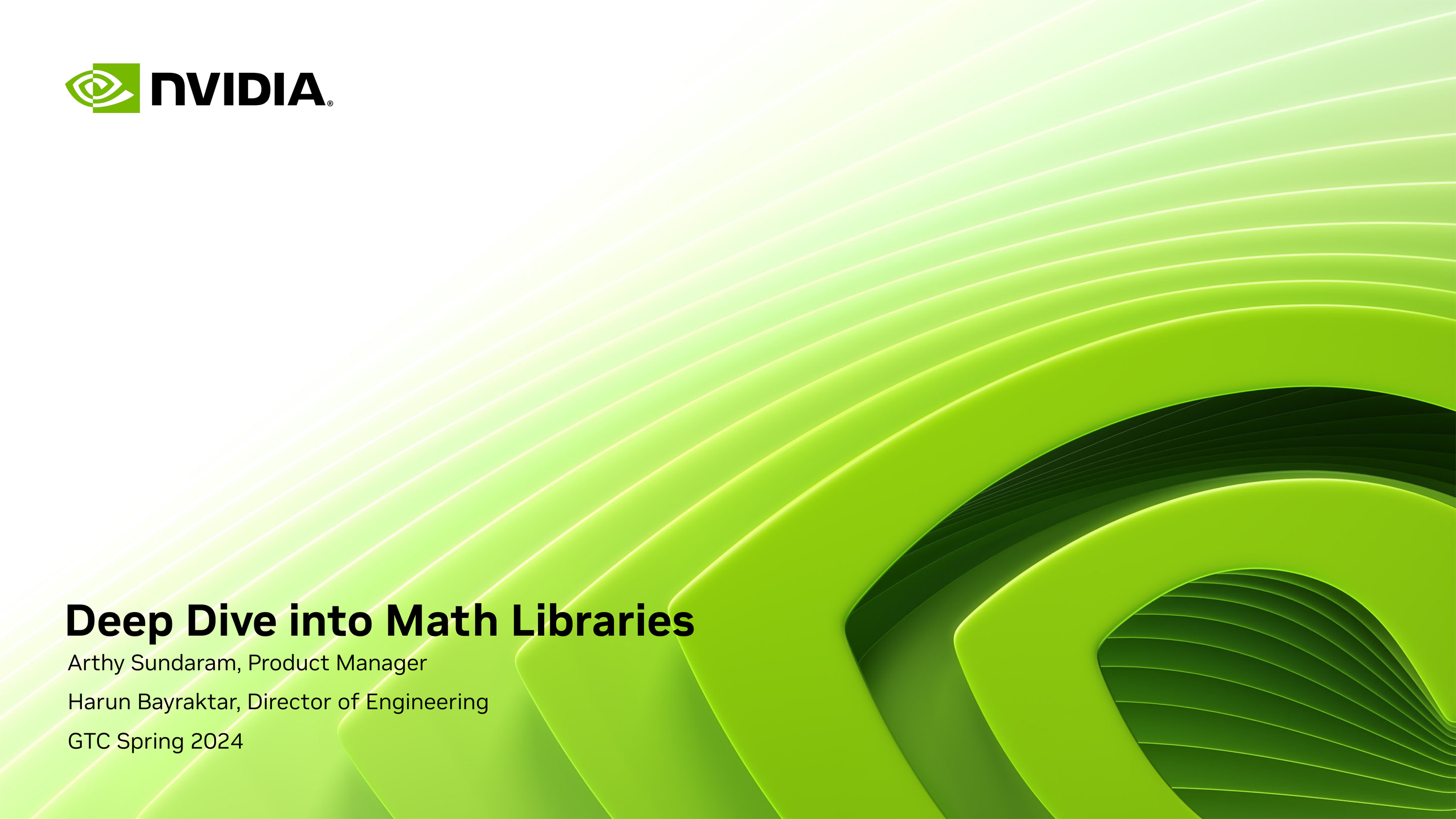


Deep Dive into Math Libraries

Arthy Sundaram, Product Manager

Harun Bayraktar, Director of Engineering

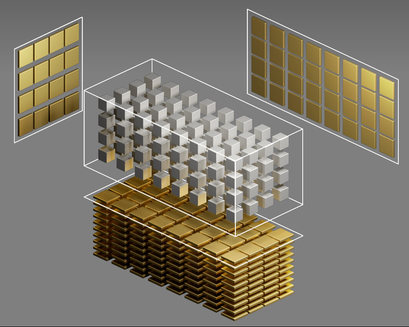
GTC Spring 2024

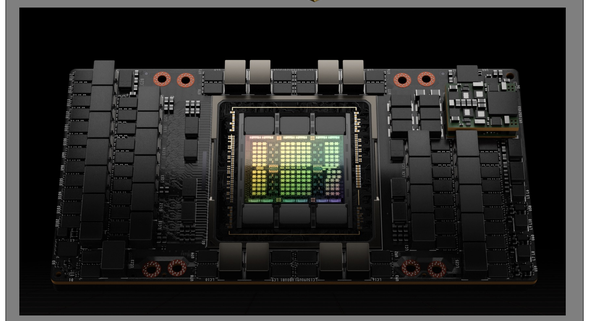
Introduction

Math Libraries are foundational to enabling acceleration that meets your needs

Seamless Acceleration

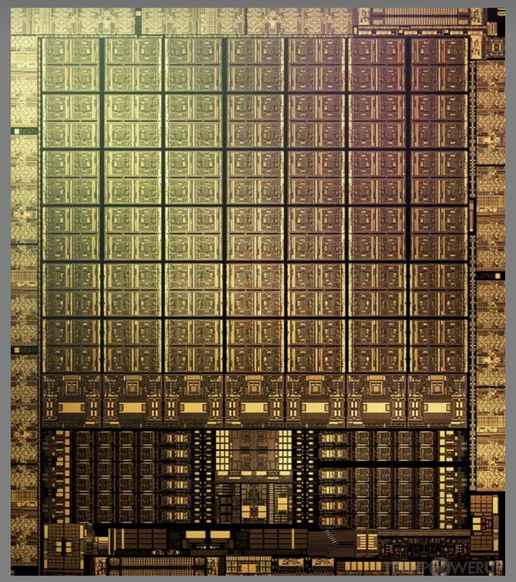
Tensor Cores, TBC, …





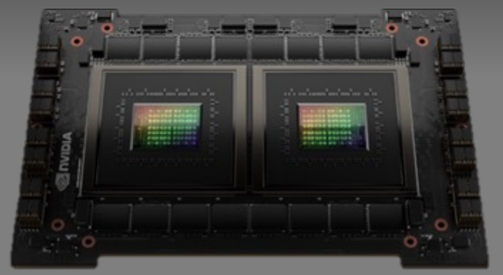
Composability

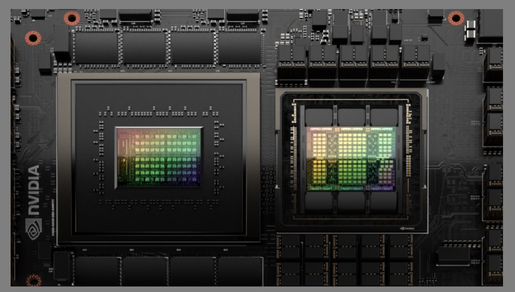
Device Extension Libraries



NVPL on Arm

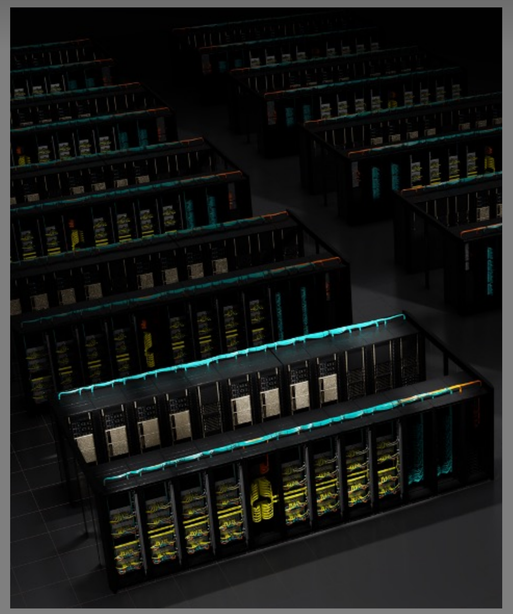
High Performance CPU Libraries





Multi-GPU & Multi-Node

Scaling Up to Supercomputers



Agenda

- Fast Fourier Transform Libraries
- Dense and Sparse Linear Algebra Libraries
- Advanced Solvers
- Image and Data Compression Libraries
- CPU Libraries
- Introducing Python APIs
- Closing Remarks

Fast Fourier Transform Libraries

cuFFT

API extensions

Callable from CPU code

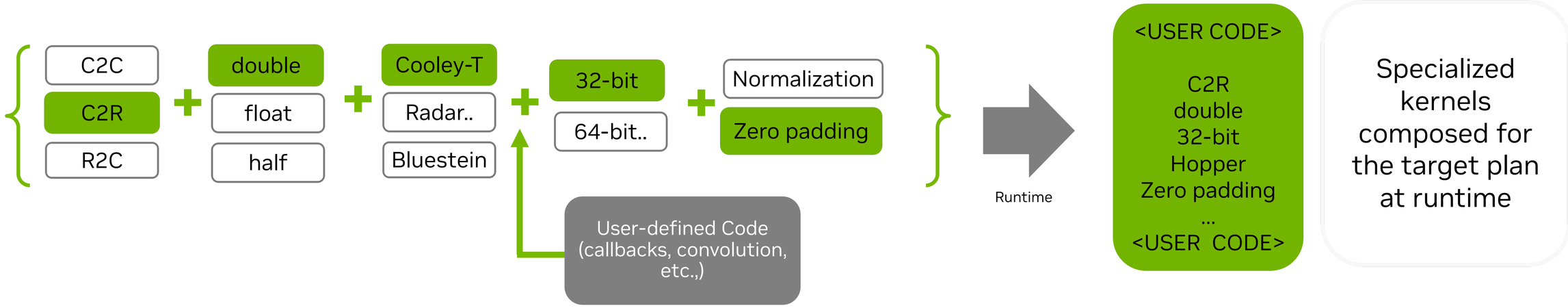
API Extensions	Description	Availability
cuFFT Host APIs	GPU accelerated FFT for **single GPU**	CUDA Toolkit/HPC SDK
cuFFTXt Host APIs	**Multi-GPU single-node** FFT	CUDA Toolkit/HPC SDK
cuFFTMp Host APIs	**Multi-GPU multi-node** FFT	HPC SDK
cuFFTDx Device APIs	**In-kernel** FFTs	Standalone download

cuFFT: Just-In-Time, Link-Time Optimization

From combinatorial explosion -> Feature rich

- Shipping dedicated (pre)-compiled kernels for every possible configuration, GPU arch is not scalable

- JIT LTO kernels are shipped as pre-compiled kernel “pieces”. Exposed possibilities for user-code integration.



cuFFT - JIT LTO

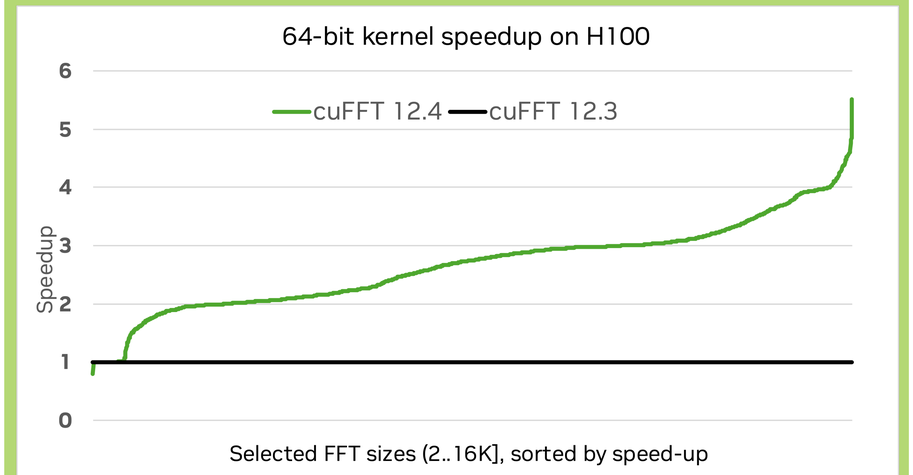
Runtime finalization of kernels and inlined-callbacks with Windows support and more!

Coming to cuFFT

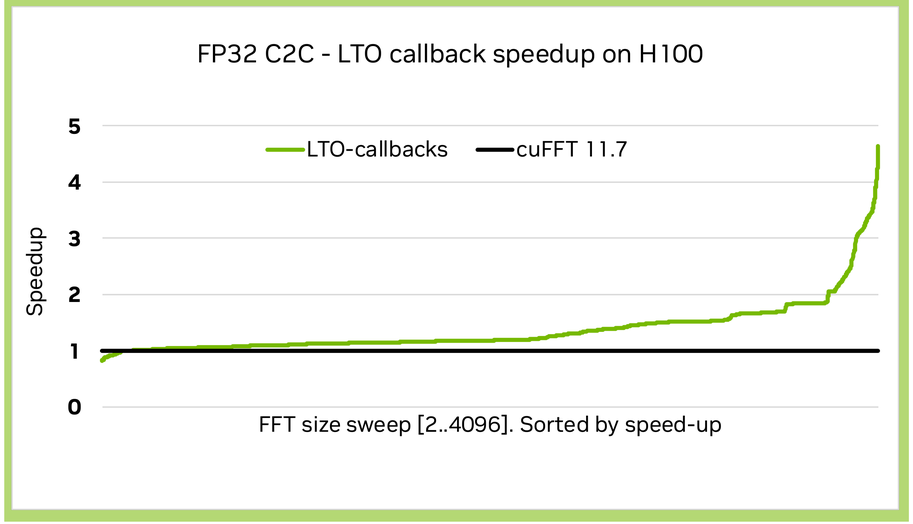
In Public Early Access now

CUDA 12.4

More Performant 64-bit kernels



More performant LTO Callbacks + Windows Support



R2C, C2R and other kernels

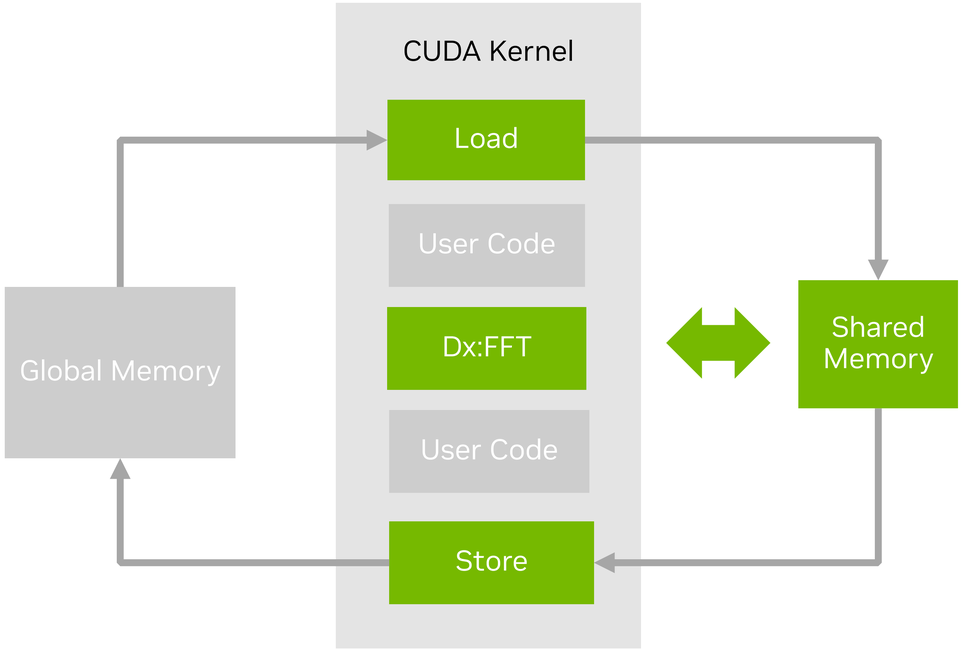
Normalization, Zero padding, Mixed-precision support

cuFFTDx

C++ library with device side APIs for fusing FFT into user kernels

cuFFTDx is a C++ header only library for inlining FFT in user kernels

- Ideal when data fits in shared memory / registers
- Fuse FFTs with custom user operations or other Dx APIs in a single kernel!

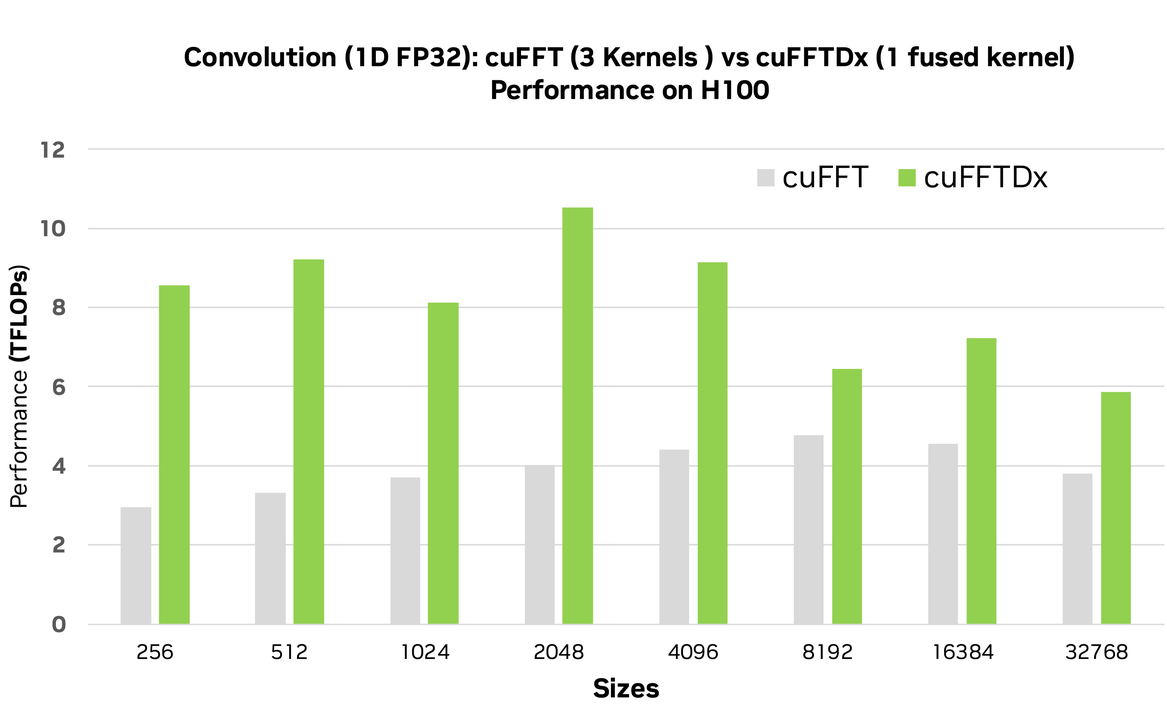


User Code:

Custom Ops for post or pre-processing for FFTs

Write your own convolution kernel using cuFFTDx for increased performance!

See Hyena convolution operator for LLM using cuFFTDx.



Dense and Sparse Linear Algebra Libraries

cuBLAS

cuBLAS

API extensions

Callable from CPU code

API Extensions	Description	Availability
cuBLAS Host APIs	**Standard BLAS APIs** for **GPUs** with additional GEMM extensions	CUDA Toolkit/HPC SDK
cuBLASLt Host APIs	**Advanced APIs for GEMM** Programmable **AI trained heuristics** for **kernel selection** and performance Variety of **mixed precision** and **epilogues**	CUDA Toolkit/HPC SDK
cuBLASXt Host APIs	cuBLAS calls for **single node multi-GPU**	CUDA Toolkit/HPC SDK

New API Extensions: cuBLASDx and cuBLASMp

Now available for download!

cuBLASMp Host APIs (Preview)	Standard BLAS APIs for **multi-GPU multi-node (MGMN).** Callable from CPU code.	Standalone download and HPC SDK
cuBLASDx Device side APIs (Preview)	Concise and easier to express and use **in-kernel GEMMs**	Standalone download

cuBLASMp Multi-Node Multi-GPU Host API (Preview)

cuBLASMp (Preview) is a high performance, **multi-process**, GPU-accelerated library for **distributed** basic dense linear algebra. cuBLASMp is available for standalone download and as part of the HPC SDK.

Download cuBLASMp

cuBLASDx Device API (Preview)

cuBLASDx (Preview) is a **device side** API extension to cuBLAS for performing BLAS calculations inside your CUDA kernel. **Fusing** numerical operations decreases the latency and improves the performance of your application.

Download cuBLASDx

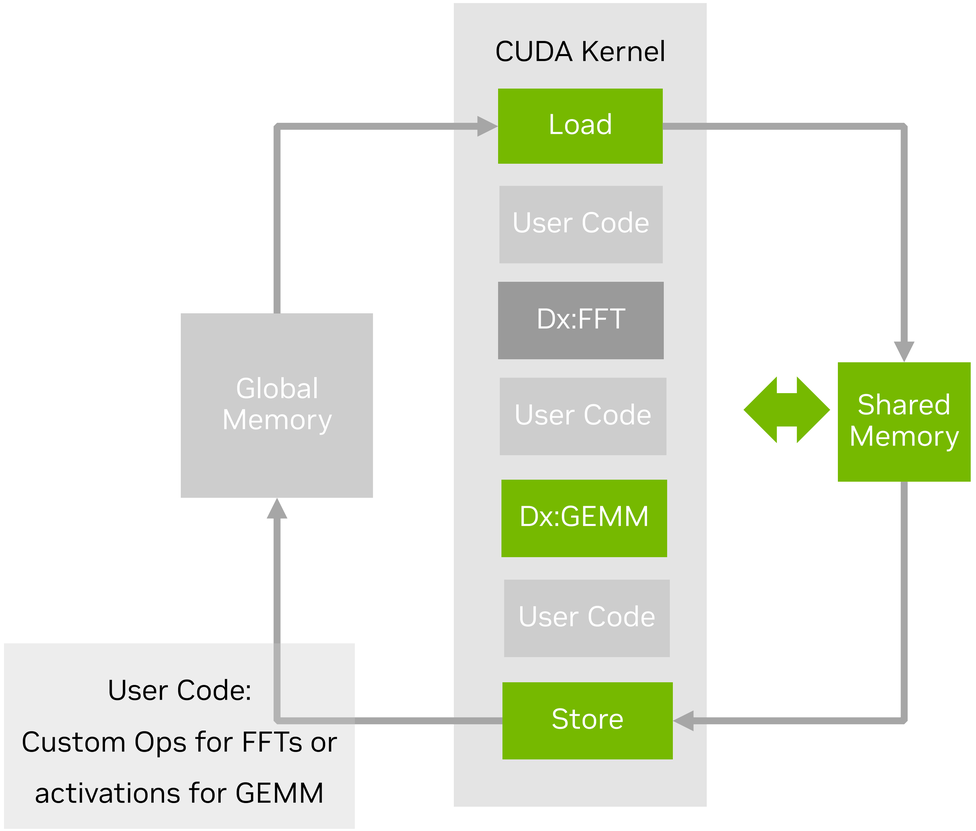
More features, capabilities, performance improvements coming!

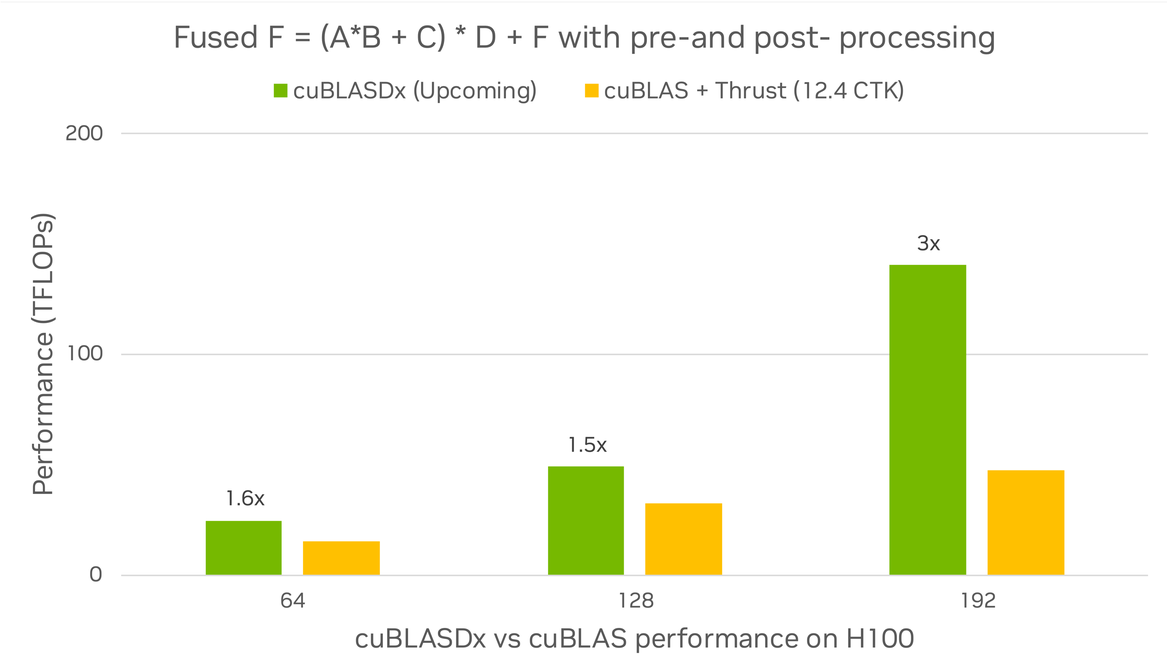
cuBLASDx

Device side APIs for in-kernel GEMMs

- Built on top of CuTE - Concise, simpler APIs for GEMMs inside CUDA kernel
- Inline alongside user-code and cuFFTDx calls
- All matrix sizes that fit into the shared memory. Real, Complex. FP16, FP32, FP64.
- **Future**: Mixed precision support and narrow precision formats

Device Side APIs for in-kernel calls



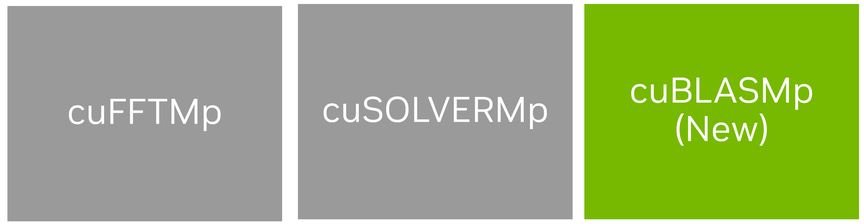


Higher GPU utilization (TFLOPs) with cuBLASDx, when fusing more than one GEMM and custom ops

cuBLASMp

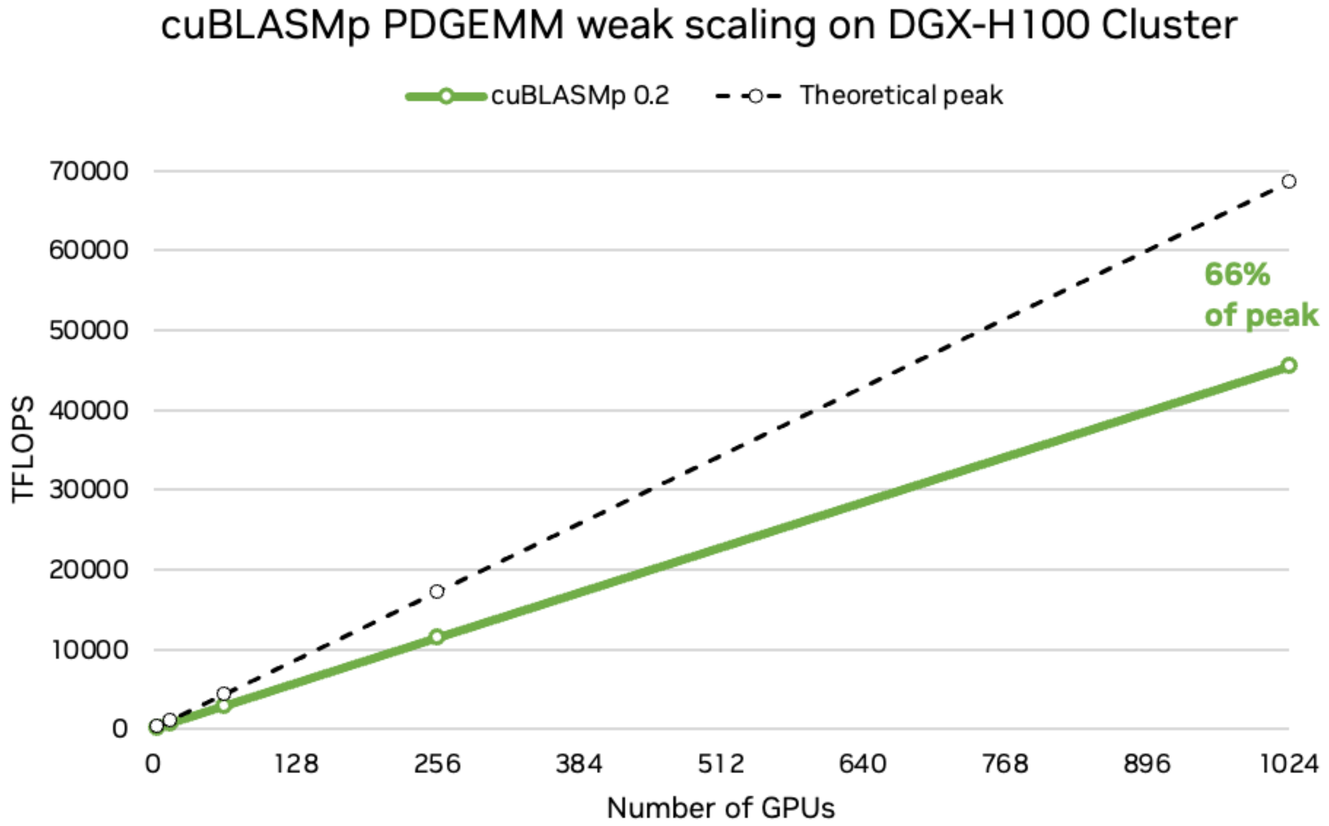
Distributed Dense Linear Algebra

Multi Node Host APIs for scale



cuBLASMp Features

- GEMM, SYRK, TRSM
- One process per GPU
- FP16, FP32, FP64
- **Future**: Mixed precision support and narrow precision formats



M=N=K=55k per GPU

cuBLAS

Family of API extensions

Callable from CPU code

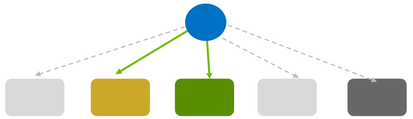
API Extensions	Description	Availability
cuBLAS Host APIs	**Standard BLAS APIs for GPUs** with additional GEMM extensions	CUDA Toolkit/HPC SDK
cuBLASLt Host APIs	**Advanced APIs for GEMM** Programmable **AI trained heuristics** for **kernel selection** and performance Variety of **mixed precision** and **epilogues**	CUDA Toolkit/HPC SDK
cuBLASXt Host APIs	cuBLAS calls for **multi-GPU single-node** from CPU code	CUDA Toolkit/HPC SDK
cuBLASMp Host APIs	**Multi-GPU multi-node .** Callable from CPU code	Standalone download and HPC SDK
cuBLASDx Device side APIs	Concise and easier to express and use **in-kernel GEMMs**	Standalone download

cuBLAS - Grouped GEMM

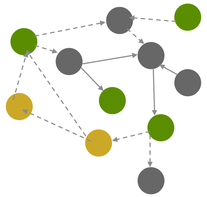
Introducing Grouped GEMM (Experimental) starting with CUDA 12.4

Single, simplified, performance APIs to group batch of matrices of variable sizes, leading dimensions

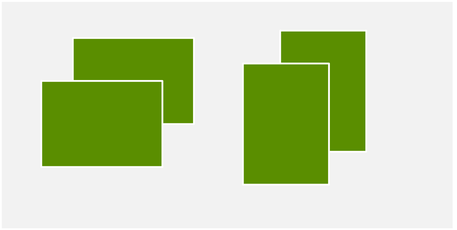
Mixture of Experts:
Activate subset of smaller models

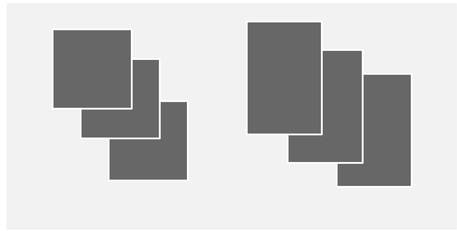


Heterogenous GNN:
Learning hierarchical relationships



Numerous dissimilar GEMMs
from different
experts or different subgraphs

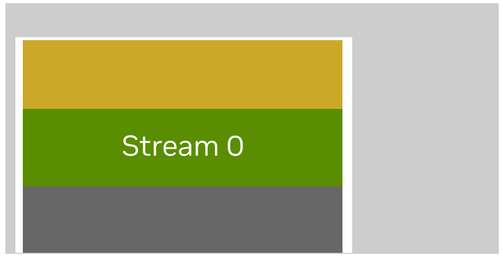




cuBLAS – Grouped GEMM vs Iterative Batched GEMMs

Single, Simplified, Performance APIs to group batches of matrices of variable sizes, leading dimensions

Grouped Batched GEMMs – concurrent kernels



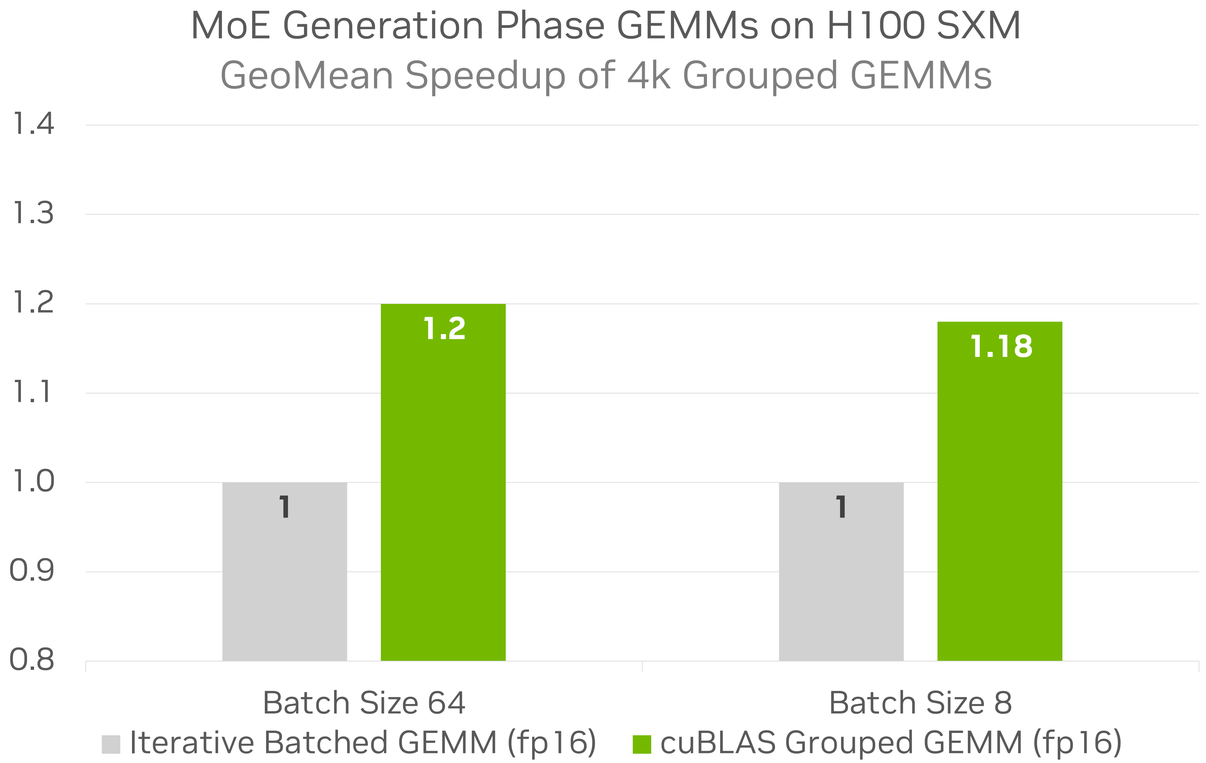
VS

Iterative calls to Batched GEMMs – sequential kernels

Single kernel submission: No launch latency, Increased GPU utilization

With FP32, TF32, and FP64

Upcoming: FP16 and BF16 support



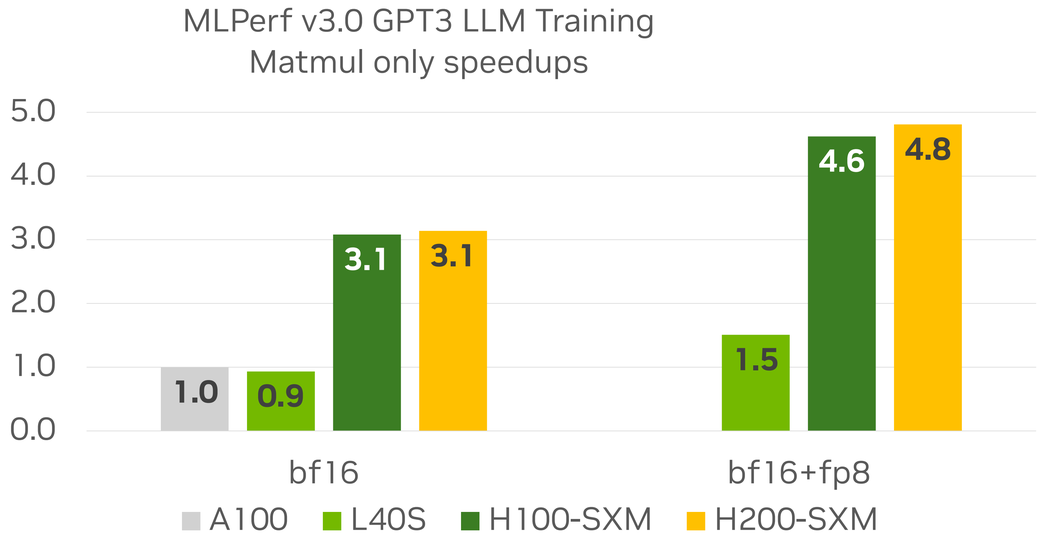
cuBLASLt

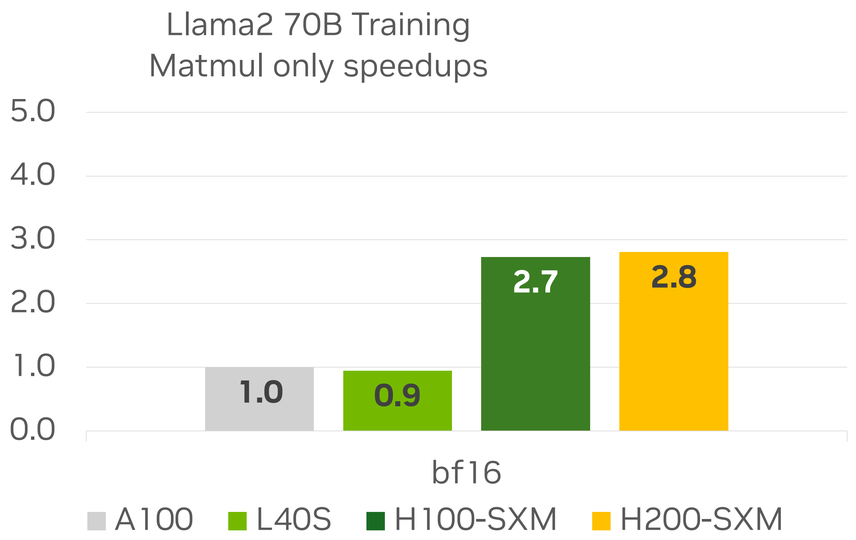
Functional Coverage and Advancing Performance

cuBLASLt Host APIs	**Advanced APIs for GEMMs** Variety of mixed precision and epilogues, kernel selection and performance	CUDA Toolkit/HPC SDK

Added support for larger m, n and batch sizes

Improved matmul performance for the latest Large Language Models





cuTENSOR

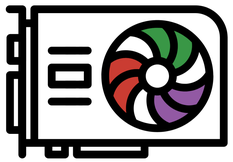
cuTENSOR – GPU accelerated tensor linear algebra library

New APIs, features and performance in 2.0

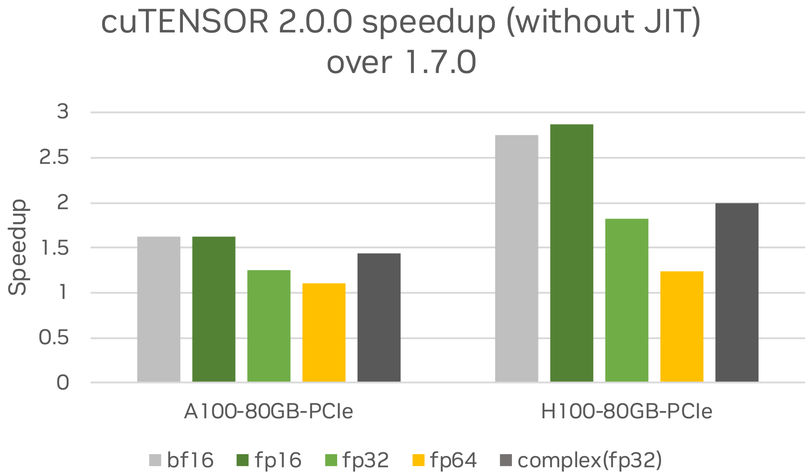
- API Uniformity: Plan based Multi-stage APIs extended for all ops
- Dynamically allocated descriptors: Arbitrarily dimensional tensor descriptors
- “Opt-in” Just-in-Time optimized kernel generation for tensor contractions
- Available for download HPC SDK , Standalone download

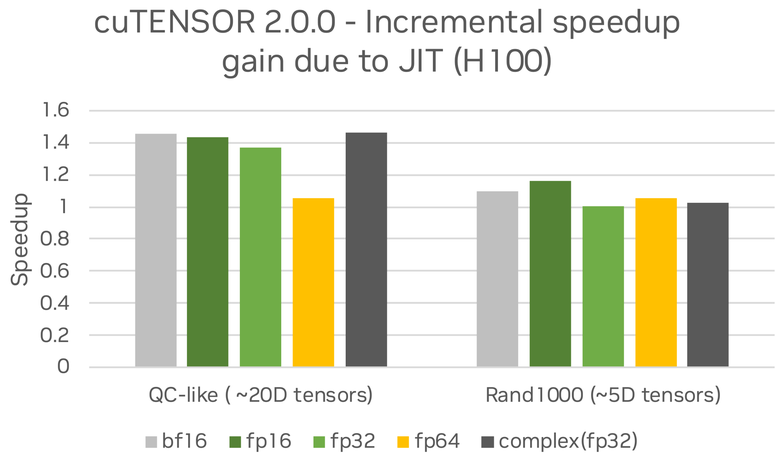
cuQuantum

CuPy



CUDA Julia





Additional Resources

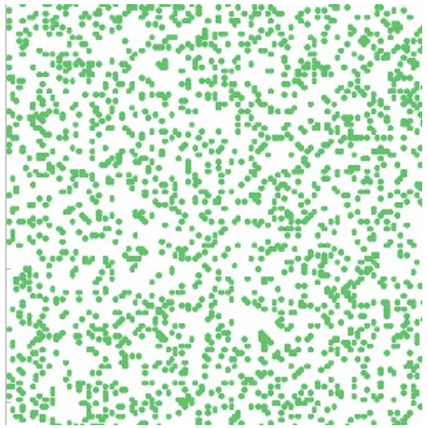
cuTENSOR 2.0: A Comprehensive Guide for Accelerating Tensor Computations
cuTENSOR 2.0: Applications and Performance
cuTENSOR 2.0: Transition to cuTENSOR 2.x guide

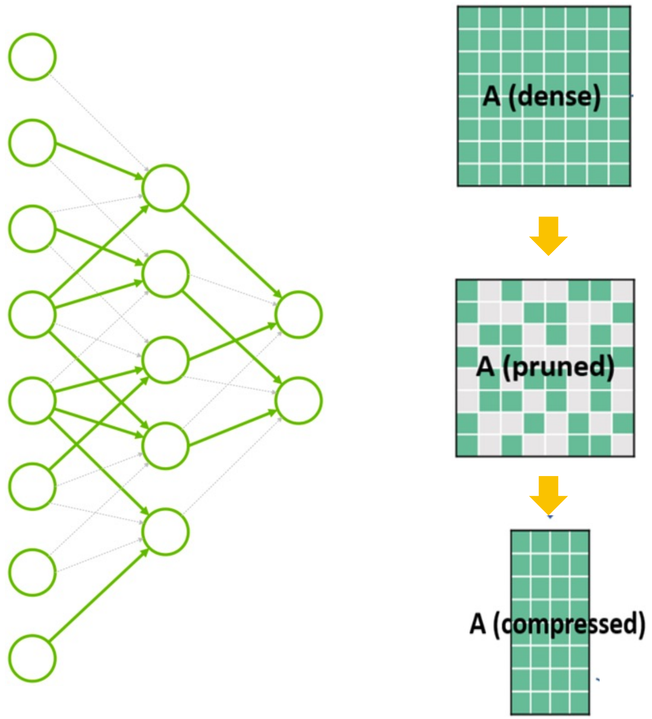
cuSPARSE

Sparsity in Action

Structured and Unstructured

cuSPARSE Host APIs	cuSPARSELt Host APIs
Unstructured Sparsity + Single API calls for standard sparse BLAS computations	**Fine-grained 2:4 Structured** Sparsity + Stateless, multi-stage APIs to access sparse tensor cores



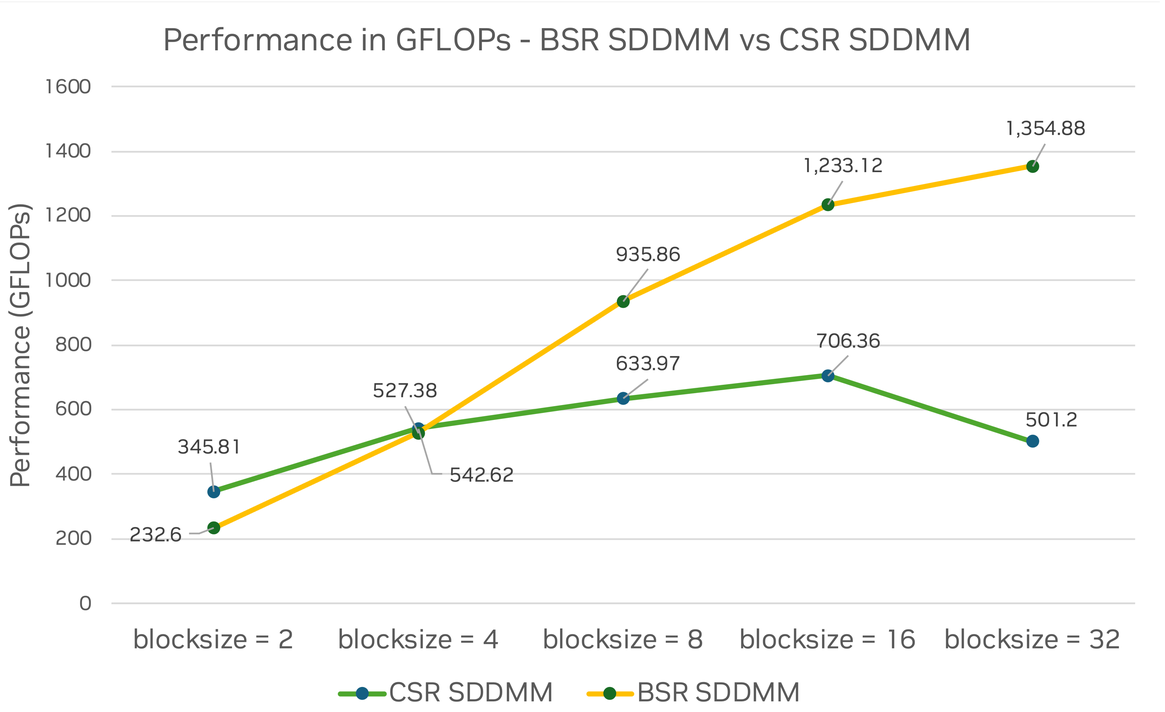


cuSPARSE

New sparsity format: Blocked Compressed Sparse Row Format

BSR/BCSR support for SDDMM
Suitable for matrices with dense block substructure

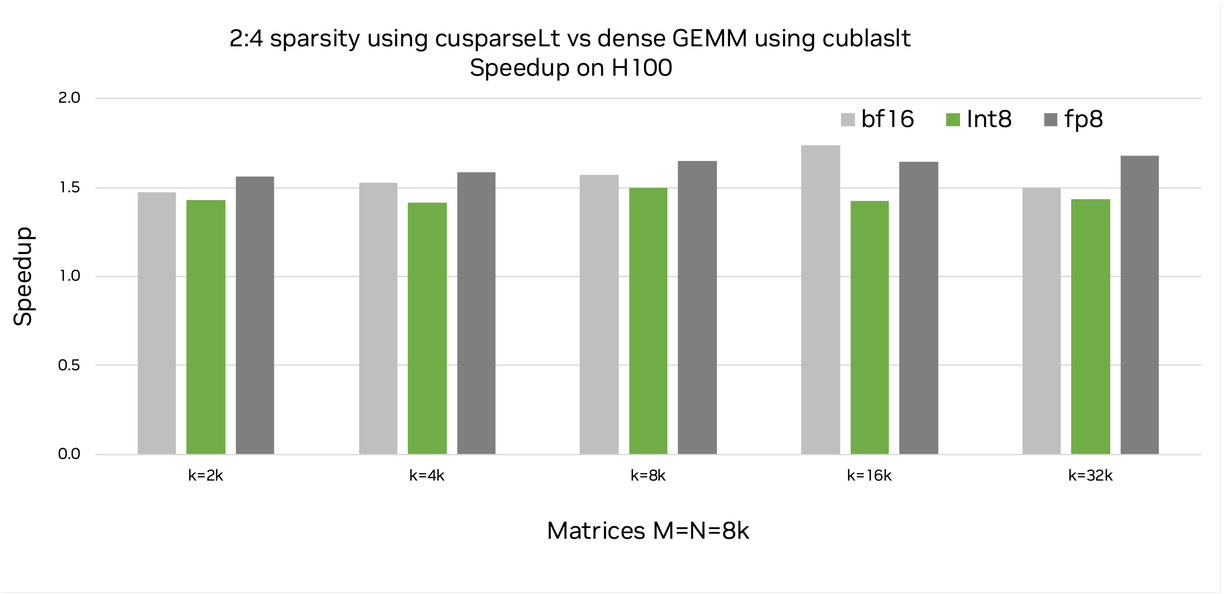
"BCSR Format can enhance the performance of sparse training and inference in PaddlePaddle" - Baidu



cuSPARSELt 0.6.0

Adds **Hopper** Sparse Tensor Cores support for 2:4 Structured Sparsity in matrices

- Hopper support with R535 TRD7 and R550 TRD1 or later drivers
- FP8 and other narrow formats with mixed precision support
  - FP8 (e4m3 and e5m2)
  - INT8, FP16, BF16, TF32 also supported
- **Future**: More matrix sizes and precision types

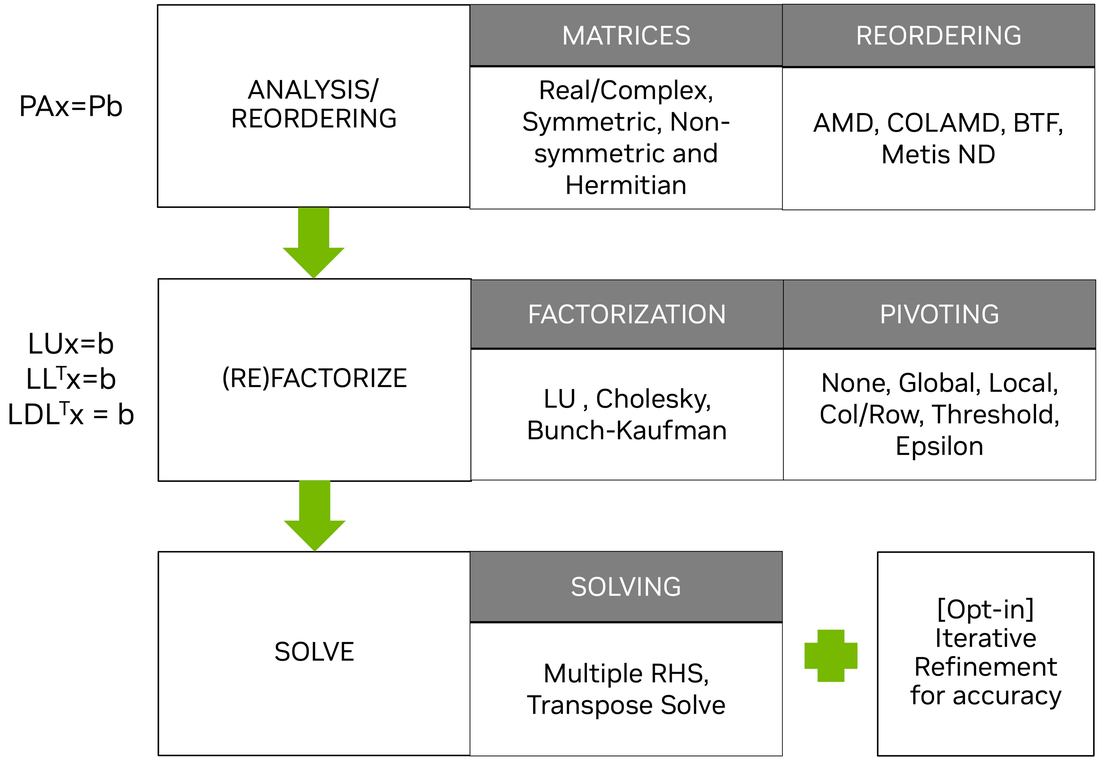


cuSPARSELt + **Ampere (0.5.x)** is NOW available via PyTorch 2.1+!

Advanced Solvers

cuDSS: Direct Sparse Solvers for NVIDIA GPUs

Version 0.2.0 of cuDSS is now available on devzone !



Windows x86, Linux x86 and Arm (Grace CPU) supported

Multi-Node Multi-GPU support in the roadmap

Honeywell uses cuDSS to accelerate UniSim Design Process Simulation

Joint talk - Anton Anders (NVIDIA), Jeffrey Renfro (Honeywell)

[S62515] GPU-Accelerating Process Simulation Performance using cuDSS

March 20, 2024 | **Today** | **10-10:50 am** PT

Image and data compression libraries

nvImageCodec

GPU-accelerated image processing

- Unified API for encoding/decoding images (C and Python)
- Available also as open source (Apache license)
- Support for batch processing of heterogenous images
- Support for 3rd party codecs, codec prioritization & fallback
- Support for most popular image formats: jpeg, jpeg2000, htj2k, tiff, bmp, png, pnm, webp
- Supported platforms
  - x86_64 (Linux, WSL)
  - arm64_sbsa (Linux)
- V0.2.0 is available for download:
  - Developer Zone (binary),
  - PyPi,
  - GitHub (open source)

nvImageCodec Support Matrix

Decoding

High

fallback priority

Low

Jpeg	Jpeg Lossless	Jpeg2k	TIFF	PNM	BMP	PNM	WebP
nvJPEG(HW engine)	nvJPEG(CUDA)	nvJPEG2000 (CUDA)	nvTIFF(CUDA)	nvPNG(CUDA)	opencv	opencv	opencv
nvJPEG(CUDA)	opencv	opencv	libtiff	opencv	nvBMP		
libturbo-jpeg			opencv				
opencv							

Encoding

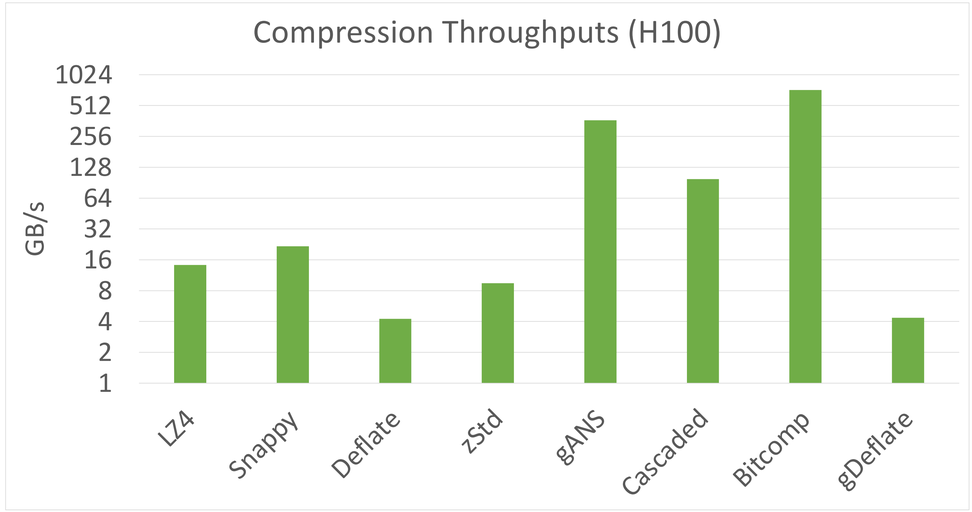
Jpeg	Jpeg Lossless	Jpeg2k	TIFF	PNM	BMP	PNM	WebP
nvJPEG(CUDA)		nvJPEG2000(CUDA)	nvTIFF(CUDA)	nvPNG(CUDA)	nvBMP	nvPNM	

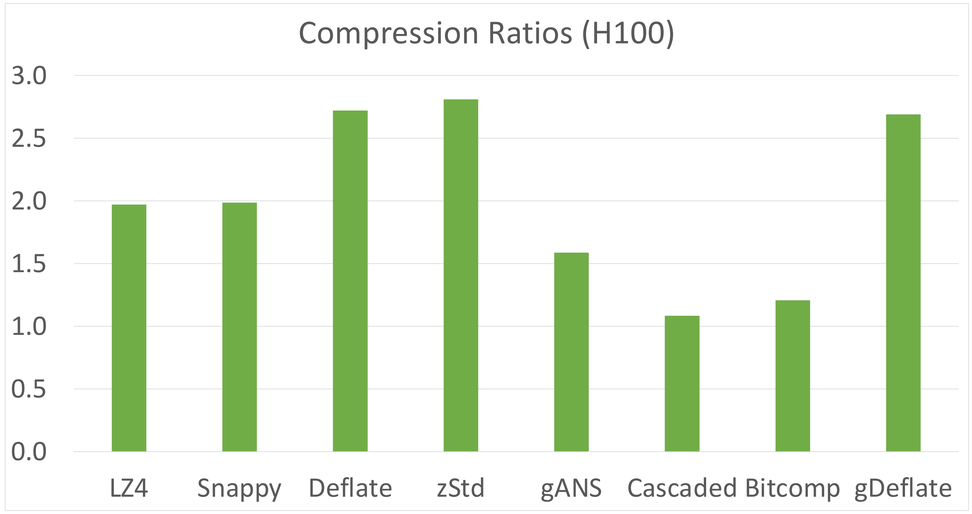
CPU GPU GPU (future)

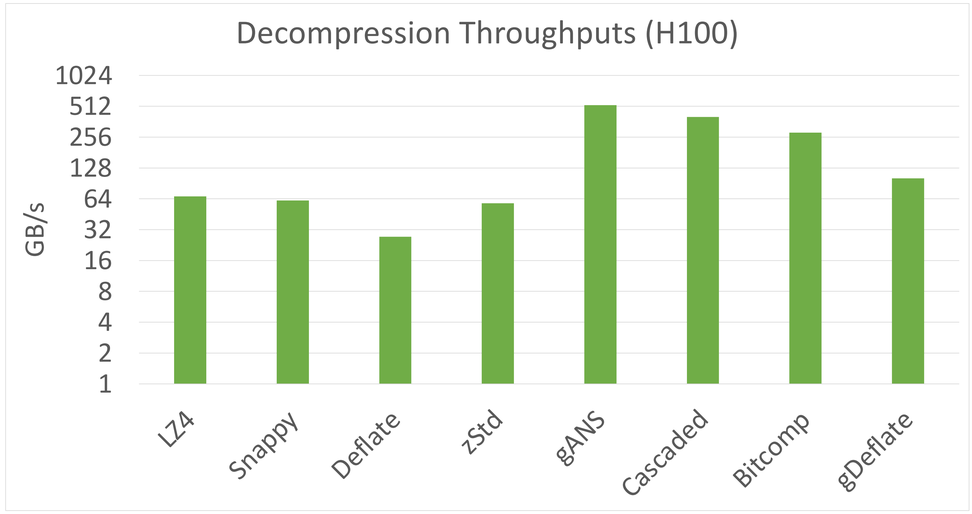
nvCOMP

GPU-accelerated data compression

- Compression algorithms
  - zSTD, deflate, gDeflate, LZ4, snappy, bitcomp, cascaded, gANS
- Integrated into (among others):
  - Spark RAPIDS (cuDF) [up to 10x E2E query speedup]
  - Parabricks (healthcare) [30% speedups for fq2bam]
  - nvTIFF [deflate]
- Available for download from Developer Zone





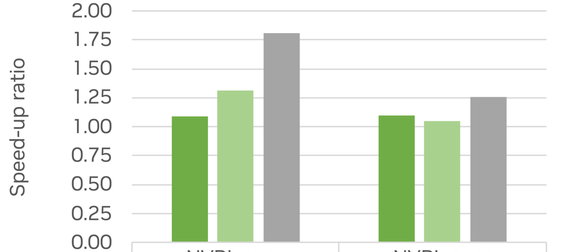


NVIDIA Performance Libraries (NVPL) for Grace

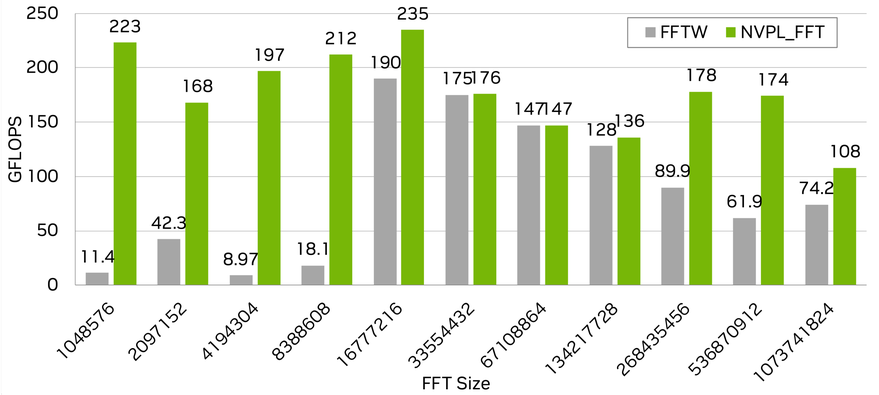
Announcing NVPL-24.03 Beta

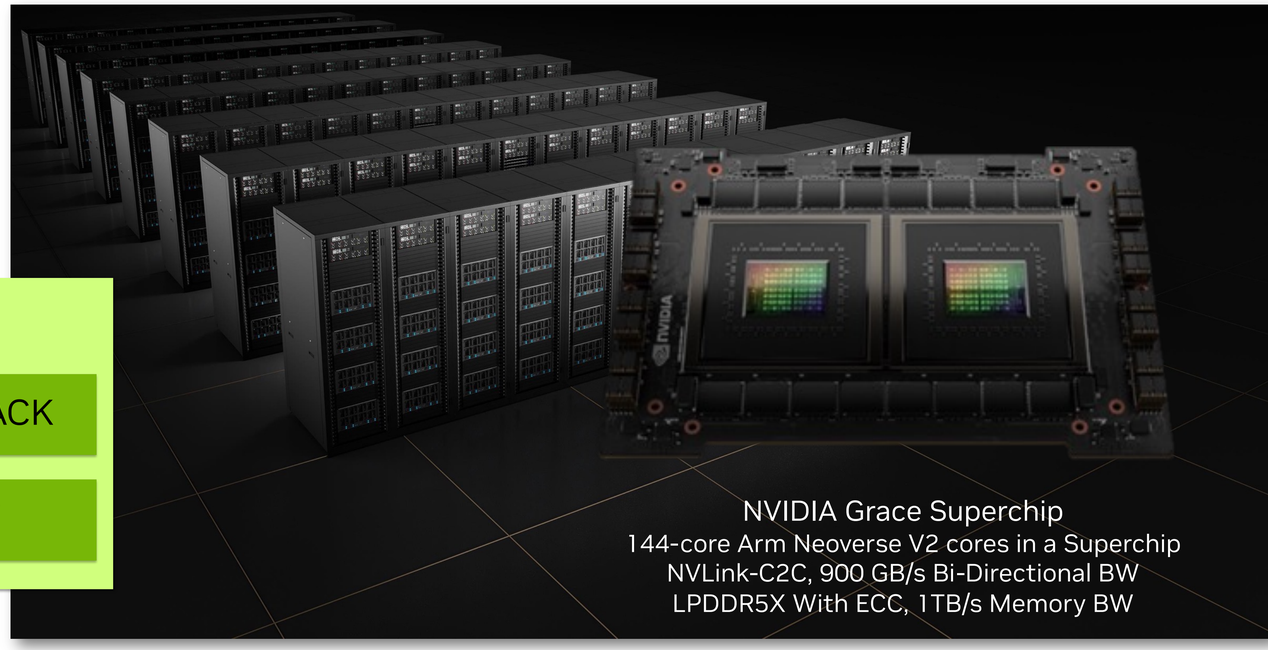
CPU Math Libraries to support HPC Applications on NVIDIA Grace

- **NVPL-24.03** is available now
  - Downloads: developer.nvidia.com/nvpl-downloads
  - Documentation: docs.nvidia.com/nvpl
  - Samples: github.com/NVIDIA/NVPLSamples
- Introducing: **NVPL Tensor**
  - API Based on cuTENSOR 2.0
  - Support for up to 64-dimensional tensors
- Other improvements since 23.11 release:
  - NVPL BLAS
    - Small DGEMM optimizations
    - +5% E2E improvements for selected HPC apps
  - NVPL FFT
    - C2C performance optimizations
    - Multi-threaded: huge sizes
  - NVPL Sparse
    - SpSV:1.8x speedup and 1.8x memory footprint reduction



	NVPL over NVPL	NVPL over OpenBLAS
NWChem	1.09	1.10
CP2K	1.31	1.05
Arpege	1.81	1.26





NVPL

BLAS	LAPACK	PBLAS	SCALAPACK
new TENSOR	SPARSE	RNG	FFTW

NVIDIA

Introducing nvmath-python

Harun Bayraktar, Leo Fang, Satya Varadhan, Leopold Cambier

Python Ecosystem for Math Library Functionality

& what is missing today?

- Over the last decade Python has become the most popular language, primarily driven by data science and AI but also making strides in scientific computing

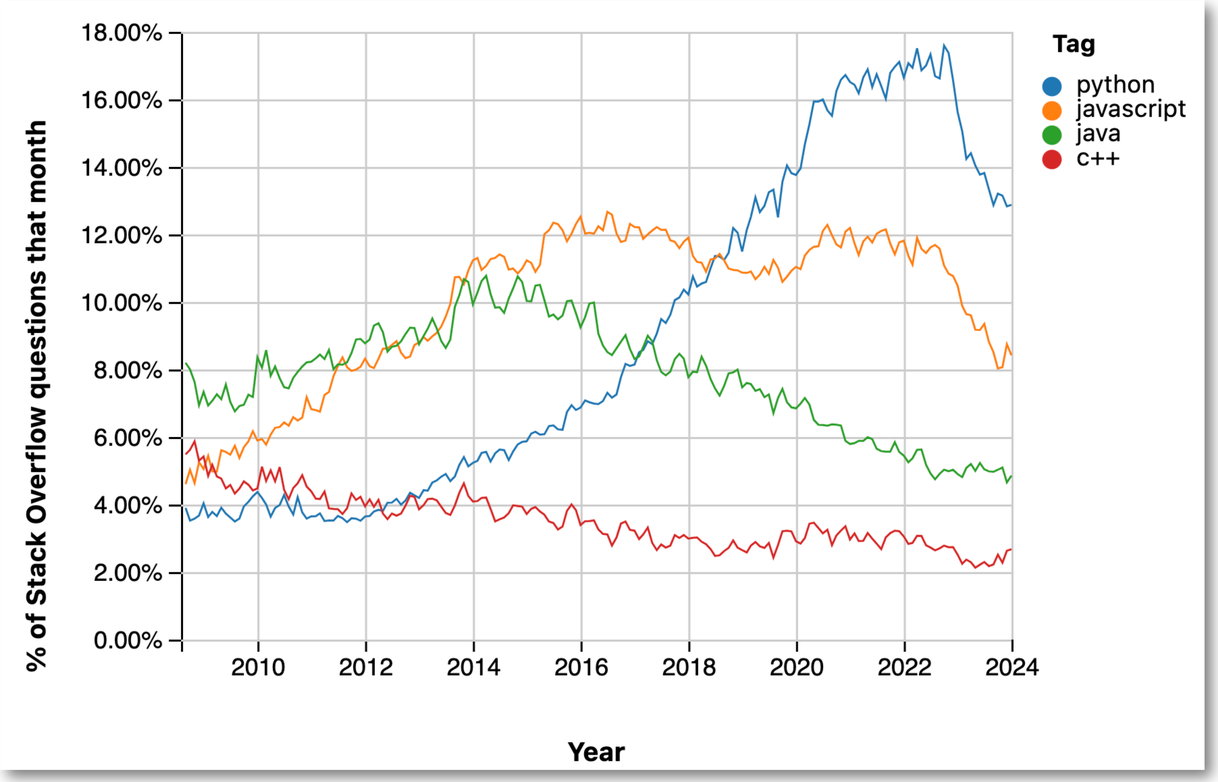
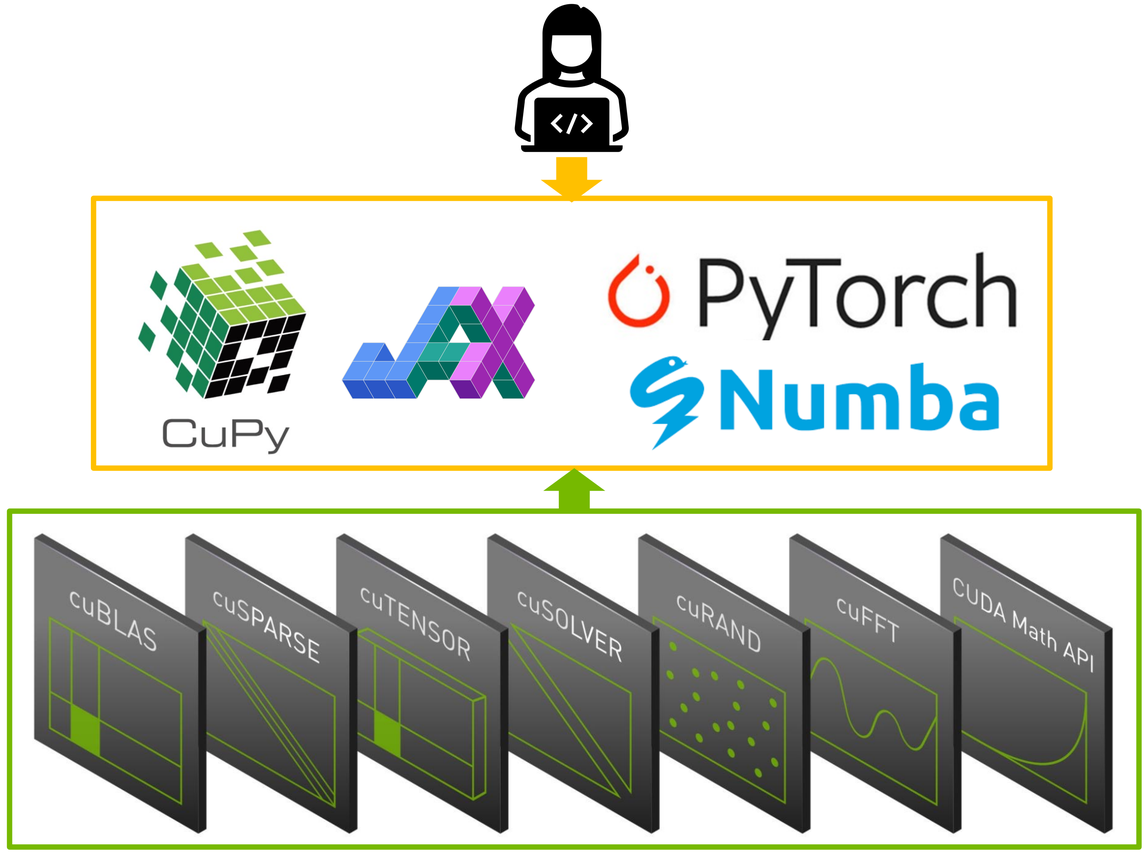


Chart from Stack overflow trends

- GPU accelerated math libraries functionality is available through popular frameworks and libraries, but functionality coverage is fragmented, and performance is not always optimal



Pathfinder: cuQuantum Python

Very successful offering of Pythonic UX for tensor network contractions

C++ sample:
279 SLOC*
(61 comment lines in addition)

low-level Python binding sample:
148 SLOC
(60 comment lines in addition)

$$r_{mxny} = \sum_{k}\sum_{u} a_{mhkn} b_{ukh} c_{xuy}$$

```
from cuquantum import contract
from numpy.random import rand

a = rand(96,64,64,96)
b = rand(96,64,64)
c = rand(64,96,64)

r = contract("mhkn,ukh,xuy->mxny", a, b, c)
```

pythonic APIs sample

*Line count obtained using the cloc utility

Introducing nvmath-python

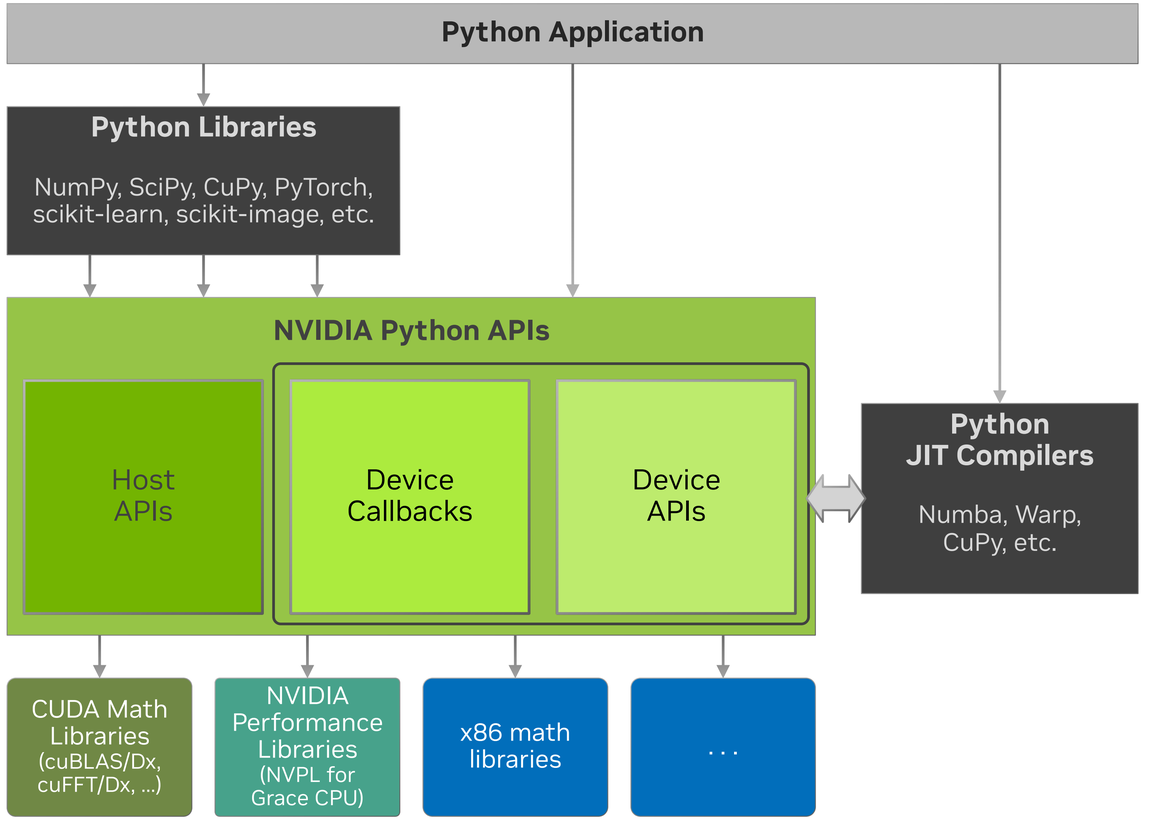
Easy Pythonic access to CUDA Math libraries functionality in a performant manner

Minimal lead-time
to access new CUDA library features from Python

Inter-operability
with core Python numeric packages

Pythonic library interface
to CUDA accelerated libraries
Pythonic APIs are built on top of low-level 1:1 Python bindings to C APIs

Platform-agnostic
Runs on GPU, CPU (x86 & Arm) and Grace-Hopper



Access to all expanded and peak-performance features
Reduced & mixed precisions
Device APIs
Functional customizations (e.g. cuFFT inline callbacks)

Unified design for high-productivity & performance
End Users
Package Developers
Kernel Authors

Warp: Advancing Simulation AI with Differentiable GPU Computing in Python [S63345]

nvmath-python demo

Example 1: Basic and advanced use of the specialized matmul APIs

$$\boldsymbol{d} = ReLU(\boldsymbol{a}\times\boldsymbol{b} + \boldsymbol{bias})$$

→ Python Summary

See: S62162_nvmath-python_fused_matmul_demo.pdf

nvmath-python demo

Example 2: Convolution three ways: host APIs, callbacks, fused kernel

$$\boldsymbol{y} = iFFT(FFT(\boldsymbol{x}_{in}) \cdot \boldsymbol{f})$$

→ Python Summary

See: S62162_nvmath-python_convolutions_demo.pdf

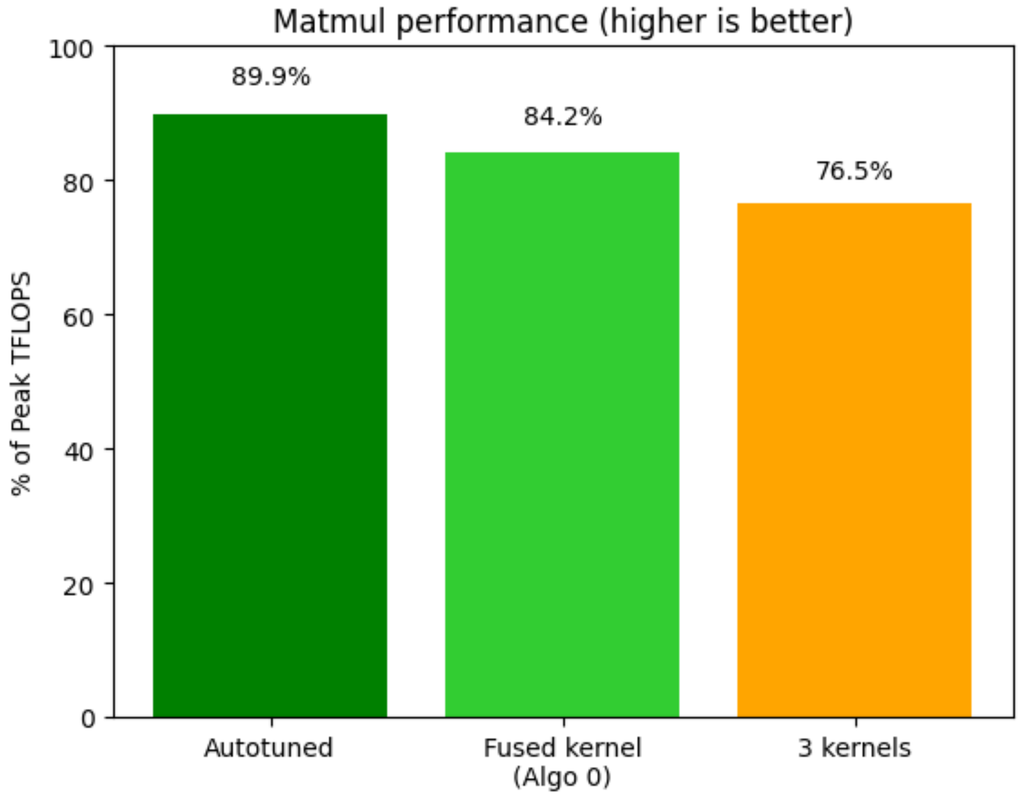
nvmath-python Demo Summary

nvmath-python Demo Summary

High-productivity environment providing flexibility to optimize performance

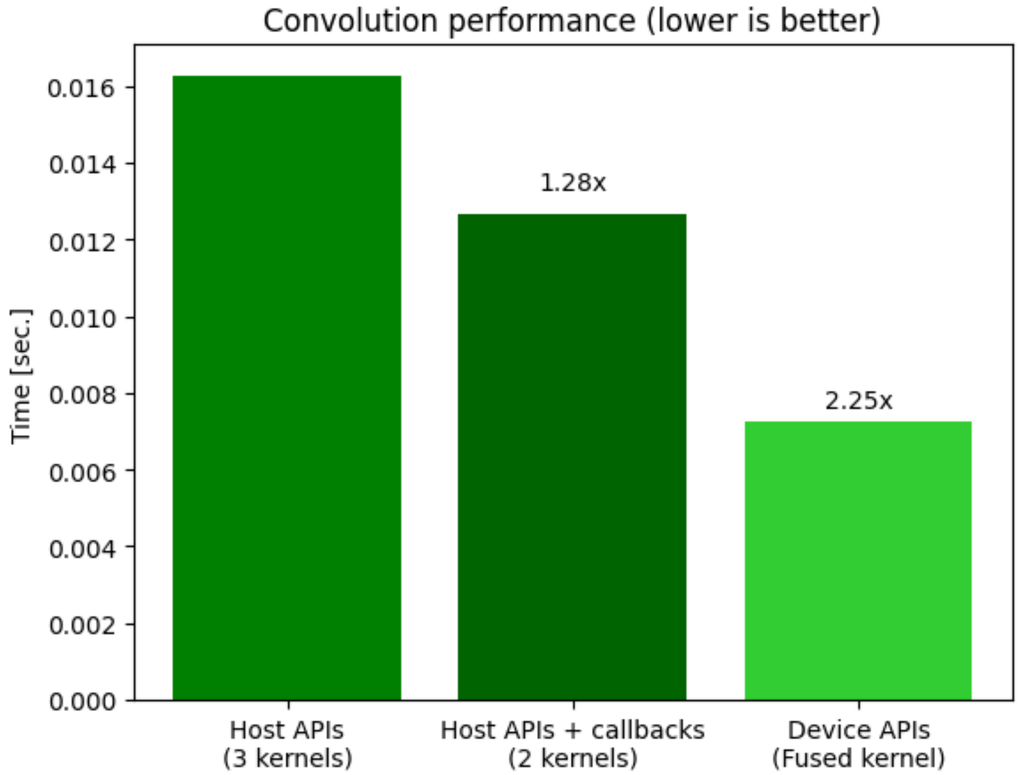
Example 1: Basic and advanced use of the specialized **matmul** APIs (uses cuBLASLt)

$$\boldsymbol{d} = ReLU(\boldsymbol{a}\times\boldsymbol{b} + \boldsymbol{bias})$$



Example 2: **Convolution** three ways: host APIs, callbacks, fused kernel (uses cuFFT and cuFFTDx)

$$\boldsymbol{y} = iFFT(FFT(\boldsymbol{x}_{in}) \cdot \boldsymbol{f})$$



When will I be able to use it?

Tentative Release Timeline

Releases will be available for pip and conda installation
Source code will be available on github

2024 H1

FFTs with callbacks
Dense Linear Algebra
Device Extensions (Numba)
GPU only

2024 H2

Support more compilers for Device Extensions
    Warp*, ...
Sparse Linear Algebra
Random Number Generator (RNG)
CPU (NVPL) and GPU

Future

Multi-GPU (MG) and Multi-GPU Multi-Node (MGMN) FFTs, Linear Algebra, **...**

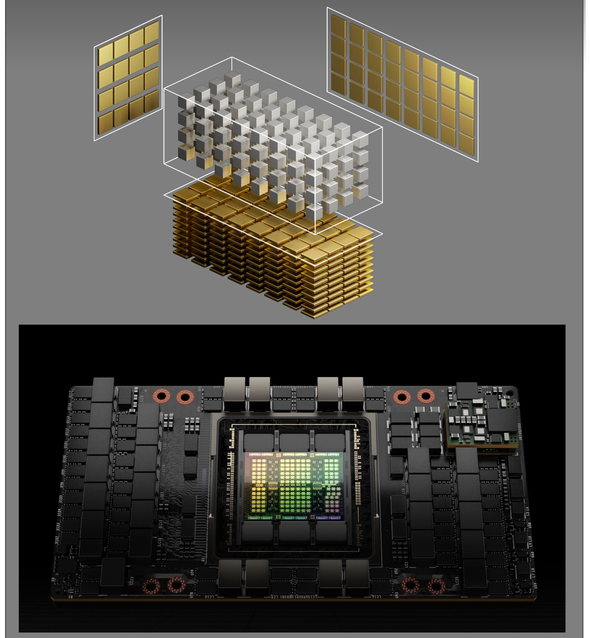
(*) Warp: Advancing Simulation AI with Differentiable GPU Computing in Python [S63345]

Future Outlook and Closing Remarks

Closing Remarks

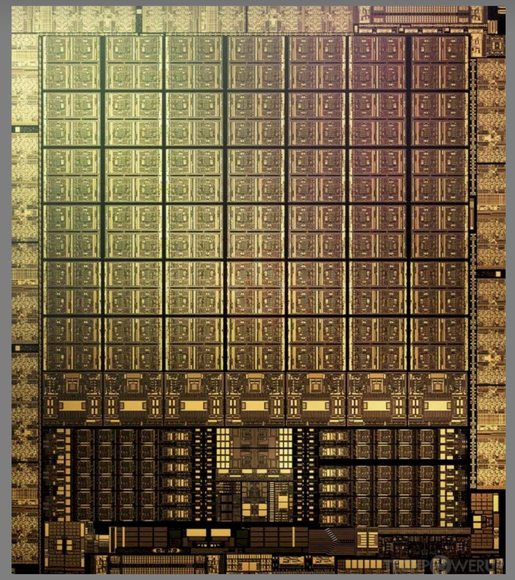
NVIDIA Math Libraries

Seamless Acceleration
Tensor Cores, TBC, …



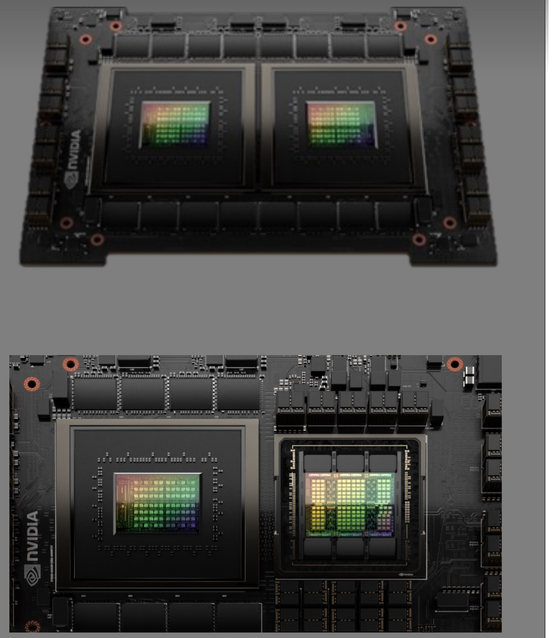
- FP8 in cuBLASLt & cuSPARSELt
- cuBLAS, cuSPARSE, cuDSS, cuTENSOR, cuFFT, nvImageCodec, nvCOMP
- nvmath-python

Composability
Device Extension Libraries



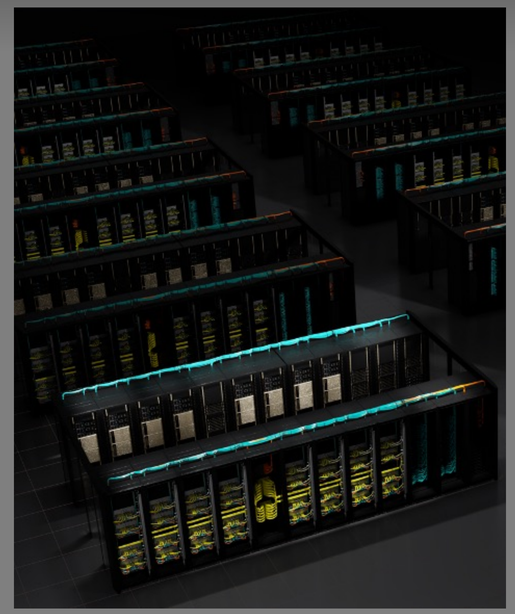
- cuFFTDx
- cuBLASDx
- Python Device Extensions

NVPL on Arm
High Performance CPU Libraries



- NVPL 24.03 release with new Tensor library

Multi-GPU & Multi-Node
Scaling Up to Supercomputers



- cuBLASMp joins cuSOLVERMp and cuFFTMp MGMN libraries

References

S61198	CUTLASS: A Performant, Flexible, and Portable Way to Target Hopper Tensor Cores
S63345	Warp: Advancing Simulation AI with Differentiable GPU Computing in Python
S61598	Scientific Computing With NVIDIA Grace and the Arm Software Ecosystem
S61203	A Deep Dive into the Latest HPC Software
S62515	GPU-Accelerating Process Simulation Performance using NVIDIA’s cuDSS Sparse Linear Systems Solver
S62275	Performance Optimization for Grace CPU Superchip
S62400	CUDA: New Features and Beyond
CWE62240	NVIDIA Math Libraries