

A Deep Dive into NVIDIA HPC Software

Jeff Larkin, NVIDIA | SC23 Exhibitor Forum

Challenges Facing HPC Developers



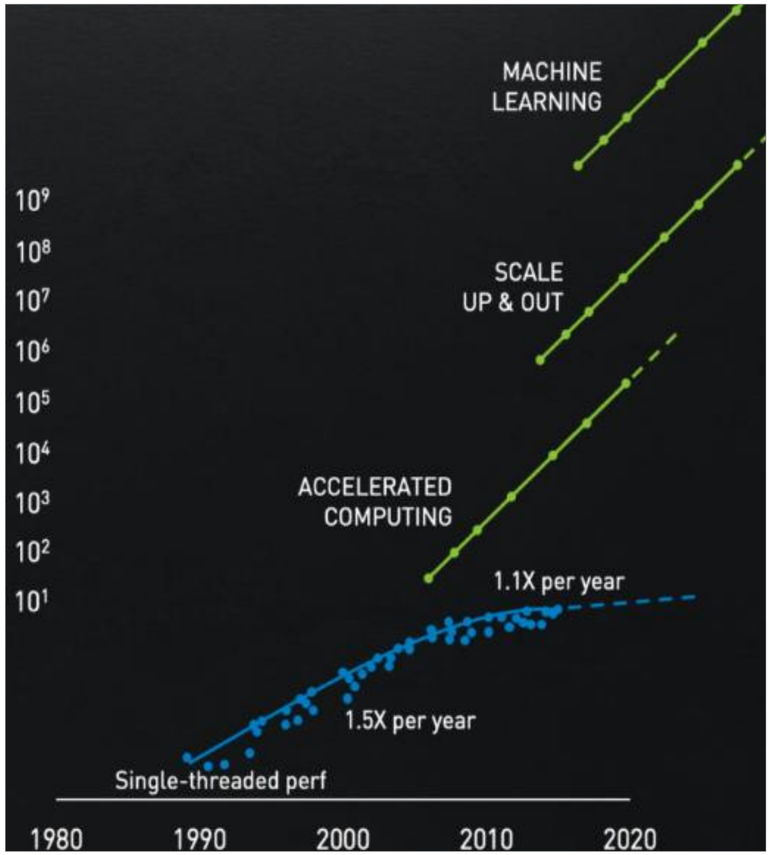
Productivity

Programmers must productively develop and maintain software for diverse environments



Performance

High performance should be within reach for domain experts

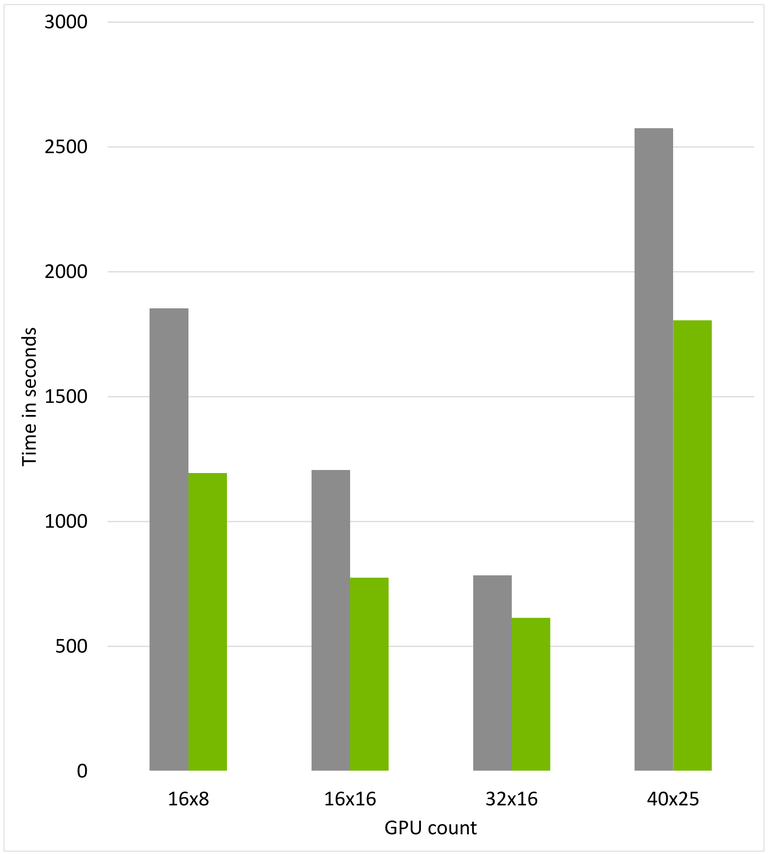


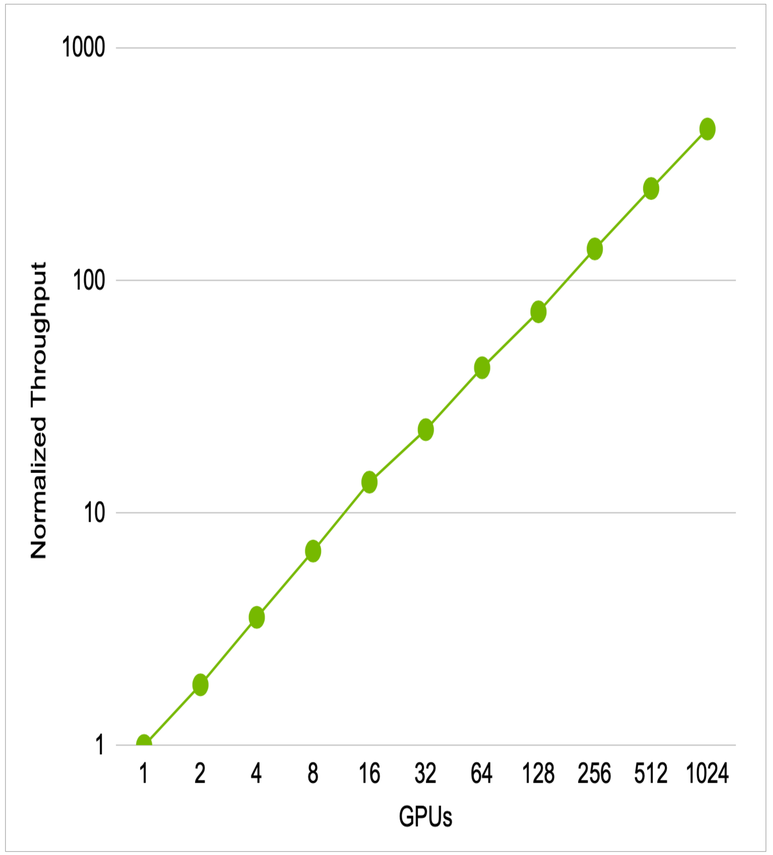
Scalability

Applications must scale to new heights

NVIDIA Software Solves HPC Challenges







Productivity

Standards-based Parallelism (StdPar) make developers more productive

Performance

Libraries and tools make performance achievable

Scalability

Scale-up libraries and tools enable more science

Programming the NVIDIA Platform

Unmatched Developer Flexibility

Languages & Programming Models	Accelerated Standard Languages	Incremental Optimization	Platform Specialization
	C++, python™, 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

Accelerated Standard Languages

Parallel performance for wherever your code runs

ISO C++

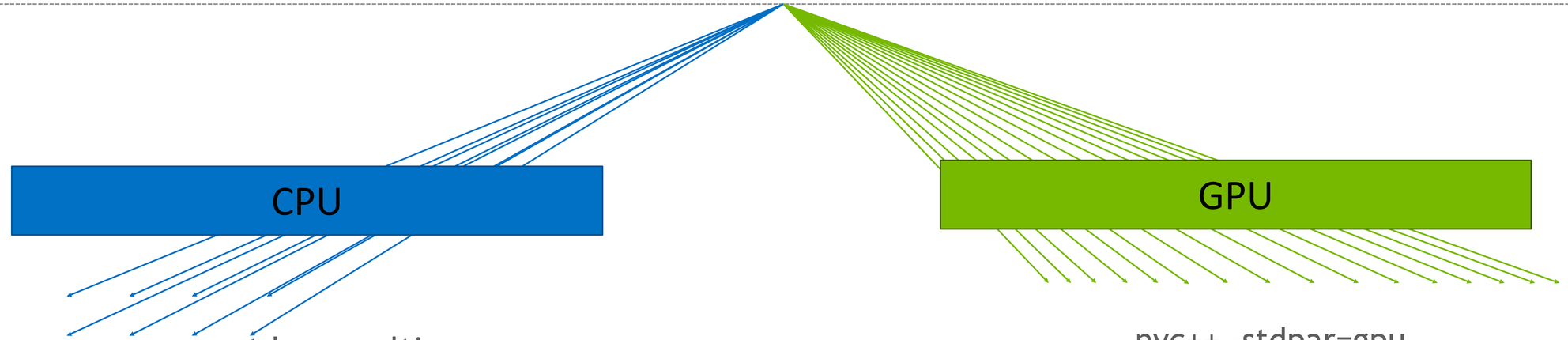
```
std::transform(par, x, x+n, y,
   y,[=](float x, float y){
      return y + a*x;
    }
);
```

ISO Fortran

```
do concurrent (i = 1:n)
   y(i) = y(i) + a*x(i)
enddo
```

Python

```
import cunumeric as np
…
def saxpy(a, x, y):
    y[:] += a*x
```



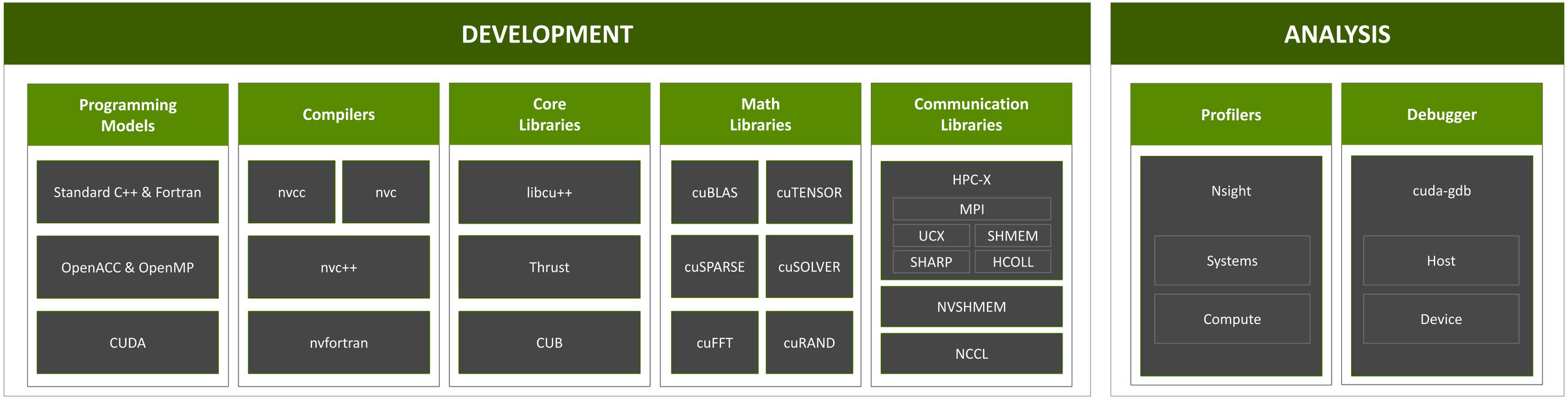
nvc++ -stdpar=multicore
nvfortran -stdpar=multicore
legate -cpus 16 saxpy.py

nvc++ -stdpar=gpu
nvfortran -stdpar=gpu
legate -gpus 1 saxpy.py

NVIDIA.

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

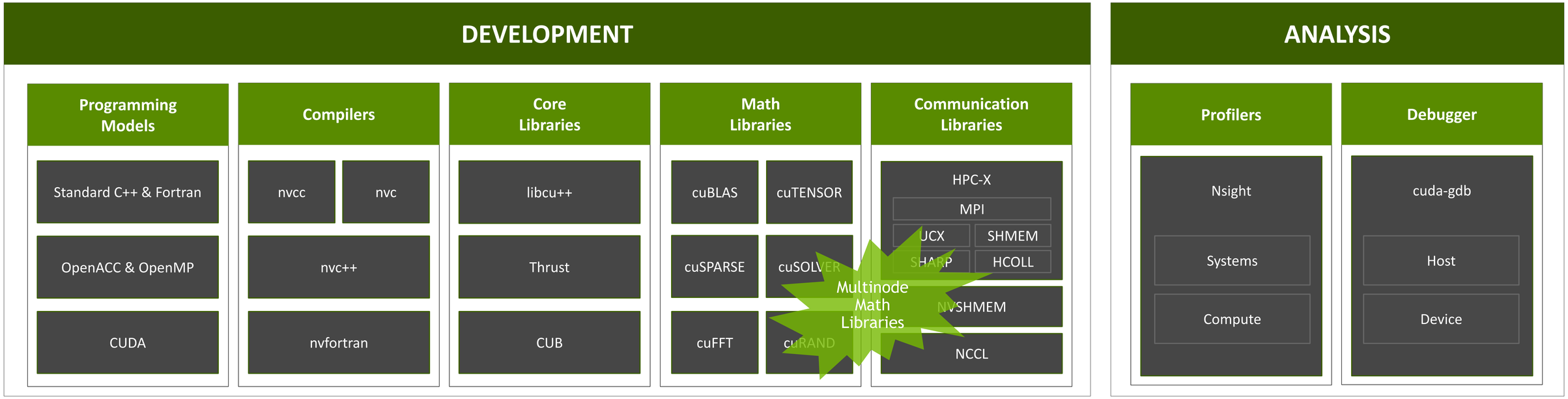
Libraries | Accelerated C++ and Fortran | Directives | CUDA

x86_64 | Arm | OpenPOWER

7-8 Releases Per Year | Freely Available

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 | OpenPOWER

7-8 Releases Per Year | Freely Available

HPC SDK 23.11

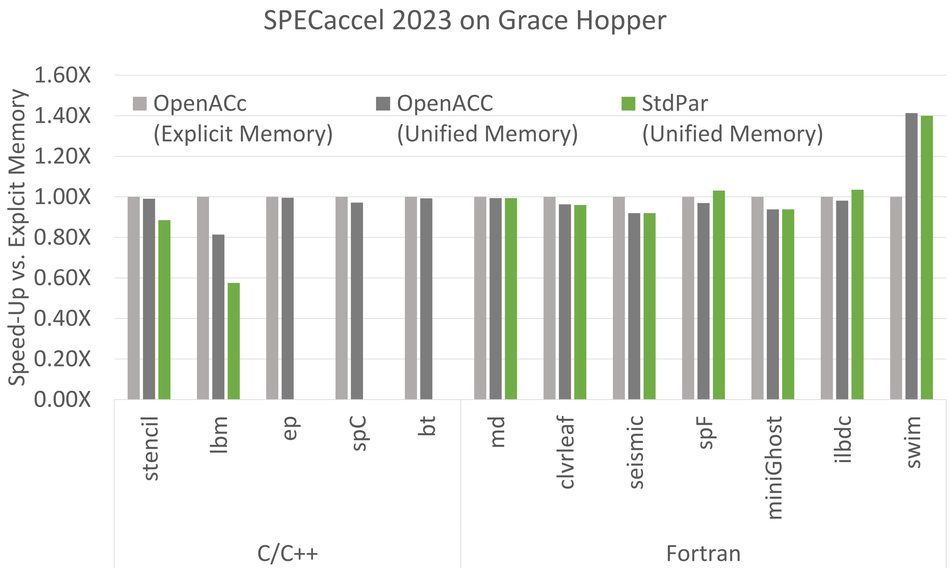
Grace Hopper, unified memory, and more

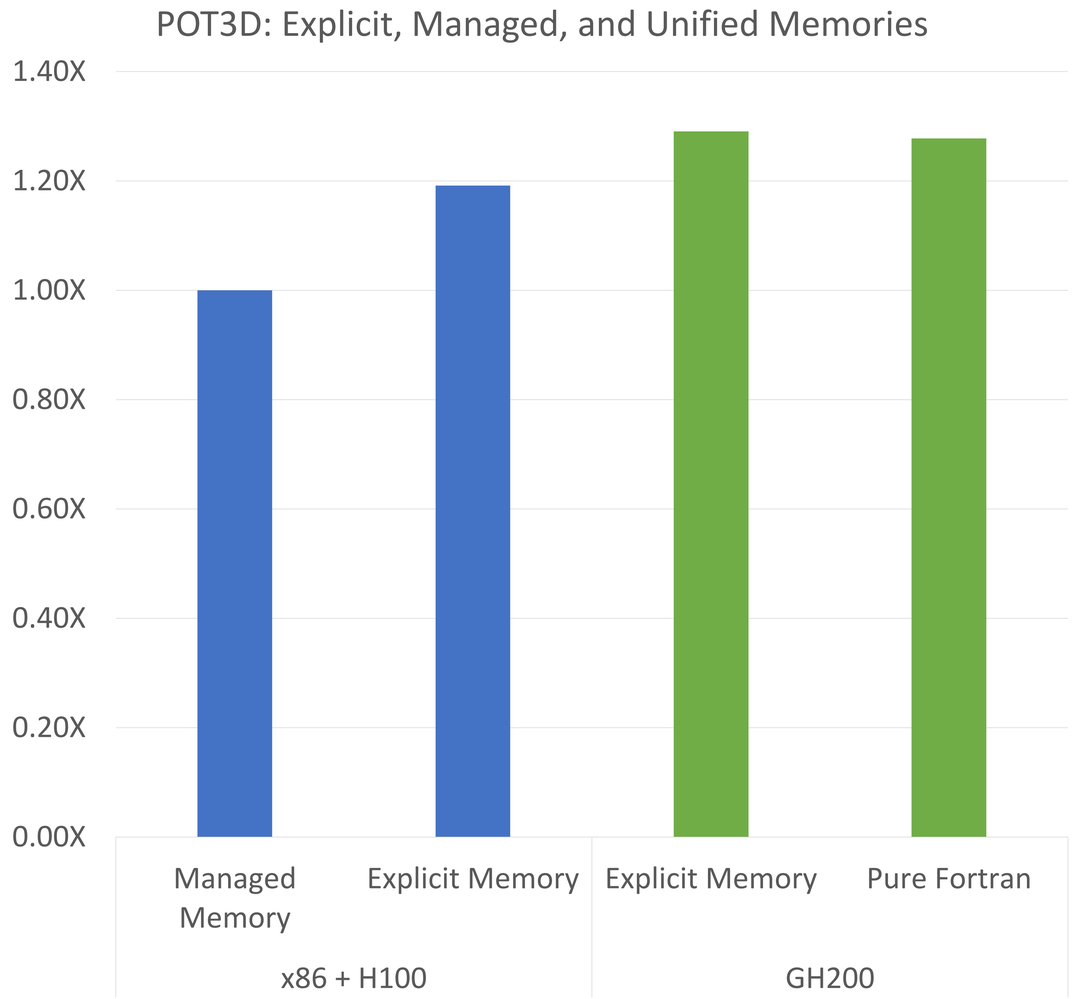
- **HPC SDK 23.11:**
  - NVTX improvements for stdpar codes
    - Now you can see your stdpar in NSight: improved tools support, developer experience, performance optimizations
  - C-Fortran Interface
    - Better multi-paradigm interoperability for mixed C, C++, and Fortran codes
    - F2008 MPI bindings 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
  - Unified memory support for stdpar, OpenACC, and CUDA C++/Fortran

- **Unified memory:**
  - C++ stdpar improvements
  - Fortran stdpar improvements
  - OpenACC improvements
  - CUDA Fortran
  - Unified Functions
  - Launch Blog Post

HPC Compilers Support Unified Memory

- 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





```
static inline
void CalcHydroConstraintForElems(Domain &domain, Index_t length,
    Index_t *regElemlist, Real_t dvovmax, Real_t& dthydro)
{
#if _OPENMP
  const Index_t threads = omp_get_max_threads();
  Index_t hydro_elem_per_thread[threads];
  Real_t dthydro_per_thread[threads];
#else
  Index_t threads = 1;
  Index_t hydro_elem_per_thread[1];
  Real_t dthydro_per_thread[1];
#endif
#pragma omp parallel firstprivate(length, dvovmax)
  {
    Real_t dthydro_tmp = dthydro ;
    Index_t hydro_elem = -1 ;
#if _OPENMP
    Index_t thread_num = omp_get_thread_num();
#else
    Index_t thread_num = 0;
#endif
#pragma omp for
    for (Index_t i = 0 ; i < length ; ++i) {
      Index_t indx = regElemlist[i] ;

      if (domain.vdov(indx) != Real_t(0.)) {
        Real_t dtdvov = dvovmax / (FABS(domain.vdov(indx))+Real_t(1.e-20)) ;

        if ( dthydro_tmp > dtdvov ) {
          dthydro_tmp = dtdvov ;
          hydro_elem = indx ;
        }
      }
    }
    dthydro_per_thread[thread_num] = dthydro_tmp ;
    hydro_elem_per_thread[thread_num] = hydro_elem ;
  }
  for (Index_t i = 1; i < threads; ++i) {
    if(dthydro_per_thread[i] < dthydro_per_thread[0]) {
      dthydro_per_thread[0] = dthydro_per_thread[i];
      hydro_elem_per_thread[0] = hydro_elem_per_thread[i];
    }
  }
  if (hydro_elem_per_thread[0] != -1) {
    dthydro = dthydro_per_thread[0] ;
  }
  return ;
}
```

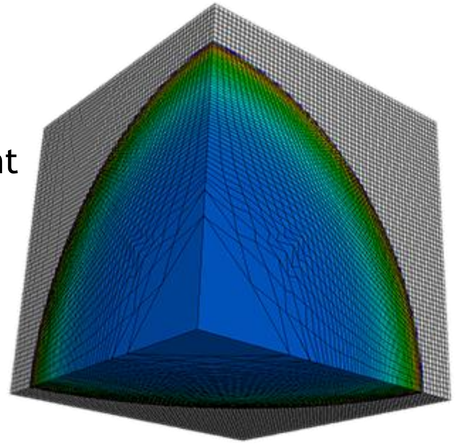
C++ with OpenMP

Lulesh with Standard C++

C++ Hydrodynamics Mini-app from LLNL

Rewritten from OpenMP to ISO C++

- More composable, compact, and elegant
- Easier to read and maintain
- ISO Standard
- Portable – nvc++, g++, icpc, MSVC, …



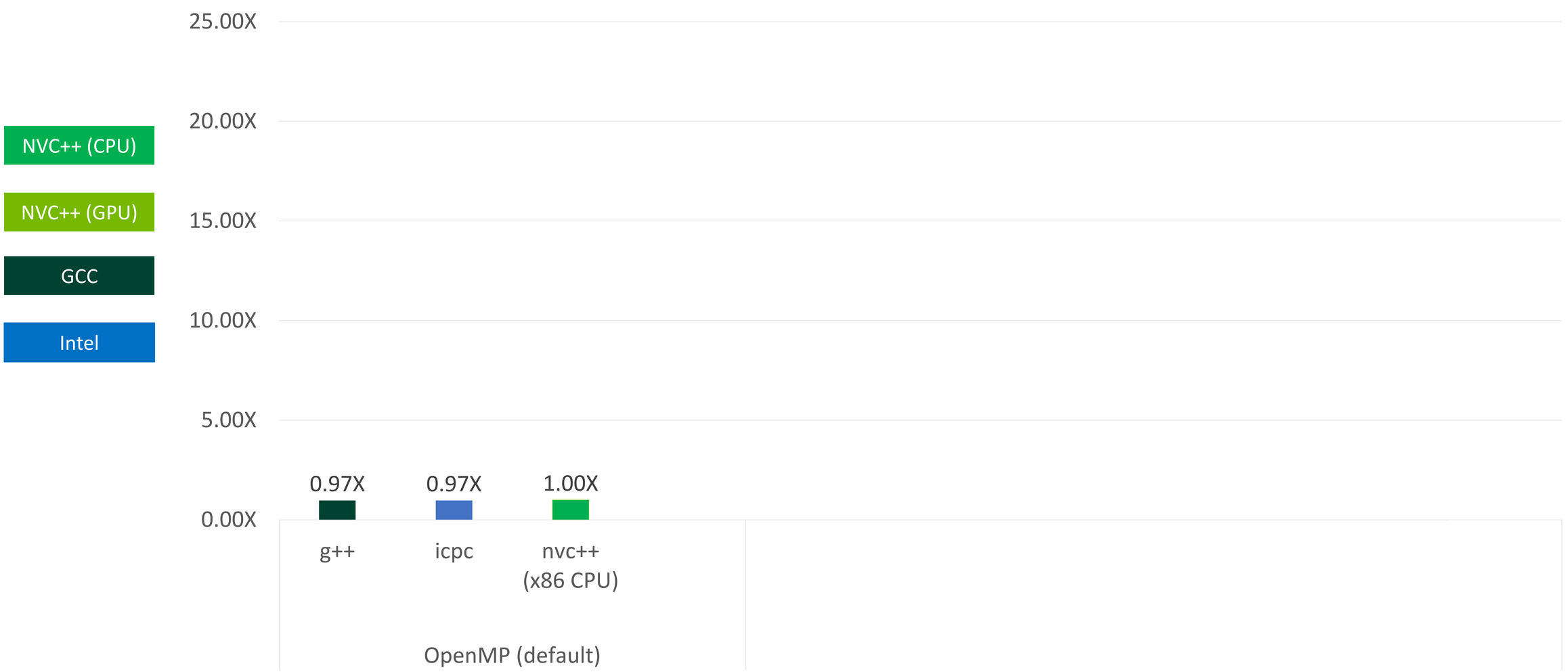
```
static inline void CalcHydroConstraintForElems(Domain &domain, Index_t length,
                Index_t *regElemlist,
                Real_t dvovmax,
                Real_t &dthydro)
{
  dthydro = std::transform_reduce(
    std::execution::par, counting_iterator(0), counting_iterator(length),
    dthydro, [](Real_t a, Real_t b) { return a < b ? a : b; },
    [=, &domain](Index_t i)
  {
      Index_t indx = regElemlist[i];
      if (domain.vdov(indx) == Real_t(0.0)) {
        return std::numeric_limits<Real_t>::max();
      } else {
        return dvovmax / (std::abs(domain.vdov(indx)) + Real_t(1.e-20));
      }
  });
}
```

Standard C++

codesign.llnl.gov/lulesh

NVIDIA.

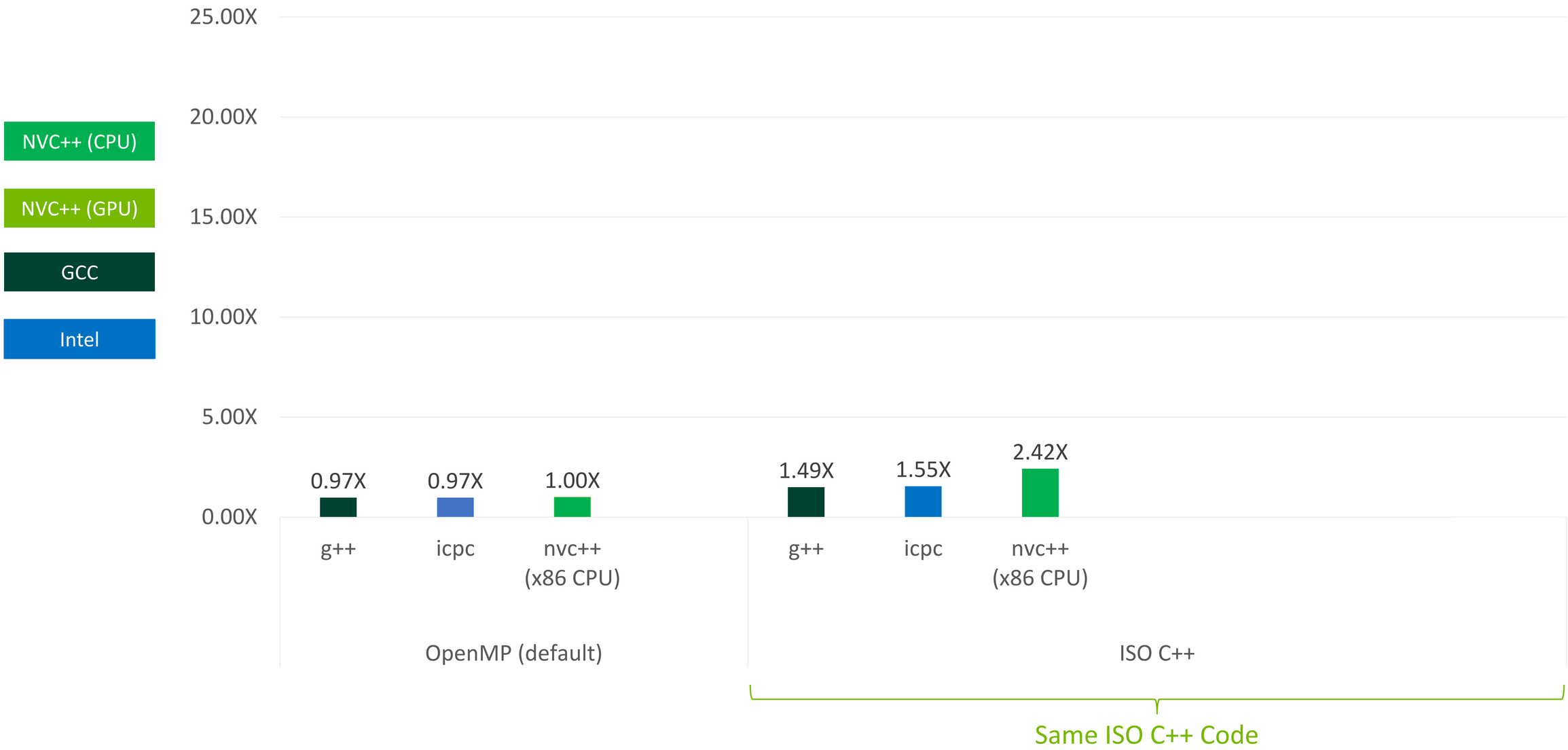
C++ Standard Parallelism



AMD EPYC 7742 CPU (64 cores), NVIDIA A100 and H100 GPUs. g++ version 10.3.0, icpc version 2021.5.0, nvc++ version 23.7

C++ Standard Parallelism

Lulesh Speed-up

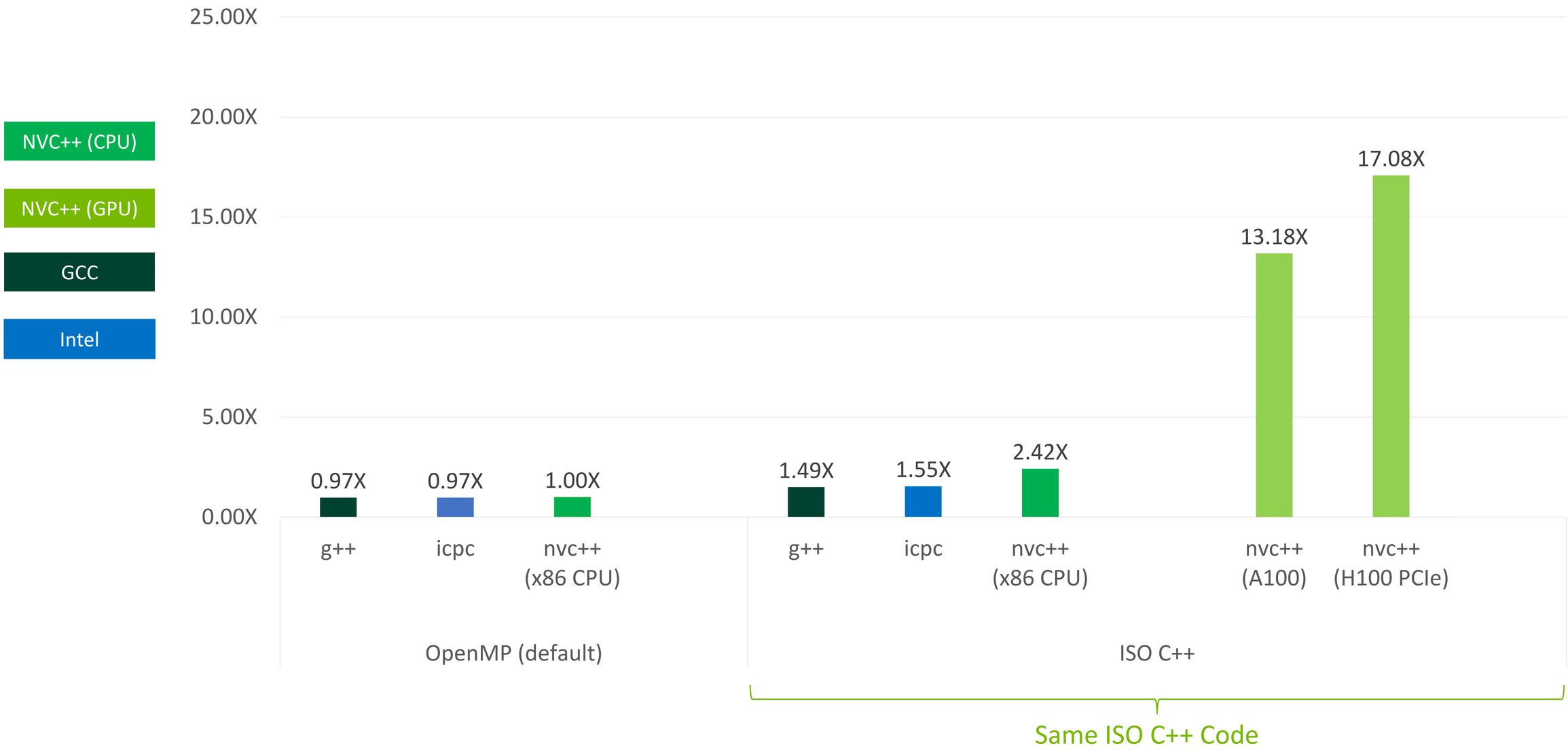


AMD EPYC 7742 CPU (64 cores), NVIDIA A100 and H100 GPUs. g++ version 10.3.0, icpc version 2021.5.0, nvc++ version 23.7

NVIDIA

C++ Standard Parallelism

Lulesh Speed-up

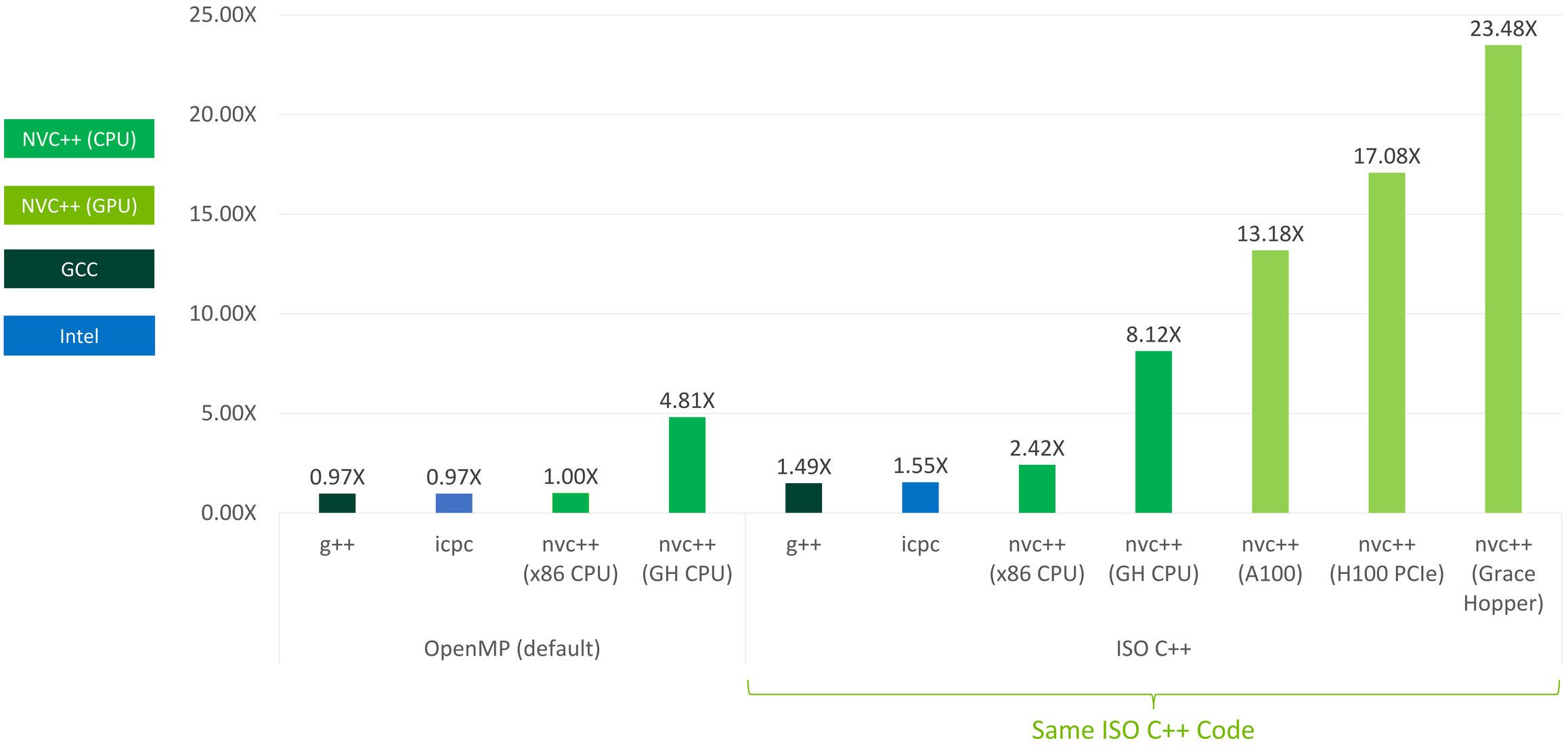


AMD EPYC 7742 CPU (64 cores), NVIDIA A100 and H100 GPUs. g++ version 10.3.0, icpc version 2021.5.0, nvc++ version 23.7

NVIDIA

C++ Standard Parallelism

Lulesh Speed-Up



AMD EPYC 7742 CPU (64 cores), NVIDIA A100 GPU. g++ version 10.3.0, icpc version 2021.5.0, nvc++ version 23.7

NVIDIA.

POT3D: GPU Computing in Pure Fortran

POT3D

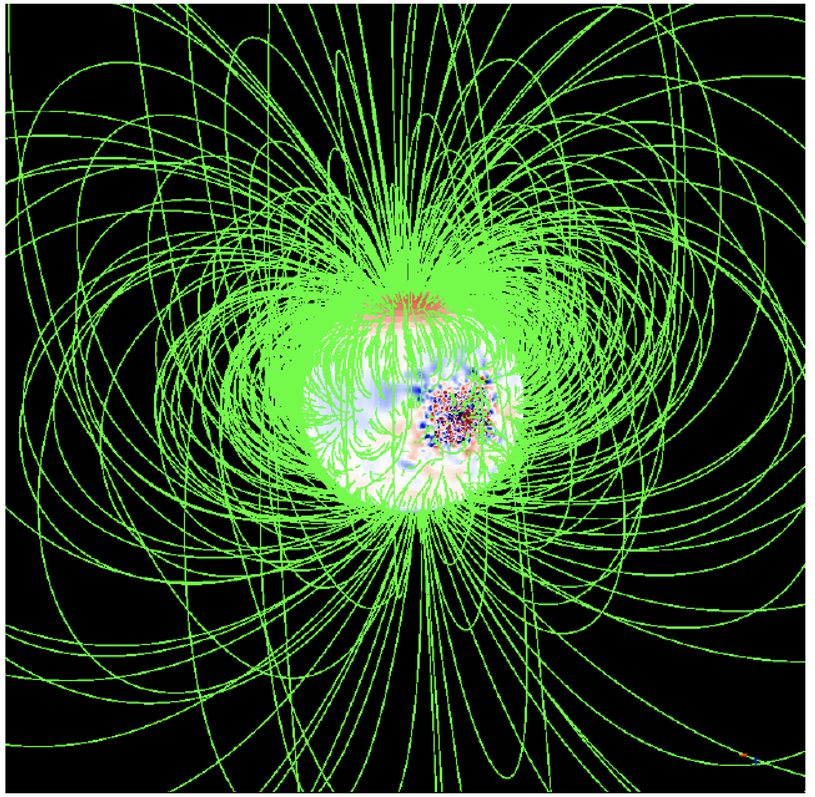
POT3D is a Fortran application for approximating solar coronal magnetic fields.

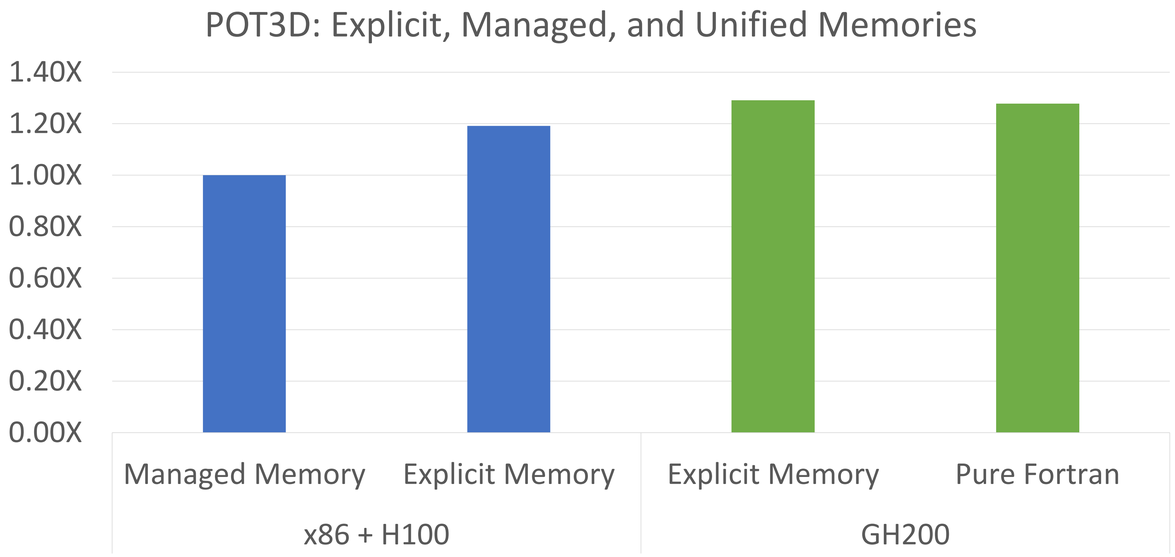
Included in the SPEChpc benchmark suite*

Existing parallelization in MPI & OpenACC

Previous experiments showed a 10% gap between pure Fortran and OpenACC, but Grace Hopper enables the same performance in pure Fortran as with tuned OpenACC.

https://github.com/predsci/POT3D



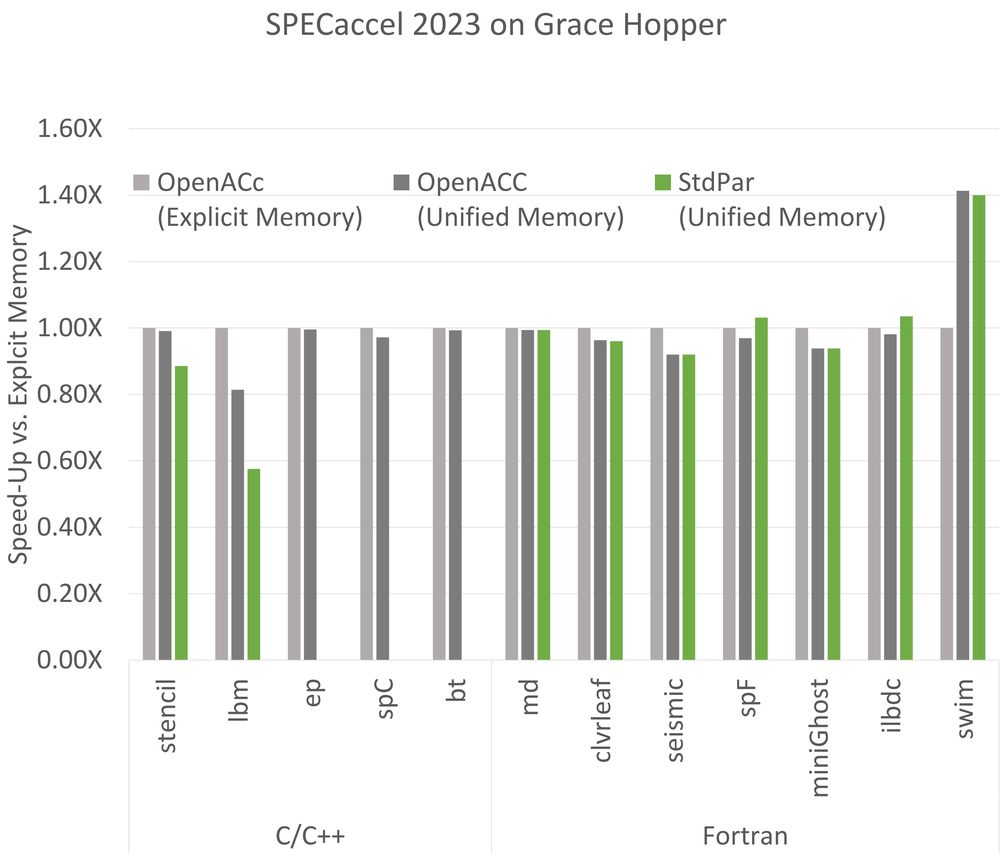


X86+H100: NVHPC 23.9, AMD EPYC 9654 + NVIDIA H100-80G HBM3 PCIe
GH200: NVHPC 23.11, NVIDIA Grace Hopper Superchip

Thank you to Predictive Science Inc.
*SPEChpc is a trademark of The Standard Performance Evaluation Corporation

SPECaccel® 2023 on Grace Hopper

- SPECaccel® is an industry-accepted benchmark suite that now includes multiple directive-based benchmarks
  - NVIDIA has additionally created language-based versions of several benchmarks for consideration.
- Standard parallelism is generally competitive to tuned-OpenACC.
- With Grace Hopper's Unified Memory, performance is generally on-par with hand-tuned data movement
- The results shown here include C++ stdpar, Fortran stdpar, and OpenACC for both C++/Fortran
- 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 October 2023 on a pre-production MGX-GH200 system using version 23.11 of the NVHPC SDK. The STDPAR versions are internal ports of the benchmarks and not part of SPECaccel 2023.

Accelerating STDPAR Adoption



abinit



BabelStream

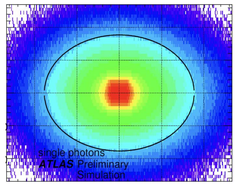
BabelStream



Cloverleaf

DIFFUSE

Diffuse



FastCaloSim

GAMESS

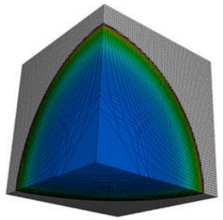
GAMESS

HipFT

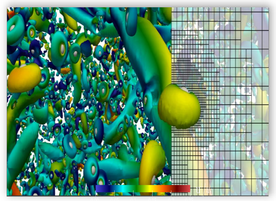
HipFT

JAEA

JAEA MiniApps



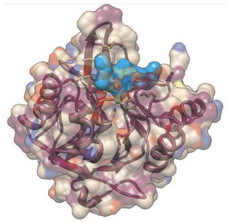
Lulesh



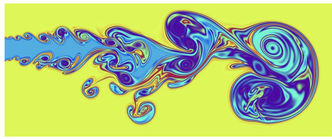
M-AIA

MAS

MAS



miniBUDE



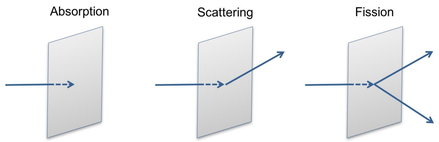
MiniWeather



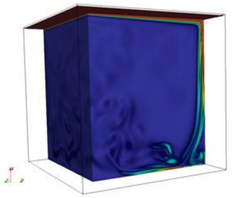
Palabos

POT3D

POT3D



Quicksilver



STLBM

RAJA

RAJAPerf

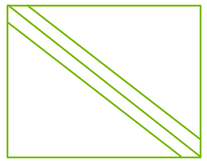
NVIDIA.

NVIDIA Math Libraries

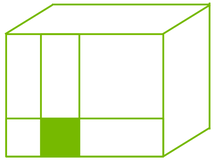
Linear Algebra, FFT, RNG, and Basic Math



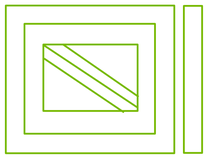
cuBLAS



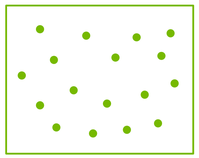
cuSPARSE



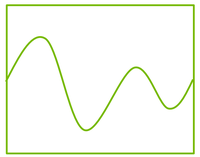
cuTENSOR



AMGX



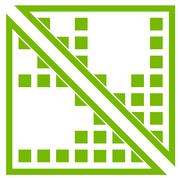
cuRAND



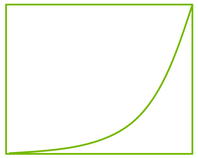
cuFFT



cuSOLVER



cuDSS



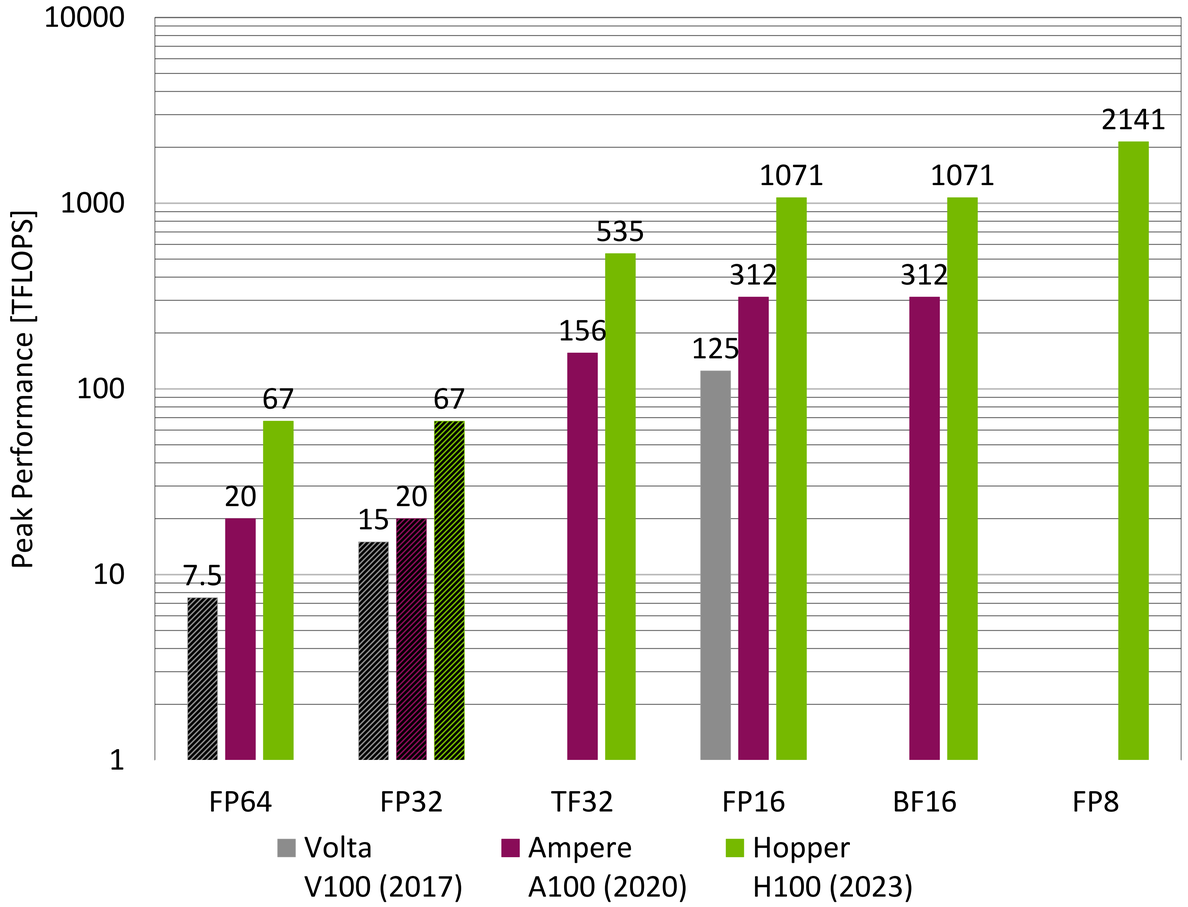
Math API

NVIDIA.

Seamless Mixed-Precision Computations

Leverage the increasing throughput of Tensor Cores

- Tensor Cores throughput is increasing with each new generation of GPU architecture



FMA performance shown for comparison (All FP32 and FP64 on Volta V100)

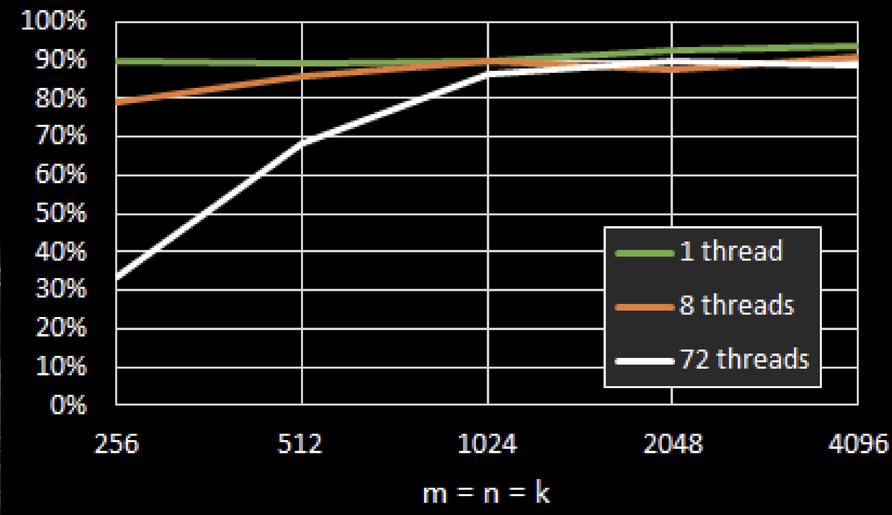
- Mixed-precision algorithms offer significant boost in performance
  - Examples:
    - Iterative refinement solver in cuSOLVERDn
  - The HPL-MxP TOP500 benchmark
  - However, many methods are not universally applicable
- There is a need for drop-in replacements for standard APIs
- We are developing Tensor Core accelerated single-precision GEMMs and tensor contractions [1]
  - Meet or exceed the numerical accuracy of IEEE754 FMA based implementations
  - Demonstrate accuracy claims through:
    - error analysis
    - scientific applications
  - Deliver significant performance benefits as well as:
    - Being more cost effective (higher effective FLOPS/Watt)
    - Reduced environmental impact

[1] https://icl.utk.edu/~luszczek/conf/2023/h3/Bayraktar-H3-Workshop-Keynote.pdf

NVIDIA Performance Libraries (NVPL)

Math Libraries Optimized for Arm CPUs

- Enable easy porting of HPC applications to NVIDIA Grace CPU based platforms to achieve industry leading performance and efficiency
  - Standard interfaces (e.g., BLAS, FFTW)
  - New interfaces (e.g., SPARSE, TENSOR)
- Public Beta release planned for Q4’2023



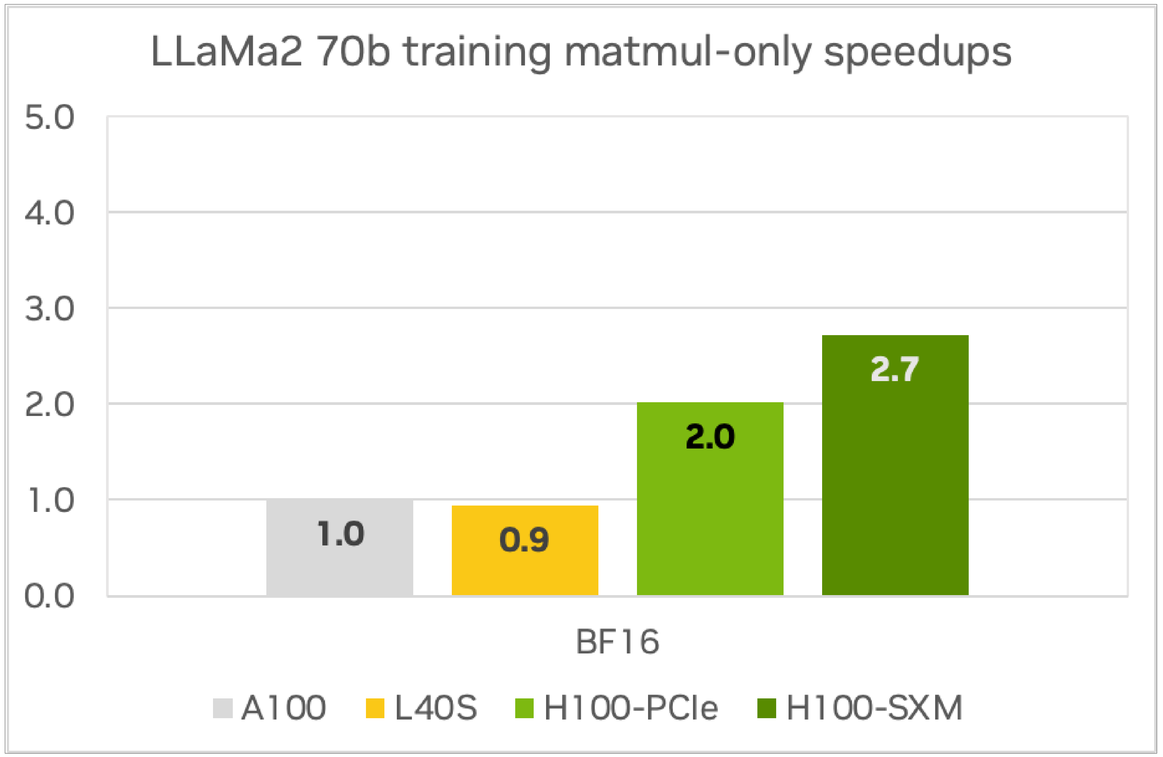


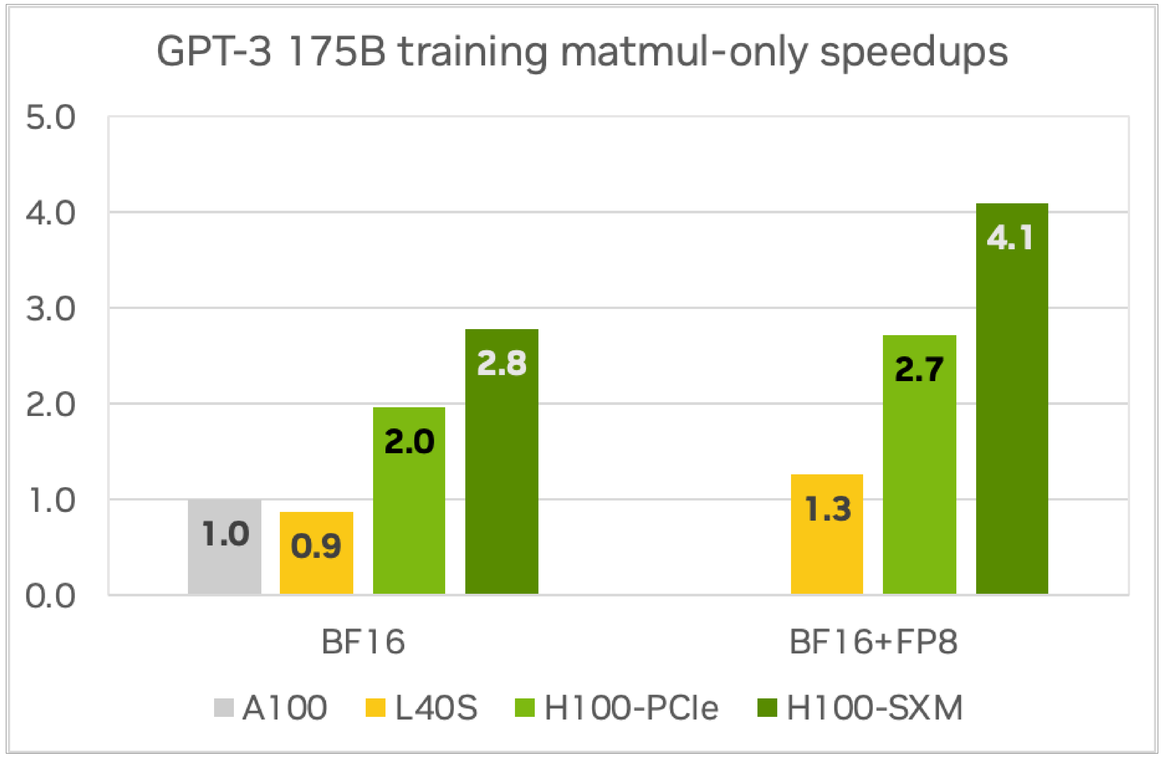
NVIDIA Grace Superchip
144-core Arm Neoverse V2 cores in a Superchip
NVLink-C2C, 900 GB/s Bi-Directional BW
LPDDR5X With ECC, 1TB/s Memory BW

BLAS	LAPACK	PBLAS	SCALAPACK
TENSOR	SPARSE	RNG	FFTW

cuBLAS LLM Matmul Performance Highlights

Comparing NVIDIA H100 and L40S to A100





cuBLAS improves performance for L40S as part of CUDA 12.3

cuDSS - A GPU accelerated Direct Sparse Solver

In Public Early Access Soon

Sparse matrices with few, O(1), non-zero elements

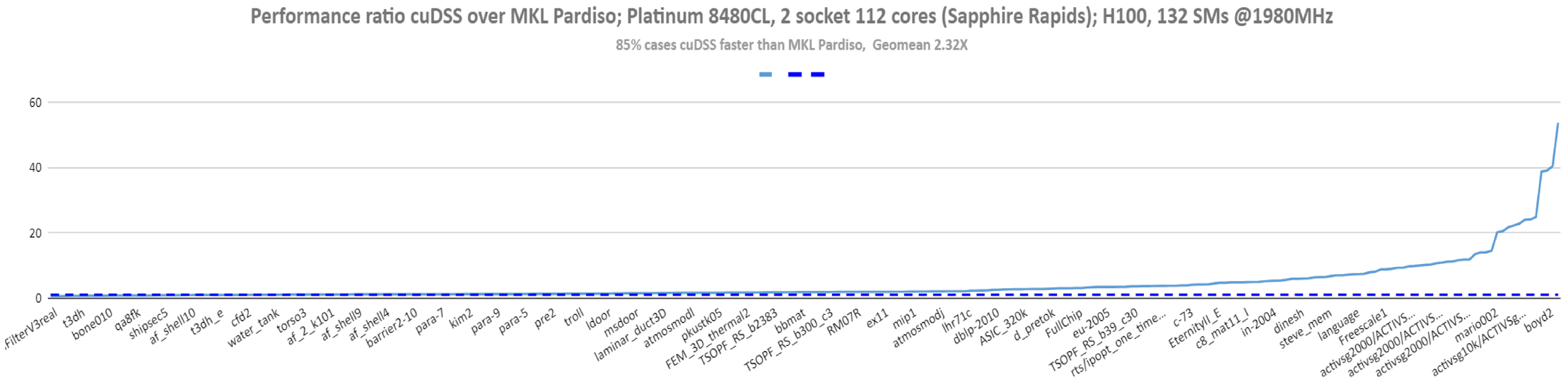
5x to 37x performance boost depending on use cases

Single GPU support for symmetric/non-symmetric, Hermitian and non-Hermitian matrices. Multi-node multi-GPU coming later.

More robust and numerically stable than iterative solver

Use cases similar to circuit simulation – real-time mapping for drive, process simulation can benefit

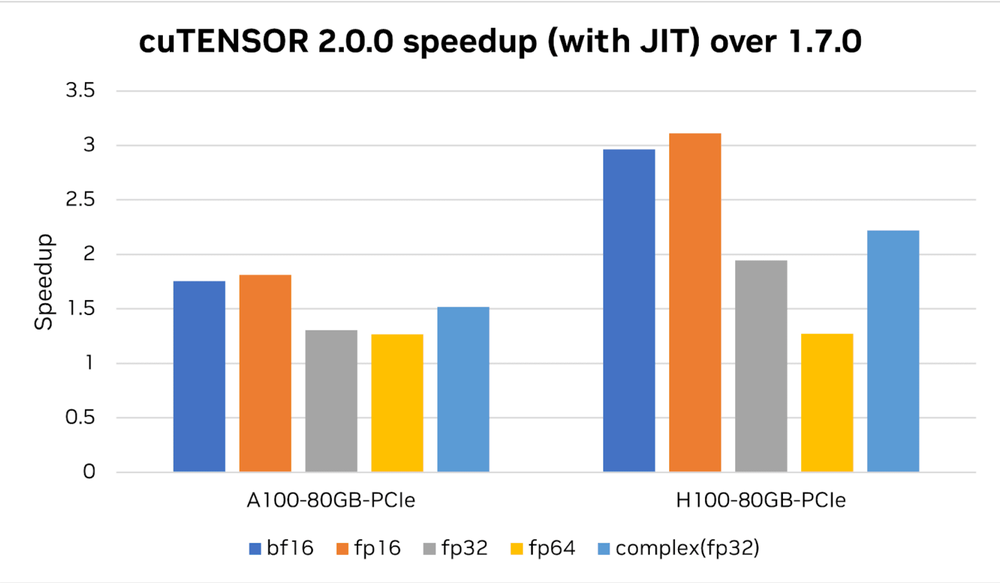
Early adopter Honeywell, for use in their UniSim Design process simulation product

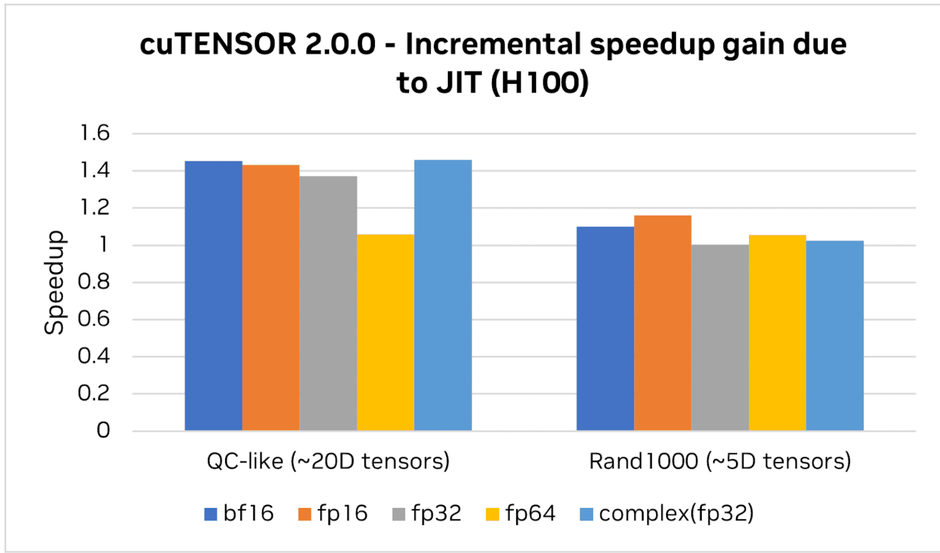


cuTensor 2.0

Performance and extensibility

- cuTENSOR 2.0 is introducing new APIs with ABI changes
- Flexibile, op-agnostic APIs and heap allocated opaque data structures ease sharing features and improvements across ops
- New Plan based APIs extends capability to all ops
- Dynamically allocated descriptors to support for arbitrarily dimensional tensors
- Introducing JIT kernel: Compose kernels for the target config for best performance for especially high dimensional tensors (See: QC-like)
- CUDA Toolkit 10.2 support removed; RHEL 7 support removed.

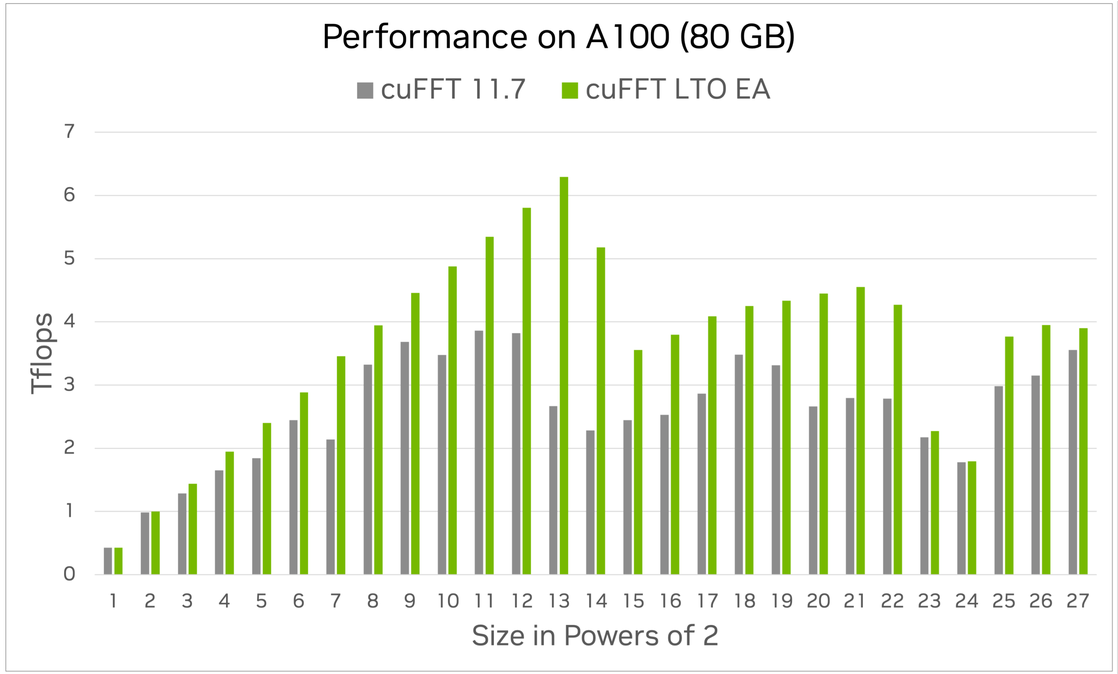


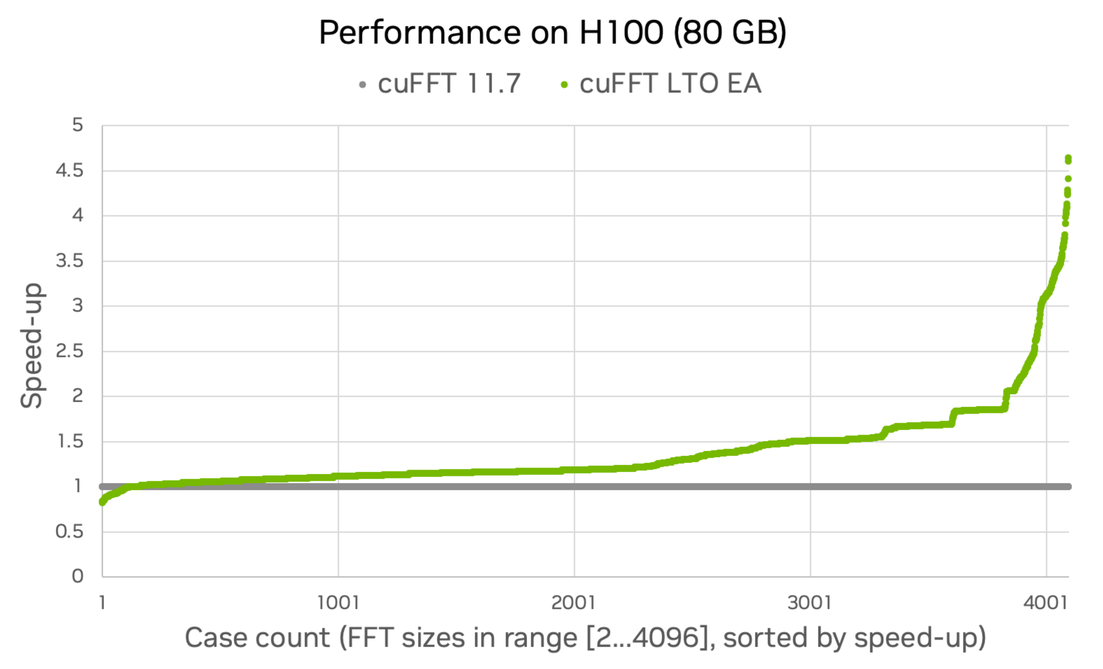


cuFFT Callbacks - LTO (In early access)

Runtime fusion with user code

- Early access version of cuFFT with Link-Time Optimization
- Enables runtime fusion of user code and cuFFT kernels
- Feedback on API and user experience is appreciated!

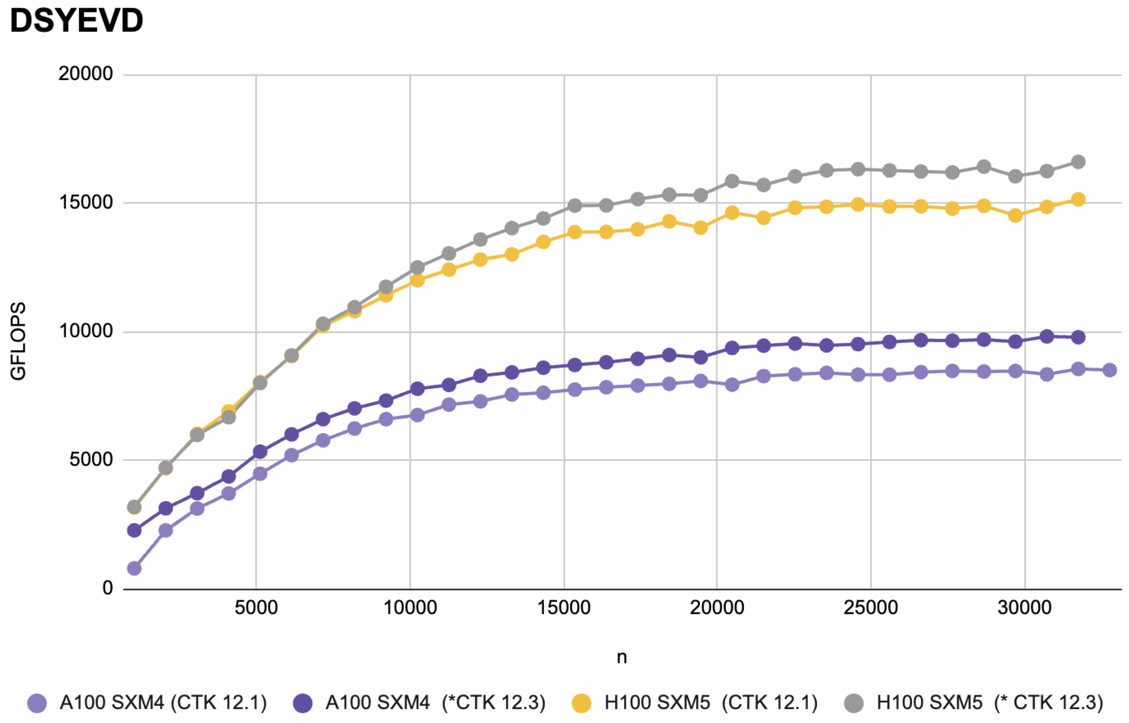




cuSOLVER

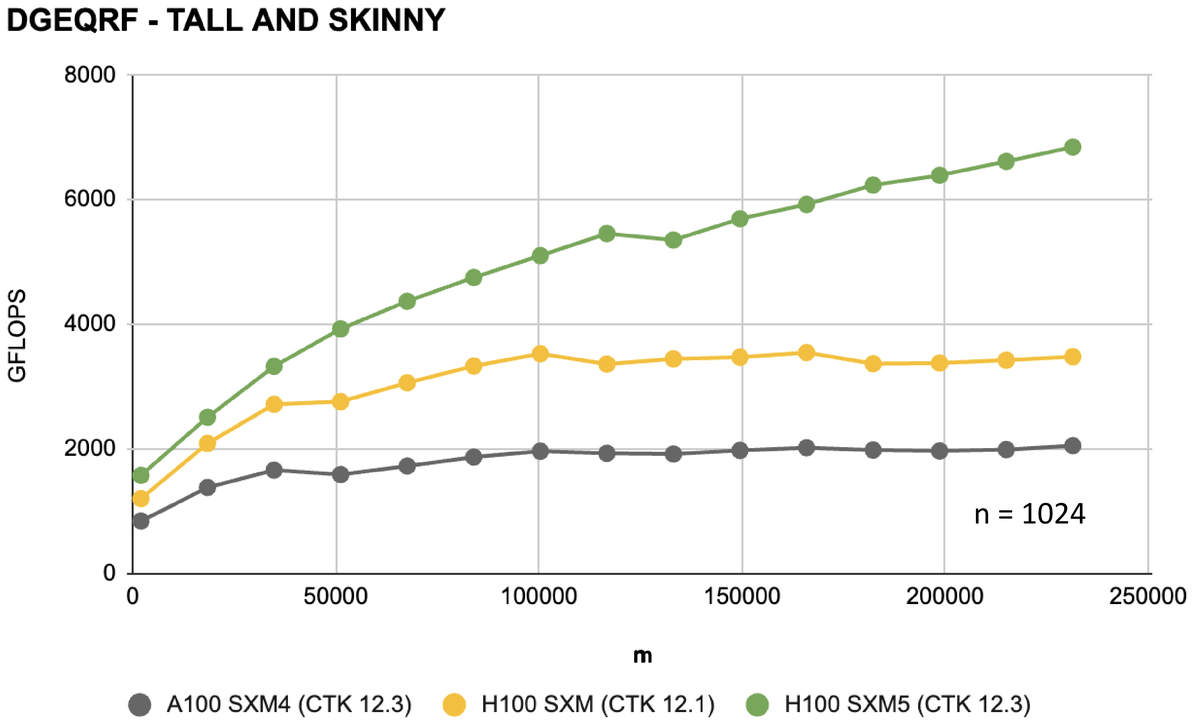
H100 performance improvements

- Real symmetric Eigen Value Decomposition performance 1.5x on H100



*CUDA 12.3: Using non-deterministic mode: new APIs for improved performance without bit-wise reproducibility

- Real QR factorization performance ~3x on H100



CUDA 12.3 : New improved QR algorithm for 'Tall and Skinny' matrices for Hopper

Multi GPU Multi Node libraries

New scalable libraries and Grace Hopper Support

cuFFTMp

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

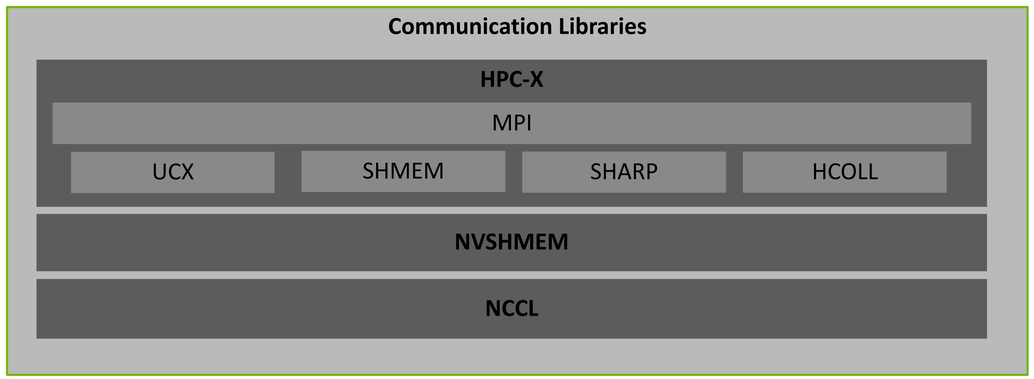
cuSOLVERMp

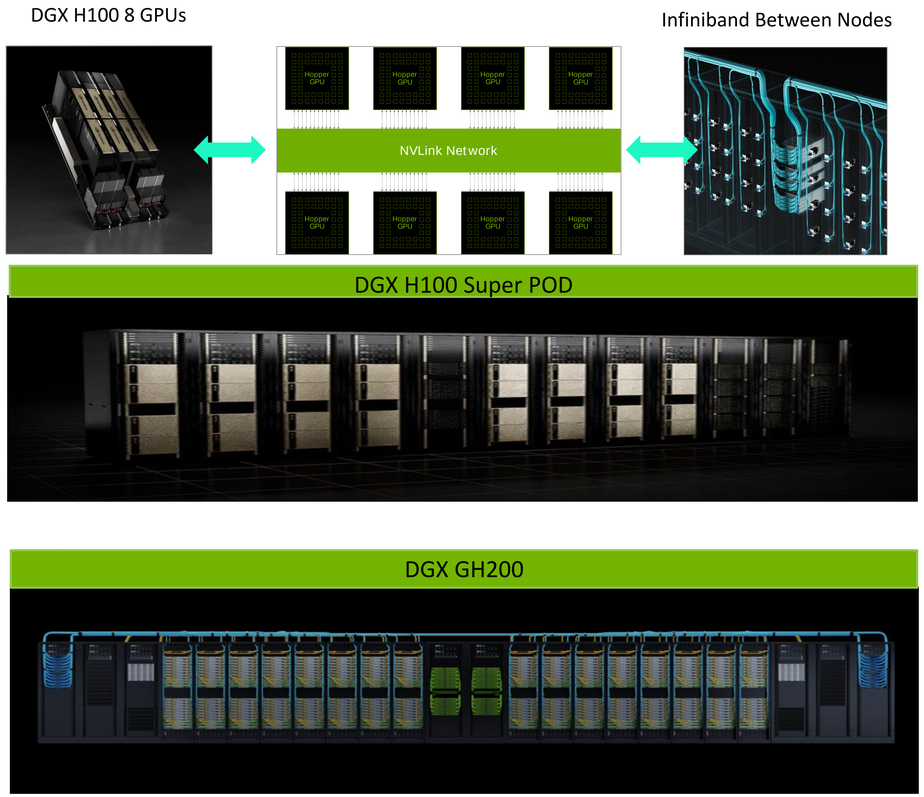
Factorization, Symmetric Eigensolver

cuBLASMp

New library
In early access

+ ARM support for Grace Hopper



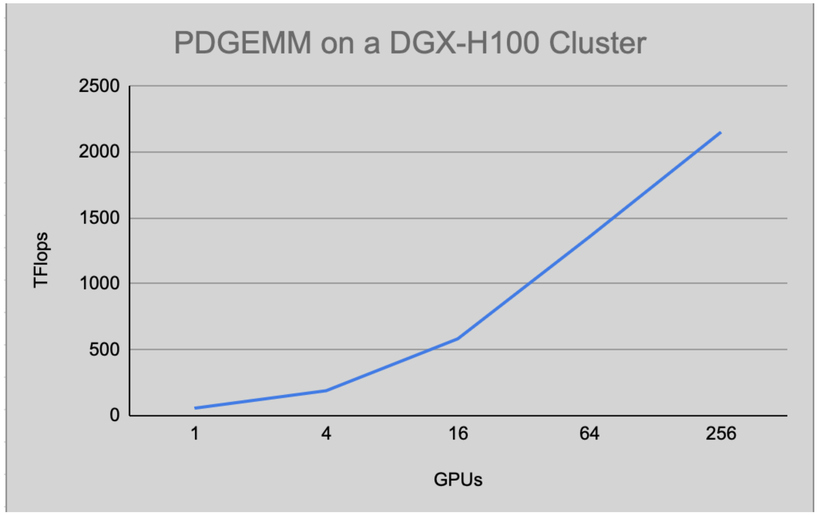


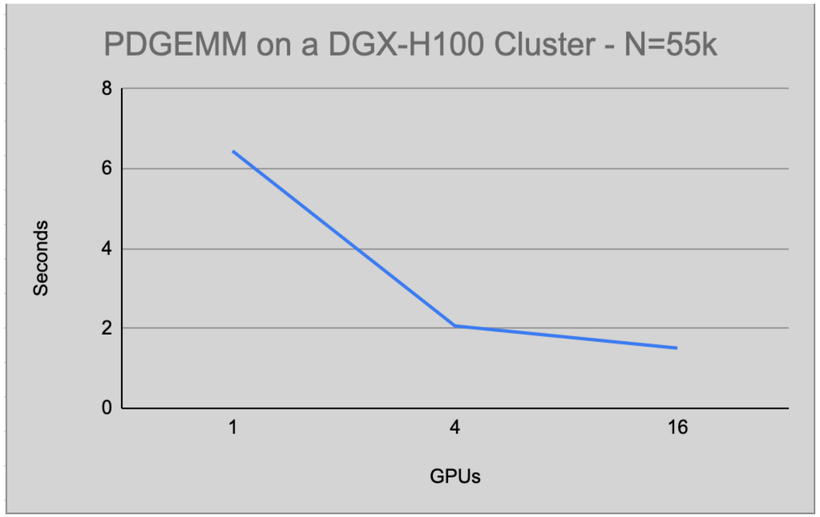
256 Grace Hopper Superchips | **1EFLOPS** AI Performance | **144TB** unified fast memory
36 L2 NVLink switches | **900 GB/s** GPU-to-GPU bandwidth | **128 TB/s** bisection bandwidth

cuBLASMp Early Access

New Distributed Dense Linear Algebra and MatMul library

- Tensor-core accelerated Multi-process, Multi-GPU extension to cuBLAS
- GEMM, SYRK/HERK and TRSM support
- One process per GPU
- PBLAS like functionalities and interfaces for ease of porting
- Configurable communication backends (UCC, NCCL, UCX, etc.,)
- **Available soon on Devzone**



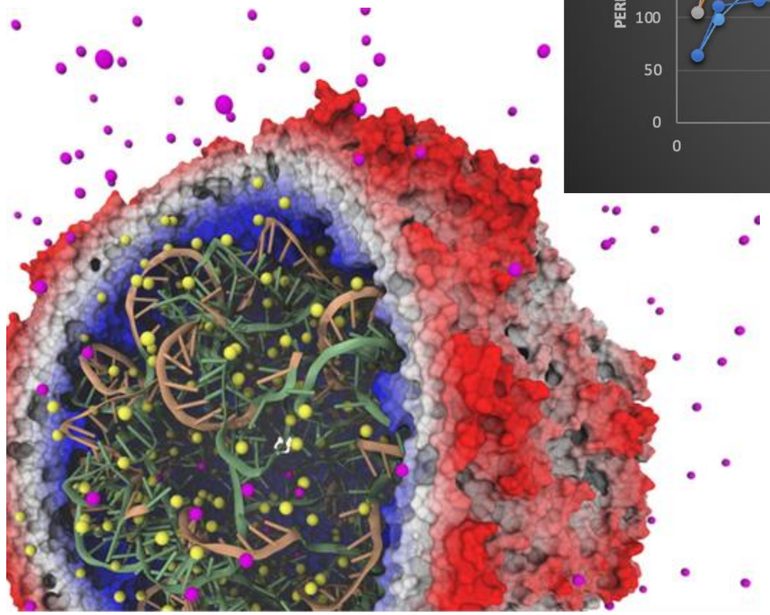


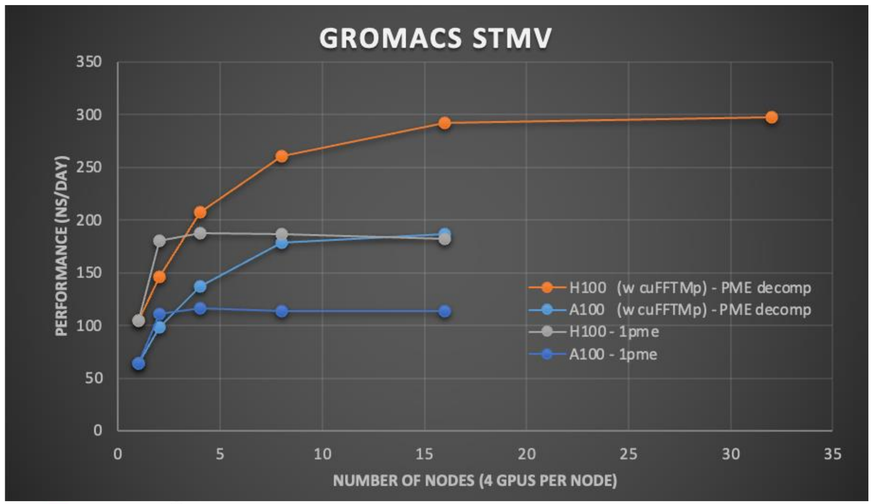
cuFFTMP Scales GROMACS

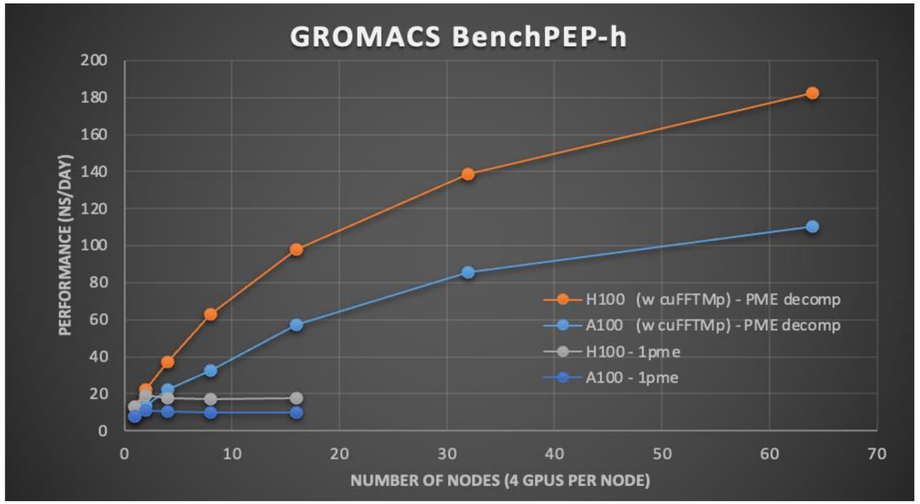
More performance improvements with Hopper

Distributed 3D FFTs

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



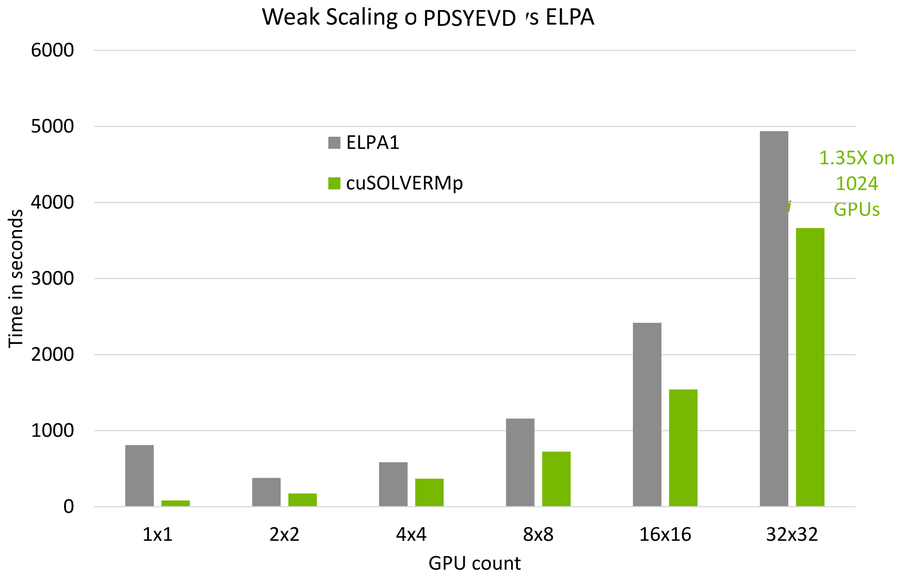




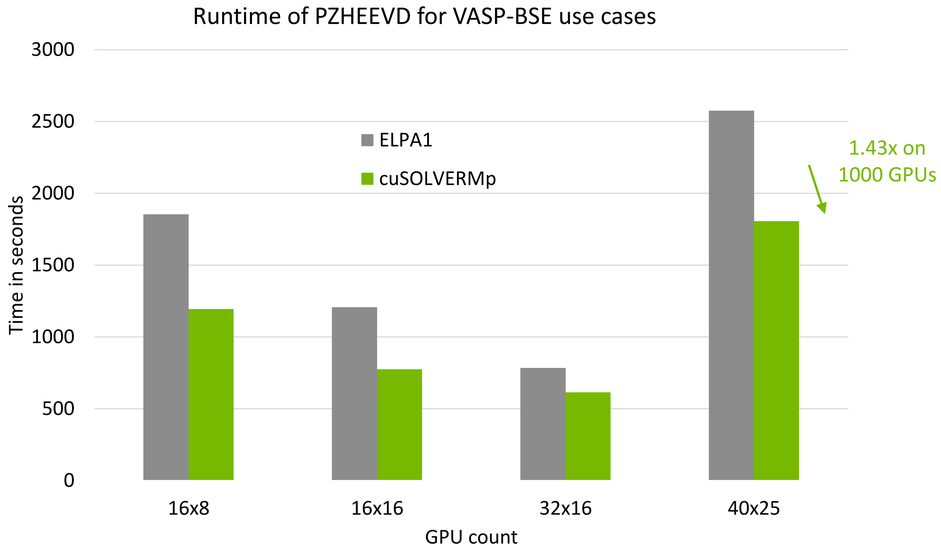
cuFFTMp available for download through HPC SDK
https://developer.nvidia.com/hpc-sdk

cuSOLVERMp Scales VASP*

State-of-the-art performance and scalability for Symmetric Eigensolvers and more



Performance measured on NVIDIA A100 DGX Super POD
~32k x 32k, real fp64 input matrix per GPU



Performance measured on NVIDIA A100 DGX Super POD
Double (fp64) complex input matrix

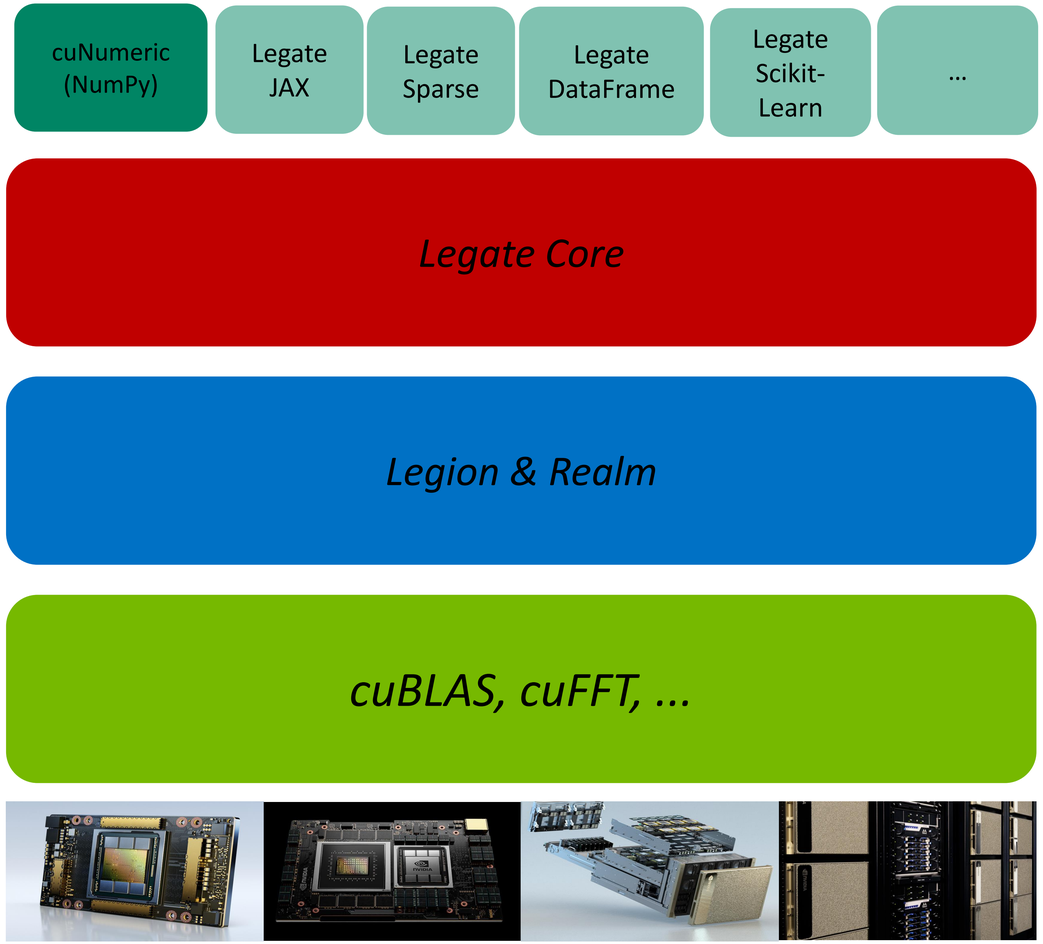
cuSOLVERMp available for download!

Large-Scale BSE Calculations for Solar-Panel Materials in VASP on GPUs with cuSolverMp

***VASP release with cuSOLVER Integration coming soon**

Legate's Mission: Accessible Accelerated Computing

Accelerated computing without the pain of complex distributed programming



Ecosystem of implicitly parallel libraries that are composable and easy-to-use

Productivity layer that accelerates library development, enables composability

Common runtime system for scalable extraction of implicit parallelism

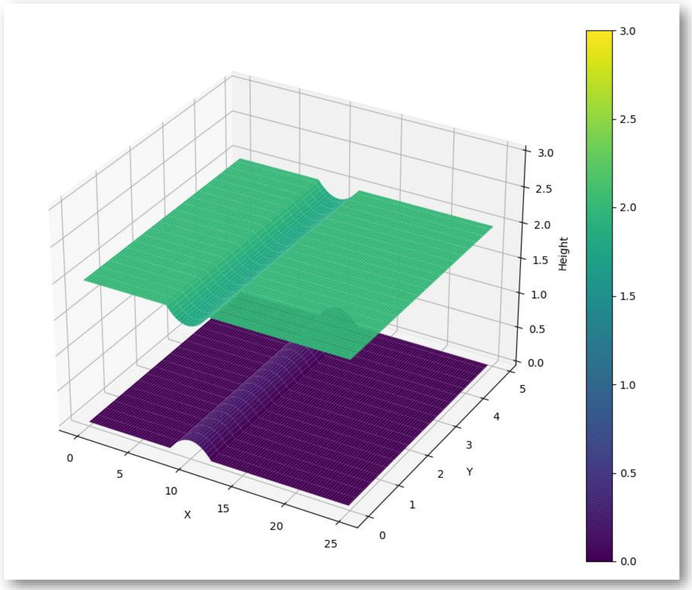
Accelerated domain libraries for excellent single-accelerator performance

CPUs, GPUs, Interconnect

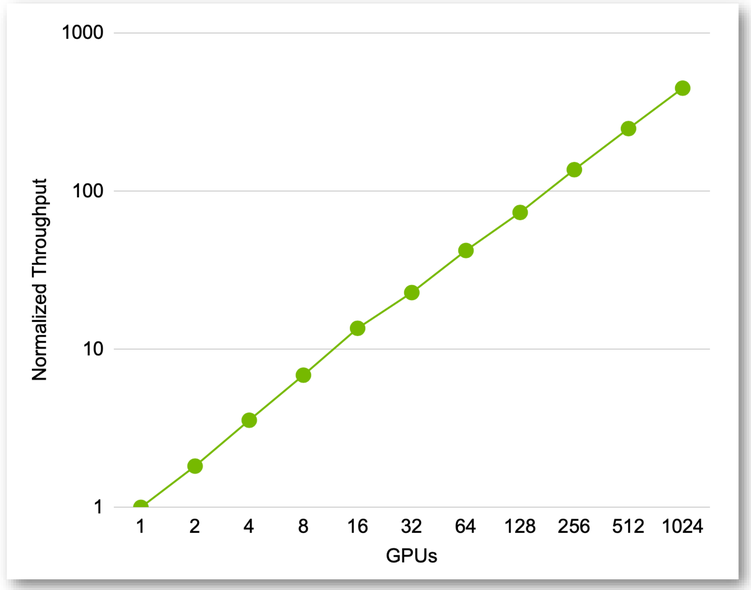
Out-of-the-Box Scaling for End-Users

TorchSWE: Porting an existing simulation to cuNumeric

- GPU accelerated solver for shallow water equations
- Original CuPy+MPI code is ported to cuNumeric by removing the MPI code
- cuNumeric allows domain scientists **with no MPI or GPU knowledge** to attain the simulation resolution (~20B points) previously out of reach to them



Topography (below) and water level (above)



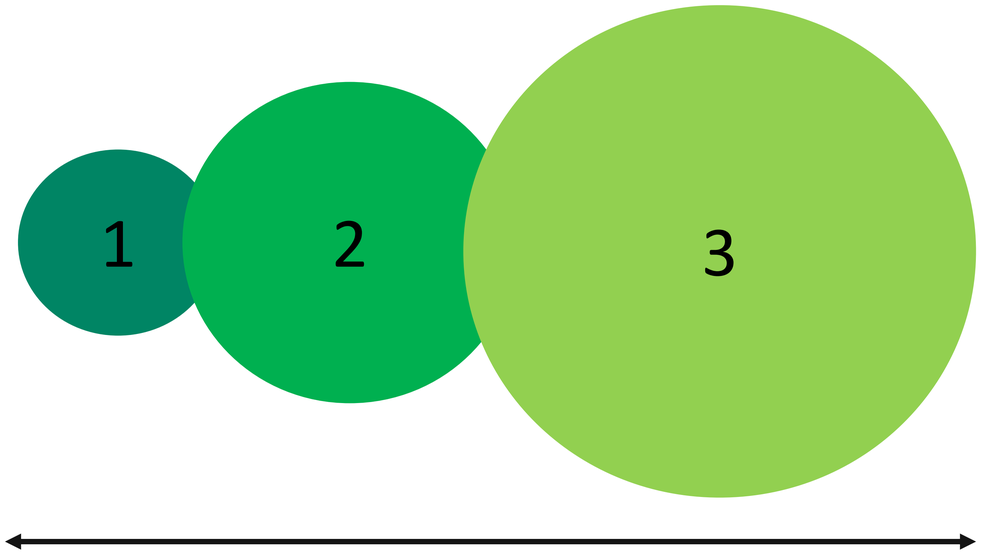
Weak scaling performance (40M grid points / GPU)

Target Audience for Legate

Three classes of users

1. “Expert class” of users
   - Highest performance at any cost. Likely committed to explicit/low-level programming APIs
2. “In-between class” of users
   - Enjoy transparent scaling/acceleration from the libraries, but occasionally need custom code or tune the libraries
3. “Domain & Data scientist class” of users
   - Have scaling needs but little computing expertise/resource to invest, productivity over performance

Target for Legate



Parallelism, data partitioning, communication, and synchronization controlled by user

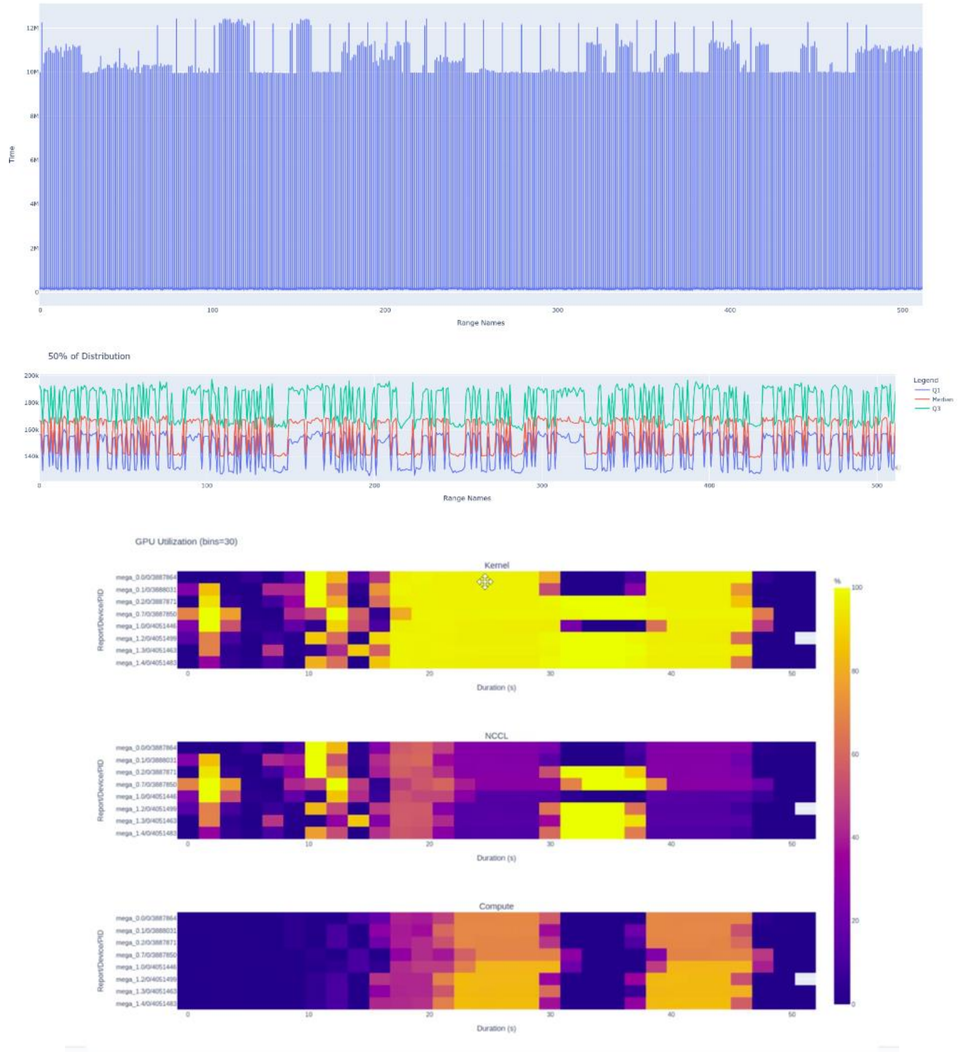
Parallelism, data partitioning, communication, and synchronization controlled by system

NVIDIA.

Scalable Profiling with Nsight Tools

CPU and GPU profiling at any scale

- NVIDIA Nsight Systems now supports multi-node profiling!
  - Networking APIs & HW metrics samples
  - Multi-report data analysis
  - Jupyter Lab integration
  - Multi-report visualization
- Analyze thousands of reports
  - No manual measure & search
  - Avoid “guess which file”
  - Scale-out processing
  - Cluster-wide conclusion
- Includes a customizable library of analysis recipes and reports.



Conclusions

- The NVIDIA HPC Software Platform makes programmers more productive
  - Unified memory simplifies code development
  - No porting required with standard languages

- NVIDIA software delivers the highest performance
  - NVIDIA HPC Compilers are free, cross-platform compilers for CPUs and GPUs
  - NVIDIA Performance Libraries provide excellent GPU and CPU performance

- NVIDIA HPC software scales science to new heights
  - Scalable math libraries enable unprecedented science
  - Legate for Python makes scaling achievable by domain experts

https://www.nvidia.com/hpc

