GPU starts the first computation and before the GPU starts a second computation of the sequence of computations, an output of the second computation in the sequence of computations representing an output of a second neuron in a second layer in the artificial neural network.

97. The Accused Products perform each step of the method of claim 21 of the ’438 Patent. To the extent the preamble is construed to be limiting, the Accused Products perform *a method of performing a sequence of computations representing an artificial neural network*, as further explained below. For instance, the Grace Hopper Superchip (GH200) “brings together the

groundbreaking performance of the NVIDIA Hopper GPU with the versatility of the NVIDIA Grace™ CPU . . . in a single Superchip." It includes the cuDNN (CUDA Deep Neural Network) library for "[d]eep ***neural networks***."

Inside NVIDIA's First GPU-CPU Superchip



The NVIDIA® GH200 Grace Hopper architecture brings together the groundbreaking performance of the NVIDIA Hopper GPU with the versatility of the NVIDIA Grace™ CPU, connected with a high bandwidth and memory coherent NVIDIA NVLink Chip-2-Chip (C2C)® interconnect in a single Superchip, and support for the new NVIDIA NVLink Switch System.

* * * * *

An extensive suite of domain-specific libraries and frameworks further accelerates main algorithms in a wide range of application domains, for example:

- Deep neural networks (cuDNN)
- Linear solvers for simulations and implicit unstructured methods (AmgX)
- Quantum computing (cuQuantum)
- Data science
- Machine learning (RAPIDS)
- Data loading and preprocessing for machine learning (DALI)
- Real-time 3D simulation and design collaboration (Omniverse)

(*See* https://resources.nvidia.com/en-us-grace-cpu/nvidia-grace-hopper (emphasis added).)

98. The "Grace Hopper Superchip is the first true heterogeneous accelerated platform for high-performance computing (HPC) and AI workloads. It accelerates applications with the strengths of both GPUs and CPUs while providing the simplest and most productive heterogeneous programming model to date."

The NVIDIA GH200 Grace Hopper Superchip is the first true heterogeneous accelerated platform for high-performance computing (HPC) and AI workloads. It accelerates applications with the strengths of both GPUs and CPUs while providing the simplest and most productive heterogeneous programming model to date, enabling scientists and engineers to focus on solving the world's most important problems. Together with NVIDIA networking technologies, NVIDIA GH200 provides the recipe for the next generation of HPC supercomputers and AI factories, enabling customers to take on larger datasets, more complex models, and new workloads, solving them more quickly than before.

(*See* https://resources.nvidia.com/en-us-grace-cpu/nvidia-grace-hopper (emphasis added).)

99. In addition, the Accused Products, including the Grace Hopper Superchip, implement CUDA, Nvidia's proprietary "parallel computing platform and programming model." CUDA further includes the CUDA Toolkit, which "includes GPU-accelerated libraries, a compiler, development tools and the CUDA runtime." As an example, the "CUDA® Deep Neural Network library (cuDNN) is a GPU-acceleration library of primitives for ***deep neural networks***."

It "provides highly tuned implementations for standard routines" for GPU-based acceleration.

CUDA Zone

CUDA® is a parallel computing platform and programming model developed by NVIDIA for general computing on graphical processing units (GPUs). With CUDA, developers are able to dramatically speed up computing applications by harnessing the power of GPUs.

In GPU-accelerated applications, the sequential part of the workload runs on the CPU – which is optimized for single-threaded performance – while the compute intensive portion of the application runs on thousands of GPU cores in parallel. When using CUDA, developers program in popular languages such as C, C++, Fortran, Python and MATLAB and express parallelism through extensions in the form of a few basic keywords.

The CUDA Toolkit from NVIDIA provides everything you need to develop GPU-accelerated applications. The CUDA Toolkit includes GPU-accelerated libraries, a compiler, development tools and the CUDA runtime.

(*See* https://developer.nvidia.com/cuda-zone (emphasis added).)

NVIDIA cuDNN

The NVIDIA CUDA® Deep Neural Network library (cuDNN) is a GPU-accelerated library of primitives for deep neural networks. cuDNN provides highly tuned implementations for standard routines such as forward and backward convolution, attention, matmul, pooling, and normalization.

(*See* https://developer.nvidia.com/cudnn (emphasis added).)

100. Nvidia GPU architectures that implement CUDA and cuDNN include the Hopper (*e.g.*, Grace Hopper Superchip (GH200), H100), Ada Lovelace, Ampere, Turing, Volta, Pascal, and Maxwell GPU architectures of the Accused Products.

1. GPU, CUDA Toolkit, and CUDA Driver Requirements

The following sections highlight the compatibility of NVIDIA® cuDNN versions with the various supported NVIDIA CUDA® Toolkit, CUDA driver, and NVIDIA hardware versions.

Table 1. GPU, CUDA Toolkit, and CUDA Driver Requirements

cuDNN Package[1]	CUDA Toolkit Version	Supports static linking?[2]	NVIDIA Driver Version		CUDA Compute Capability	Supported NVIDIA Hardware
			Linux	Windows		
cuDNN 8.9.6 for CUDA 12.x	12.2	Yes	>=525.60.13	>=527.41	9.0[3]	
	12.1	No			8.9[4]	NVIDIA Hopper™[5]
	12.0				8.6	NVIDIA Ada Lovelace architecture[6]
cuDNN 8.9.6 for CUDA 11.x	11.8	Yes	>= 450.80.02	>=452.39	8.0	NVIDIA Ampere architecture
	11.7	No			7.5	NVIDIA Turing™
	11.6				7.0	NVIDIA Volta™
	11.5				6.1	NVIDIA Pascal™
	11.4				6.0	NVIDIA Maxwell[6]
	11.3				5.0	
	11.2[7]					
	11.1[8]					
	11.0[9]					

(*See* https://docs.nvidia.com/deeplearning/cudnn/archives/cudnn-896/support-matrix/index.html (emphasis added).)

101. The Accused Products perform a method that includes *receiving, at a central processing unit (CPU), first input data acquired from an external system in real time*. For instance, as illustrated below, a diagram describing the architecture of the Grace Hopper Superchip depicts a CPU ("Grace CPU") with "[u]p to 72 cores" that receives and sends input and output data via "High-Speed IO" ("PCIe-5"). The Grace CPU is further depicted as being coupled to memory "CPU LPDDR5X" up to 480GB via a link up to "500 GB/s."

Figure 1 shows the logical overview of the NVIDIA GH200 Grace Hopper Superchip and Table 1 lists its key features.

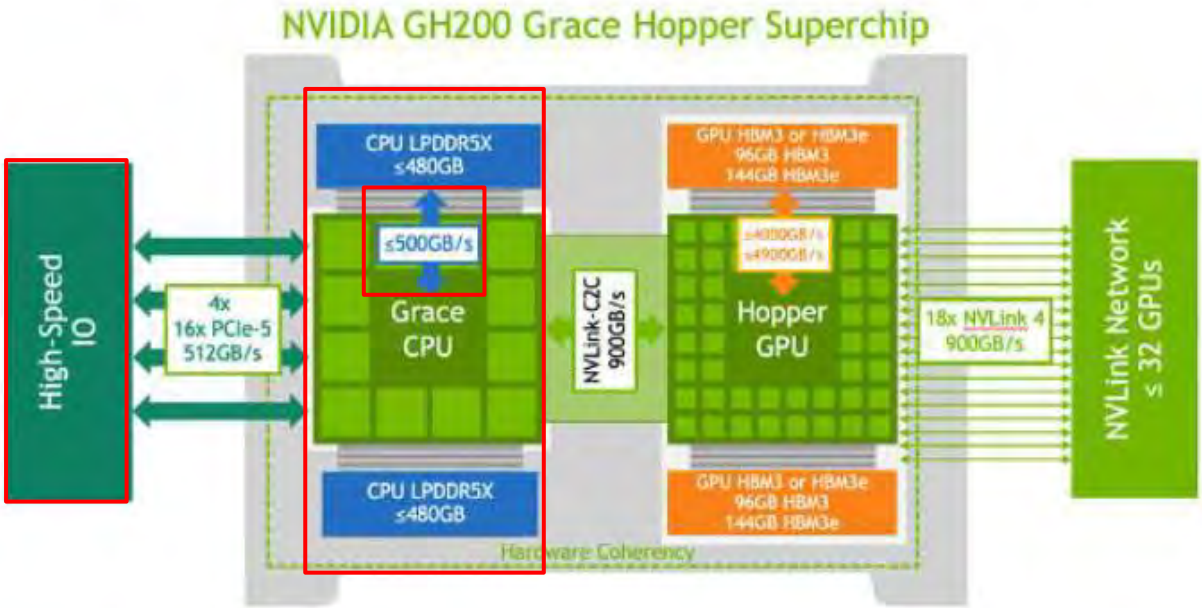


Figure 1. NVIDIA GH200 Grace Hopper Superchip Logical Overview

Table 1. NVIDIA GH200 Grace Hopper Superchip Key Features

Feature	Description
Grace CPU cores (number)	Up to 72 cores
CPU LPDDR5X bandwidth (GB/s)	Up to 500GB/s
GPU HBM bandwidth (GB/s)	4TB/s HBM3 4.9TB/s HBM3e
NVLink-C2C bandwidth (GB/s)	900GB/s total, 450GB/s per direction
CPU LPDDR5X capacity (GB)	Up to 480GB
GPU HBM capacity (GB)	96GB HBM3 144GB HBM3e
PCIe Gen 5 Lanes	64x

* * * * *

NVLink-C2C enables applications to oversubscribe the GPU's memory and directly utilize NVIDIA Grace CPU's memory at high bandwidth. With up to 480GB of LPDDR5X CPU memory per Grace Hopper Superchip, the GPU has direct high-bandwidth access to an additional 480GB of memory. Combined with NVIDIA NVLink Switch System, all GPU threads running on up to 32 NVLink-connected GPUs on NVIDIA GH200 NVL32 can now access up to 19.5TB of memory at high bandwidth. Fourth generation NVLink allows

(*See* https://resources.nvidia.com/en-us-grace-cpu/nvidia-grace-hopper (emphasis added).)

102. In a related diagram example, "CPU PHYSICAL MEMORY" (LPDDR5X) is illustrated as being accessed by a CPU ("CPU-resident access").

In NVIDIA Grace Hopper Superchip-based systems, Address Translation Service (ATS) enables the CPU and GPU to share a single per-process page table, enabling all CPU and GPU threads to access all system-allocated memory (Figure 8), which can reside on physical CPU or GPU memory. The CPU heap, CPU thread stack, global variables,

memory-mapped files, and inter-process memory are accessible to all CPU and GPU threads.

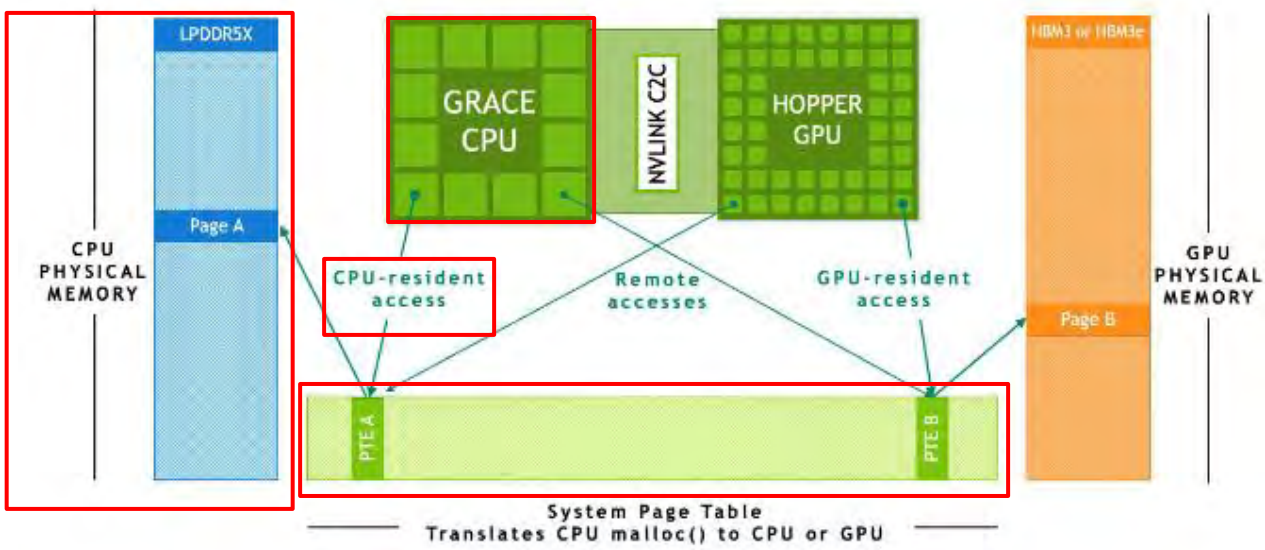


Figure 8. ATS in an NVIDIA Grace Hopper Superchip System

(*See* https://resources.nvidia.com/en-us-grace-cpu/nvidia-grace-hopper (emphasis added).)

103. As previously stated, the Accused Products implement CUDA and specialized GPU-acceleration libraries such as cuDNN. "CUDA® is a parallel computing platform and

programming model developed by NVIDIA for general computing on graphical processing units (GPUs)" including "GPU-accelerated applications." In GPU-accelerated applications, "***the sequential part of the workload runs on the CPU*** – which is optimized for single-threaded performance – while the compute intensive portion of the application runs on thousands of GPU cores in parallel."

CUDA Zone

CUDA® is a parallel computing platform and programming model developed by NVIDIA for general computing on graphical processing units (GPUs). With CUDA, developers are able to dramatically speed up computing applications by harnessing the power of GPUs.

In GPU-accelerated applications, the sequential part of the workload runs on the CPU – which is optimized for single-threaded performance – while the compute intensive portion of the application runs on thousands of GPU cores in parallel. When using CUDA, developers program in popular languages such as C, C++, Fortran, Python and MATLAB and express parallelism through extensions in the form of a few basic keywords.

The CUDA Toolkit from NVIDIA provides everything you need to develop GPU-accelerated applications. The CUDA Toolkit includes GPU-accelerated libraries, a compiler, development tools and the CUDA runtime.

(*See* https://developer.nvidia. com/cuda-zone (emphasis added).)

104. The "CUDA programming model" implements programming functions and instructions for CPUs and GPUs. "The host is the CPU available in the system" and "system memory associated with the CPU is called host memory." As an example, the first main CUDA program execution step is "[c]opy[ing] the ***input data from host [CPU] memory***."

Let me introduce two keywords widely used in CUDA programming model: *host* and *device*.

The host is the CPU available in the system. The system memory associated with the CPU is called host memory. The GPU is called a device and GPU memory likewise called device memory.

To execute any CUDA program, there are three main steps:

- Copy the input data from host memory to device memory, also known as host-to-device transfer.
- Load the GPU program and execute, caching data on-chip for performance.
- Copy the results from device memory to host memory, also called device-to-host transfer.

(*See* https://developer.nvidia.com/blog/cuda-refresher-cuda-programming-model/ (emphasis added).)

105. The Accused Products perform a method that includes *initializing, by a controller operably coupled to a graphics processing unit (GPU), textures and shaders in a memory operably coupled to the GPU*. For instance, as illustrated below, a GPU ("Hopper GPU") for the Grace Hopper Superchip is depicted as accessing both CPU and GPU memory using "NVLINK" for both accessing and storing data. Indeed, "NVIDIA GH200 is designed to accelerate applications with exceptionally large memory footprints." As illustrated, the Hopper GPUs are illustrated coupled to "GPU HBM3" high bandwidth memory or "GPU HBM3e" high bandwidth memory.

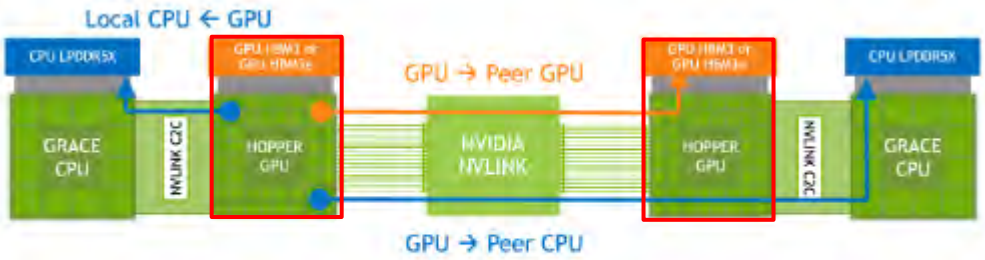


Figure 5. Memory Accesses across NVLink-connected Grace Hopper Superchips

(*See* https://resources.nvidia.com/en-us-grace-cpu/nvidia-grace-hopper (emphasis added).)

106. The GPU architecture of the Accused Products implements a ***controller***. As an example, the Hopper-GPU architecture includes "GPU processing clusters" and "***texture*** processing clusters" and implements "HBM3 memory ***controllers***" including "12 512-bit memory ***controllers***" coupled GPU memory including "6 HBM3 or HBM2e stacks," "80 GB HBM3, 5 HBM3 stacks," and "80 GB HBM2e, 5 HBM2e stacks."

The NVIDIA GH100 GPU is composed of multiple GPU processing clusters (GPCs), texture processing clusters (TPCs), streaming multiprocessors (SMs), L2 cache, and HBM3 memory controllers.

The full implementation of the GH100 GPU includes the following units:

- 8 GPCs, 72 TPCs (9 TPCs/GPC), 2 SMs/TPC, 144 SMs per full GPU
- 128 FP32 CUDA Cores per SM, 18432 FP32 CUDA Cores per full GPU
- 4 fourth-generation Tensor Cores per SM, 576 per full GPU
- 6 HBM3 or HBM2e stacks, 12 512-bit memory controllers
- 60 MB L2 cache
- Fourth-generation NVLink and PCIe Gen 5

The NVIDIA H100 GPU with SXM5 board form-factor includes the following units:

- 8 GPCs, 66 TPCs, 2 SMs/TPC, 132 SMs per GPU
- 128 FP32 CUDA Cores per SM, 16896 FP32 CUDA Cores per GPU
- 4 fourth-generation Tensor Cores per SM, 528 per GPU
- 80 GB HBM3, 5 HBM3 stacks, 10 512-bit memory controllers
- 50 MB L2 cache
- Fourth-generation NVLink and PCIe Gen 5

The NVIDIA H100 GPU with a PCIe Gen 5 board form-factor includes the following units:

- 7 or 8 GPCs, 57 TPCs, 2 SMs/TPC, 114 SMs per GPU
- 128 FP32 CUDA Cores/SM, 14592 FP32 CUDA Cores per GPU
- 4 fourth-generation Tensor Cores per SM, 456 per GPU
- 80 GB HBM2e, 5 HBM2e stacks, 10 512-bit memory controllers
- 50 MB L2 cache
- Fourth-generation NVLink and PCIe Gen 5

(*See* https://developer.nvidia.com/blog/nvidia-hopper-architecture-in-depth/ (emphasis added).)

107. In addition, CUDA includes the exemplary NPP (Nvidia Performance Primitives) library "for performing CUDA accelerated processing" and "performing CUDA accelerated processing for 2D image and signal processing."

CUDA Toolkit Documentation 12.6

Develop, Optimize and Deploy GPU-Accelerated Apps

The NVIDIA® CUDA® Toolkit provides a development environment for creating high performance GPU-accelerated applications. With the CUDA Toolkit, you can develop, optimize, and deploy your applications on GPU-accelerated embedded systems, desktop workstations, enterprise data centers, cloud-based platforms and HPC supercomputers. The toolkit includes GPU-accelerated libraries, debugging and optimization tools, a C/C++ compiler, and a runtime library to deploy your application.

* * * * *

NPP

NVIDIA NPP is a library of functions for performing CUDA accelerated processing. The initial set of functionality in the library focuses on imaging and video processing and is widely applicable for developers in these areas. NPP will evolve over time to encompass more of the compute heavy tasks in a variety of problem domains. The NPP library is written to maximize flexibility, while maintaining high performance.

(*See* https://docs.nvidia.com/cuda/index.html (emphasis added).)

What is NPP ?

NVIDIA NPP is a library of functions for performing CUDA accelerated 2D image and signal processing.

The primary set of functionality in the library focuses on image processing and is widely applicable for developers in these areas. NPP will evolve over time to encompass more of the compute heavy tasks in a variety of problem domains. The NPP library is written to maximize flexibility, while maintaining high performance.

(*See* https://docs.nvidia.com/cuda/npp/introduction.html (emphasis added).)

108. As an example, the exemplary CUDA NPP library passes image data using "[a] pointer to the image's underlying data type" and "[a] line step in bytes." In this example, the pointer is passed "to the underlying pixel data type" and the pointer and line step are passed individually for processing involving "image data."

Image Data

Image data is passed to and from NPPI primitives via a pair of parameters:

1. A pointer to the image's underlying data type.
2. A line step in bytes (also sometimes called line stride).

The general idea behind this fairly low-level way of passing image data is ease-of-adoption into existing software projects:

- Passing a raw pointer to the underlying pixel data type, rather than structured (by color) channel pixel data allows usage of the function in a wide variety of situations avoiding risky type cast or expensive image data copies.
- Passing the data pointer and line step individually rather than a higher-level image struct again allows for easy adoption by not requiring a specific image representation and thus avoiding awkward packing and unpacking of image data from the host application to an NPP specific image representation.

(*See* https://docs.nvidia.com/cuda/npp/introduction.html#nppi_conventions_lb_1passing_image_data (emphasis added).)

109. As another example, the exemplary CUDA NPP library implements function for image color conversion. These functions "manipulat[e] an image's color model and sampling format" and "can be found in the nppicc [NVIDIA Performance Primitives Image Color Conversion] library." As shown, these functions save "application load time" and "CUDA runtime."

Image Color Conversion Functions

Routines manipulating an image's color model and sampling format.

These functions can be found in the nppicc library. Linking to only the sub-libraries that you use can significantly save link time, application load time, and CUDA runtime startup time when using dynamic libraries.

(*See* https://docs.nvidia.com/cuda/npp/image_color_conversion.html#image-color-model-conversion-functions (emphasis added).)

110. The Accused Products perform a method that includes *transferring the first input*

data received by the CPU to the memory operably coupled to the GPU. For instance, the "CUDA programming model" implements programming functions and instructions for CPUs and GPUs. "The host is the CPU available in the system" and "system memory associated with the CPU is called host memory." "The GPU is called a device and GPU memory likewise called device memory." As an example, the first main CUDA program execution step is "[c]opy[ing] the ***input data from host [CPU] memory to device [GPU] memory***, also known as host-to-device transfer."

> Let me introduce two keywords widely used in CUDA programming model: *host* and *device*.
>
> The host is the CPU available in the system. The system memory associated with the CPU is called host memory. The GPU is called a device and GPU memory likewise called device memory.
>
> To execute any CUDA program, there are three main steps:
>
> - Copy the input data from host memory to device memory, also known as host-to-device transfer.
> - Load the GPU program and execute, caching data on-chip for performance.
> - Copy the results from device memory to host memory, also called device-to-host transfer.

(*See* https://developer.nvidia.com/blog/cuda-refresher-cuda-programming-model/ (emphasis added).)

111. The Accused Products perform a method that includes *performing, by the graphics processing unit (GPU), a first computation in the sequence of computations on the first input data based on the textures and shaders to generate first output data, computations in the sequence of computations representing respective layers of neurons in the artificial neural network, an output of the first computation in the sequence of computations representing an output of a first neuron in a first layer in the artificial neural network*. For instance, the CUDA platform programming implemented in the Accused Products utilizes the GPU and GPU memory. As an example, after the "host-to-device transfer" (host (CPU) memory to device (GPU) memory), the second main

step is "[l]oad the GPU program and execute, caching data on-chip for performance."

Let me introduce two keywords widely used in CUDA programming model: *host* and *device*.

The host is the CPU available in the system. The system memory associated with the CPU is called host memory. The GPU is called a device and GPU memory likewise called device memory.

To execute any CUDA program, there are three main steps:

- Copy the input data from host memory to device memory, also known as host-to-device transfer.
- Load the GPU program and execute, caching data on-chip for performance.
- Copy the results from device memory to host memory, also called device-to-host transfer.

(*See* https://developer.nvidia.com/blog/cuda-refresher-cuda-programming-model/ (emphasis added).)

112. In addition, as previously stated, cuDNN is a CUDA "GPU-acceleration library of primitives for deep neural networks." (*See* https://developer.nvidia.com/cudnn.) The exemplary cuDNN release notes below demonstrate computations implemented for RNNs and related data being transferred to GPU memory. As shown, users do "not need to transfer [an] array [from RNN data descriptors] to device memory; the operation will be performed automatically by RNN APIs."

cuDNN Release Notes

NVIDIA CUDA Deep Neural Network (cuDNN) is a GPU-accelerated library of primitives for deep neural networks. It provides highly tuned implementations of routines arising frequently in DNN applications. These release notes describe the key features, software enhancements and improvements, and known issues for the NVIDIA cuDNN 8.9.3 and earlier releases.

* * * * *

- RNN and multihead attention API calls may exhibit nondeterministic behavior when the cuDNN library is built with CUDA Toolkit 10.2 or higher. This is the result of a new buffer management and heuristics in the cuBLAS library. As described in Results Reproducibility, numerical results may not be deterministic when cuBLAS APIs are launched in more than one CUDA stream using the same cuBLAS handle. This happens when two buffer sizes (16 KB and 4 MB) are used in the default configuration.
  When a larger buffer size is not available at runtime, instead of waiting for a buffer of that size to be released, a smaller buffer may be used with a different GPU kernel. The kernel selection may affect numerical results. The user can eliminate the nondeterministic behavior of cuDNN RNN and multihead attention APIs, by setting a single buffer size in the `CUBLAS_WORKSPACE_CONFIG` environmental variable, for example, `:16:8` or `:4096:2`.
  The first configuration instructs cuBLAS to allocate eight buffers of 16 KB each in GPU memory while the second setting creates two buffers of 4 MB each. The default buffer configuration in cuBLAS 10.2 and 11.0 is `:16:8:4096:2`, that is, we have two buffer sizes. In earlier cuBLAS libraries, such as cuBLAS 10.0, it used the `:16:8` non-adjustable configuration. When buffers of only one size are available, the behavior of cuBLAS calls is deterministic in multi-stream setups.

* * * * *

- cuDNN 8.9.1 added tensor alignment checks to instance norm and layer norm engines to prevent IMA issues.

- Starting in cuDNN 8.9.1, the `const int32_t devSeqLengths[]` argument in `cudnnRNNForward()`, `cudnnRNNBackwardData_v8()`, and `cudnnRNNBackwardWeights_v8()` APIs will be ignored. All three functions will source variable sequence length arrays from RNN data descriptors, configured through the `seqLengthArray` parameter of `cudnnSetRNNDataDescriptor()`. The user does not need to transfer this array to device memory; the operation will be performed automatically by RNN APIs. This refinement simplifies the usage of cuDNN RNN APIs. It is also a workaround for random crashes in multi-GPU RNN training on TensorFlow. Replacing earlier versions of cuDNN 8.x shared libraries with cuDNN 8.9.1 will eliminate those crashes without forcing the user to switch the TensorFlow version. The cause of intermittent corruptions of `devSeqLengths[]`, fed to RNN APIs, is still being investigated.

(*See* https://docs.nvidia.com/deeplearning/cudnn/archives/cudnn-893/release-notes/index.html#abstract (emphasis added).)

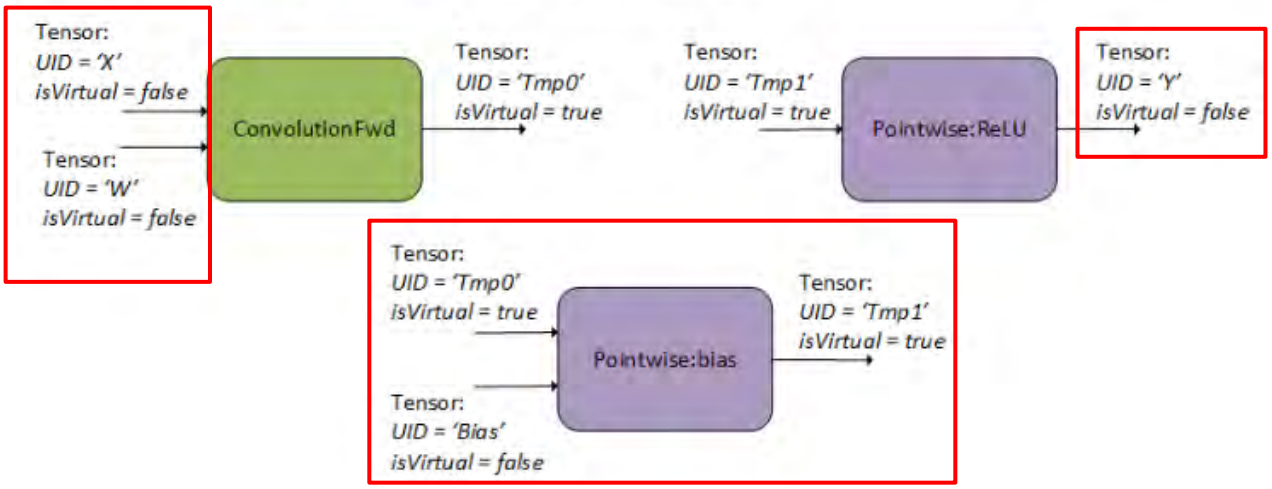
113. Relatedly, cuDNN operations below exemplify tensors being used as inputs and outputs (*e.g.*, *Tmp0*). Exemplary "cuDNN operations take tensors as input and produce tensors as output." As part of CUDA, these cuDNN operations implement computer tasks performed by a GPU.

2.2. Tensors and Layouts

Whether using the graph API or the legacy API, cuDNN operations take tensors as input and produce tensors as output.

* * * * *

Figure 6. A set of operation descriptors the user passes to the operation graph



(*See* https://docs.nvidia.com/deeplearning/cudnn/archives/cudnn-881/developer-guide/index.html#tensors-layouts (emphasis added).)

114. The Accused Products perform a method that includes *storing, in the memory operably coupled to the GPU, the first input data and the first output data*. For instance, as illustrated below, a diagram describing the architecture of the Grace Hopper Superchip depicts a GPU ("Hopper GPU") in communication with GPU memory ("GPUHBM3 or HBm3e" high bandwidth memory).

Figure 1 shows the logical overview of the NVIDIA GH200 Grace Hopper Superchip and Table 1 lists its key features.

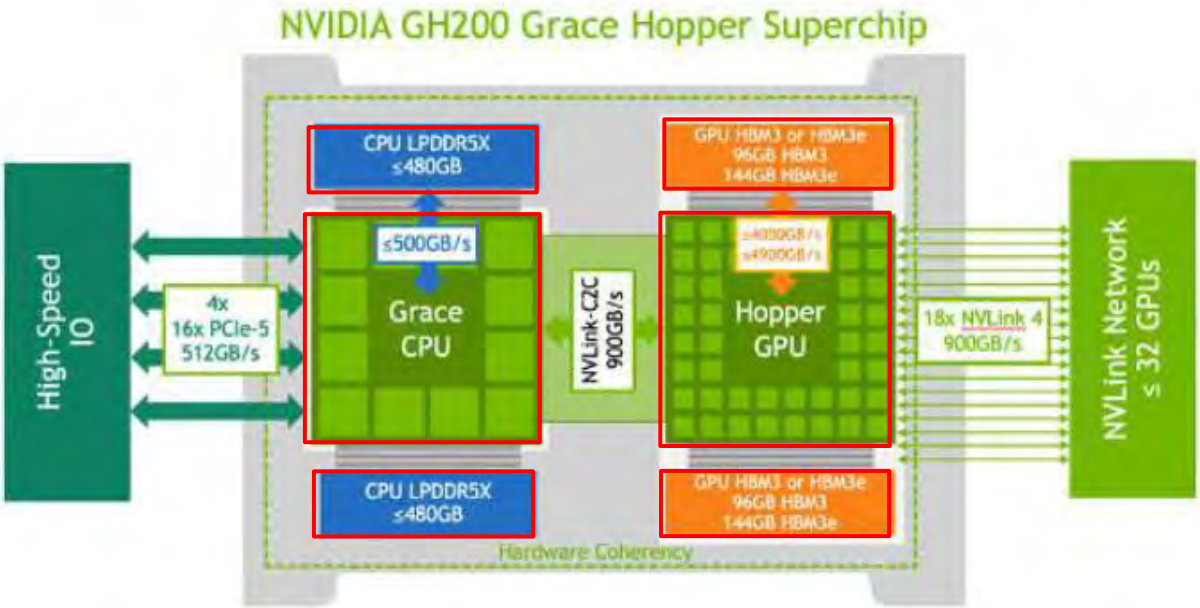


Figure 1. NVIDIA GH200 Grace Hopper Superchip Logical Overview

Table 1. NVIDIA GH200 Grace Hopper Superchip Key Features

Feature	Description
Grace CPU cores (number)	Up to 72 cores
CPU LPDDR5X bandwidth (GB/s)	Up to 500GB/s
GPU HBM bandwidth (GB/s)	4TB/s HBM3 4.9TB/s HBM3e
NVLink-C2C bandwidth (GB/s)	900GB/s total, 450GB/s per direction
CPU LPDDR5X capacity (GB)	Up to 480GB
GPU HBM capacity (GB)	96GB HBM3 144GB HBM3e
PCIe Gen 5 Lanes	64x

(*See* https://resources.nvidia.com/en-us-grace-cpu/nvidia-grace-hopper (emphasis added).)

115. As previously stated, the "CUDA programming model" implements programming functions and instructions for CPUs (host) and GPUs (device). For example, after the "host-to-device transfer" (CPU to GPU) first main step and "[l]oad[ing] the GPU program and execut[ing]"

and "caching data on-chip for performance" for the second main step, the "results" are stored on GPU "device memory."

> Let me introduce two keywords widely used in CUDA programming model: *host* and *device*.
>
> The host is the CPU available in the system. The system memory associated with the CPU is called host memory. The GPU is called a device and GPU memory likewise called device memory.
>
> To execute any CUDA program, there are three main steps:
>
> - Copy the input data from host memory to device memory, also known as host-to-device transfer.
> - Load the GPU program and execute, caching data on-chip for performance.
> - Copy the results from device memory to host memory, also called device-to-host transfer.

(*See* https://developer.nvidia.com/blog/cuda-refresher-cuda-programming-model/ (emphasis added.)

116. The Accused Products perform a method that includes *transferring second input data acquired from the external system in real time into the memory operably coupled to the GPU after the GPU starts the first computation and before the GPU starts a second computation of the sequence of computations, an output of the second computation in the sequence of computations representing an output of a second neuron in a second layer in the artificial neural network*. For instance, as illustrated below, a diagram describing the architecture of the Grace Hopper Superchip depicts a CPU ("Grace CPU") in communication with a GPU ("Hopper GPU") via "NVLink-C2C" (chip-to-chip). "High-Speed IO" input and output data is received by the CPU via "PCIe-5."

Figure 1 shows the logical overview of the NVIDIA GH200 Grace Hopper Superchip and Table 1 lists its key features.

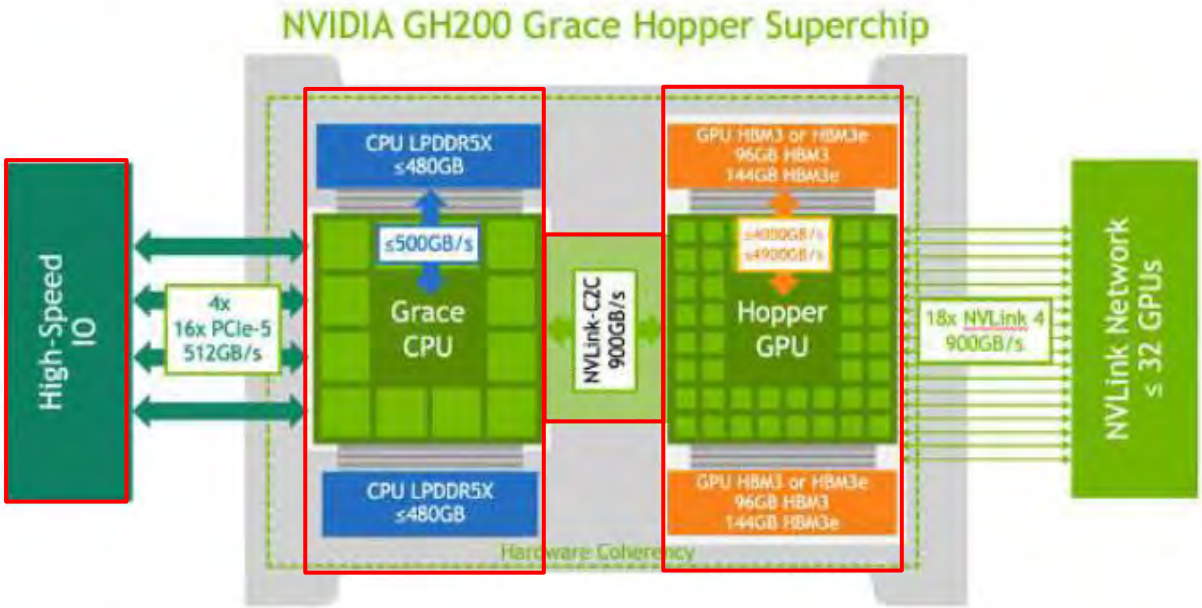


Figure 1. NVIDIA GH200 Grace Hopper Superchip Logical Overview

Table 1. NVIDIA GH200 Grace Hopper Superchip Key Features

Feature	Description
Grace CPU cores (number)	Up to 72 cores
CPU LPDDR5X bandwidth (GB/s)	Up to 500GB/s
GPU HBM bandwidth (GB/s)	4TB/s HBM3 4.9TB/s HBM3e
NVLink-C2C bandwidth (GB/s)	900GB/s total, 450GB/s per direction
CPU LPDDR5X capacity (GB)	Up to 480GB
GPU HBM capacity (GB)	96GB HBM3 144GB HBM3e
PCIe Gen 5 Lanes	64x

(*See* https://resources.nvidia.com/en-us-grace-cpu/nvidia-grace-hopper (emphasis added).)

117. Furthermore, the Accused Products, including the Grace Hopper Superchip, implement libraries and SDKs designed for neural networks that "are created from large numbers of identical neurons [that] are highly parallel by nature." The Accused Products implement

cuDNN, a library "makes it easy to obtain state-of-the-art performance with Deep Neural Networks," and TensorRT, a platform accelerator and runtime for optimizing, validating, and deploying neural networks for inference (e.g., applying knowledge from a trained neural network model and inferring a result).

Accelerating Artificial Neural Networks with GPUs

State-of-the-art Neural Networks can have from millions to well over one billion parameters to adjust via back-propagation. They also require a large amount of training data to achieve high accuracy, meaning hundreds of thousands to millions of input samples will have to be run through both a forward and backward pass. Because neural nets are created from large numbers of identical neurons they are highly parallel by nature. This parallelism maps naturally to GPUs, which provide a significant computation speed-up over CPU-only training.

GPUs have become the platform of choice for training large, complex Neural Network-based systems because of their ability to accelerate the systems. Because of the increasing importance of Neural networks in both industry and academia and the key role of GPUs, NVIDIA has a library of primitives called cuDNN that makes it easy to obtain state-of-the-art performance with Deep Neural Networks.

The parallel nature of inference operations also lend themselves well for execution on GPUs. To optimize, validate, and deploy networks for inference, NVIDIA has an inference platform accelerator and runtime called TensorRT. TensorRT delivers low-latency, high-throughput inference and tunes the runtime application to run optimally across different families of GPUs.

(*See* https://developer.nvidia.com/discover/artificial-neural-network (emphasis added).)

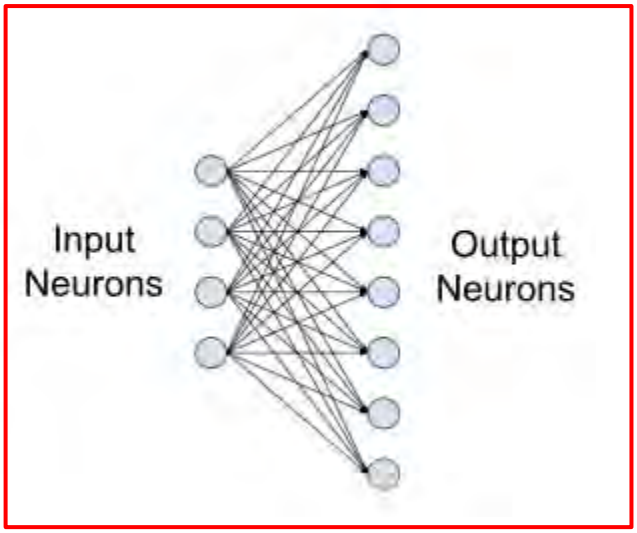
118. An example below illustrates an exemplary neural network the Accused Products are designed to accelerate using parallel computations. "Input" (four) and "Output" (eight) neurons are depicted below in a full-connected or linear layer structure in which all of the input neurons depicted in a first layer are connected to all of the output neurons depicted in a second layer. Computations for the neural network are performed using, for example, "NVIDIA Matrix Multiplication." Examples of inputs and outputs for forward propagation, activation gradient computation, and weight gradient computation (as matrix by matrix multiplications) are shown below.

- As a rough guideline, choose batch sizes and neuron counts greater than 128 to avoid being limited by memory bandwidth (NVIDIA® A100-SXM4-80GB; this threshold is similar for other A100 and V100 GPUs); see **Batch Size**.

2. Fully-Connected Layer

Fully-connected layers, also known as linear layers, connect every input neuron to every output neuron and are commonly used in neural networks.

Figure 1. Example of a small fully-connected layer with four input and eight output neurons.



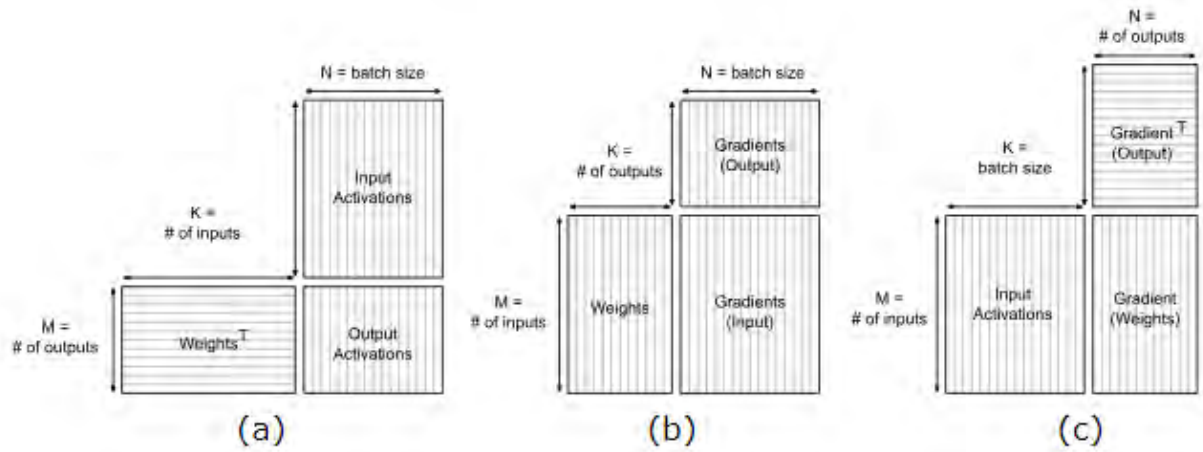
Three parameters define a fully-connected layer: batch size, number of inputs, and number of outputs. Forward propagation, activation gradient computation, and weight gradient computation are directly expressed as matrix-matrix multiplications. How the three parameters map to GEMM dimensions (General Matrix Multiplication, background in the **NVIDIA Matrix Multiplication Background User's Guide**) varies among frameworks, but the underlying principles are the same. For the purposes of the discussion, we adopt the convention used by PyTorch and Caffe where A contains the weights and B the activations. In TensorFlow, matrices take the opposite roles, but the performance principles are the same.

Table 1. Mapping of inputs, outputs, and batch size to GEMM parameters M, N, K.

Computation Phase	M	N	K
Forward Propagation	Number of outputs	Batch size	Number of inputs
Activation Gradient	Number of inputs	Batch size	Number of outputs
Weight Gradient	Number of inputs	Number of outputs	Batch size

* * * * *

Figure 2. Dimensions of equivalent GEMMs for (a) forward propagation, (b) activation gradient, and (c) weight gradient computations of a fully-connected layer.



(*See* https://docs.nvidia.com/deeplearning/performance/dl-performance-fully-connected/index.html#performance (annotations added).)

119. Indeed, the cuDNN GPU-acceleration library of the Accused Products implement operations that "take tensors as input and produce tensors as output."

2.2. Tensors and Layouts

Whether using the graph API or the legacy API, cuDNN operations take tensors as input and produce tensors as output.

(*See* https://docs.nvidia.com/deeplearning/cudnn/archives/cudnn-881/developer-guide/index.html#tensors-layouts (emphasis added).)

120. Each claim in the '438 Patent recites an independent invention. Neither claim 21, described above, nor any other individual claim is representative of all claims in the '438 Patent.

121. Defendant has been aware of the '438 Patent since at least the filing of the original Complaint. Defendant has been aware of the technology patented by the '438 Patent since at least 2007, when the inventors of the Asserted Patents first discussed their patented technologies with Mr. Sanford Russell, then the CTO of Nvidia. At the time, the inventors asked Defendant to collaborate with them on training neural networks using Nvidia's GPUs. Defendant informed the inventors, through Mr. Russell, that it was not interested in the collaboration. Defendant has also cited an ancestor of the '438 Patent in its own patent portfolio since at least June 28, 2010 (*See* https://patents.google.com/patent/US8648867B2/en?oq=8648867#citedBy; https://patentimages.storage.googleapis.com/ee/13/e9/61df149c3fddc7/US8922566.pdf; https://patentcenter.uspto.gov/applications/13335850/displayReferences/referenceForms?application= (Nvidia U.S. Appl. No. 13/335,850 August 12, 2014, List of References Cited by Examiner).)

122. Starting in or around 2016, the inventors of the Asserted Patents held multiple discussions with Nvidia to invest in or purchase their AI company, Neurala, Inc., and all its assets, including the '438 Patent family. These discussions included at least Mr. Alvin Lin, an Nvidia Senior Director of Business Development, and Mr. Jeff Herbst, then an Nvidia Vice President of Business Development and head of Nvidia's Inception GPU Ventures, in or around September 6,

2016. In or around October 2016, Nvidia, through its representatives, initiated discussions with the inventors to invest in Neurala, Inc. for approximately $10 million.

123. The inventors also discussed their patented technology, including the underlying technology and family to the ’438 Patent (including U.S. Patent No. 9,189,828, the patent the ’438 Patent reissued from), with Defendant’s representatives at Nvidia’s artificial intelligence conference in or around June 2017. On or about June 26, 2017, Defendant received materials from the inventors, in lieu of a meeting on or about June 29, that identified patents related to the ’438 Patent and described the technology in detail. Defendant had previously stated it was interested in the inventors’ solutions. Defendant also featured the inventors on its website as members of Defendant’s start-up incubator on or about September 25, 2019.

Computer Vision / Video Analytics

Inception Spotlight: AI Startup Neurala Sees 7X Speedup with NGC

Sep 25, 2019 0 Like Discuss (0)

By Nefi Alarcon

* * * * *

To help businesses develop custom computer vision solutions quickly, Neurala, a member of NVIDIA’s start-up incubator Inception, has developed Brain Builder, a cloud platform that provides data scientists and developers that are new to deep learning with the ability to quickly and easily train neural networks.

(*See* https://developer.nvidia.com/blog/inception-spotlight-ai-startup-neurala-sees-7x-speedup-with-ngc/ (September 25, 2019); *see also* https://www.youtube.com/watch?v=-WBtxGLoQNs (“Neurala Accelerating AI Video Annotation with NGC Containers” posted by Defendant’s YouTube account).)

124. Neural AI and/or its predecessors-in-interest have satisfied all statutory obligations required to collect pre-filing damages for the full period allowed by law for infringement of the ’438 Patent.

125. Defendant directly infringes at least claim 21 of the ’438 Patent, either literally or under the doctrine of equivalents, by performing the steps described above. For example, Defendant performs the claimed method in an infringing manner as described above by implementing the Accused Products as part of its accelerated computing operations and running corresponding software that implements the infringing performance. Defendant also performs the claimed method in an infringing manner when testing the operation of the Accused Products and corresponding systems. As another example, Defendant performs the claimed method when providing or administering services to third parties, customers, and partners using the Accused Products.

126. Defendant’s partners, customers, and users of its Accused Products and corresponding systems and services directly infringe at least claim 21 of the ’438 Patent, literally or under the doctrine of equivalents, at least by using the Accused Products and corresponding systems and services, as described above.

127. Defendant has actively induced and is actively inducing infringement of at least claim 21 of the ’438 Patent with specific intent to induce infringement, and/or willful blindness to the possibility that its acts induce infringement, in violation of 35 U.S.C. § 271(b). For example, Defendant encourages and induces customers to use Nvidia’s CUDA platform in a manner that infringes claim 21 of the ’438 Patent at least by offering and providing software that performs a method that infringes claim 21 when installed and operated by the customer using the Accused Products, and by engaging in activities relating to selling, marketing, advertising, promotion,

installation, support, and distribution of the Accused Products.

128. Defendant encourages, instructs, directs, and/or requires third parties—including its certified partners and/or customers—to perform the claimed method using the software, platform, services, and systems in infringing ways, as described above.

129. Defendant further encourages and induces its customers to infringe claim 21 of the ’438 Patent: 1) by making its accelerated computing and data center services available on its website, providing applications that allow users to access those services, widely advertising those services, and providing technical support and instructions to users (*see* https://www.nvidia.com/en-us/data-center/data-center-gpus/gpu-test-drive/); and 2) through activities relating to marketing, advertising, promotion, installation, support, and distribution of the Accused Products, including its CUDA platform, and services in the United States. (*See* https://www.nvidia.com/en-us/; *see* https://www.nvidia.com/en-us/about-nvidia/partners/; https://www.nvidia.com/en-us/data-center/where-to-buy/; https://www.nvidia.com/en-us/data-center/where-to-buy-tesla/.)

130. For example, Defendant shares instructions, guides, and manuals, which advertise and instruct third parties on how to use its hardware and platform as described above, including at least customers and partners. (*See* https://docs.nvidia.com/cuda/cuda-c-programming-guide/.) Defendant also provides customer service and technical support to purchasers of the Accused Products and corresponding systems and services, which directs and encourages customers to perform certain actions that use the Accused Products in an infringing manner. (*See* https://www.nvidia.com/en-us/support/; https://www.nvidia.com/en-us/support/enterprise/services/.)

131. Defendant and/or Defendant’s partners recommend and sell the Accused Products

and provide technical support for the installation, implementation, integration, and ongoing operation of the Accused Products for each individual customer. On information and belief, each customer enters into a contractual relationship with Defendant and/or one of Defendant's partners, which obligates each customer to perform certain actions in order to use the Accused Products. (*See* https://www.nvidia.com/en-us/agreements/; https://www.nvidia.com/en-us/agreements/cloud-services/nvidia-cloud-agreement/; https://www.nvidia.com/en-us/agreements/cloud-services/service-specific-terms-for-nvidia-dgx-cloud/.) Further, in order to receive the benefit of Defendant's and/or its partner's continued technical support and their specialized knowledge and guidance of the operability of the Accused Products, each customer must continue to use the Accused Products in a way that infringes the '438 Patent. (*See* https://www.nvidia.com/en-us/support/.)

132. Further, as the entity that provides installation, implementation, and integration of the Accused Products in addition to ensuring the Accused Product remains operational for each customer through ongoing technical support, on information and belief, Defendant and/or Defendant's partners affirmatively aid and abet each customer's use of the Accused Products in a manner that performs the claimed method of, and infringes, the '438 Patent.

133. Defendant also contributes to the infringement of its partners, customers, and users of the Accused Products by providing within the United States or importing into the United States the Accused Products, which are for use in practicing, and under normal operation practice, the methods, systems, and devices claimed in the Asserted Patents, constituting a material part of the inventions claimed, and not a staple article or commodity of commerce suitable for substantial non-infringing uses. Indeed, as shown above, the Accused Products and the example functionality have no substantial non-infringing uses but are specifically designed to practice the '438 Patent.

134. On information and belief, the infringing actions of each partner, customer, and/or user of the Accused Products are attributable to Defendant. For example, on information and belief, Defendant directs and controls the activities or actions of its partners or others in connection with the Accused Products by contractual agreement or otherwise requiring partners or others to provide information and instructions to customers who acquire the Accused Products which, when followed, results in infringement. Defendant further directs and controls the operation of devices executing the Accused Products by programming the software which, when executed by a customer or user, performs the claimed method of at least claim 21 of the ’438 Patent.

135. Plaintiff has suffered and continues to suffer damages as a result of Defendant’s infringement of the ’438 Patent. Defendant is therefore liable to Plaintiff under 35 U.S.C. § 284 for damages in an amount that adequately compensates Plaintiff for Defendant’s infringement, but no less than a reasonable royalty.

136. Defendant’s infringement of the ’438 Patent is knowing and willful. Defendant acquired actual knowledge of the patent that the ’438 Patent reissued from, and its family, since at least 2017 and has acquired additional knowledge of the ’438 Patent since at least the filing of this lawsuit.

137. On information and belief, despite Defendant’s knowledge of the Asserted Patents and Plaintiff’s patented technology, Defendant made the deliberate decision to sell products and services that it knew infringe these patents. Defendant’s continued infringement of the ’438 Patent with knowledge of the ’438 Patent constitutes willful infringement.

THIRD CAUSE OF ACTION
(INFRINGEMENT OF THE ’461 PATENT)

138. Plaintiff realleges and incorporates by reference the allegations of the preceding paragraphs of this Complaint.

139. Defendant has infringed and continues to infringe one or more claims of the ’461 Patent in violation of 35 U.S.C. § 271 in this District and elsewhere in the United States and will continue to do so. The Accused Products, including features of, *e.g.*, the Grace Hopper Superchip (GH200), at least when used for their ordinary and customary purposes, practice each element of at least claim 21 of the ’461 Patent as demonstrated below.

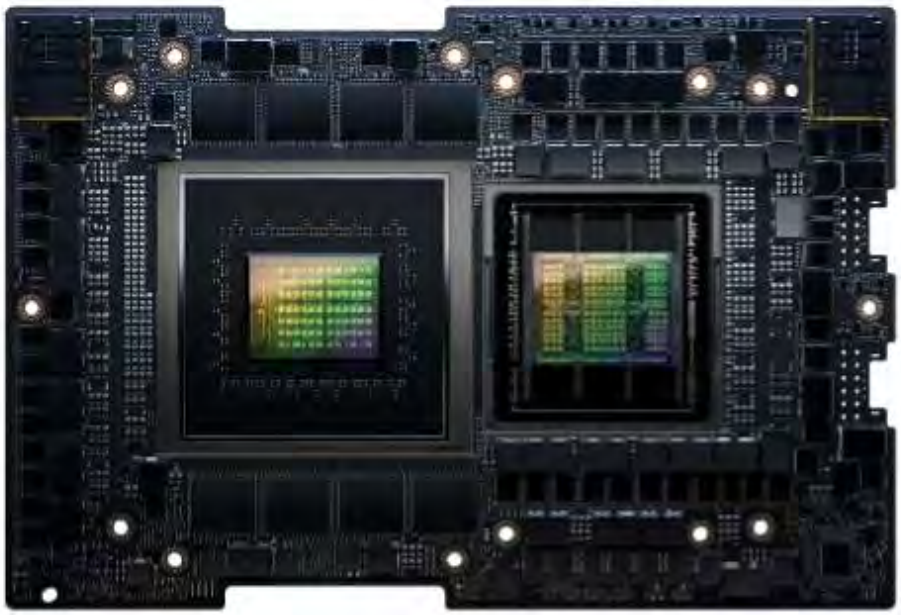
140. For example, claim 21 of the ’461 Patent recites:

> 21. A method of executing computations representing an artificial neural network on a computer system comprising at least one central processing unit (CPU), a processing unit, a first memory partition, and a second memory partition, the method comprising:
>
> executing, by the at least one CPU, a user interaction stream, the user interaction stream controlling transfer of inputs to the artificial neural network to the first memory partition and the second memory partition;
>
> executing, by the processing unit, a computational stream, the computational stream controlling data exchange between the user interaction stream and the computational stream during execution of the computations representing the artificial neural network;
>
> shifting control of a data exchange between the user interaction stream and the computational stream to the computational stream in response to starting execution of the computations representing the artificial neural network;
>
> shifting control of the data exchange between the user interaction stream and the computational stream to the user interaction stream in response to completion or interruption of the computations representing the artificial neural network;
>
> queueing a user command received by the user interaction stream during execution of the computations representing the artificial neural network; and
>
> executing the user command during execution of the computations representing the artificial neural network at times determined by the computational stream.

141. The Accused Products perform each step of the method of claim 21 of the ’461

Patent. To the extent the preamble is construed to be limiting, the Accused Products perform *a method of executing computations representing an artificial neural network on a computer system comprising at least one central processing unit (CPU), a processing unit, a first memory partition, and a second memory partition*, as further explained below. For instance, the Grace Hopper Superchip (GH200) "brings together the groundbreaking performance of the NVIDIA Hopper GPU with the versatility of the NVIDIA Grace™ CPU . . . in a single Superchip." It includes the cuDNN (CUDA Deep Neural Network) library for "[d]eep ***neural networks***."

Inside NVIDIA's First GPU-CPU Superchip



The NVIDIA® GH200 Grace Hopper architecture brings together the groundbreaking performance of the NVIDIA Hopper GPU with the versatility of the NVIDIA Grace™ CPU, connected with a high bandwidth and memory coherent NVIDIA NVLink Chip-2-Chip (C2C)® interconnect in a single Superchip, and support for the new NVIDIA NVLink Switch System.

* * * * *

An extensive suite of domain-specific libraries and frameworks further accelerates main algorithms in a wide range of application domains, for example:

- Deep neural networks (cuDNN)
- Linear solvers for simulations and implicit unstructured methods (AmgX)
- Quantum computing (cuQuantum)
- Data science
- Machine learning (RAPIDS)
- Data loading and preprocessing for machine learning (DALI)
- Real-time 3D simulation and design collaboration (Omniverse)

(*See* https://resources.nvidia.com/en-us-grace-cpu/nvidia-grace-hopper (emphasis added).)

142. As illustrated below, a diagram describing the architecture of the Grace Hopper Superchip depicts a CPU ("Grace CPU") with "[u]p to 72 cores" and CPU memory ("CPU LPDDR5X") and a GPU ("Hopper GPU") and GPU memory ("GPUHBM3 or HBm3e").

Figure 1 shows the logical overview of the NVIDIA GH200 Grace Hopper Superchip and Table 1 lists its key features.

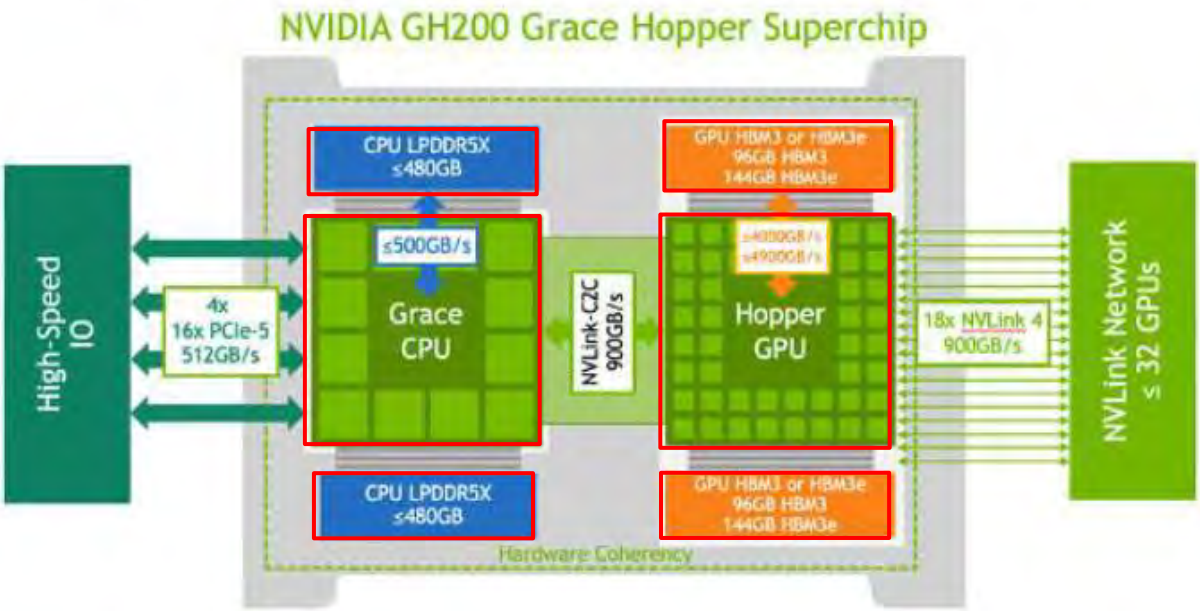


Figure 1. NVIDIA GH200 Grace Hopper Superchip Logical Overview

Table 1. NVIDIA GH200 Grace Hopper Superchip Key Features

Feature	Description
Grace CPU cores (number)	Up to 72 cores
CPU LPDDR5X bandwidth (GB/s)	Up to 500GB/s
GPU HBM bandwidth (GB/s)	4TB/s HBM3 4.9TB/s HBM3e
NVLink-C2C bandwidth (GB/s)	900GB/s total, 450GB/s per direction
CPU LPDDR5X capacity (GB)	Up to 480GB
GPU HBM capacity (GB)	96GB HBM3 144GB HBM3e
PCIe Gen 5 Lanes	64x

(*See* https://resources.nvidia.com/en-us-grace-cpu/nvidia-grace-hopper (emphasis added).)

143. The "Grace Hopper Superchip is the first true heterogeneous accelerated platform for high-performance computing (HPC) and AI workloads. It accelerates applications with the strengths of both GPUs and CPUs while providing the simplest and most productive heterogeneous programming model to date."

The NVIDIA GH200 Grace Hopper Superchip is the first true heterogeneous accelerated platform for high-performance computing (HPC) and AI workloads. It accelerates applications with the strengths of both GPUs and CPUs while providing the simplest and most productive heterogeneous programming model to date, enabling scientists and engineers to focus on solving the world's most important problems. Together with NVIDIA networking technologies, NVIDIA GH200 provides the recipe for the next generation of HPC supercomputers and AI factories, enabling customers to take on larger datasets, more complex models, and new workloads, solving them more quickly than before.

(*See* https://resources.nvidia.com/en-us-grace-cpu/nvidia-grace-hopper (emphasis added).)

144. In addition, the Accused Products, including the Grace Hopper Superchip, implement CUDA, Nvidia's proprietary "parallel computing platform and programming model." CUDA further includes the CUDA Toolkit, which "includes GPU-accelerated libraries, a compiler, development tools and the CUDA runtime." As an example, the "CUDA® Deep Neural

Network library (cuDNN) is a GPU-acceleration library of primitives for ***deep neural networks***." It "provides highly tuned implementations for standard routines" for GPU-based acceleration.

CUDA Zone

CUDA® is a parallel computing platform and programming model developed by NVIDIA for general computing on graphical processing units (GPUs). With CUDA, developers are able to dramatically speed up computing applications by harnessing the power of GPUs.

In GPU-accelerated applications, the sequential part of the workload runs on the CPU – which is optimized for single-threaded performance – while the compute intensive portion of the application runs on thousands of GPU cores in parallel. When using CUDA, developers program in popular languages such as C, C++, Fortran, Python and MATLAB and express parallelism through extensions in the form of a few basic keywords.

The CUDA Toolkit from NVIDIA provides everything you need to develop GPU-accelerated applications. The CUDA Toolkit includes GPU-accelerated libraries, a compiler, development tools and the CUDA runtime.

(*See* https://developer.nvidia.com/cuda-zone (emphasis added).)

NVIDIA cuDNN

The NVIDIA CUDA® Deep Neural Network library (cuDNN) is a GPU-accelerated library of primitives for deep neural networks. cuDNN provides highly tuned implementations for standard routines such as forward and backward convolution, attention, matmul, pooling, and normalization.

(*See* https://developer.nvidia.com/cudnn (emphasis added).)

145. Nvidia GPU architectures that implement CUDA and cuDNN include the Hopper (*e.g.*, Grace Hopper Superchip (GH200), H100), Ada Lovelace, Ampere, Turing, Volta, Pascal, and Maxwell GPU architectures of the Accused Products.

1. GPU, CUDA Toolkit, and CUDA Driver Requirements

The following sections highlight the compatibility of NVIDIA® cuDNN versions with the various supported NVIDIA CUDA® Toolkit, CUDA driver, and NVIDIA hardware versions.

Table 1. GPU, CUDA Toolkit, and CUDA Driver Requirements

cuDNN Package[1]	CUDA Toolkit Version	Supports static linking?[2]	NVIDIA Driver Version		CUDA Compute Capability	Supported NVIDIA Hardware
			Linux	Windows		
cuDNN 8.9.6 for CUDA 12.x	12.2	Yes	>=525.60.13	>=527.41	9.0[3]	
	12.1	No			8.9[4]	NVIDIA Hopper™[5]
	12.0				8.6	NVIDIA Ada Lovelace architecture[6]
cuDNN 8.9.6 for CUDA 11.x	11.8	Yes	>= 450.80.02	>=452.39	8.0	
	11.7	No			7.5	NVIDIA Ampere architecture
	11.6				7.0	NVIDIA Turing™
	11.5				6.1	NVIDIA Volta™
	11.4				6.0	NVIDIA Pascal™
	11.3				5.0	NVIDIA Maxwell[6]
	11.2[7]					
	11.1[8]					
	11.0[9]					

(*See* https://docs.nvidia.com/deeplearning/cudnn/archives/cudnn-896/support-matrix/index.html (emphasis added).)

146. The Accused Products perform a method that includes *executing, by the at least one CPU, a user interaction stream, the user interaction stream controlling transfer of inputs to the artificial neural network to the first memory partition and the second memory partition*. For instance, as shown in the Grace Hopper Superchip architecture diagram below, the Grace Hopper Superchip is illustrated below with a CPU ("GRACE CPU"). The CPU "share[s] a single per-process page table" with a GPU ("Hopper GPU"), "enabling all CPU and GPU threads to access all system-allocated memory." The CPU is depicted as coupled to the GPU via "NVLINK C2C [chip-to-chip]," and can access the "System Page Table" and "CPU PHYSICAL MEMORY" via "CPU-resident access" and "GPU PHYSICAL MEMORY" via "[r]emote access" and "PTE [page table entry] B." The GPU can also access the System Page Table, and it can access "GPU PHYSICAL MEMORY" via "GPU-resident access" and "CPU PHYSICAL MEMORY" via "[r]emote access" and "PTE A." Moreover, the "System Page Table" "[t]ranslates CPU malloc()

[memory allocation] to CPU or GPU." "The CPU heap, CPU thread stack, global variables memory-mapped files, and inter-process memory are accessible to all CPU and GPU threads."

In NVIDIA Grace Hopper Superchip-based systems, Address Translation Service (ATS) enables the CPU and GPU to share a single per-process page table, enabling all CPU and GPU threads to access all system-allocated memory (Figure 8), which can reside on physical CPU or GPU memory. The CPU heap, CPU thread stack, global variables,

memory-mapped files, and inter-process memory are accessible to all CPU and GPU threads.

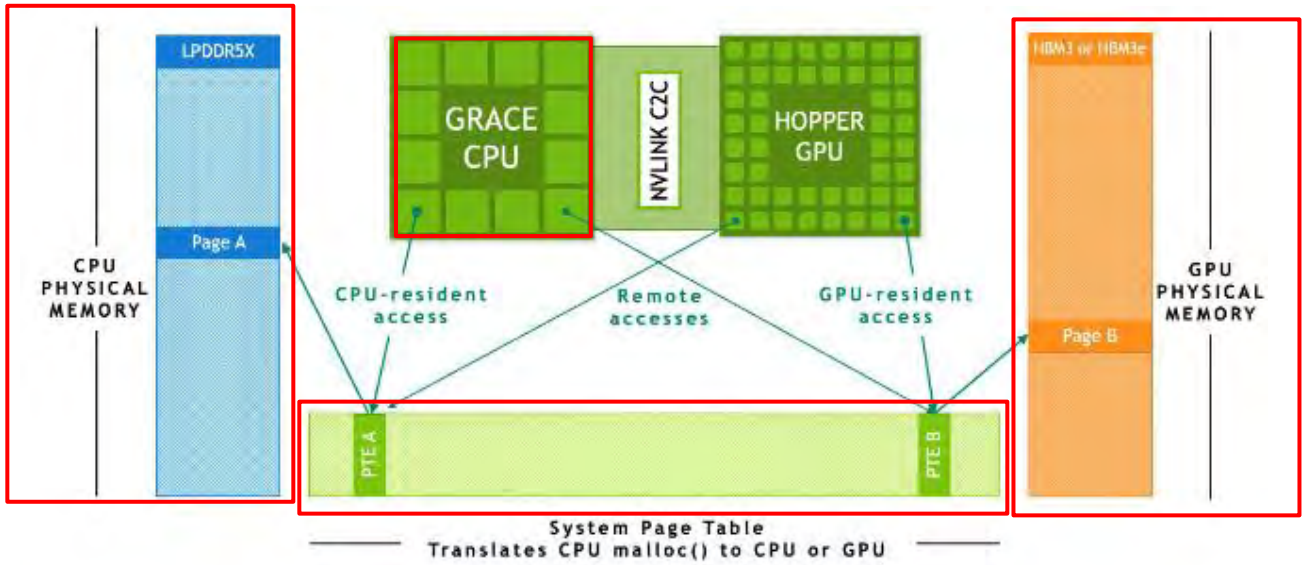


Figure 8. ATS in an NVIDIA Grace Hopper Superchip System

(*See* https://resources.nvidia.com/en-us-grace-cpu/nvidia-grace-hopper (emphasis added).)

147. The "CUDA programming model" implements programming functions and instructions for CPUs and GPUs. "The host is the CPU available in the system" and "system memory associated with the CPU is called host memory." "The GPU is called a device and GPU memory likewise called device memory." As an example, the first main CUDA program execution step is "[c]opy[ing] the ***input data from host [CPU] memory to device [GPU] memory***, also known as host-to-device transfer."

Let me introduce two keywords widely used in CUDA programming model: *host* and *device*.

The host is the CPU available in the system. The system memory associated with the CPU is called host memory. The GPU is called a device and GPU memory likewise called device memory.

To execute any CUDA program, there are three main steps:

- Copy the input data from host memory to device memory, also known as host-to-device transfer.
- Load the GPU program and execute, caching data on-chip for performance.
- Copy the results from device memory to host memory, also called device-to-host transfer.

(*See* https://developer.nvidia.com/blog/cuda-refresher-cuda-programming-model/ (emphasis added).)

148. The Accused Products perform a method that includes *executing, by the processing unit, a computational stream, the computational stream controlling data exchange between the user interaction stream and the computational stream during execution of the computations representing the artificial neural network*. For instance, the "CUDA programming model" implements programming functions and instructions for CPUs and GPUs. As previously stated, the host is the CPU and the device is the GPU. After "[c]opy[ing] the input data from host [CPU] memory to device [GPU] memory," the second main CUDA program execution step is "***[l]oad[ing] the GPU program and execut[ing]***."

Let me introduce two keywords widely used in CUDA programming model: *host* and *device*.

The host is the CPU available in the system. The system memory associated with the CPU is called host memory. The GPU is called a device and GPU memory likewise called device memory.

To execute any CUDA program, there are three main steps:

- Copy the input data from host memory to device memory, also known as host-to-device transfer.
- Load the GPU program and execute, caching data on-chip for performance.
- Copy the results from device memory to host memory, also called device-to-host transfer.

(*See* https://developer.nvidia.com/blog/cuda-refresher-cuda-programming-model/ (emphasis added).)

149. Indeed, the Grace Hopper Superchip "is designed to accelerate applications" using "Extended GPU Memory." As depicted in the gram of the Grace Hopper architecture below, a GPU ("HOPPER GPU") can access "Local CPU," "Peer CPU," and "Peer GPU" memory via "NVLink."

Accelerating Applications with Extended GPU Memory

The NVIDIA GH200 is designed to accelerate applications with exceptionally large memory footprints, larger than the capacity of the HBM3 / HBM3e and LPDDR5X memory of a single superchip (see the NVIDIA GH200 Accelerated Applications section below).

The Extended GPU Memory (EGM) feature over the high-bandwidth NVLink-C2C enables GPUs to access all the system memory efficiently. EGM provides up to 19.5TBs system memory in a multi-node NVSwitch-connected system. With EGM, physical memory in the system can be allocated to be accessible from any GPU thread. All GPUs can access EGM at the minimum of GPU-GPU NVLink or NVLink-C2C speed.

Memory accesses within a Grace Hopper Superchip configuration go through the local high-bandwidth NVLink-C2C at 900GB/s total. Remote memory accesses are performed via GPU NVLink, and depending on the memory being accessed, also NVLink-C2C as shown in Figure 5. With EGM, GPU threads can now access all memory resources available over the NVSwitch fabric, both LPDDR5X and HBM3 or HBM3e, unidirectionally at 450GB/s.

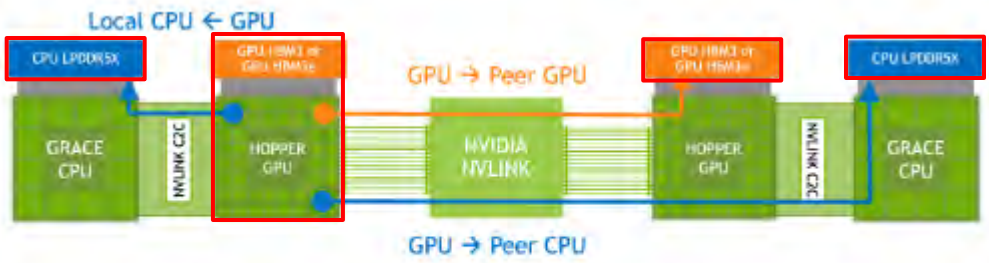


Figure 5. Memory Accesses across NVLink-connected Grace Hopper Superchips

(*See* https://resources.nvidia.com/en-us-grace-cpu/nvidia-grace-hopper (emphasis added).)

150. For instance, exemplary CUDA library cuDNN function "cudnnSetRNNDescriptor_v8" "initializes a previously created RNN [recurrent neural network] descriptor object." This function "store[s] ***all information*** needed to compute the total number of adjustable ***weights/biases in the RNN model***." In addition, the parameters "dirMode," "inputMode," and "datatype" confirm the exchange of calculations and values between the hidden layers of an RNN.

7.2.49. cudnnSetRNNDescriptor_v8()

This function initializes a previously created RNN descriptor object. The RNN descriptor configured by `cudnnSetRNNDescriptor_v8()` was enhanced to store all information needed to compute the total number of adjustable weights/biases in the RNN model.

```
cudnnStatus_t cudnnSetRNNDescriptor_v8(
 cudnnRNNDescriptor_t rnnDesc,
 cudnnRNNAlgo_t algo,
 cudnnRNNMode_t cellMode,
 cudnnRNNBiasMode_t biasMode,
 cudnnDirectionMode_t dirMode,
 cudnnRNNInputMode_t inputMode,
 cudnnDataType_t dataType,
 cudnnDataType_t mathPrec,
 cudnnMathType_t mathType,
 int32_t inputSize,
 int32_t hiddenSize,
 int32_t projSize,
 int32_t numLayers,
    cudnnDropoutDescriptor_t dropoutDesc,
    uint32_t auxFlags);
```

* * * * *

dirMode

Input. Specifies the recurrence pattern: `CUDNN_UNIDIRECTIONAL` or `CUDNN_BIDIRECTIONAL`. In bidirectional RNNs, the hidden states passed between physical layers are concatenations of forward and backward hidden states.

inputMode

Input. Specifies how the input to the RNN model is processed by the first layer. When `inputMode` is `CUDNN_LINEAR_INPUT`, original input vectors of size `inputSize` are multiplied by the weight matrix to obtain vectors of `hiddenSize`. When `inputMode` is `CUDNN_SKIP_INPUT`, the original input vectors to the first layer are used as is without multiplying them by the weight matrix.

dataType

Input. Specifies data type for RNN weights/biases and input and output data.

(*See* https://docs.nvidia.com/deeplearning/cudnn/archives/cudnn-891/pdf/cuDNN-API.pdf (emphasis added).)

151. The Accused Products perform a method that includes *shifting control of a data exchange between the user interaction stream and the computational stream to the computational stream in response to starting execution of the computations representing the artificial neural network*. For instance, the "CUDA programming model" implements programming functions and instructions for CPUs (host) and GPUs (device). As an example, the "host-to-device transfer" (CPU to GPU) first main step, the second main step is "[l]oad the GPU program and ***execute***" and the third main step is "***[c]opy the results from device [GPU] memory to host [CPU] memory***, also known as device-to-host transfer" (GPU to CPU).

Let me introduce two keywords widely used in CUDA programming model: *host* and *device*.

The host is the CPU available in the system. The system memory associated with the CPU is called host memory. The GPU is called a device and GPU memory likewise called device memory.

To execute any CUDA program, there are three main steps:

- Copy the input data from host memory to device memory, also known as host-to-device transfer.
- Load the GPU program and execute, caching data on-chip for performance.
- Copy the results from device memory to host memory, also called device-to-host transfer.

(*See* https://developer.nvidia.com/blog/cuda-refresher-cuda-programming-model/ (emphasis added.)

152. As previously stated, the Grace Hopper Superchip "is designed to accelerate applications" using "Extended GPU Memory" and the GPU can access local/peer CPU and peer GPU memory via "NVLink." The Grace Hopper Superchip's Extended GPU Memory feature "enables GPUs to access all the system memory efficiently" and "physical memory in the system can be allocated to be accessible from any GPU thread."

Accelerating Applications with Extended GPU Memory

The NVIDIA GH200 is designed to accelerate applications with exceptionally large memory footprints, larger than the capacity of the HBM3 / HBM3e and LPDDR5X memory of a single superchip (see the NVIDIA GH200 Accelerated Applications section below).

The Extended GPU Memory (EGM) feature over the high-bandwidth NVLink-C2C enables GPUs to access all the system memory efficiently. EGM provides up to 19.5TBs system memory in a multi-node NVSwitch-connected system. With EGM, physical memory in the system can be allocated to be accessible from any GPU thread. All GPUs can access EGM at the minimum of GPU-GPU NVLink or NVLink-C2C speed.

Memory accesses within a Grace Hopper Superchip configuration go through the local high-bandwidth NVLink-C2C at 900GB/s total. Remote memory accesses are performed via GPU NVLink, and depending on the memory being accessed, also NVLink-C2C as shown in Figure 5. With EGM, GPU threads can now access all memory resources available over the NVSwitch fabric, both LPDDR5X and HBM3 or HBM3e, unidirectionally at 450GB/s.

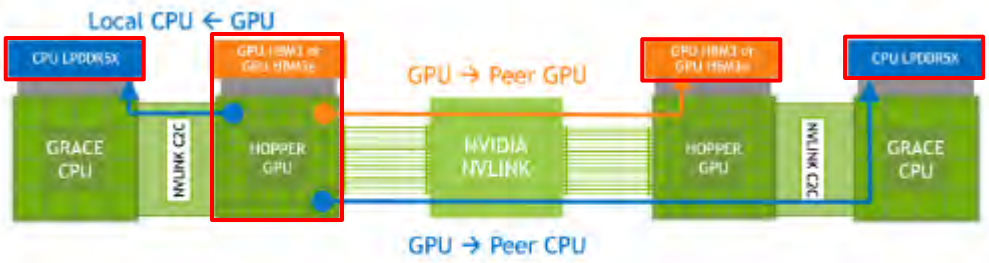


Figure 5. Memory Accesses across NVLink-connected Grace Hopper Superchips

(*See* https://resources.nvidia.com/en-us-grace-cpu/nvidia-grace-hopper (emphasis added).)

153. The Accused Products perform a method that includes *shifting control of the data exchange between the user interaction stream and the computational stream to the user interaction stream in response to completion or interruption of the computations representing the artificial neural network*. For instance, after the "CUDA programming model" "host-to-device transfer" (CPU to GPU) and GPU program load and execution steps, the third main step is "[c]opy the results from device [GPU] memory to host [CPU] memory, also known as ***device-to-host transfer***" (GPU to CPU). The "***host-to-device transfer***" (CPU to GPU) first main step can be reintroduced for additional computations.

Let me introduce two keywords widely used in CUDA programming model: *host* and *device*.

The host is the CPU available in the system. The system memory associated with the CPU is called host memory. The GPU is called a device and GPU memory likewise called device memory.

To execute any CUDA program, there are three main steps:

- Copy the input data from host memory to device memory, also known as host-to-device transfer.
- Load the GPU program and execute, caching data on-chip for performance.
- Copy the results from device memory to host memory, also called device-to-host transfer.

(*See* https://developer.nvidia.com/blog/cuda-refresher-cuda-programming-model/ (emphasis

added.)

154. In addition, as shown in the Grace Hopper Superchip architecture diagram below, the CPU ("GRACE CPU") "share[s] a single per-process page table" with a GPU ("Hopper GPU"), "enabling all CPU and GPU threads to access all system-allocated memory." "The CPU heap, CPU thread stack, global variables memory-mapped files, and inter-process memory are accessible to all CPU and GPU threads."

In NVIDIA Grace Hopper Superchip-based systems, Address Translation Service (ATS) enables the CPU and GPU to share a single per-process page table, enabling all CPU and GPU threads to access all system-allocated memory (Figure 8), which can reside on physical CPU or GPU memory. The CPU heap, CPU thread stack, global variables, memory-mapped files, and inter-process memory are accessible to all CPU and GPU threads.

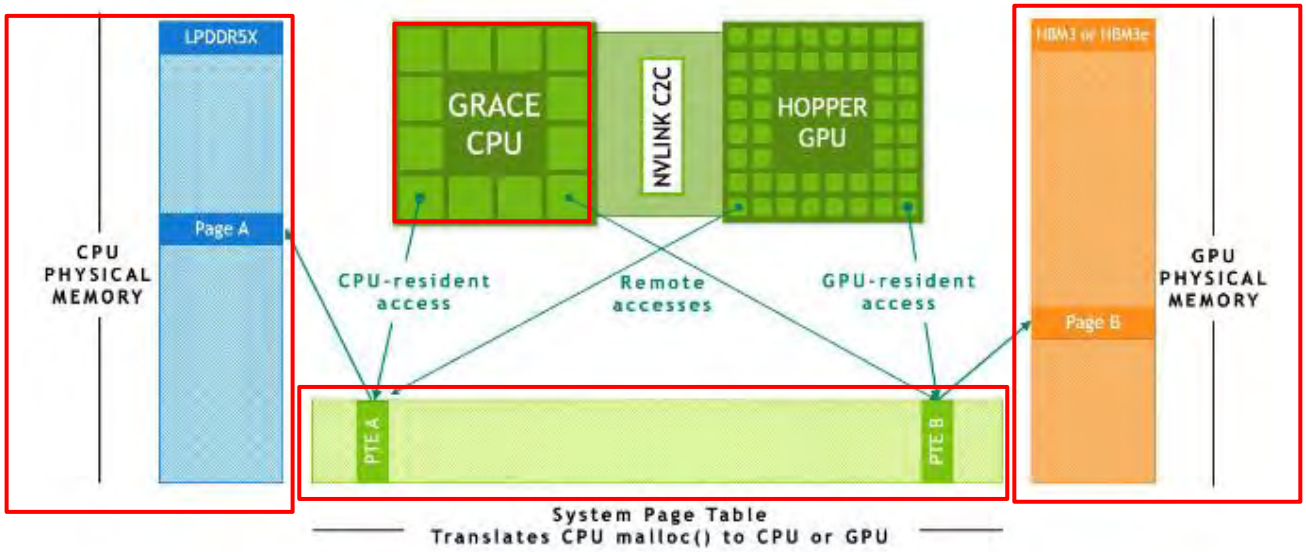


Figure 8. ATS in an NVIDIA Grace Hopper Superchip System

(*See* https://resources.nvidia.com/en-us-grace-cpu/nvidia-grace-hopper (emphasis added).)

155. The Accused Products perform a method that includes *queueing a user command received by the user interaction stream during execution of the computations representing the artificial neural network*. For instance, as shown by publicly available CUDA toolkit

documentation, CUDA implements exemplary "memory management functions" that "[c]op[y] data between host [CPU] and device [GPU]." This includes CUDA functions "cudaMemcpy" and "cudaMemcpyAsync."

6.11. Memory Management

This section describes the memory management functions of the CUDA runtime application programming interface.

Some functions have overloaded C++ API template versions documented separately in the C++ API Routines module.

Functions

* * * * *

```
__host__ cudaError_t cudaMemcpy ( void* dst , const void* src
```
Copies data between host and device.

* * * * *

```
__host__ __device__ cudaError_t cudaMemcpyAsync ( void* dst , const void
```
Copies data between host and device.

(*See* https://docs.nvidia.com/cuda/cuda-runtime-api/group__CUDART__MEMORY.html (emphasis added).)

156. As an example, exemplary CUDA memory management function "cudaMemcpyAsync" "[c]opies count bytes [data] from the memory area pointed to by src [source memory address pointer] to the memory area pointed to by dst [destination memory address pointer], where kind [type of transfer] specifies the direction of the copy." Destinations includes "cudaMemcpyHostToDevice [CPU to device GPU], cudaMemcpyDeviceToHost [GPU to CPU], cudaMemcpyDeviceToDevice [GPU to GPU]. Because the function "cudaMemcpyAsync() is asynchronous with respect to the host, [] the call may return before the copy is complete. The copy can optionally be associated to a stream [identified stream] by passing a non-zero stream argument."

__host__ __device__ cudaError_t cudaMemcpyAsync(void* dst , const void* src , size_t count , cudaMemcpyKind kind , cudaStream_t stream = 0)

Copies data between host and device.

Parameters

`dst`
- Destination memory address

`src`
- Source memory address

`count`
- Size in bytes to copy

`kind`
- Type of transfer

`stream`
- Stream identifier

Returns

cudaSuccess, cudaErrorInvalidValue, cudaErrorInvalidMemcpyDirection

Description

Copies `count` bytes from the memory area pointed to by `src` to the memory area pointed to by `dst`, where `kind` specifies the direction of the copy, and must be one of cudaMemcpyHostToHost, cudaMemcpyHostToDevice, cudaMemcpyDeviceToHost, cudaMemcpyDeviceToDevice, or cudaMemcpyDefault. Passing cudaMemcpyDefault is recommended, in which case the type of transfer is inferred from the pointer values. However, cudaMemcpyDefault is only allowed on systems that support unified virtual addressing.

The memory areas may not overlap. Calling cudaMemcpyAsync() with `dst` and `src` pointers that do not match the direction of the copy results in an undefined behavior.

cudaMemcpyAsync() is asynchronous with respect to the host, so the call may return before the copy is complete. The copy can optionally be associated to a stream by passing a non-zero `stream` argument. If `kind` is cudaMemcpyHostToDevice or cudaMemcpyDeviceToHost and the `stream` is non-zero, the copy may overlap with operations in other streams.

The device version of this function only handles device to device copies and cannot be given local or shared pointers.

(*See* https://docs.nvidia.com/cuda/cuda-runtime-api/group__CUDART__MEMORY.html (emphasis added).)

157. The Accused Products perform a method that includes *executing the user command during execution of the computations representing the artificial neural network at times determined by the computational stream*. For instance, as shown by exemplary and publicly available CUDA toolkit documentation, CUDA implements "memory management functions" that "[c]op[y] data between host [CPU] and device [GPU]."

6.11. Memory Management

This section describes the memory management functions of the CUDA runtime application programming interface.

Some functions have overloaded C++ API template versions documented separately in the C++ API Routines module.

Functions

* * * * *

```
__host__ cudaError_t cudaMemcpy ( void* dst , const void* src
```
Copies data between host and device.

* * * * *

```
__host__ __device__ cudaError_t cudaMemcpyAsync ( void* dst , const void
```
Copies data between host and device.

(*See* https://docs.nvidia.com/cuda/cuda-runtime-api/group__CUDART__MEMORY.html (emphasis added).)

158. As an example, exemplary CUDA memory management function "cudaMemcpyAsync" "[c]opies count bytes [data] from the memory area pointed to by src [source memory address pointer] to the memory area pointed to by dst [destination memory address pointer], where kind [type of transfer] specifies the direction of the copy." Because the function "cudaMemcpyAsync() is asynchronous with respect to the host, [] the call may return before the copy is complete. The copy can optionally be associated to a stream [identified stream]."

__host__ __device__ cudaError_t cudaMemcpyAsync (void* dst , const void* src , size_t count , cudaMemcpyKind kind , cudaStream_t stream = 0)

Copies data between host and device.

Parameters

`dst`
- Destination memory address

`src`
- Source memory address

`count`
- Size in bytes to copy

`kind`
- Type of transfer

`stream`
- Stream identifier

Returns

cudaSuccess, cudaErrorInvalidValue, cudaErrorInvalidMemcpyDirection

Description

Copies `count` bytes from the memory area pointed to by `src` to the memory area pointed to by `dst`, where `kind` specifies the direction of the copy, and must be one of cudaMemcpyHostToHost, cudaMemcpyHostToDevice, cudaMemcpyDeviceToHost, cudaMemcpyDeviceToDevice, or cudaMemcpyDefault. Passing cudaMemcpyDefault is recommended, in which case the type of transfer is inferred from the pointer values. However, cudaMemcpyDefault is only allowed on systems that support unified virtual addressing.

The memory areas may not overlap. Calling cudaMemcpyAsync() with `dst` and `src` pointers that do not match the direction of the copy results in an undefined behavior.

cudaMemcpyAsync() is asynchronous with respect to the host, so the call may return before the copy is complete. The copy can optionally be associated to a stream by passing a non-zero `stream` argument. If `kind` is cudaMemcpyHostToDevice or cudaMemcpyDeviceToHost and the `stream` is non-zero, the copy may overlap with operations in other streams.

The device version of this function only handles device to device copies and cannot be given local or shared pointers.

(*See* https://docs.nvidia.com/cuda/cuda-runtime-api/group__CUDART__MEMORY.html (emphasis added).)

159. In another example, CUDA implements "CUDA-specific memory APIs [that] provide users with guarantees about where the memory resides, which threads can access it, whether it is migratable, and many other features that enable users to extract all the performance the hardware has to offer."

CUDA-specific memory APIs provide users with guarantees about where the memory resides, which threads can access it, whether it is migratable, and many other features that enable users to extract all the performance the hardware has to offer. Applications can hint the system about their memory access patterns, for example, using CUDA and/or NUMA APIs, to enable the users to perform application-specific optimizations. NUMA memory hints enable applications to inform the runtime about their memory access patterns.

(*See* https://resources.nvidia.com/en-us-grace-cpu/nvidia-grace-hopper (emphasis added).)

160. Each claim in the ’461 Patent recites an independent invention. Neither claim 21, described above, nor any other individual claim is representative of all claims in the ’461 Patent.

161. Defendant has been aware of the ’461 Patent since at least the filing of the original Complaint. Defendant has been aware of the technology patented by the ’461 Patent since at least 2007, when the inventors of the Asserted Patents first discussed their patented technologies with Mr. Sanford Russell, then the CTO of Nvidia. At the time, the inventors asked Defendant to collaborate with them on training neural networks using Nvidia’s GPUs. Defendant informed the inventors, through Mr. Russell, that it was not interested in the collaboration. Defendant has also cited an ancestor of the ’461 Patent in its own patent portfolio since at least June 28, 2010 (*See* https://patents.google.com/patent/US8648867B2/en?oq=8648867#citedBy; https://patentimages.storage.googleapis.com/ee/13/e9/61df149c3fddc7/US8922566.pdf; https://patentcenter.uspto.gov/applications/13335850/displayReferences/referenceForms?application= (Nvidia U.S. Appl. No. 13/335,850 August 12, 2014, List of References Cited by Examiner).)

162. Starting in or around 2016, the inventors of the Asserted Patents held multiple discussions with Nvidia to invest in or purchase their AI company, Neurala, Inc., and all its assets, including the ’461 Patent family. These discussions included at least Mr. Alvin Lin, an Nvidia Senior Director of Business Development, and Mr. Jeff Herbst, then an Nvidia Vice President of Business Development and head of Nvidia’s Inception GPU Ventures, in or around September 6, 2016. In or around October 2016, Nvidia, through its representatives, initiated discussions with the inventors to invest in Neurala, Inc. for approximately $10 million.

163. The inventors also discussed their patented technology, including the underlying technology and family to the ’461 Patent, with Defendant’s representatives at Nvidia’s artificial intelligence conference in or around June 2017. On or about June 26, 2017, Defendant received

materials from the inventors, in lieu of a meeting on or about June 29, that identified patents related to the ’461 Patent and described the technology in detail. Defendant had previously stated it was interested in the inventors’ solutions. Defendant also featured the inventors on its website as members of Defendant’s start-up incubator on or about September 25, 2019.

Computer Vision / Video Analytics

Inception Spotlight: AI Startup Neurala Sees 7X Speedup with NGC

Sep 25, 2019 0 Like Discuss (0)

By Nefi Alarcon

* * * * *

To help businesses develop custom computer vision solutions quickly, Neurala, a member of NVIDIA’s start-up incubator Inception, has developed Brain Builder, a cloud platform that provides data scientists and developers that are new to deep learning with the ability to quickly and easily train neural networks.

(*See* https://developer.nvidia.com/blog/inception-spotlight-ai-startup-neurala-sees-7x-speedup-with-ngc/ (September 25, 2019); *see also* https://www.youtube.com/watch?v=-WBtxGLoQNs (“Neurala Accelerating AI Video Annotation with NGC Containers” posted by Defendant’s YouTube account).)

164. Neural AI and/or its predecessors-in-interest have satisfied all statutory obligations required to collect pre-filing damages for the full period allowed by law for infringement of the ’461 Patent.

165. Defendant directly infringes at least claim 21 of the ’461 Patent, either literally or under the doctrine of equivalents, by performing the steps described above. For example, Defendant performs the claimed method in an infringing manner as described above by

implementing the Accused Products as part of its accelerated computing operations and running corresponding software that implements the infringing performance. Defendant also performs the claimed method in an infringing manner when testing the operation of the Accused Products and corresponding systems. As another example, Defendant performs the claimed method when providing or administering services to third parties, customers, and partners using the Accused Products.

166. Defendant's partners, customers, and users of its Accused Products and corresponding systems and services directly infringe at least claim 21 of the '461 Patent, literally or under the doctrine of equivalents, at least by using the Accused Products and corresponding systems and services, as described above.

167. Defendant has actively induced and is actively inducing infringement of at least claim 21 of the '461 Patent with specific intent to induce infringement, and/or willful blindness to the possibility that its acts induce infringement, in violation of 35 U.S.C. § 271(b). For example, Defendant encourages and induces customers to use Nvidia's CUDA platform in a manner that infringes claim 21 of the '461 Patent at least by offering and providing software that performs a method that infringes claim 21 when installed and operated by the customer using the Accused Products, and by engaging in activities relating to selling, marketing, advertising, promotion, installation, support, and distribution of the Accused Products.

168. Defendant encourages, instructs, directs, and/or requires third parties—including its certified partners and/or customers—to perform the claimed method using the software, platform, services, and systems in infringing ways, as described above.

169. Defendant further encourages and induces its customers to infringe claim 21 of the '461 Patent: 1) by making its accelerated computing and data center services available on its

website, providing applications that allow users to access those services, widely advertising those services, and providing technical support and instructions to users (*see* https://www.nvidia.com/en-us/data-center/data-center-gpus/gpu-test-drive/); and 2) through activities relating to marketing, advertising, promotion, installation, support, and distribution of the Accused Products, including its CUDA platform, and services in the United States. (*See* https://www.nvidia.com/en-us/; *see* https://www.nvidia.com/en-us/about-nvidia/partners/; https://www.nvidia.com/en-us/data-center/where-to-buy/; https://www.nvidia.com/en-us/data-center/where-to-buy-tesla/.)

170. For example, Defendant shares instructions, guides, and manuals, which advertise and instruct third parties on how to use its hardware and platform as described above, including at least customers and partners. (*See* https://docs.nvidia.com/cuda/cuda-c-programming-guide/.) Defendant also provides customer service and technical support to purchasers of the Accused Products and corresponding systems and services, which directs and encourages customers to perform certain actions that use the Accused Products in an infringing manner. (*See* https://www.nvidia.com/en-us/support/; https://www.nvidia.com/en-us/support/enterprise/services/.)

171. Defendant and/or Defendant's partners recommend and sell the Accused Products and provide technical support for the installation, implementation, integration, and ongoing operation of the Accused Products for each individual customer. On information and belief, each customer enters into a contractual relationship with Defendant and/or one of Defendant's partners, which obligates each customer to perform certain actions in order to use the Accused Products. (*See* https://www.nvidia.com/en-us/agreements/; https://www.nvidia.com/en-us/agreements/cloud-services/nvidia-cloud-agreement/; https://www.nvidia.com/en-

us/agreements/cloud-services/service-specific-terms-for-nvidia-dgx-cloud/.) Further, in order to receive the benefit of Defendant's and/or its partner's continued technical support and their specialized knowledge and guidance of the operability of the Accused Products, each customer must continue to use the Accused Products in a way that infringes the '461 Patent. (*See* https://www.nvidia.com/en-us/support/.)

172. Further, as the entity that provides installation, implementation, and integration of the Accused Products in addition to ensuring the Accused Product remains operational for each customer through ongoing technical support, on information and belief, Defendant and/or Defendant's partners affirmatively aid and abet each customer's use of the Accused Products in a manner that performs the claimed method of, and infringes, the '461 Patent.

173. Defendant also contributes to the infringement of its partners, customers, and users of the Accused Products by providing within the United States or importing into the United States the Accused Products, which are for use in practicing, and under normal operation practice, the methods, systems, and devices claimed in the Asserted Patents, constituting a material part of the inventions claimed, and not a staple article or commodity of commerce suitable for substantial non-infringing uses. Indeed, as shown above, the Accused Products and the example functionality have no substantial non-infringing uses but are specifically designed to practice the '461 Patent.

174. On information and belief, the infringing actions of each partner, customer, and/or user of the Accused Products are attributable to Defendant. For example, on information and belief, Defendant directs and controls the activities or actions of its partners or others in connection with the Accused Products by contractual agreement or otherwise requiring partners or others to provide information and instructions to customers who acquire the Accused Products which, when followed, results in infringement. Defendant further directs and controls the operation of devices

executing the Accused Products by programming the software which, when executed by a customer or user, performs the claimed method of at least claim 21 of the ’461 Patent.

175. Plaintiff has suffered and continues to suffer damages as a result of Defendant’s infringement of the ’461 Patent. Defendant is therefore liable to Plaintiff under 35 U.S.C. § 284 for damages in an amount that adequately compensates Plaintiff for Defendant’s infringement, but no less than a reasonable royalty.

176. Defendant’s infringement of the ’461 Patent is knowing and willful. Defendant acquired actual knowledge of the family of the ’461 Patent since at least 2017 and has acquired additional knowledge of the ’461 Patent since at least the filing of this lawsuit.

177. On information and belief, despite Defendant’s knowledge of the Asserted Patents and Plaintiff’s patented technology, Defendant made the deliberate decision to sell products and services that it knew infringe these patents. Defendant’s continued infringement of the ’461 Patent with knowledge of the ’461 Patent constitutes willful infringement.

PRAYER FOR RELIEF

WHEREFORE, Plaintiff respectfully requests the following relief:

a) That this Court adjudge and decree that Defendant has been, and is currently, infringing each of the Asserted Patents;

b) That this Court award Plaintiff damages to compensate for Defendant's past and future infringement of the Asserted Patents, through the life of the Asserted Patents;

c) That this Court award Plaintiff pre- and post-judgment interest on such;

d) That this Court order an accounting of damages incurred by Plaintiff from six years prior to the date this lawsuit was filed through entry of a final, non-appealable judgment;

e) That this Court determine that this patent infringement case is exceptional and award Plaintiff its costs and attorneys' fees incurred in this action;

f) That this Court award increased damages under 35 U.S.C. § 284; and

g) That this Court award such other relief as the Court deems just and proper.

DEMAND FOR JURY TRIAL

Plaintiff respectfully requests a trial by jury on all issues triable thereby.

DATED: December 12, 2024

By:*/s/ Mark D. Siegmund*

Mark D. Siegmund
Texas Bar No. 24117055
CHERRY JOHNSON SIEGMUND JAMES PLLC
Bridgeview Center
7901 Fish Pond Road, 2nd Floor
Waco, Texas 76710
Telephone: (254) 732-2242
Facsimile: (866) 627-3509
msiegmund@cjsjlaw.com

Christopher C. Campbell
KING & SPALDING LLP
1700 Pennsylvania Avenue, NW
Suite 900
Washington, DC 20006
Telephone: (202) 626-5578
Facsimile: (202) 626-3737
ccampbell@kslaw.com

Britton F. Davis
Brian Eutermoser (*pro hac vice to be filed)*
KING & SPALDING LLP
1401 Lawrence Street
Suite 1900
Denver, CO 80202
Telephone: (720) 535-2300
Facsimile: (720) 535-2400
bfdavis@kslaw.com
beutermoser@kslaw.com

Attorneys for Plaintiff Neural AI, LLC

CERTIFICATE OF SERVICE

The undersigned does hereby certify that a true and correct copy of the foregoing document was served on all counsel of record via the Court's electronic filing system on this 12th day of December 2024.

By:*/s/ Mark D. Siegmund*
Mark D. Siegmund