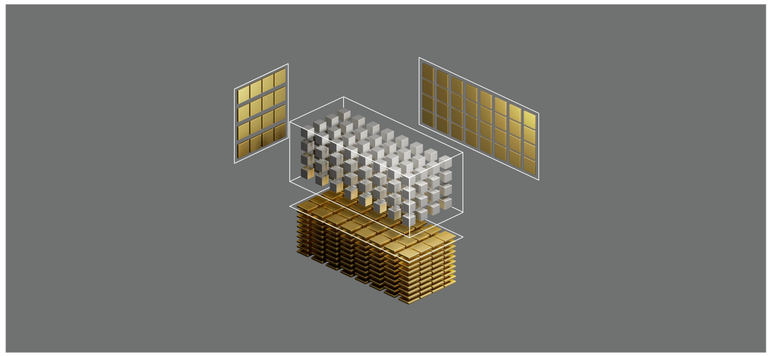
A Deep Dive into the Latest HPC Software

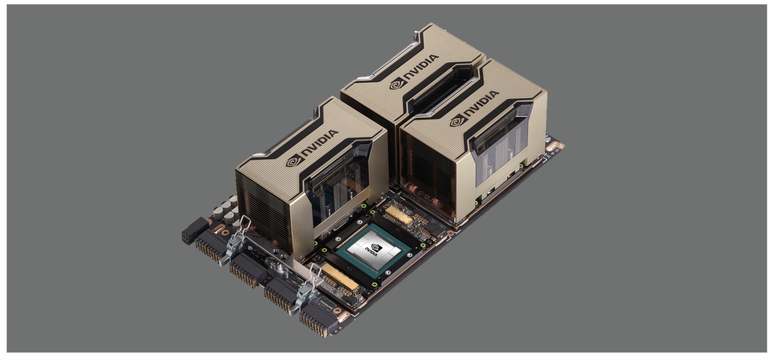
Jeff Larkin | GTC 2024

NVIDIA HPC Software

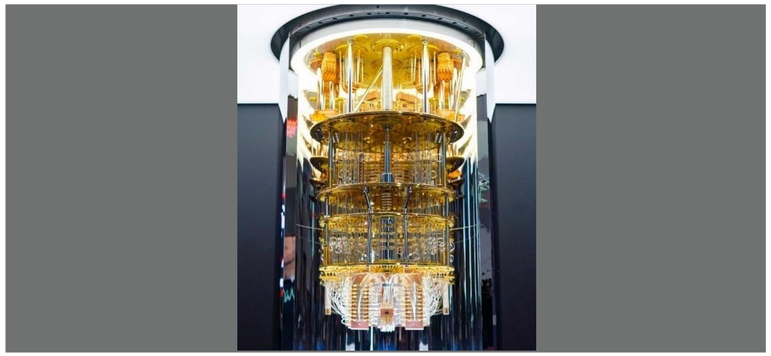
Major Initiatives



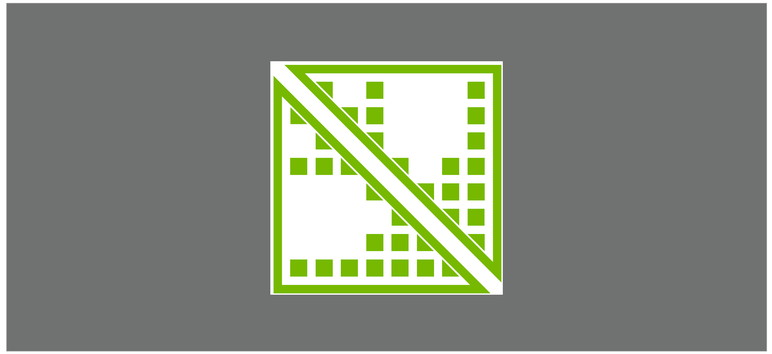
Seamless Adoption
Support from standard languages and libraries



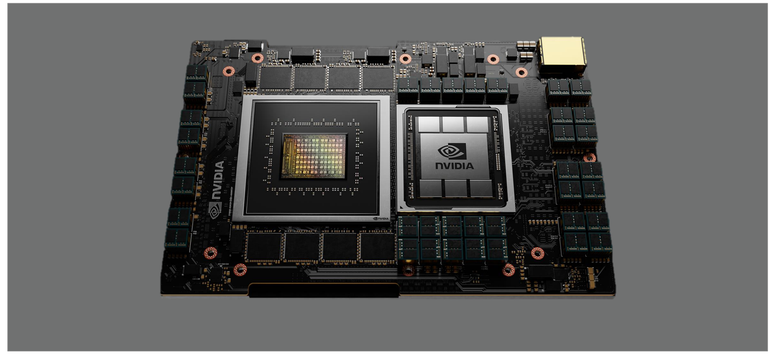
Scalable Solutions
Libraries, Tools, and Runtimes at System Scale



Quantum Accelerated Supercomputing
Defining the Quantum Accelerated Supercomputer



Domain Libraries
A rising tide for domain scientists



Mature Arm Ecosystem
Compilers, Libraries, Applications, Ecosystem

CUDA in More Languages
Python on GPUs and at scale, more languages to come

Agenda

- Improving Programmer Productivity with HPC SDK
- Enhancing Support for Python Developers
- Evolving NVIDIA’s Math Libraries
- Enabling Grace Hopper
- Creating the Ecosystem for Quantum Accelerated Supercomputing

Improving Programmer Productivity with HPC SDK

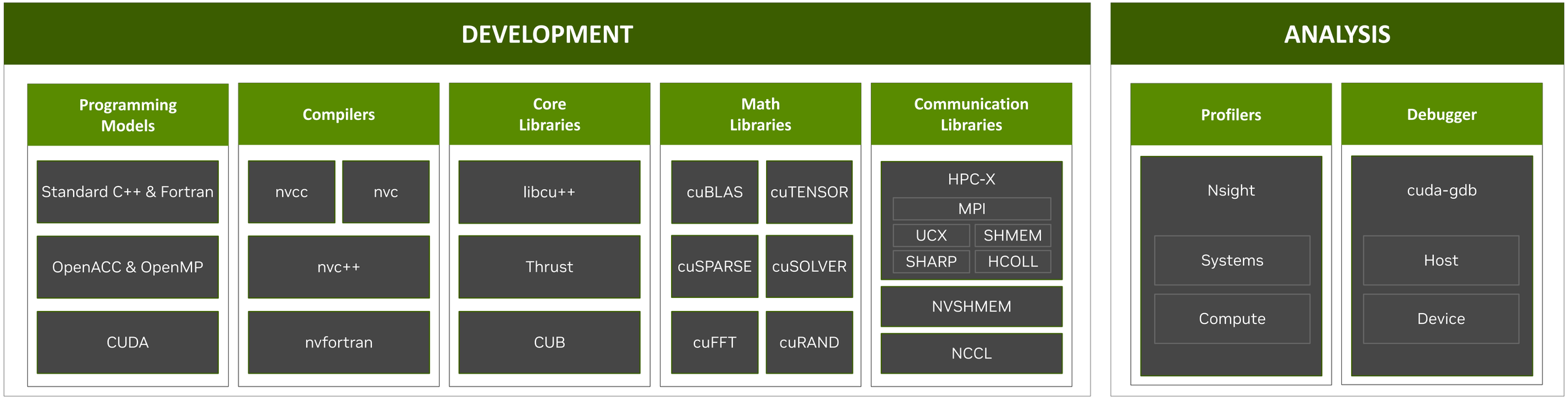
Programming the NVIDIA Platform

Unmatched Developer Flexibility

Languages & Programming Models	Accelerated Standard Languages	Incremental Optimization	Platform Specialization
	  Fortran	OpenACC More Science, Less Programming OpenMP	NVIDIA CUDA C++ | Fortran | Python
Libraries & Frameworks	Acceleration Libraries: Core, Math, Communication, Data Analytics, AI, DSLs		
Hardware	GPU	CPU	Interconnect
Wherever You Run	PCs	Workstations	On Prem	Cloud	At the Edge

NVIDIA HPC SDK

Available at developer.nvidia.com/hpc-sdk, on NGC, via Spack, and in the Cloud



Develop for the NVIDIA Platform: GPU, CPU and Interconnect

Libraries | Accelerated C++ and Fortran | Directives | CUDA

x86_64 | Arm

6 Releases Per Year | Freely Available

HPC SDK Updates

Grace Hopper, unified memory, and more

- **HPC SDK 23.11:**
  - Unified memory support for stdpar, OpenACC, and CUDA C++/Fortran
  - NVTX improvements for stdpar codes
    - Now you can see your stdpar in NSight: improved tools support, developer experience, performance optimizations
  - C-Fortran Interface
    - Better multi-paradigm interoperability for mixed C, C++, and Fortran codes
    - F2008 MPI bindings for nvfortran
  - C++20 Coroutines for CPU
    - Future GPU support will enable alternative async models for stdpar
  - Support for Grace Hopper in all bundled components
    - Compilers, Math Libraries, Networking, Tools.
  - HPC-X is the default MPI implementation optimized for NV platform
  - Grace(/Arm) performance (-tp=neoverse-v2)
    - Re-engineered vectorizer, intrinsics, system math library functions

- **HPC SDK 24.3:**
  - Improved compile speed for nvc++
    - Up to 1.15x - 2x faster for some workloads
  - Unified memory support for OpenMP Target Offload
  - Integrated NVIDIA Performance Library (NVPL) for Grace CPUs
  - CUDA Fortran `unified` attribute

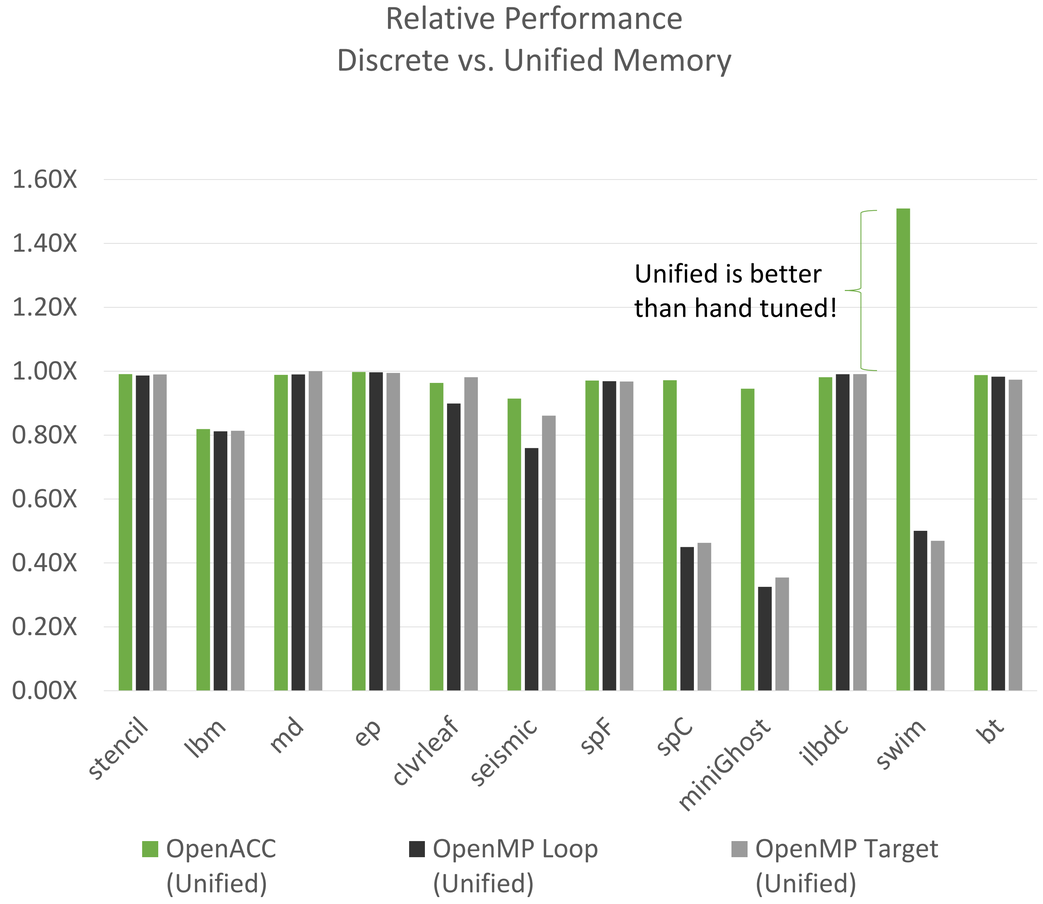
Unified Memory

- C++ stdpar improvements
- Fortran stdpar improvements
- OpenACC improvements
- CUDA Fortran
- OpenMP Target Offload
- Unified Functions

SPECaccel® 2023 on Grace Hopper

Performance on Unified Memory

- SPECaccel® is an industry-accepted benchmark suite that now includes multiple directive-based benchmarks
- With Grace Hopper's Unified Memory, performance is generally on-par with hand-tuned data movement
- We expect performance to mature with continuing development for Unified Memory in the NVIDIA software stack



SPEC® and the benchmark name SPECaccel® are registered trademarks of the Standard Performance Evaluation Corporation. For more information about SPECaccel 2023, see www.spec.org/accel2023/

Results run March 2024 with NVIDIA HPC SDK 24.3.

Vienna Ab initio Simulation Package (VASP)

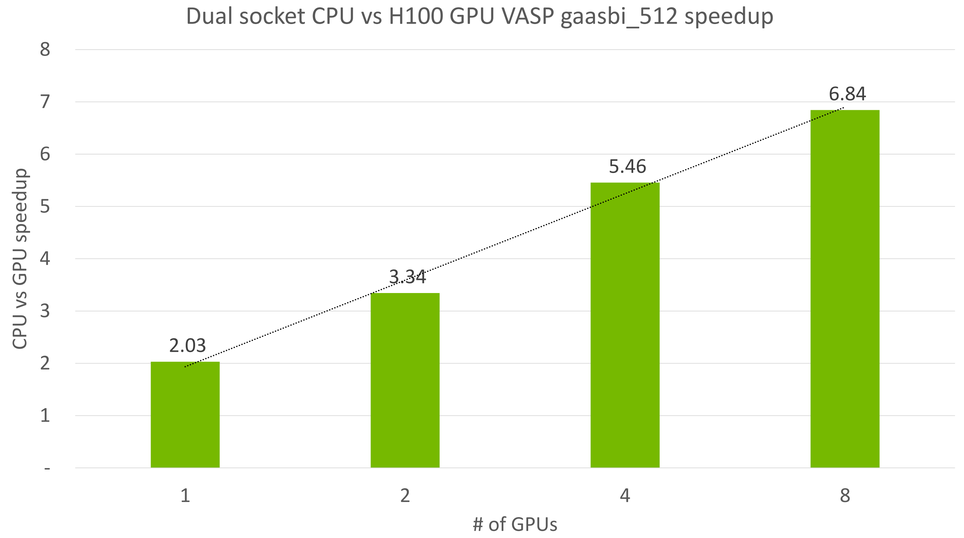
An efficient implementation in OpenACC for material science

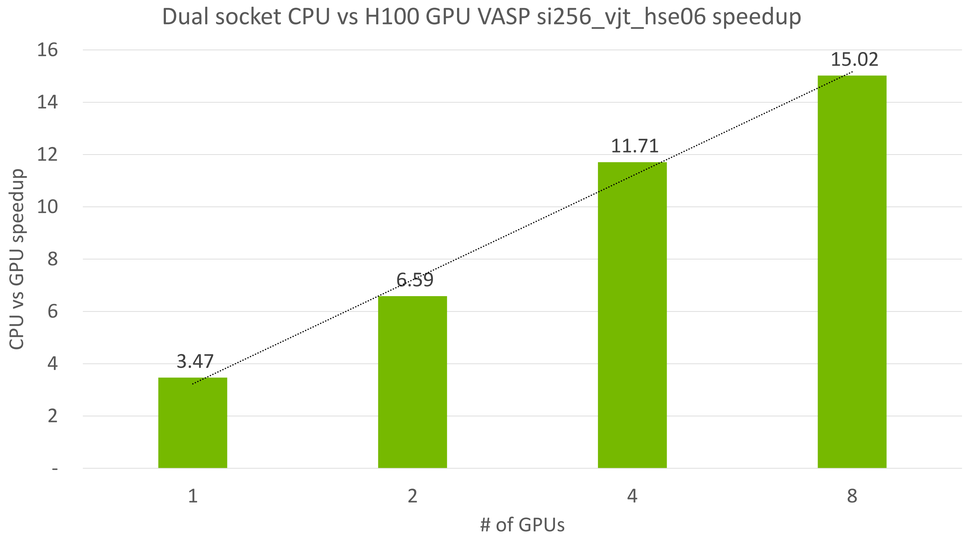
- **Use case:**
  VASP is a computer program for atomic-scale materials modeling, such as electronic structure calculations and quantum-mechanical molecular dynamics, from first principles.

- **Implementation:**

  OpenACC directives have been extensibility used in combination with NVIDIA libraries to maximize performance and scalability. Benchmarks are performed using:
  - VASP 6.4.1
  - NVIDIA HPC SDK's 23.9
  - CUDA 12.2
  - NVIDIA Magnum IO (MPI and NCCL)
  - cuBLAS
  - cuSOLVER
  - cuFFT
  - Future VASP Release: cuSOLVERMp, cuBLASMp

- **Deep dive:**
  - Optimize Energy Efficiency of Multi-Node VASP Simulations with NVIDIA Magnum IO



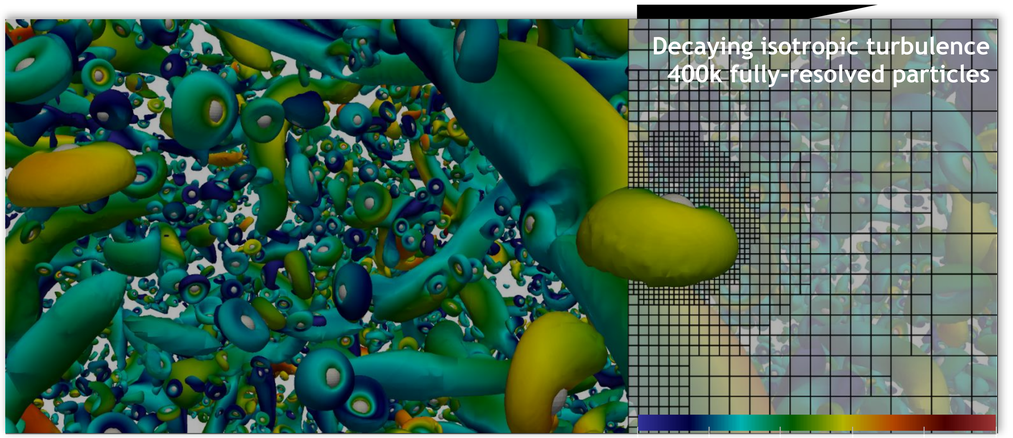


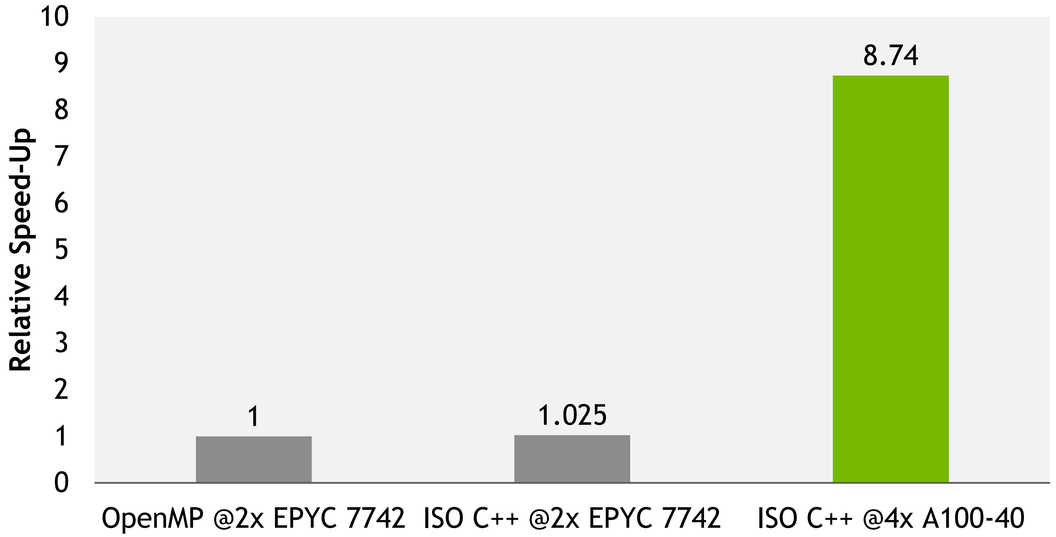
M-AIA

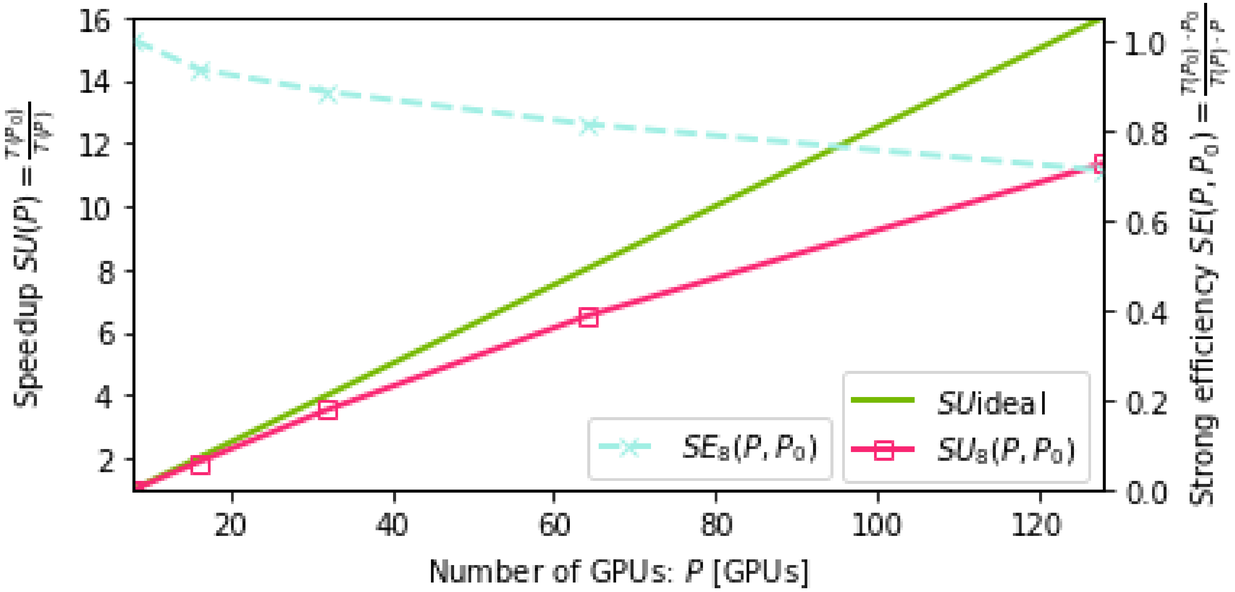
Multi-physics simulation framework developed at the Institute of Aerodynamics, RWTH Aachen University

MAIA

- Hierarchical grids, complex moving geometries
- Adaptive meshing, load balancing
- Numerical methods: FV, DG, LBM, FEM, Level-Set, ...
- Physics: aeroacoustics, combustion, biomedical, ...
- Developed by ~20 PhDs (Mech. Eng.), ~500k LOC++
- **Programming model:** MPI + ISO C++ parallelism







ABINIT: Materials Science without Porting

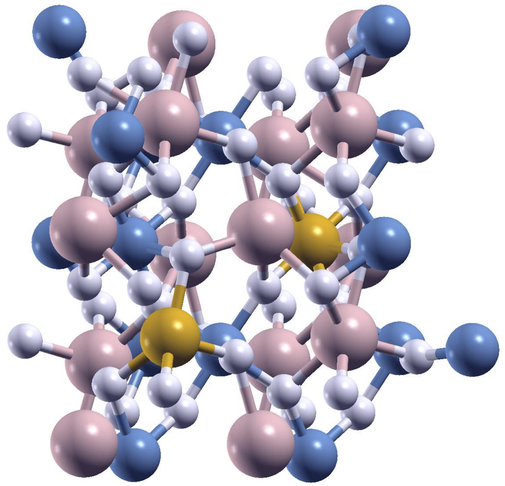
Drop-In Math Libraries + Standard Parallelism

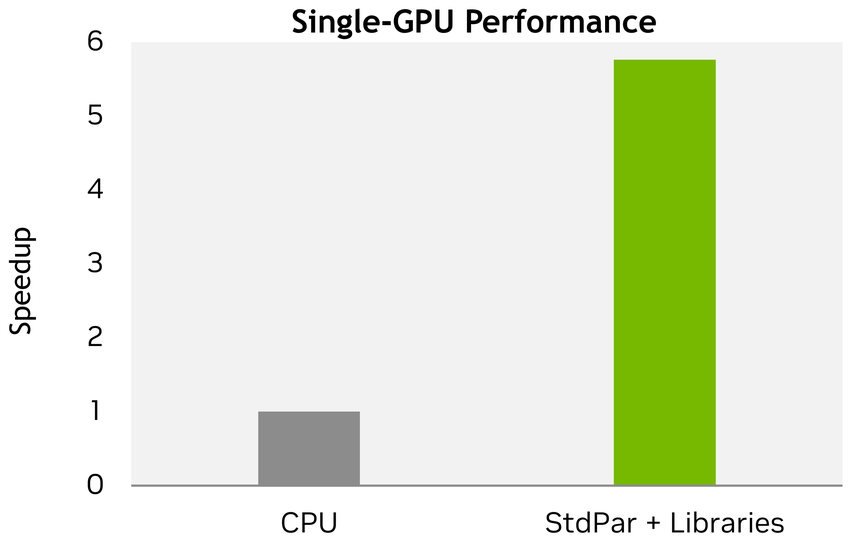
ABINIT

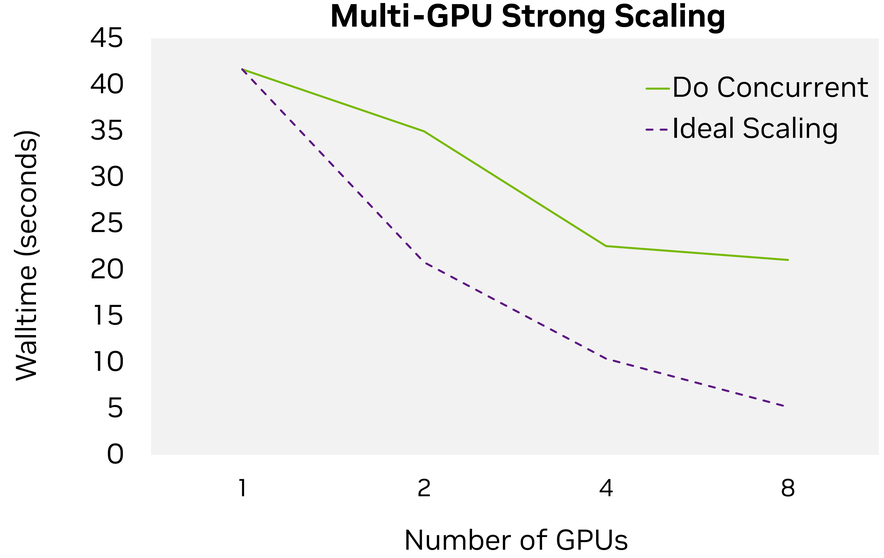
Widely-used software package for the *ab initio* study of material properties based on Kohn-Sham Density Functional Theory with pseudopotentials and a plane wave basis.

Fortran with MPI and optional OpenMP parallelization.

SCF method with LOBPCG eigen-solver re-written with ***Drop-In Math Libraries*** and **StdPar (*DO CONCURRENT*)**.





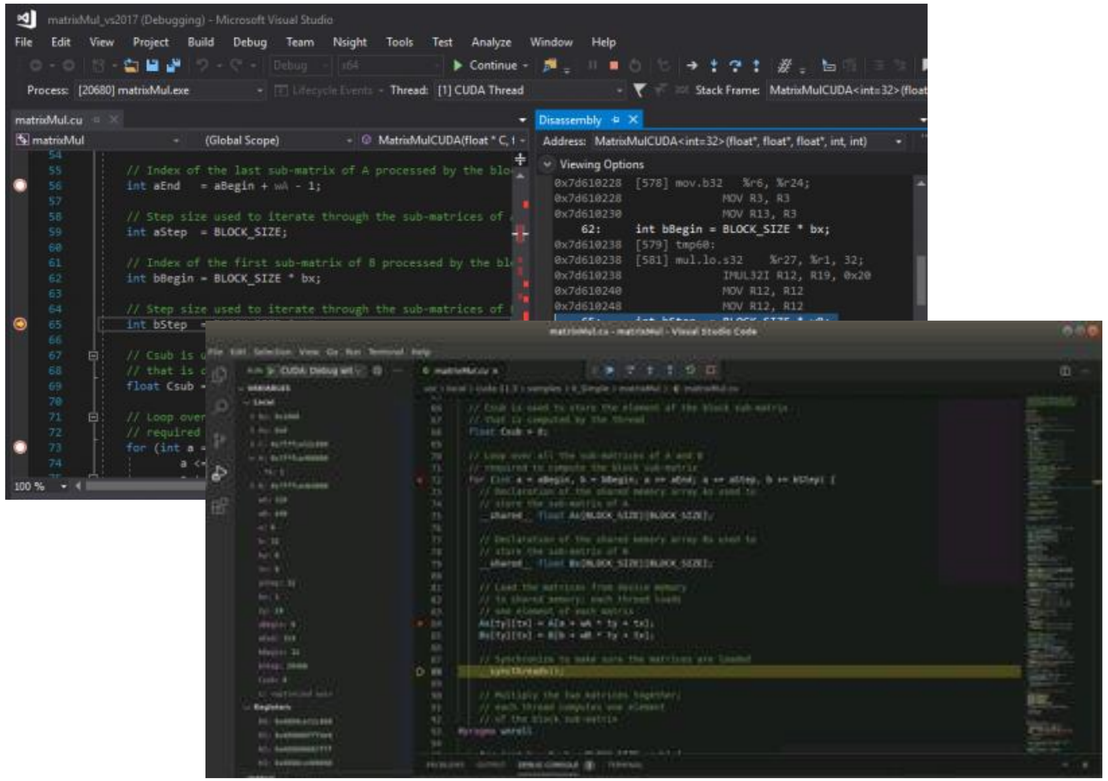


CPU: 2X Intel® Xeon® Platinum 8480C CPU. GPU: NVIDIA H100. NVIDIA HPC SDK 24.1.

Developer Tools Ecosystem

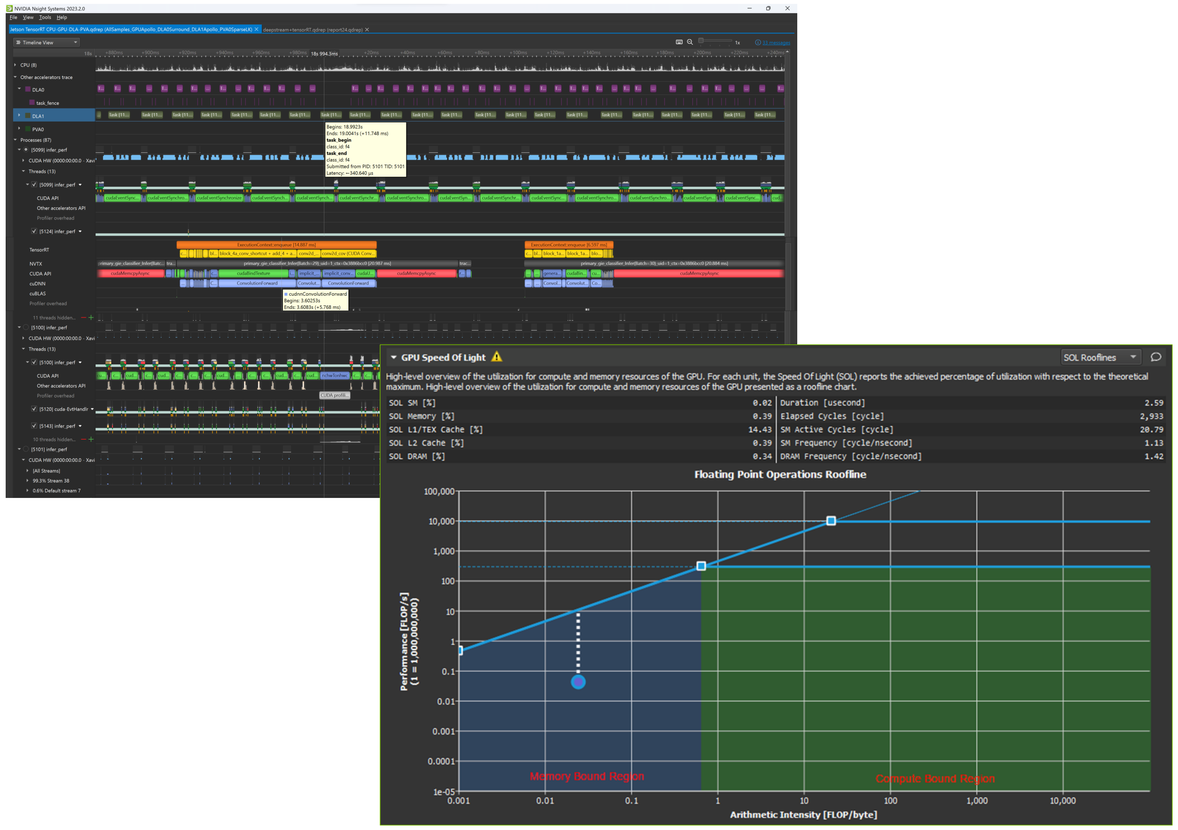
Debuggers & IDE Integrations

cuda-gdb, Nsight Visual Studio Edition,
Nsight Visual Studio **Code** Edition, Nsight Eclipse Edition



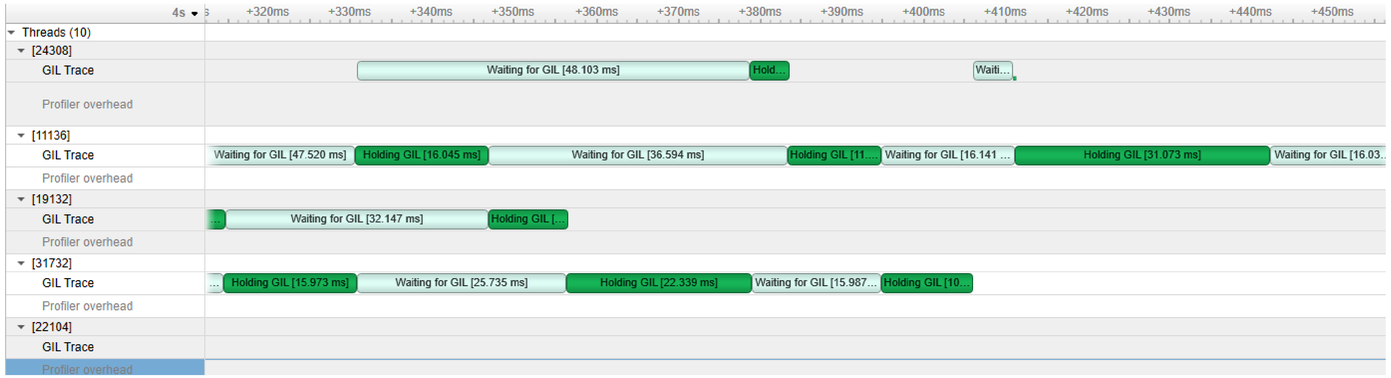
Profilers

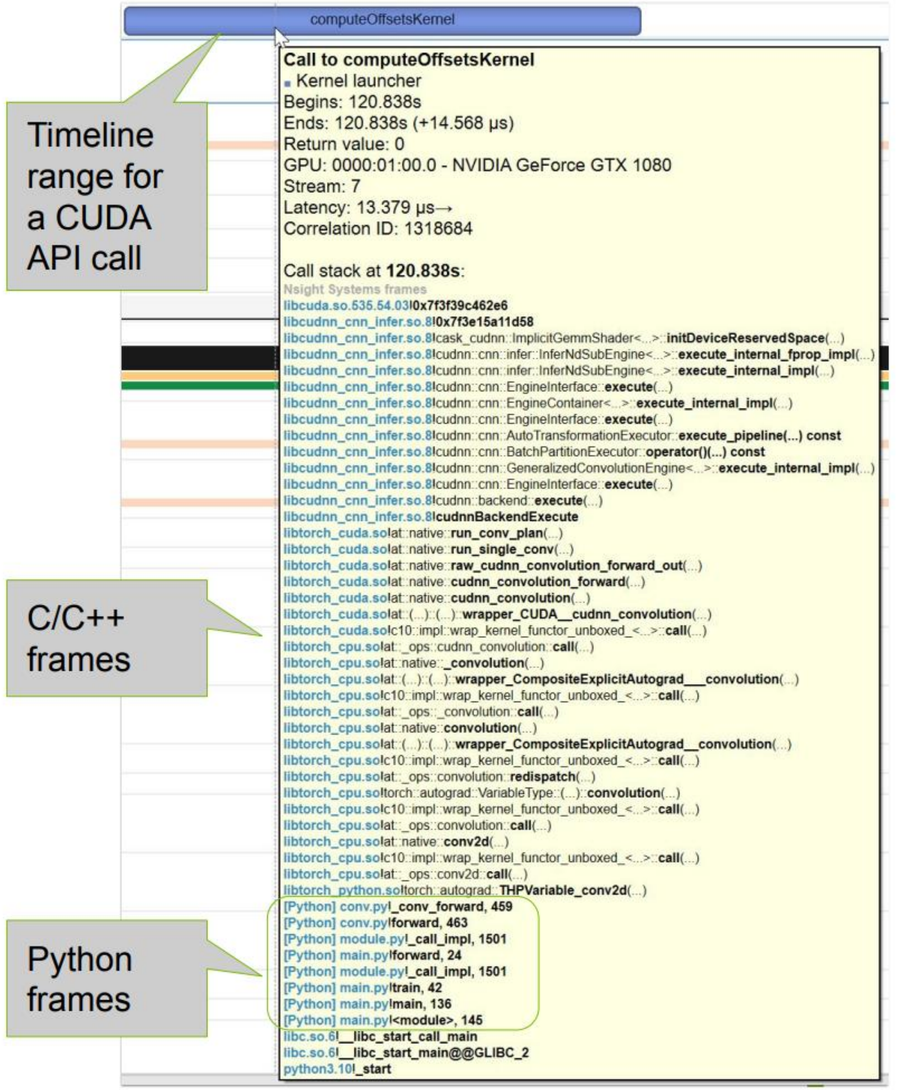
Nsight Systems, Nsight Compute, CUPTI, NVIDIA
Tools eXtension (NVTX)



Python Profiling Updates

- Python Call Stacks Samples and CUDA API Backtrace
  - Identify where you are and how you got there
- Global Interpreter Lock (GIL) trace
  - Common performance limiter in Python
- See annotated code ranges built into in popular frameworks and libraries such as:
  - RAPIDS, Spark, CV-CUDA, and more…

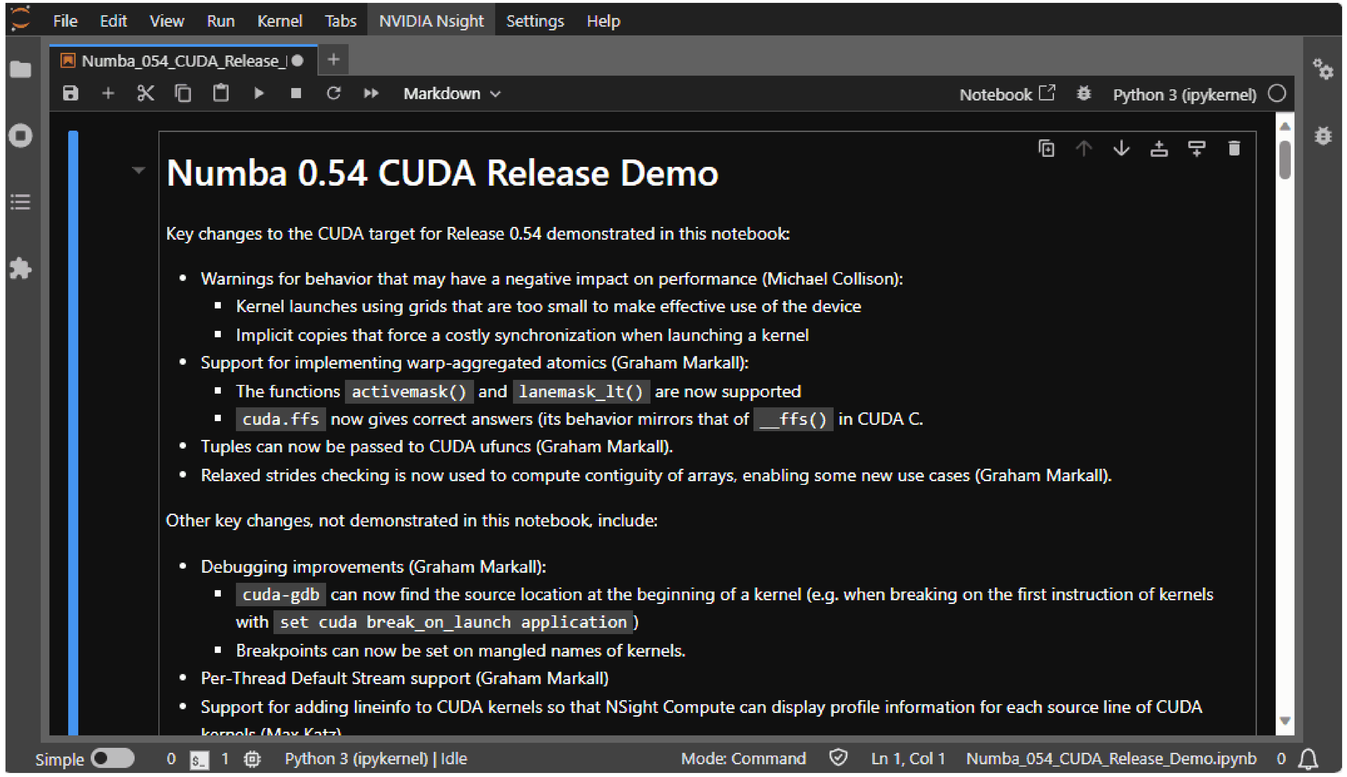




Jupyter Lab Integration Updates

- Extension to Jupyter Lab
- Profile individual Jupyter cells
- Text-based results can be viewed directly in Jupyter
- Launch **new** remote GUI streaming container directly in Jupyter Lab
  - Servers without X, Windowing Manager, …
  - Container with X, WM, & WebRTC server
  - Dockerfile inside Nsight Systems Installer

- See it in action:
  - DLIT61667: Profilers, Python, and Performance: Nsight Tools for Optimizing Modern CUDA Workloads

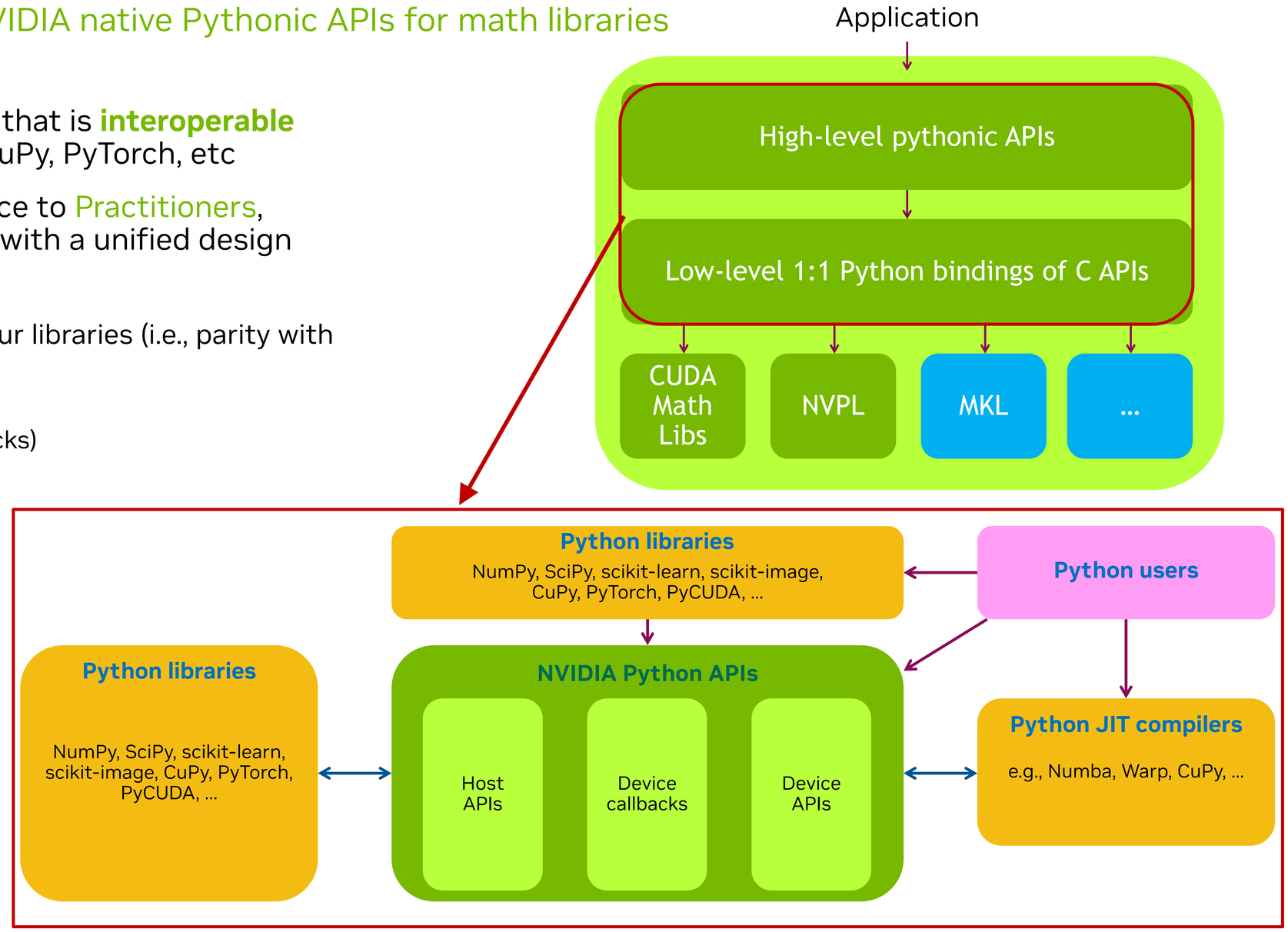


Enhancing Support for Python Developers

Introducing: nvmath-python (Preview)

NVIDIA native Pythonic APIs for math libraries

- Deliver a pythonic & user-friendly library that is **interoperable** with core numerical packages: NumPy, CuPy, PyTorch, etc
- Deliver high-productivity and performance to Practitioners, Package Developers and Kernel Authors with a unified design
- Key features:
  - Ensuring extended features offered by our libraries (i.e., parity with our C offering) such as:
    - Reduced and mixed-precision
    - Function customizations (e.g., FFT callbacks)
    - Kernel development productivity through device APIs
  - Platform-agnostic single codebase
    - e.g. Grace, Grace-Hopper, x86-Hopper
  - Easy access to CUDA Math Libraries functionality in a performant manner.
  - Shrinking lead time for Python users to access the latest library features



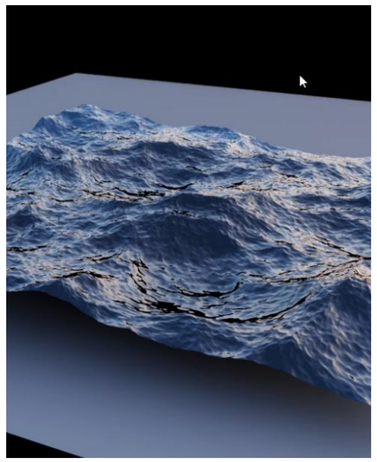
See Deep Dive into NVIDIA Math Libraries [S62162] for more details.

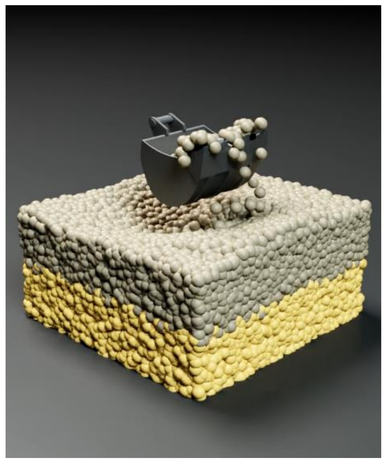
Warp 1.0

Differentiable Accelerated Computing for Python

1. GPU Kernels in Python
   - Write CUDA kernels in 100% Python syntax
   - Runtime JIT compilation
   - Fast developer iteration
2. Spatial Computing
   - Rich vector math library
   - Mesh processing and queries
   - Sparse volumes (OpenVDB)
   - Hash grids
3. Differentiable programming
   - Auto-differentiation
   - Forward + backward kernel generation
   - Interop with DL frameworks (e.g.: Torch, JAX)
4. Omniverse integration
   - USD import/export
   - Runtime extensions for Isaac/USD Composer

```python
import warp as wp

@wp.kernel
def integrate(p: wp.array(dtype=wp.vec3),
              v: wp.array(dtype=wp.vec3),
              f: wp.array(dtype=wp.vec3),
              m: wp.array(dtype=float)):

    # thread id
    tid = wp.tid()

    # Semi-implicit Euler step
    v[tid] = v[tid] + (f[tid] * m[tid] + wp.vec3(0.0, -9.8, 0.0)) * dt
    x[tid] = x[tid] + v[tid] * dt

# kernel launch
wp.launch(integrate, dim=1024, inputs=[x, v, f, …], device="cuda:0")
```





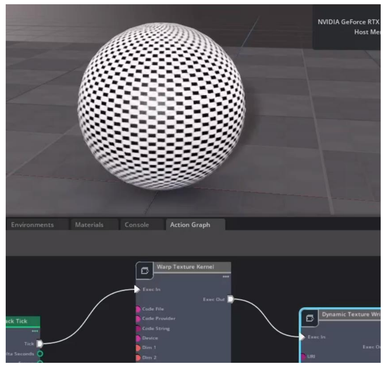
https://github.com/NVIDIA/warp | `pip install warp-lang`

See Warp: Advancing Simulation AI with Differentiable GPU Computing in Python [S63345] for more Information.

Warp Use Cases

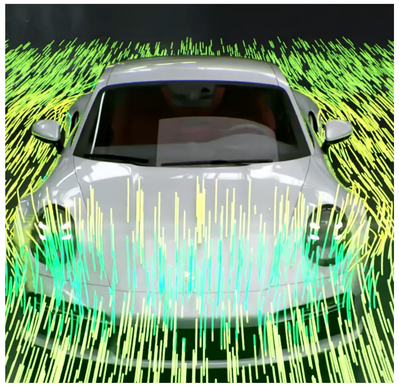
- Warp provides the building blocks for ISVs to create, accelerate and extend their own simulators
- Not providing out-of-the-box CFD solvers, but make it easy to write their own

Data Processing



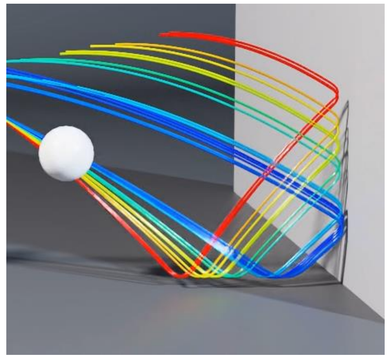
- Mesh processing
- Image processing
- Synthetic data generation

Simulation



- Rigid body dynamics
- Elasticity
- Fluid flow

Training



- Neural dynamics
- Parameter estimation
- Trajectory optimization
- Inverse problems

Scripting

```
@wp.kernel
def initialize_particles(
    particle_x: wp.array(dtype=wp.vec3), sm
):
    tid = wp.tid()

    # grid size
    nr_x = wp.int32(width / 4.0 / smoothing
    nr_y = wp.int32(height / smoothing_leng
    nr_z = wp.int32(length / 4.0 / smoothin

    # calculate particle position
    z = wp.float(tid % nr_z)
    y = wp.float((tid // nr_z) % nr_y)
    x = wp.float((tid // (nr_z * nr_y)) % n
```

- Loss/reward functions
- Custom forces
- Custom behaviors
- Custom boundaries

Large Scale Fluid Simulation with Warp

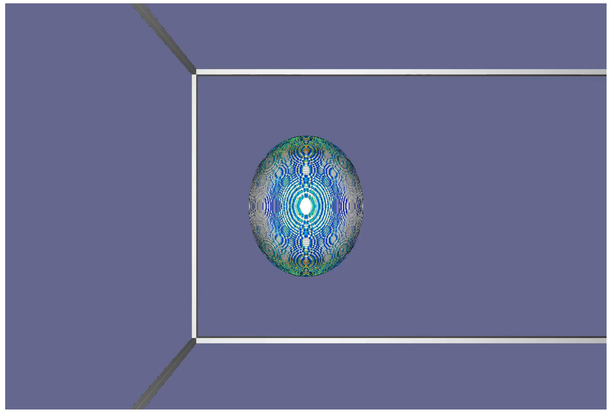
XLB is a Lattice Boltzmann fluid dynamics solver developed by Autodesk Research.

Developed in Python using JAX with key features:

- Easily extendable to add new methods
- Scalable across distributed systems
- ML friendly allowing integration with physics-based ML models
- Topological optimization and inverse problems

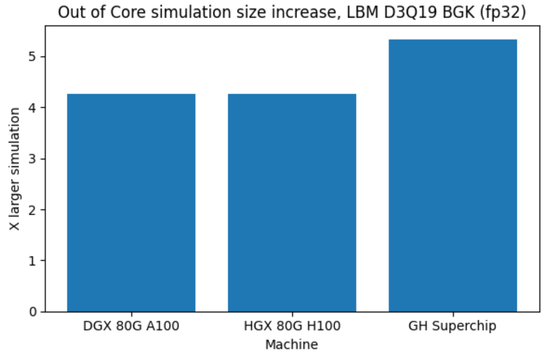
XLB + Warp

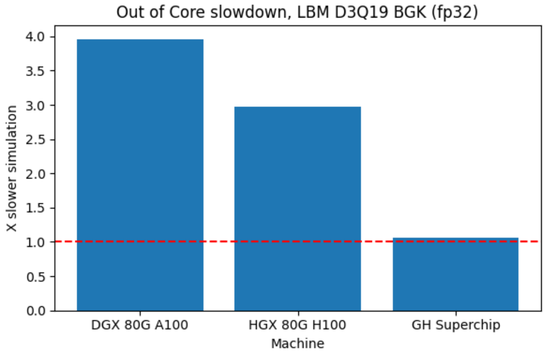
XLB is being restructured to allow Warp as a compute backend. This results in **~3.8x speed improvement and ~3.1x larger simulations**



XLB + GH200 + Warp + nvCOMP

Combining XLB, Warp, nvCOMP and our Grace Hopper super chips allows for efficient out of core memory simulations. This allows for **~5x larger simulations** with minimal impact on performance.





Ecosystem for Accessible Accelerated Computing

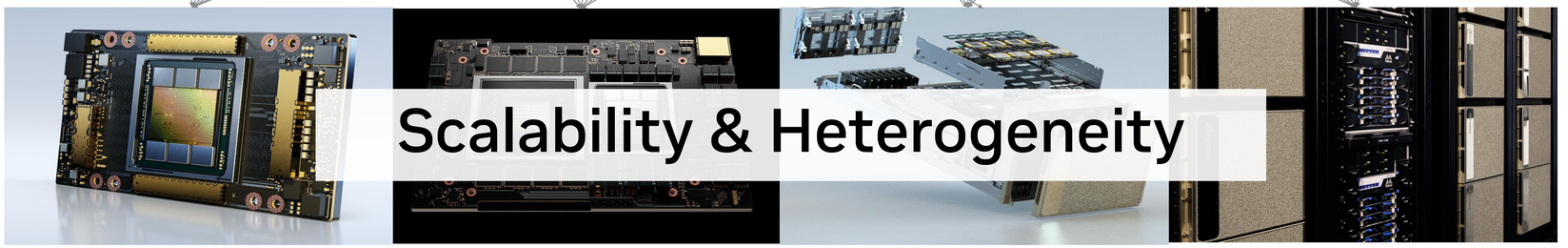
Key challenge: composability and scalability together at the same time

Target domain programming APIs to accelerate

Composability & Productivity

NumPy SciPy pandas scikit learn dmlc XGBoost . . .

NVIDIA hardware with various computing capacities



Scalability & Heterogeneity

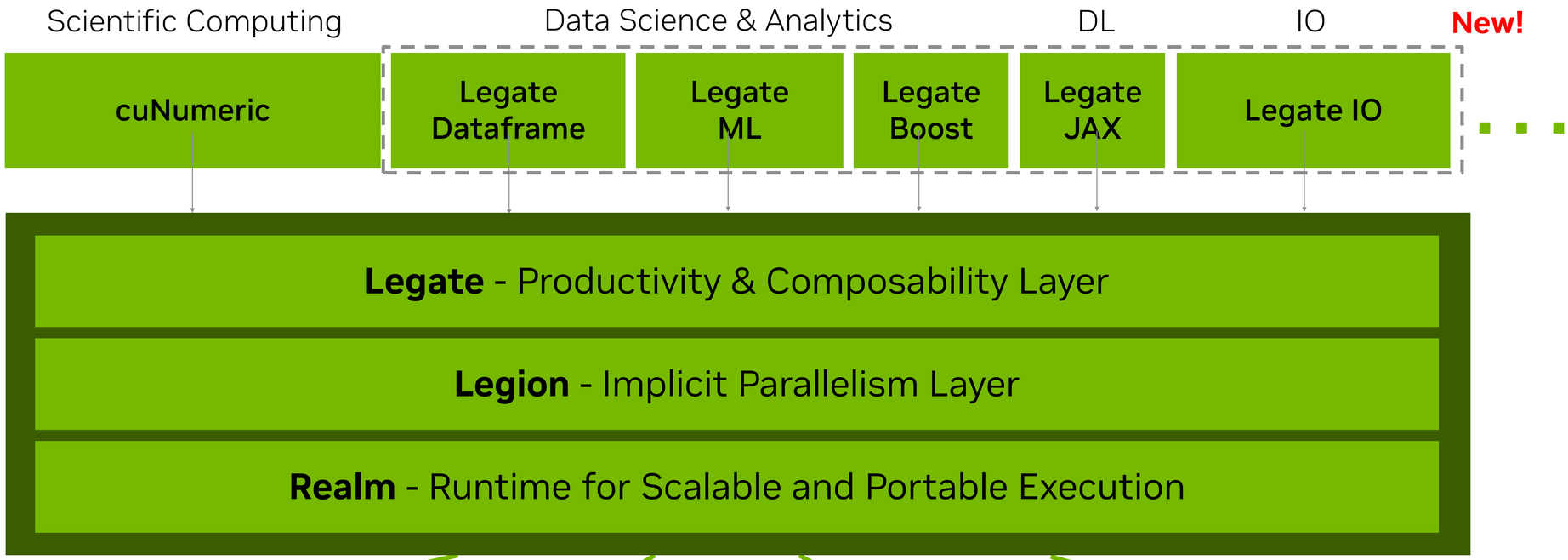
Single GPU | Mixed CPU/GPU | Single-Node Multi-GPU | Cloud & Supercomputer

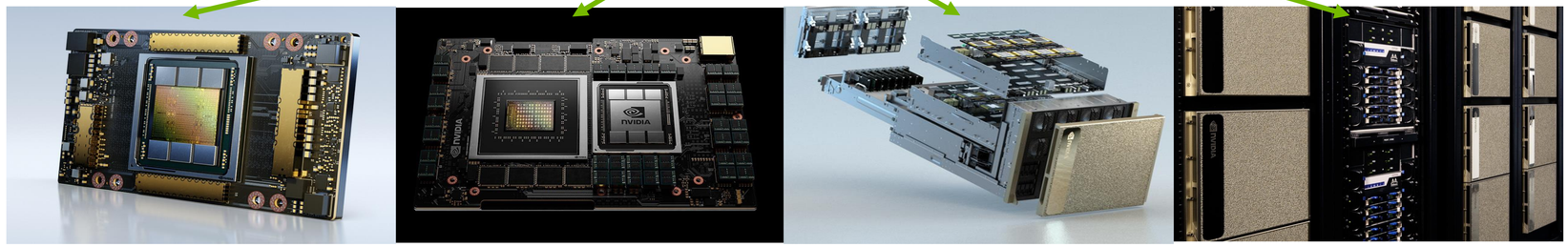
Legate Ecosystem

Composable and accelerated libraries for scientific computing and beyond

Foundational Legate libraries

Core programming framework for productive library development





Single GPU | Mixed CPU/GPU | Single-Node Multi-GPU | Cloud & Supercomputer

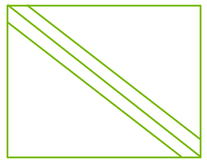
See Legate: A Productive Programming Framework for Composable, Scalable, Accelerated Libraries [S62262] for more details.

Evolving NVIDIA’s Math Libraries

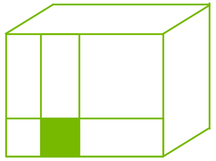
NVIDIA Math Libraries

Linear Algebra, FFT, RNG, and Basic Math

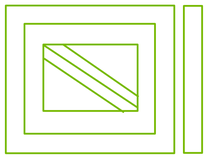
cuBLAS



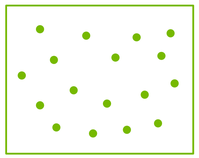
cuSPARSE



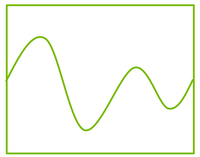
cuTENSOR



AMGX

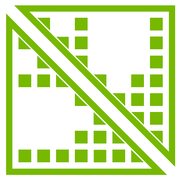


cuRAND

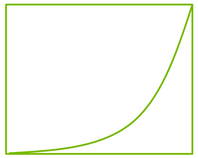


cuFFT

cuSOLVER



cuDSS



Math API

Multi GPU Multi Node APIs

Scalable and Grace Hopper Support

cuFFTMp

2D and 3D **FFTs Decompositions**: Slab (1D), Pencil (2D)

cuSOLVERMp

Factorization, Symmetric Eigensolver

cuBLASMp

GEMM, TRSM, SYRK at scale for FP16, FP32, FP64 types

Communication Libraries

HPC-X

MPI

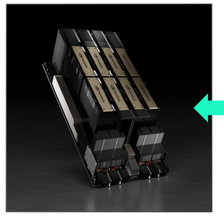
UCX | SHMEM | SHARP | HCOLL

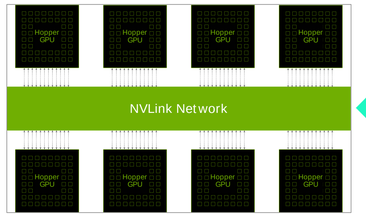
NVSHMEM

NCCL

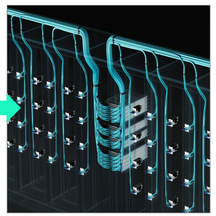
X86 + ARM support

DGX H100 8 GPUs

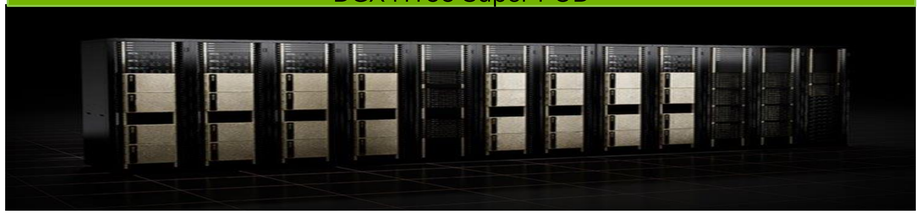




Infiniband Between Nodes



DGX H100 Super POD



DGX GH200

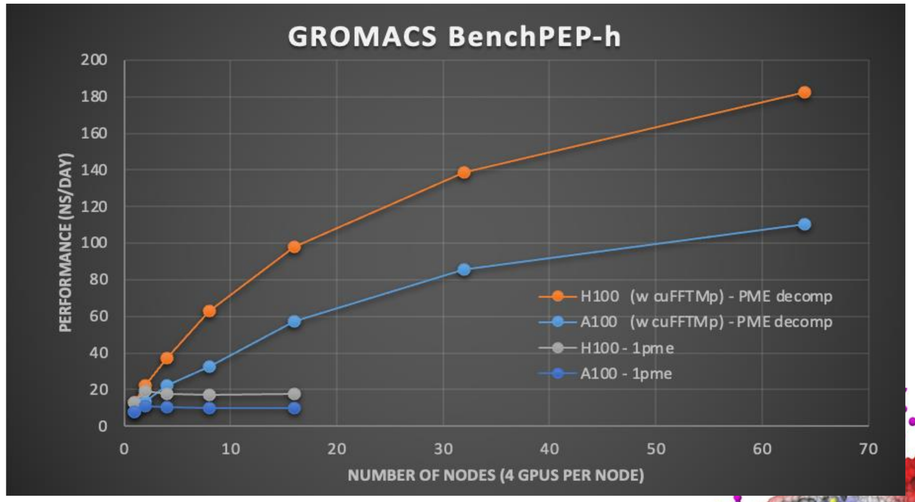
256 Grace Hopper Superchips | **1EFLOPS** AI Performance | **144TB** unified fast memory
36 L2 NVLink switches | **900 GB/s** GPU-to-GPU bandwidth | **128 TB/s** bisection bandwidth

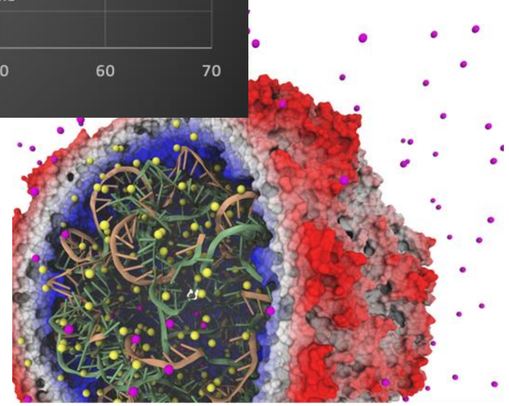
Multi-GPU, Multi-Node Libraries Scale Science

cuFFTMp and cuSOLVERMp

Integrating cuFFTMp scales GROMACS PME solvers for faster, larger simulations.

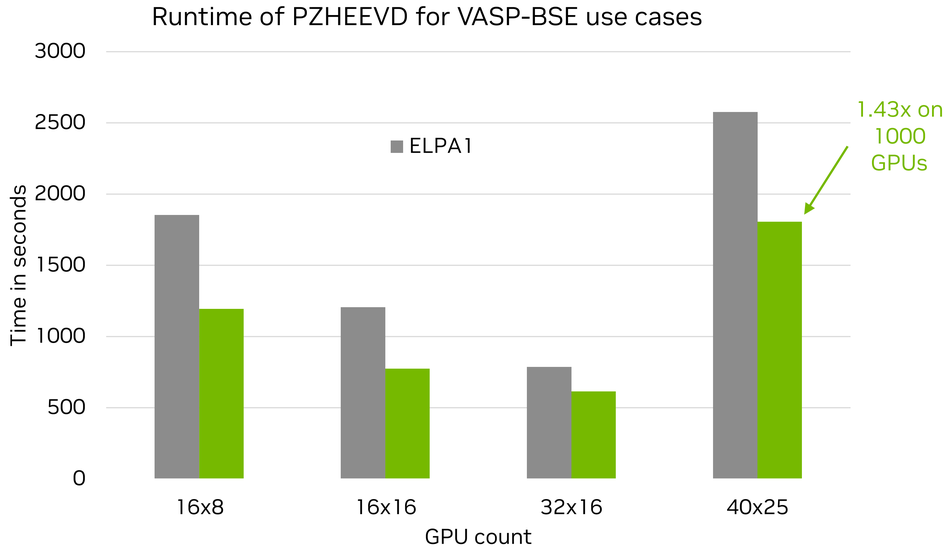
- **3x** or more performance on Hopper with cuFFTMp scaling on multi node configurations





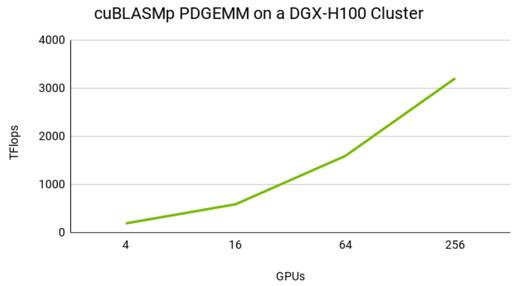
Integrating cuSOLVERMp into VASP for Distributed Symmetrical Eigensolver enables largerBSE calculation than ELPA library.

- **1.43X** on 1000X GPUs



New API Extensions: cuBLASDx and cuBLASMp

cuBLASDx Device side APIs (Preview)	Call **GEMM** and fuse with custom epilogues, GEMMs or FFTs **inside CUDA kernel** Concise and easier to express and use	Standalone download
cuBLASMp Host APIs (Preview)	NETLIB BLAS for **multi-node multi-GPU.** Callable from CPU code.	Standalone download and HPC SDK



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!

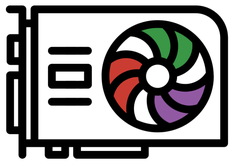
cuTENSOR

New APIs, features and performance in 2.0

- API Uniformity: Plan based Multi-stage APIs akin extended for all ops
- Dynamically allocated descriptors: Arbitrarily dimensional tensor descriptors
- “Opt-in” Just in Time ‘optimized’ kernel generation for tensor contractions
- Significant boost in performance relative to 1.x, and additional performance from JIT
- Available for download HPC SDK , Standalone download

CuPy

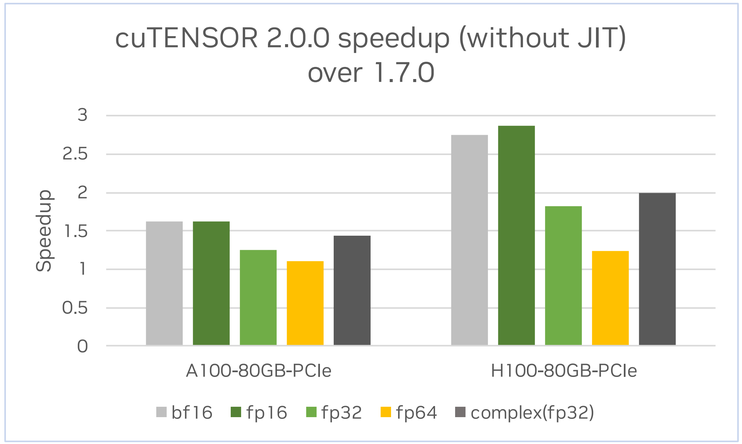
cuQuantum

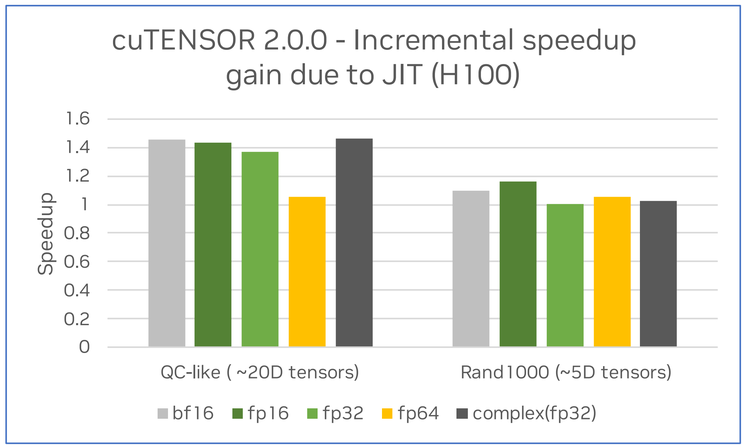


CUDA Julia

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



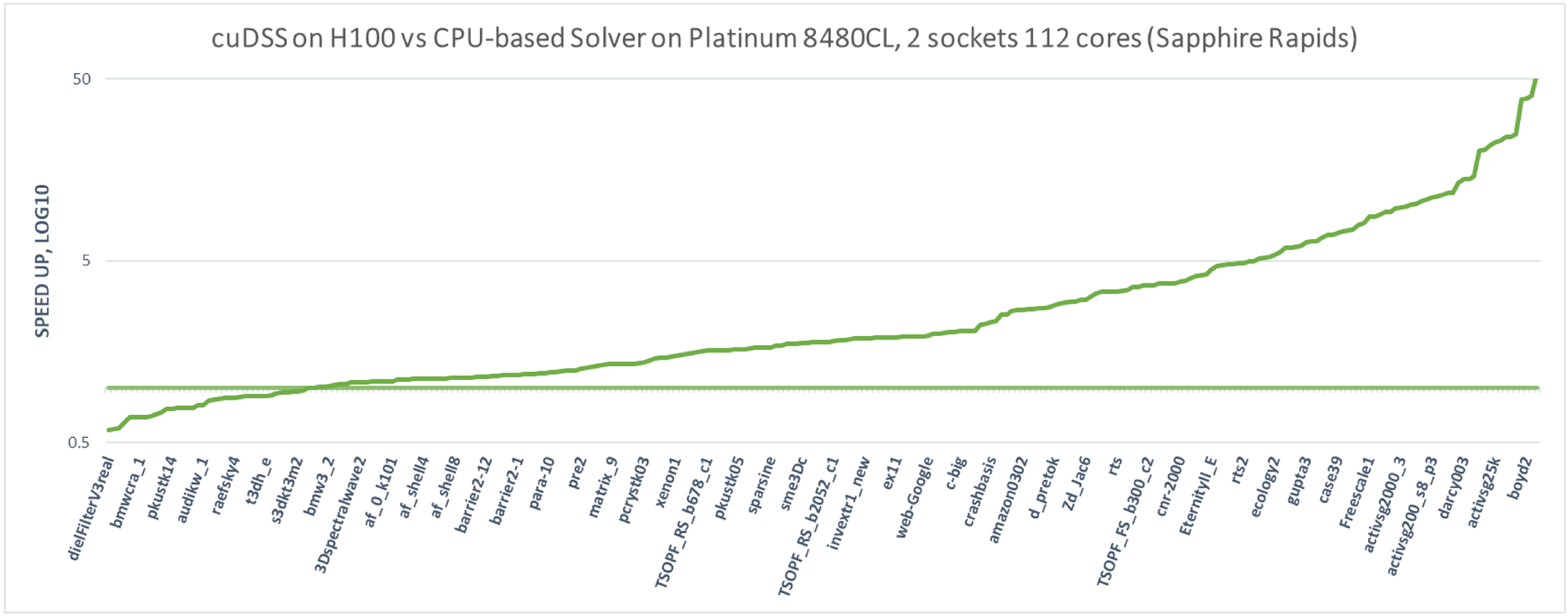


cuDSS

Preview of new Direct Sparse Solvers for NVIDIA GPUs is now available!

Solves sparse linear systems for sparse coefficients (A) on NVIDIA GPUs

Robust alternative to iterative solvers for use in Robotics, Self-driving, Process simulation, Circuit Simulation etc.,



See more at S62515 GPU-Accelerating Process Simulation Performance using NVIDIA's cuDSS Sparse Linear Systems Solver

Enabling Grace Hopper

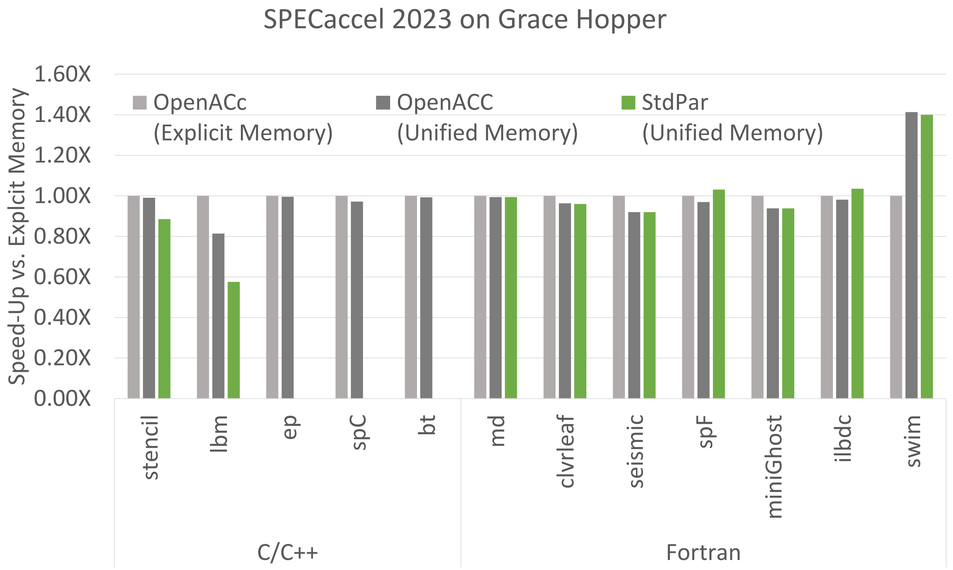
The Grace Hopper Advantage for Developers

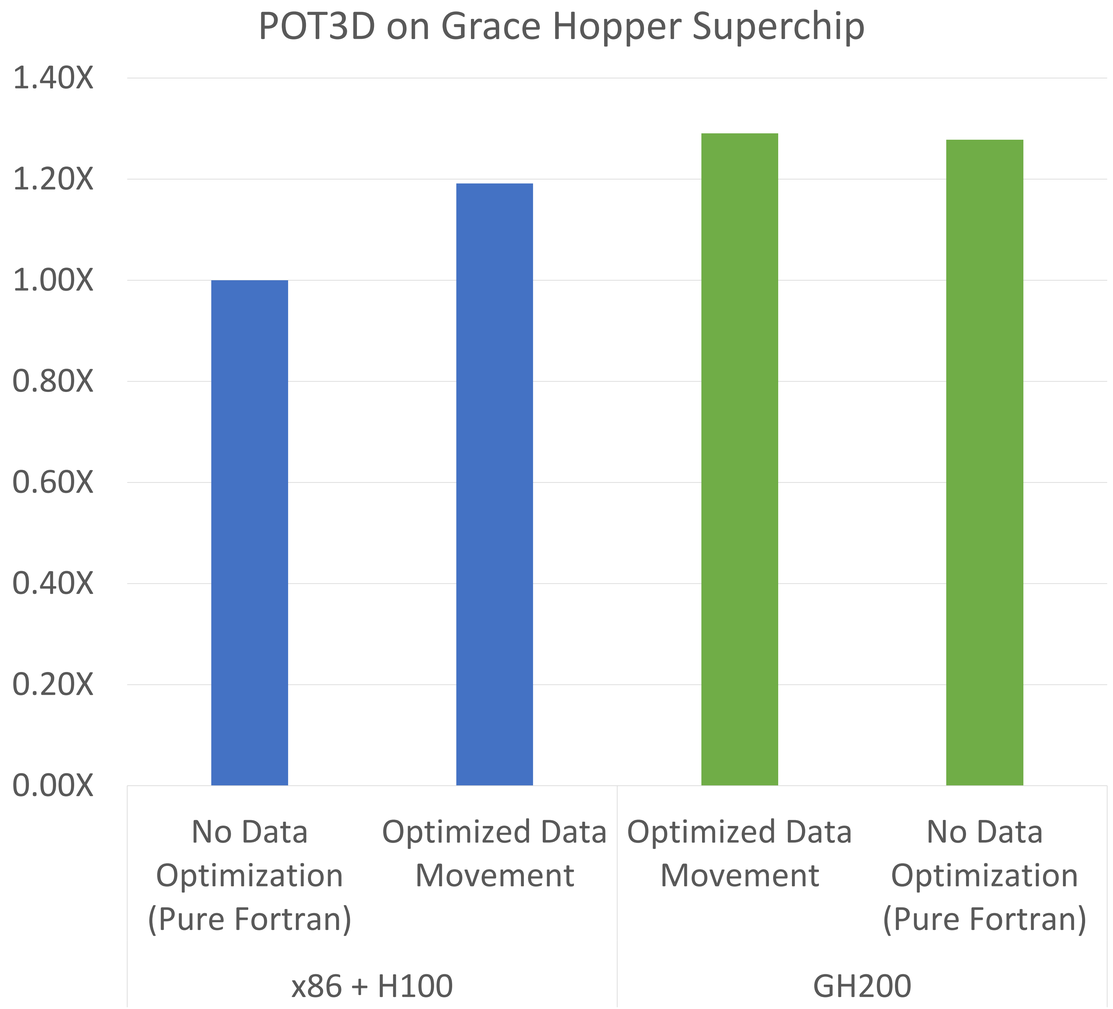
- Existing GPU applications require no changes for Grace Hopper
  - No new APIs
  - No restructuring
  - No new programming model
  - Developers who choose to can optimize for the Grace Hopper platform
- Existing GPU applications (fully or partially ported) will run better on Grace Hopper
  - Data migration no longer required, may still be a performance optimization
  - When data migrations happen, they happen faster due to C2C interconnect
  - CPU code will benefit from higher bandwidth memory, high thread performance, coherent accesses
  - Existing, stable Unified Memory APIs may be used for performance optimization
- Non-GPU applications will run unmodified and benefit from Grace architecture
- Porting from CPU to GPU is made simpler by Grace Hopper
  - Coherent Memory Subsystem
  - C2C interconnect
  - Programming model choice
- Some new capabilities may be unlocked
  - Larger data sets
  - Workflows that utilize both halves

HPC Compilers Support Unified Memory

Unified Memory Enhances Developer Productivity

- The NVIDIA HPC Compilers now support unified memory systems.
  - Grace Hopper
  - Linux HMM on x86
- No need for explicit data management
- Standard Parallelism (StdPar) works for ***all*** memories.
- Composable with CUDA for memory optimization





NVIDIA Performance Libraries

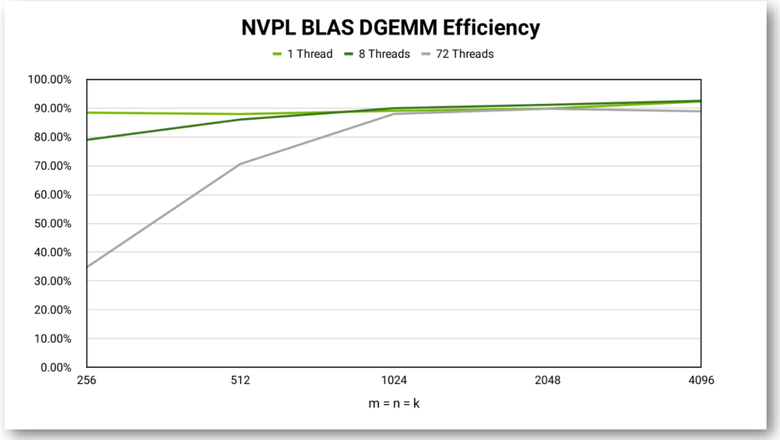
Optimized math libraries for NVIDIA CPUs

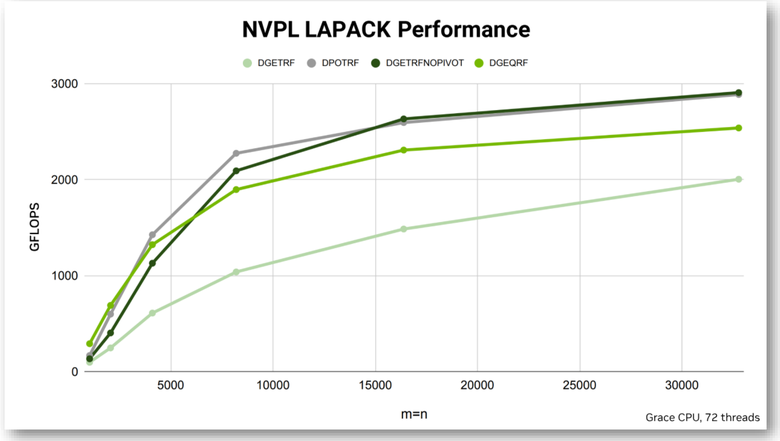
- Easily port applications to NVIDIA's datacenter CPUs
- Drop-in replacement for any math library implementing standard interfaces (e.g. Netlib, FFTW)
- New interfaces for high-performance libraries
- www.developer.nvidia.com/nvpl

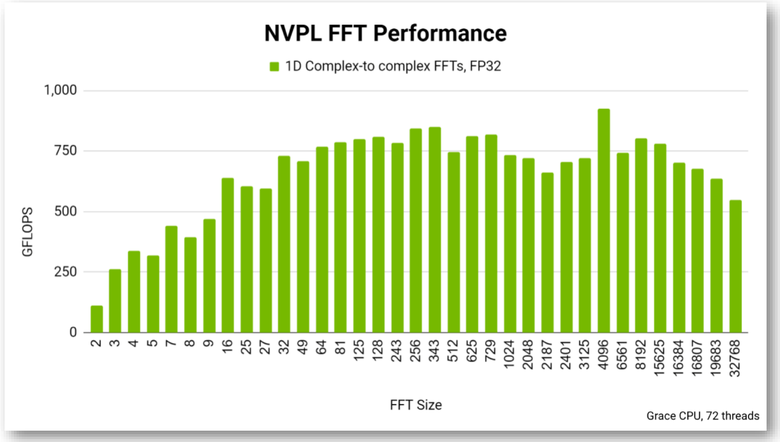
BLAS	LAPACK	PBLAS	SCALAPACK
TENSOR	SPARSE	RAND	FFT

Download Now

www.developer.nvidia.com/nvpl



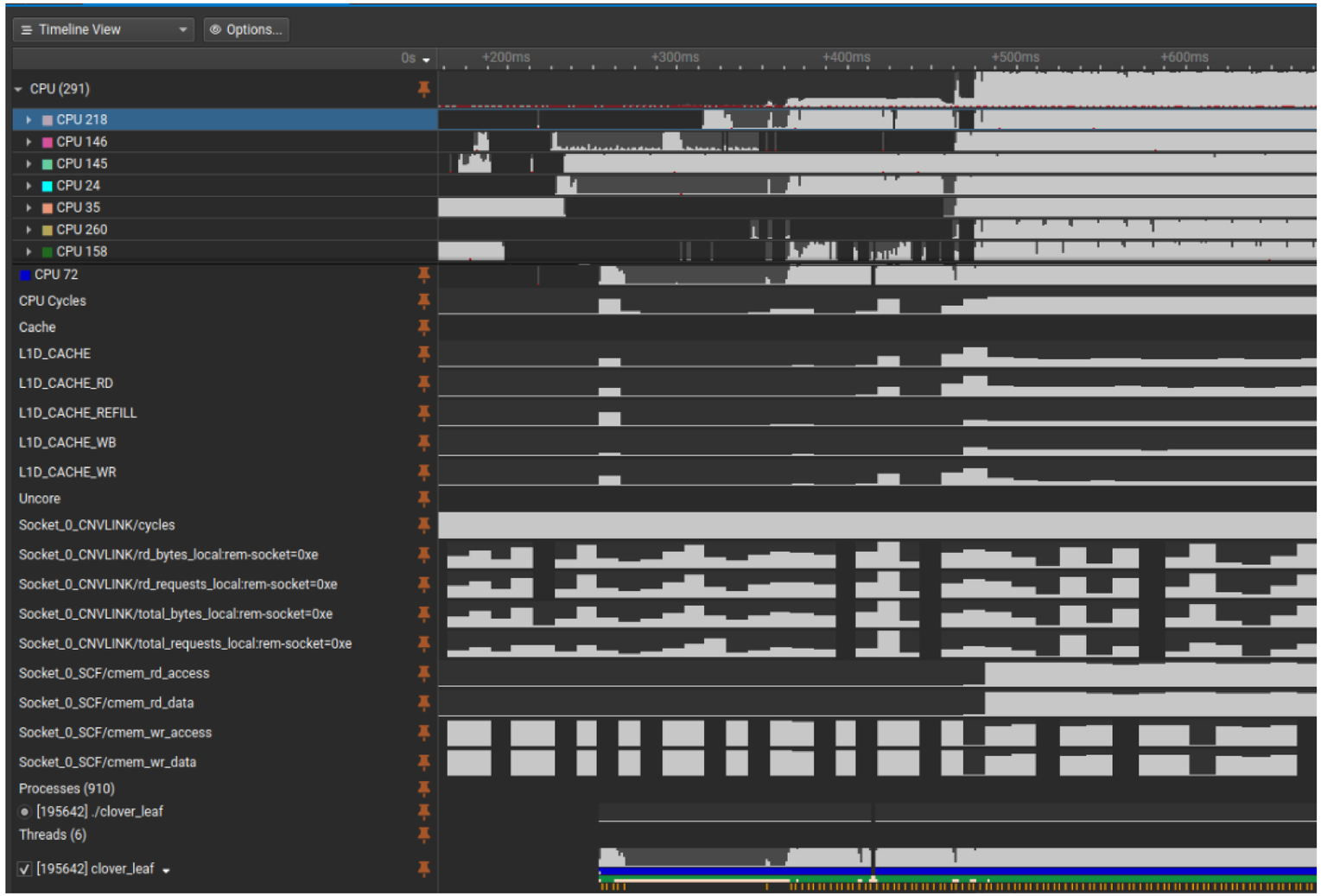




Grace Host Profiling

Hardware Counters and Metrics

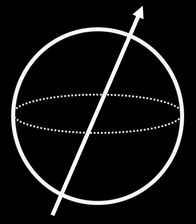
- CPU Core and Uncore Events
  - Sampled for each CPU
  - Visualize parallelism effects
  - Cache hit/miss, instructions retired, etc…
  - L3 Coherency Fabric
  - Socket to socket traffic
- Variable sampling frequencies supported
- Timeline correlated with all other data
  - GPU vs. CPU idle times and metrics
  - Data movement
  - Zoom and filter



Creating the Ecosystem for Quantum Accelerated Supercomputing

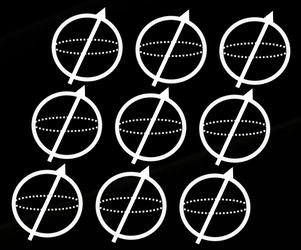
Quantum Challenges

What's Standing Between Today and Useful Quantum Computing?



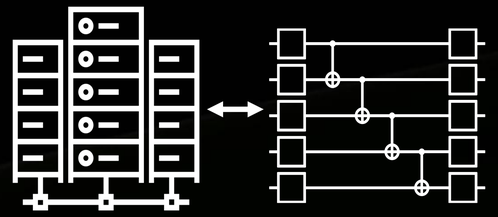
Qubit Fidelity

99.99% 2-Qubit Gate Fidelity



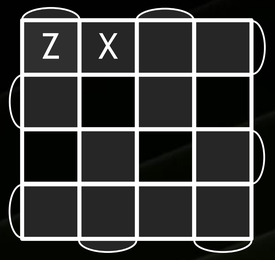
Qubit Scale

100k-1M+ Qubits for FTQC



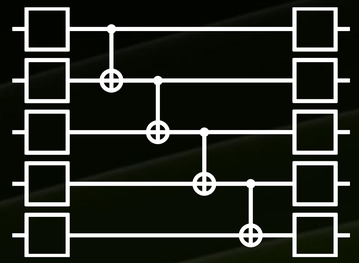
HPC Integration

Sub-Microsecond HPC-QC Latency



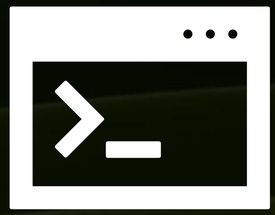
Error Correction

Methods that Scale to Large Quantum Systems



Algorithms

Algorithms with Exponential Speed-up



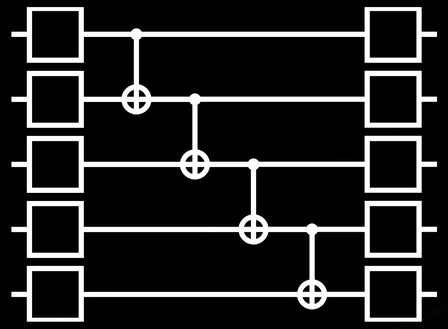
Developer Tools

Integrate with Scientific Computing
Familiar to non-Quantum Physicists

See Defining the Quantum-Accelerated Supercomputer [S62139] for more details

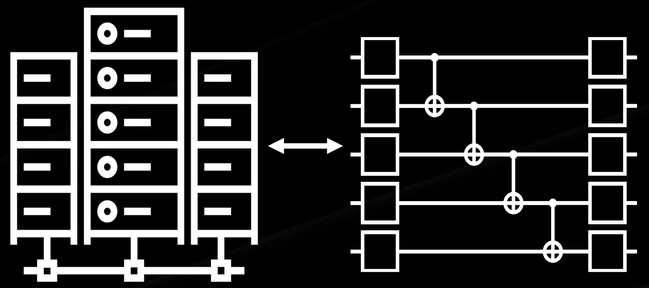
NVIDIA Quantum

Powering the Global Quantum Computing Community



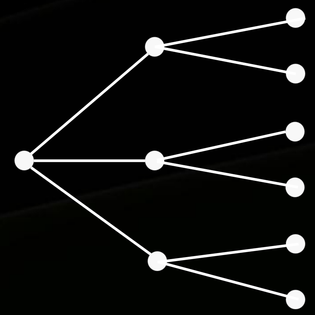
Simulation

Algorithm Design, Resource Estimation, QPU Design



HPC Quantum Integration

Integrated Applications, QEC, Sub-Microsecond Latency



AI for Quantum

QEC, Calibration, Algorithms

CUDA-Q

Libraries | Programming Model | Tools | Infrastructure

cuQuantum

Quantum Simulation

DGX-Q

Quantum Integrated Computing

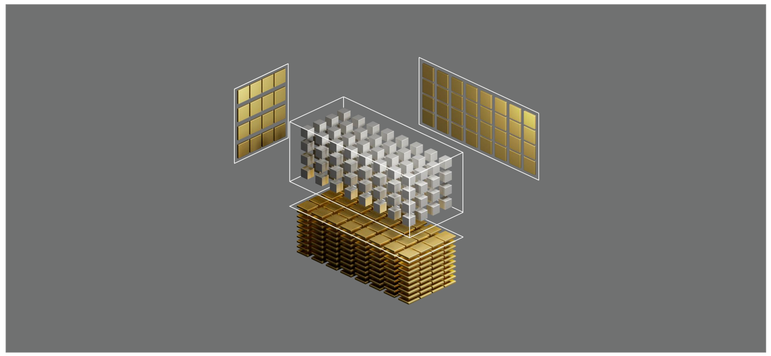
GPU Supercomputing

NVIDIA

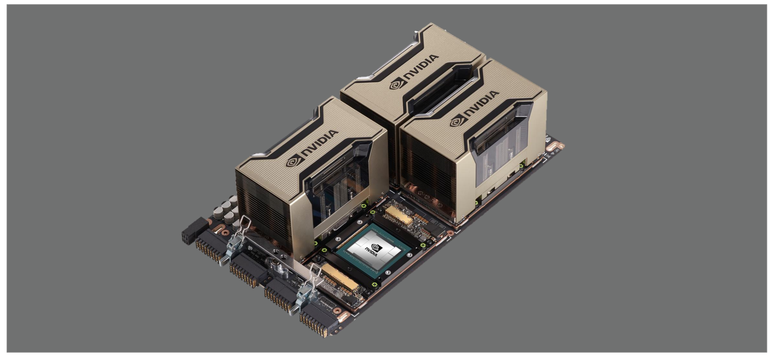
Takeaways

NVIDIA HPC Software

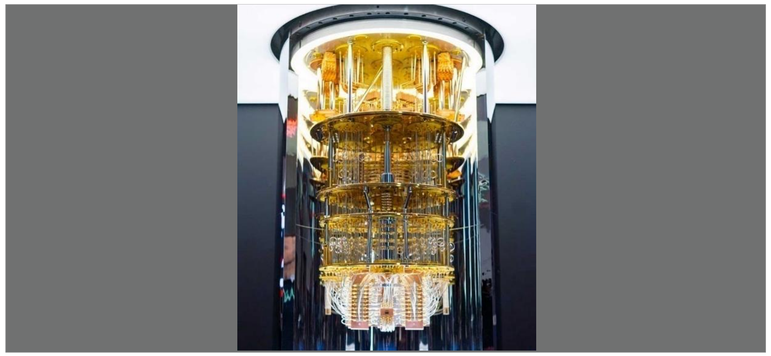
Takeaways



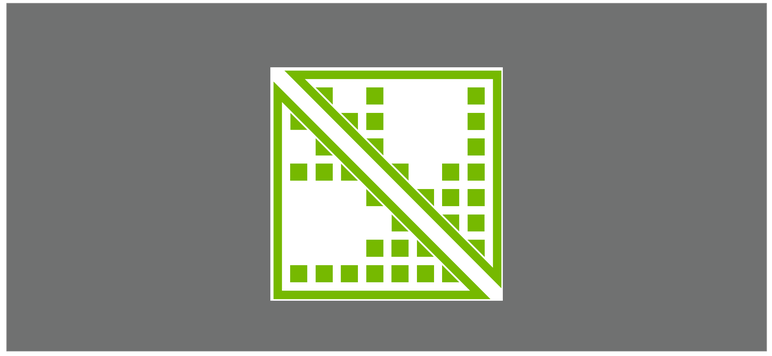
Standard languages and the HPC SDK make accelerated computing seamless



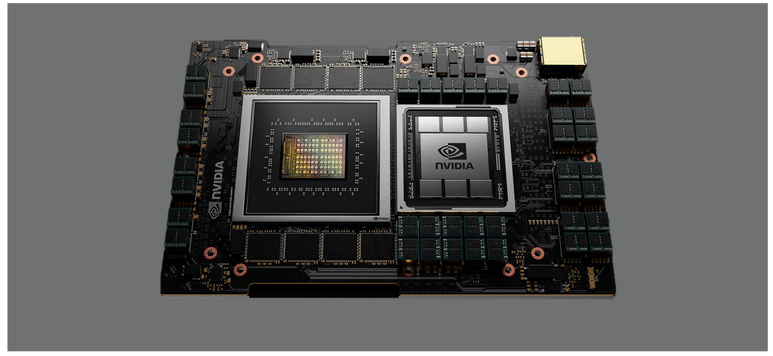
Multi-node, multi-GPU libraries enables science at breakthrough scale



NVIDIA Quantum enables Quantum Accelerated Supercomputing



NVIDIA Math Libraries are GPU accelerated with new features and improved performance



NVIDIA HPC SDK and NVPL make accelerated computing more productive on Grace

New libraries and tools support Python as a first-class language for accelerated computing

More Talks at GTC

- Demystify CUDA Debugging and Performance with Powerful Developer Tools [S62256]
- Deep Dive into NVIDIA Math Libraries [S62162]
- GPU-Accelerating Process Simulation Performance using NVIDIA’s cuDSS Sparse Linear Systems Solver [S62515]
- Defining the Quantum Accelerated Supercomputer [S62139]
- No More Porting: Accelerated Computing With Standard C++, Fortran, and Python [S61204]
- Scientific Computing With NVIDIA Grace and the Arm Software Ecosystem [S61598]
- Accelerating Scientific Workflows With the NVIDIA Grace Hopper Platform [S62337]
- Legate: A Productive Programming Framework for Composable, Scalable, Accelerated Libraries [S62262]
- CUDA: New Features and Beyond [S62400]