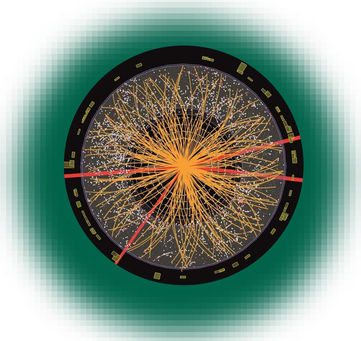
PORTABLE PARALLELISM IN THE POST EXASCALE ERA

The Great Thing About Portable Parallel Programming Standards is that there are SO Many to Choose From

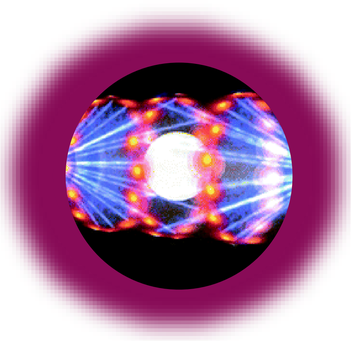
EXPANDING UNIVERSE OF HPC

EDGE

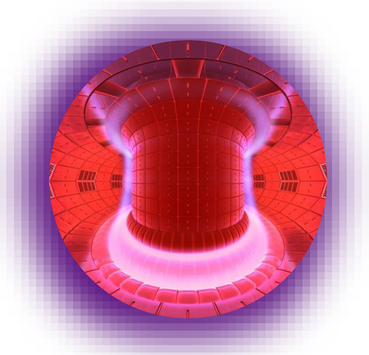
HPC * AI



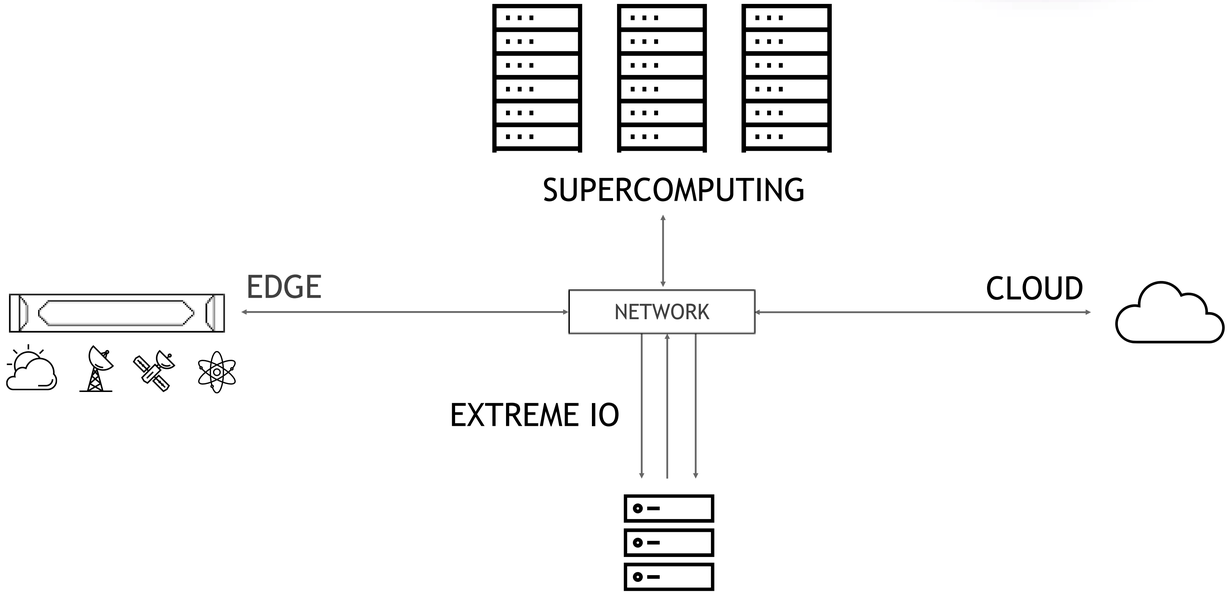
SIMULATION



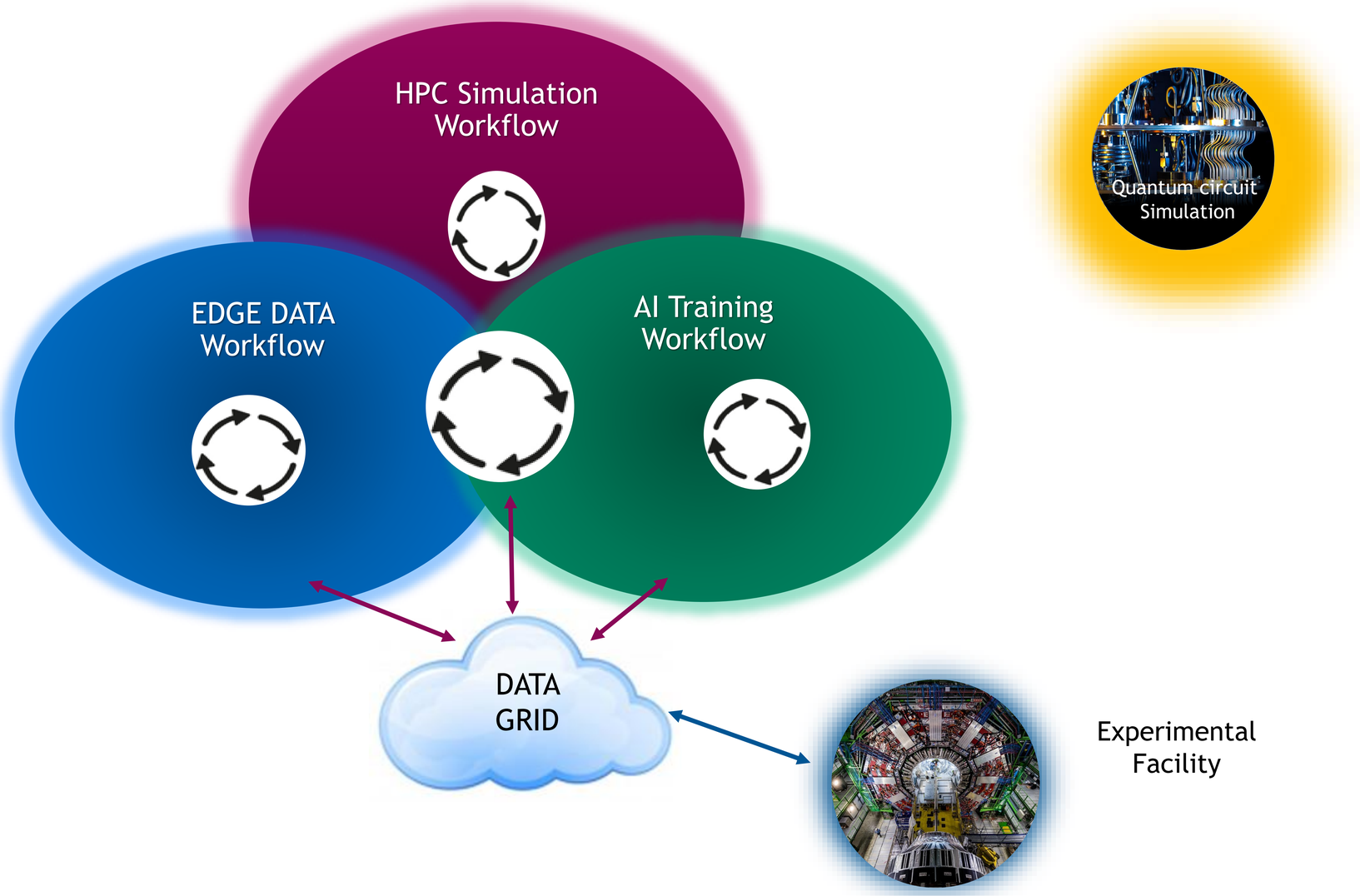
DIGITAL TWIN



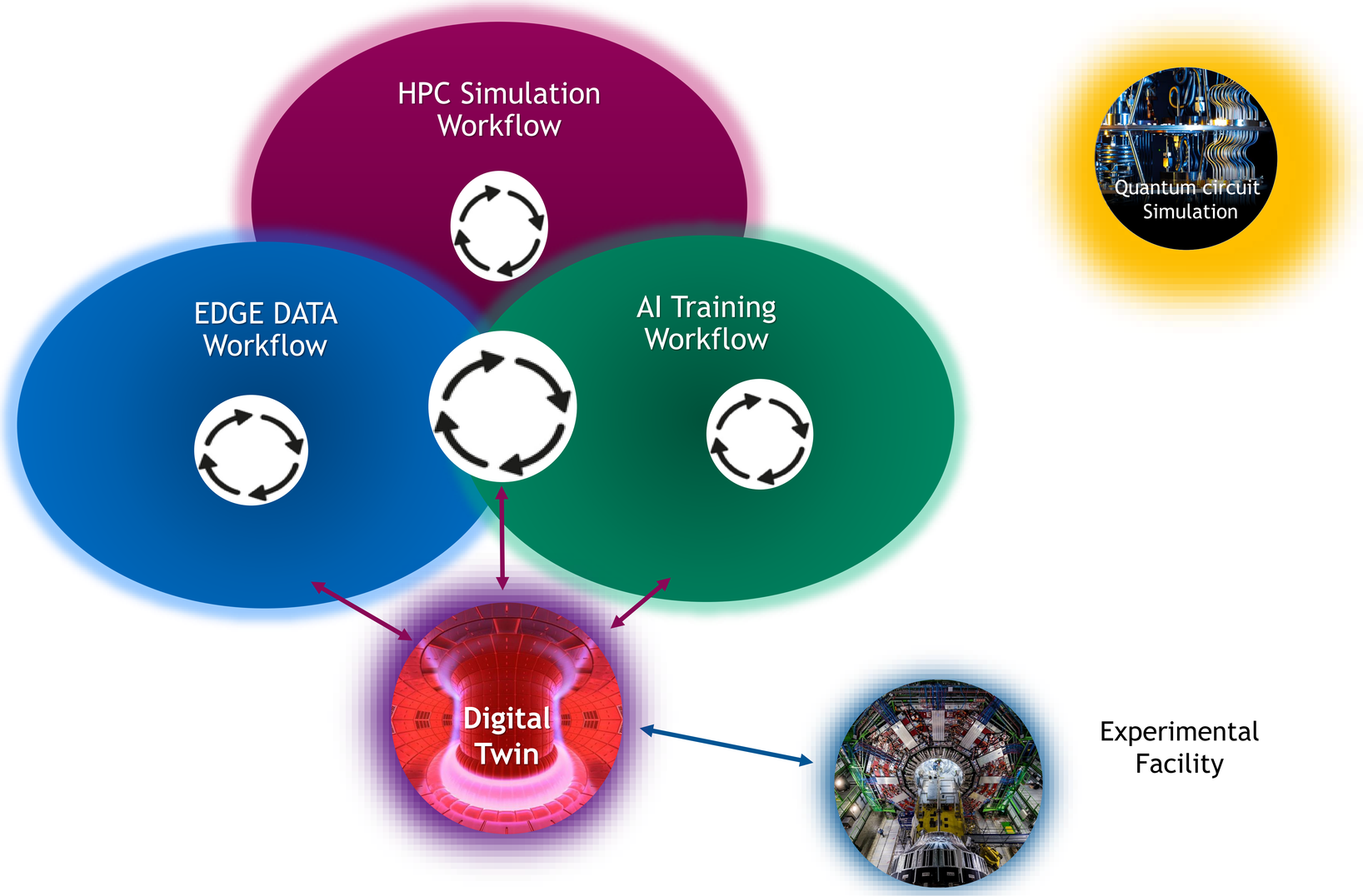
QUANTUM COMPUTING



EMERGING WORKFLOWS



EMERGING WORKFLOWS



PORTABLE PARALLEL LANGUAGES AND TOOLS

On PCs

On Prem

At the Edge

In the Public Cloud

Standard Languages

python

Directives

OpenACC

OpenMP

Platform Specific Languages

NVIDIA

CUDA

Templated Abstraction Libraries and Dynamic Programming

KOKKOS, RAJA, JULIA

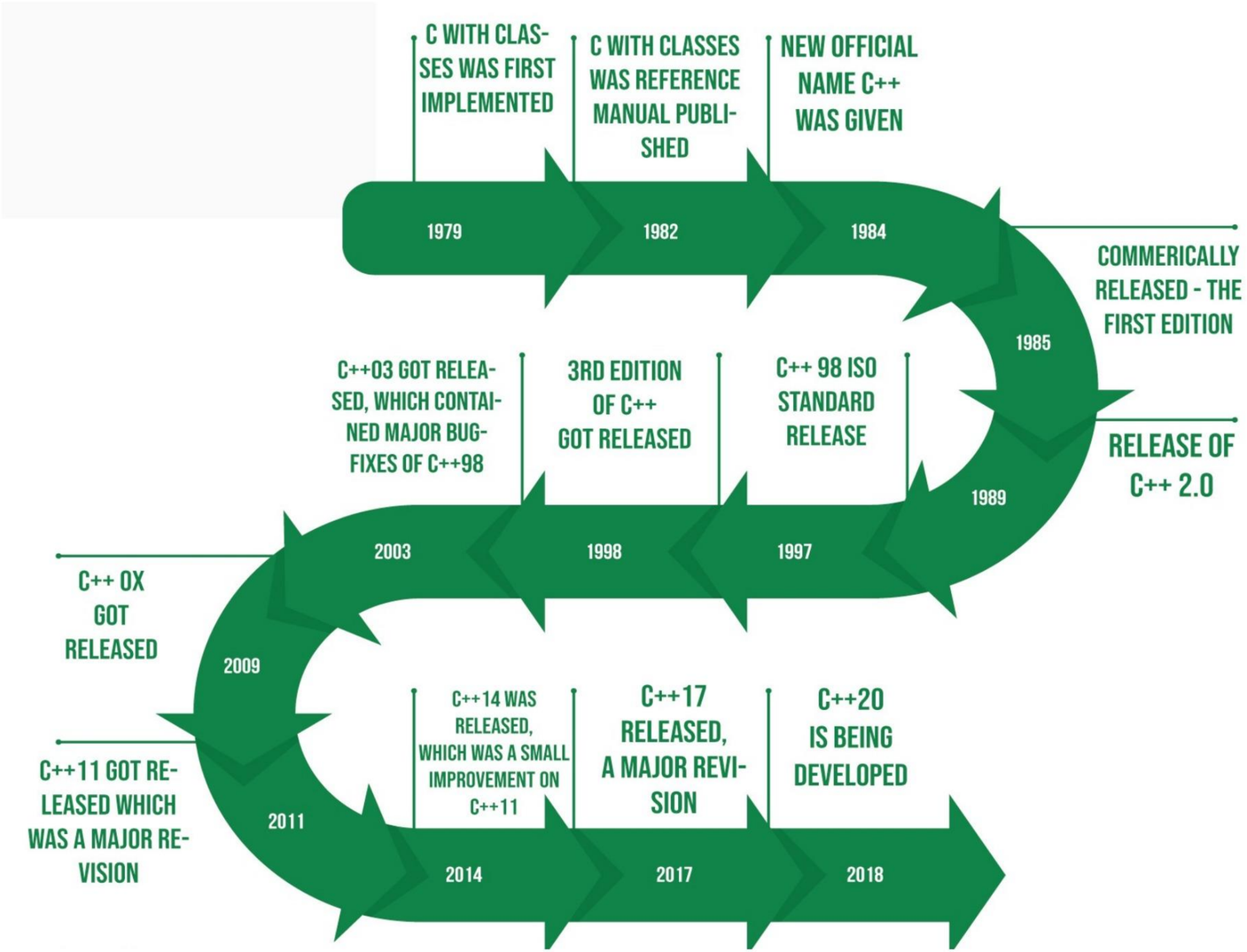
AI Frameworks

(PyTorch, Keras, TensorFlow, etc.)

Acceleration Libraries

(AI, Data Analytics, Algebra, Quantum, Communication)

History of C++


C WITH CLASSES WAS FIRST IMPLEMENTED
1979
C WITH CLASSES WAS REFERENCE MANUAL PUBLISHED
1982
NEW OFFICIAL NAME C++ WAS GIVEN
1984
COMMERICALLY RELEASED - THE FIRST EDITION
1985
RELEASE OF C++ 2.0
1989
C++ 98 ISO STANDARD RELEASE
1997
3RD EDITION OF C++ GOT RELEASED
1998
C++03 GOT RELEASED, WHICH CONTAINED MAJOR BUGFIXES OF C++98
2003
C++ 0X GOT RELEASED
2009
C++11 GOT RELEASED WHICH WAS A MAJOR REVISION
2011
C++14 WAS RELEASED, WHICH WAS A SMALL IMPROVEMENT ON C++11
2014
C++17 RELEASED, A MAJOR REVISION
2017
C++20 IS BEING DEVELOPED
2018

PARALLELISM IN C++ ROADMAP

C++ 11

- Memory model enhancements
- Lambdas

C++ 14

- Atomics extensions
- Generic Lambda Expressions

C++ 17

- Parallel algorithms
- Forward progress guarantees
- Memory model clarifications

C++ 20

- Scalable synchronization library
- Ranges
- Span

C++ PIPELINE

- Mdspan
- Range-based parallel algorithms
- Extended floating-point types
- Linear algebra algorithms
- Asynchronous parallel algorithms
- Senders-receivers

How users run C++ code on GPUs today

Co-designed with V100 hardware support

N-dimensional loops and usability

Custom algorithms and async. control flow

Extended C++ interface to BLAS/Lapack

General usability of performance provided by executors

General parallelism user facing feature

LULESH HYDRODYNAMICS

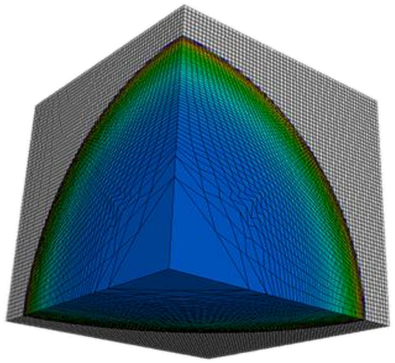
```cpp
static inline
void CalcHydroConstraintForElems(Domain &domain, Index_t length,
    Index_t *regElemlist, Real_t dvovmax, Real_t& dthydro)
{
#if _OPENMP
  const Index_t threads = omp_get_max_threads();
  Index_t hydro_elem_per_thread[threads];
  Real_t dthydro_per_thread[threads];
#else
  Index_t 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

- Composable, compact and elegant
- Easy to read and maintain
- ISO Standard
- Portable – nvc++, g++, icpc, MSVC, …



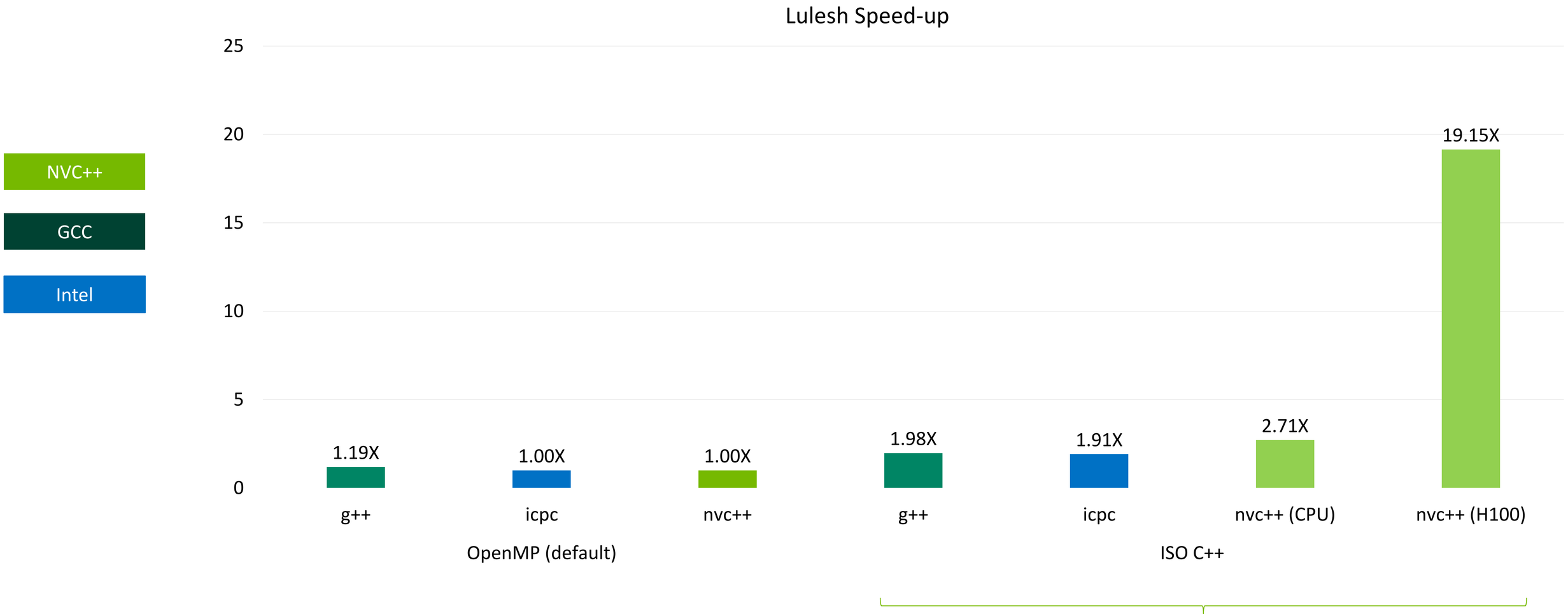
```cpp
static inline void CalcHydroConstraintForElems(Domain &domain, Index_t length,
                Index_t *regElemlist,
                Real_t dvovmax,
                Real_t &dthydro)
{
  dthydro = std::transform_reduce(
    std::execution::par, counting_iterator(0), counting_iterator(length),
    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

C++ STANDARD PARALLELISM

Lulesh Performance



Same ISO C++ Code

HPC PROGRAMMING IN ISO FORTRAN

ISO is the place for portable concurrency and parallelism

Preview support available now in NVFORTRAN

Fortran 2018

Fortran Array Intrinsics

- NVFORTRAN 20.5
- Accelerated matmul, reshape, spread, …

DO CONCURRENT

- NVFORTRAN 20.11
- Auto-offload & multi-core

Co-Arrays

- Not currently available
- Accelerated co-array images

Fortran 202x

DO CONCURRENT Reductions

- NVFORTRAN 21.11
- `REDUCE` subclause added
- Support for `+`, `*`, `MIN`, `MAX`, `IAND`, `IOR`, `IEOR`.
- Support for `.AND.`, `.OR.`, `.EQV.`, `.NEQV` on `LOGICAL` values

GAMESS

Computational Chemistry with Fortran Do Concurrent

- GAMESS is a popular Quantum Chemistry application.
- More than 40 years of development in Fortran and C
- MPI + OpenMP baseline code
- Hartree-Fock rewritten in Do Concurrent

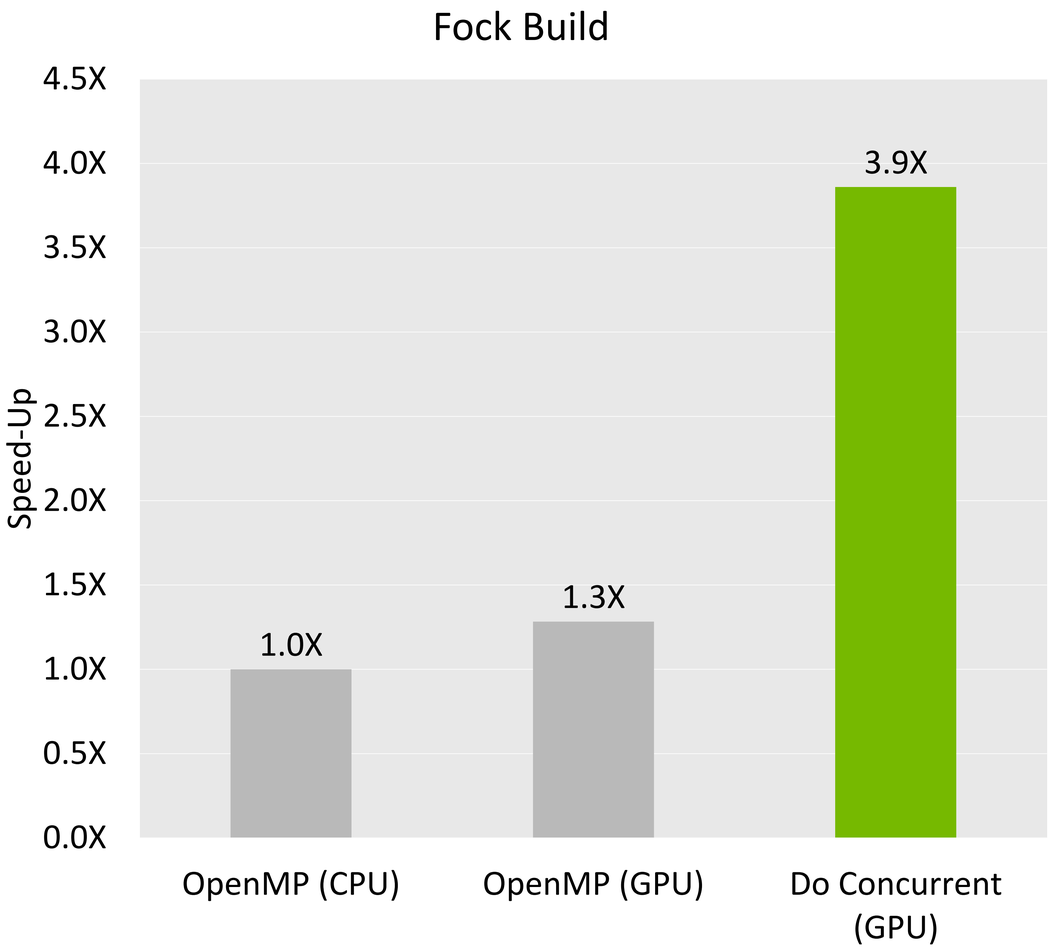
```
!pre-sorting, screening

!$omp target teams distribute &
            parallel do &
!$omp shared() private()
do iquart = 1, ssdd_quarts
!recover shell index
ish=IDX(s_sh)
jsh=IDX(s_sh)
ksh=IDX(d_sh)
lsh=IDX(d_sh)
  !compute ints
  !digest ints
enddo
!$omp end target teams distribute &
          parallel do
```

```
!pre-sorting, screening



DO CONCURRENT (iquart=1::ssdd_quarts)&
   SHARED() LOCAL()
!recover shell index
ish=IDX(s_sh)
jsh=IDX(s_sh)
ksh=IDX(d_sh)
lsh=IDX(d_sh)
  !compute ints
  !digest ints
enddo
```



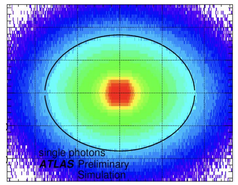
nvfortran 22.7, NVIDIA A100 GPU, AMD “Milan” CPU

* Courtesy of Melisa Alkan, Iowa State University. Not yet published.

ACCELERATING STDPAR ADOPTION

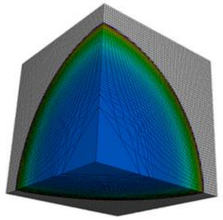
BabelStream

Cloverleaf

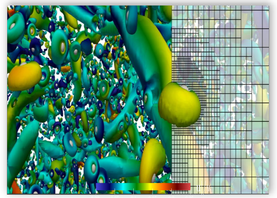


FastCaloSim

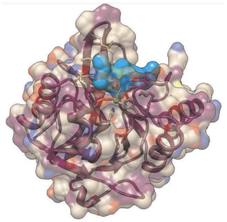
GAMESS



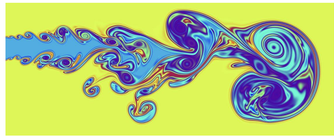
Lulesh



M-AIA



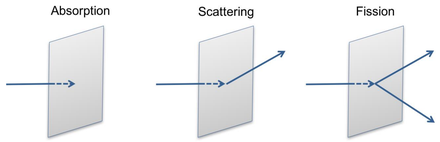
miniBUDE



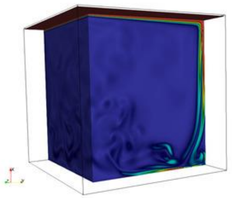
MiniWeather

Palabos

POT3D



Quicksilver

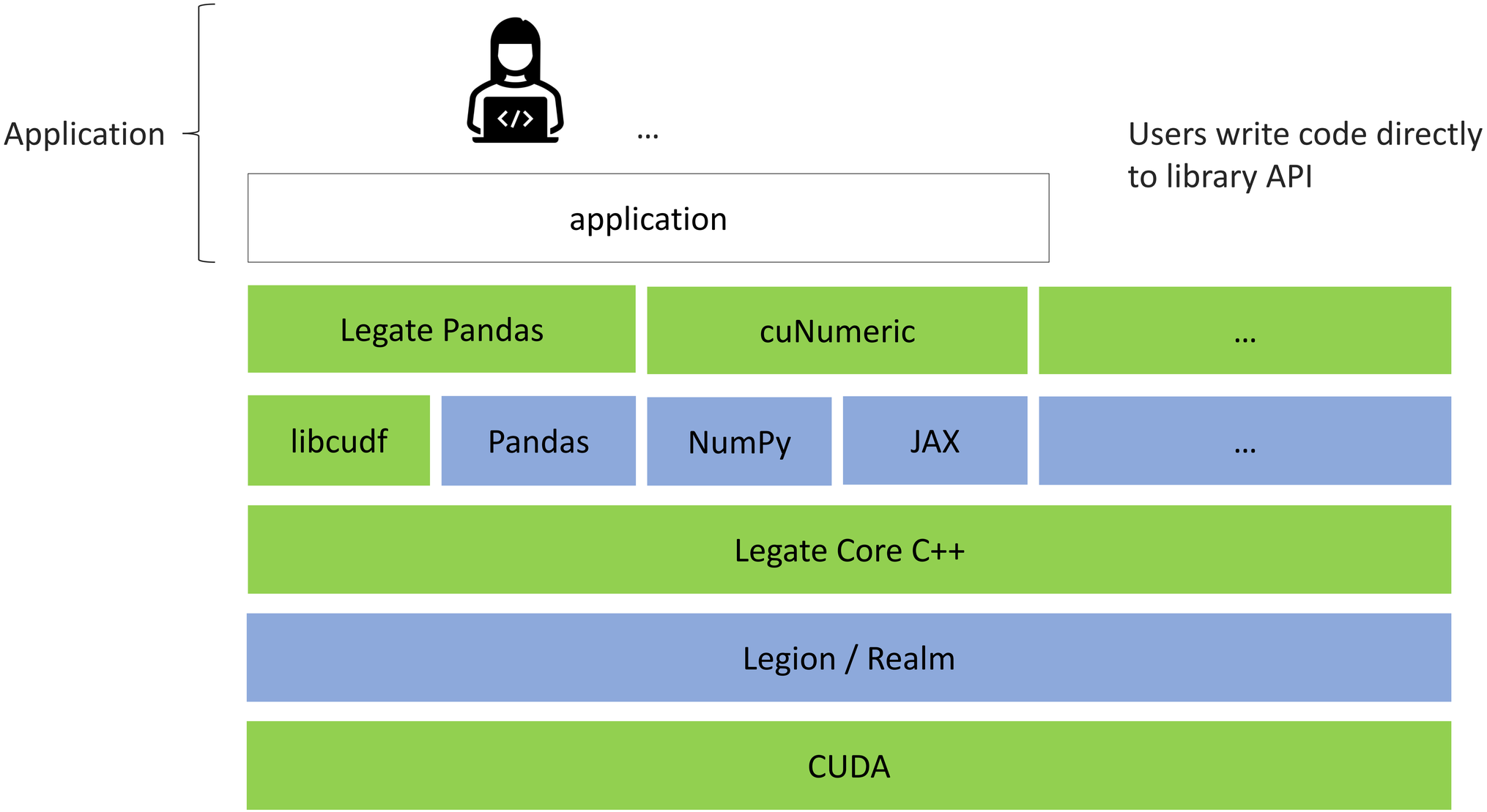


STLBM

RAJAPerf

LEGATE ECOSYSTEM ARCHITECTURE FOR PYTHON

Scalable implementations of popular domain-specific APIs



ENABLING CLOUD-NATIVE WORKFLOWS

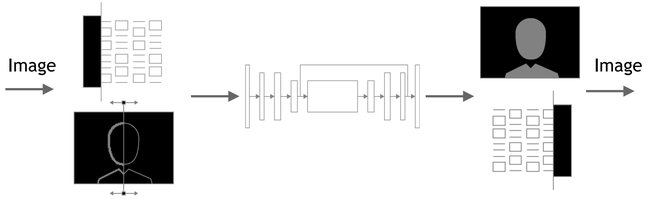
Logistics	Conversational AI	Speech AI	Recommenders	Cybersecurity	Communications	Robotics	Medical Imaging	Medical Devices	Video Analytics	Customer Service	Autonomous Vehicles

CV-CUDA

Open-source, accelerated pre- and post-processing for cloud-scale computer vision

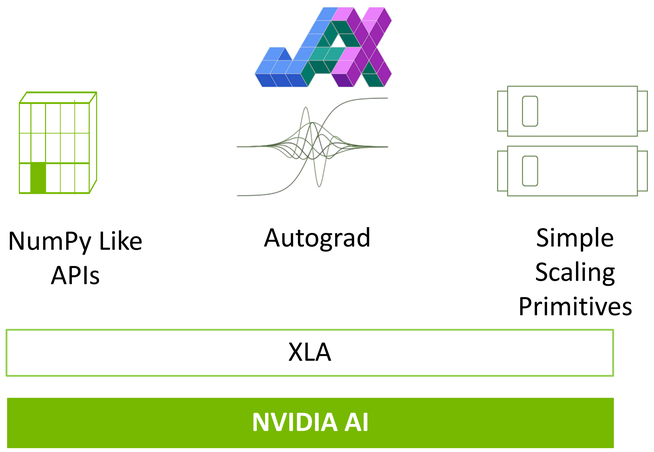
Accelerated Pre- & Post Processing at Cloud Scale

Segmentation CNN



Apply for EA at developer.nvidia.com/cv-cuda/early-access

Accelerating JAX



JAX Container will be available as private EA Q4'2022

Apply Now at developer.nvidia.com/jax-container-early-access

GPU-optimized PyG & DGL Container

PyG

NEW!

DGL DEEP GRAPH LIBRARY

Apply for EA at developer.nvidia.com/gnn-frameworks

ENABLING CLOUD-NATIVE WORKFLOWS

Logistics	Conversational AI	Speech AI	Recommenders	Cybersecurity	Communications	Robotics	Medical Imaging	Medical Devices	Video Analytics	Customer Service	Autonomous Vehicles
CUOPT	NEMO	RIVA	MERLIN	MORPHEUS	MAXINE	ISAAC	MONAI	HOLOSCAN	METROPOLIS	TOKKIO	DRIVE

Triton Inference Server

- Any Framework
- Any Model
- Any Query Type
- Any Processor
- Any Deployment Platform
- Any Deployment Location

RAPIDS

INTEGRATED WITH 100+ OPEN-SOURCE TOOLKITS

Apache Spark, BERTopic, dask, HoloViz, InterpretML, MARS, mlflow, MONAI+, Numba, PyCaret, RAY, scanpy

Kubernetes and Kubeflow

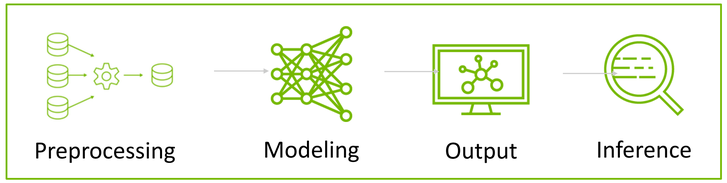
Easily launch multi-node workflows on Kubernetes and Kubeflow

Cloud

Available In All Major Public Clouds

NVIDIA Morpheus

GPU-accelerated, AI Cybersecurity Framework



Morpheus enables cybersecurity workflows:

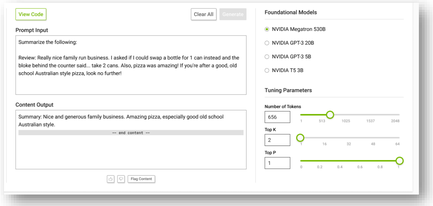
- Digital fingerprinting users and machines
- Sensitive Information detection
- Crypto mining malware detection
- Phishing detection
- Fraud detection

NGC

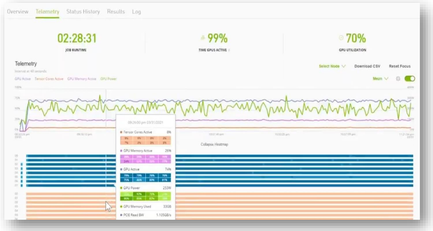
Portal to AI/HPC services, software, support

NGC Catalog

Cloud Services
End-to-End AI development

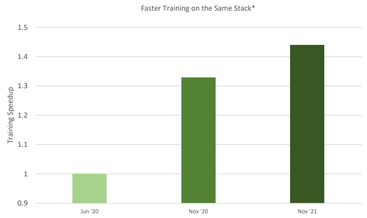


AI Services for NLP, biology, speech

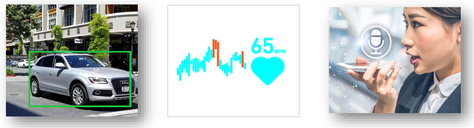


AI Workflow Management & Support

Performance Optimized
Tested across GPU-accelerated platforms



Monthly sw container updates

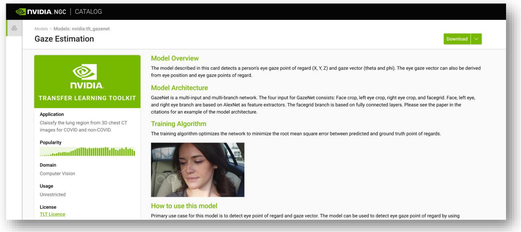


SOTA models

Fully Transparent
Quickly find and deploy the right sw

vulnerabilities	OS package	Medium	(CVE-2021-3995) libmount1
vulnerabilities	OS package	Medium	(CVE-2021-3995) fdisk
vulnerabilities	OS package	Medium	(CVE-2019-9152) hdf5-helpers
vulnerabilities	OS package	Medium	(CVE-2018-17233) hdf5-helpers

Detailed security scan reports



Model resumes

Accelerates Development
Focus on building, not setup

jupyter

One click deploy from NGC

aws
Google Cloud
Microsoft Azure
ORACLE CLOUD
Alibaba Cloud
OVHcloud

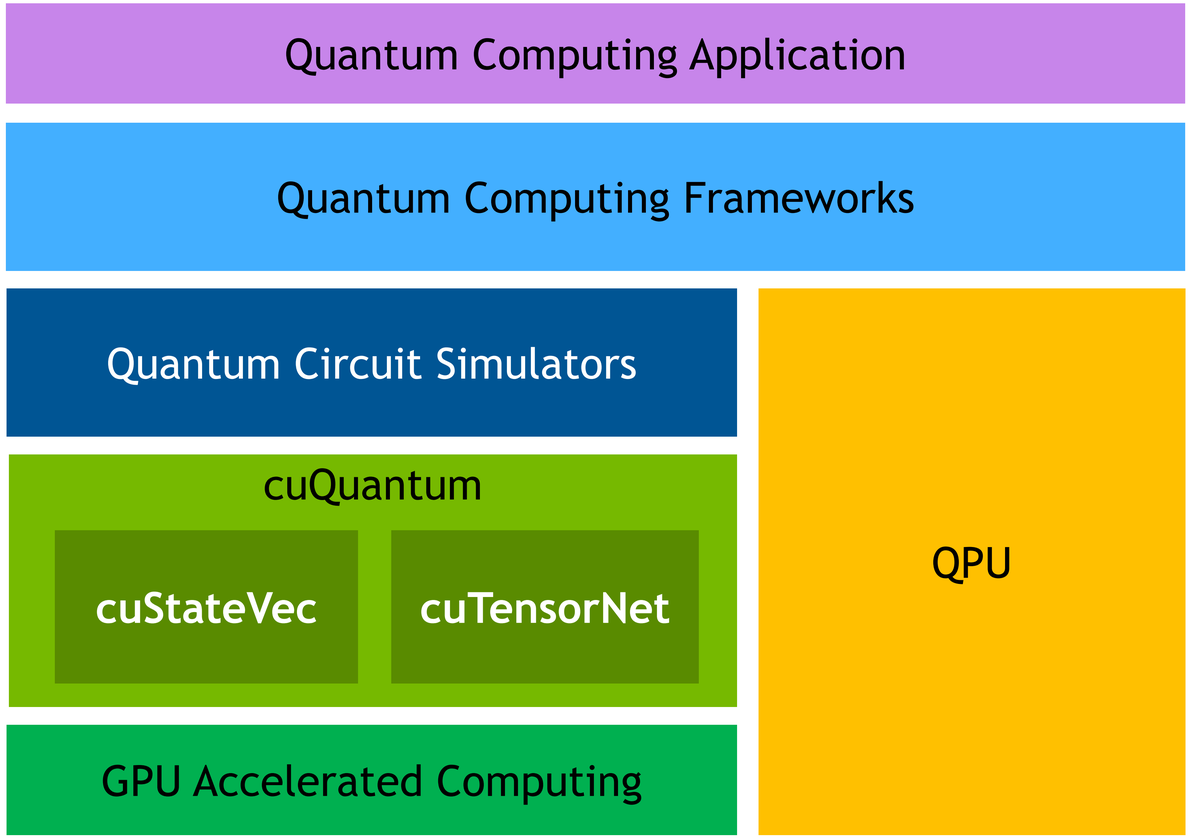
Develop once. Deploy anywhere w/ NVIDIA VMI

ngc.nvidia.com

NVIDIA

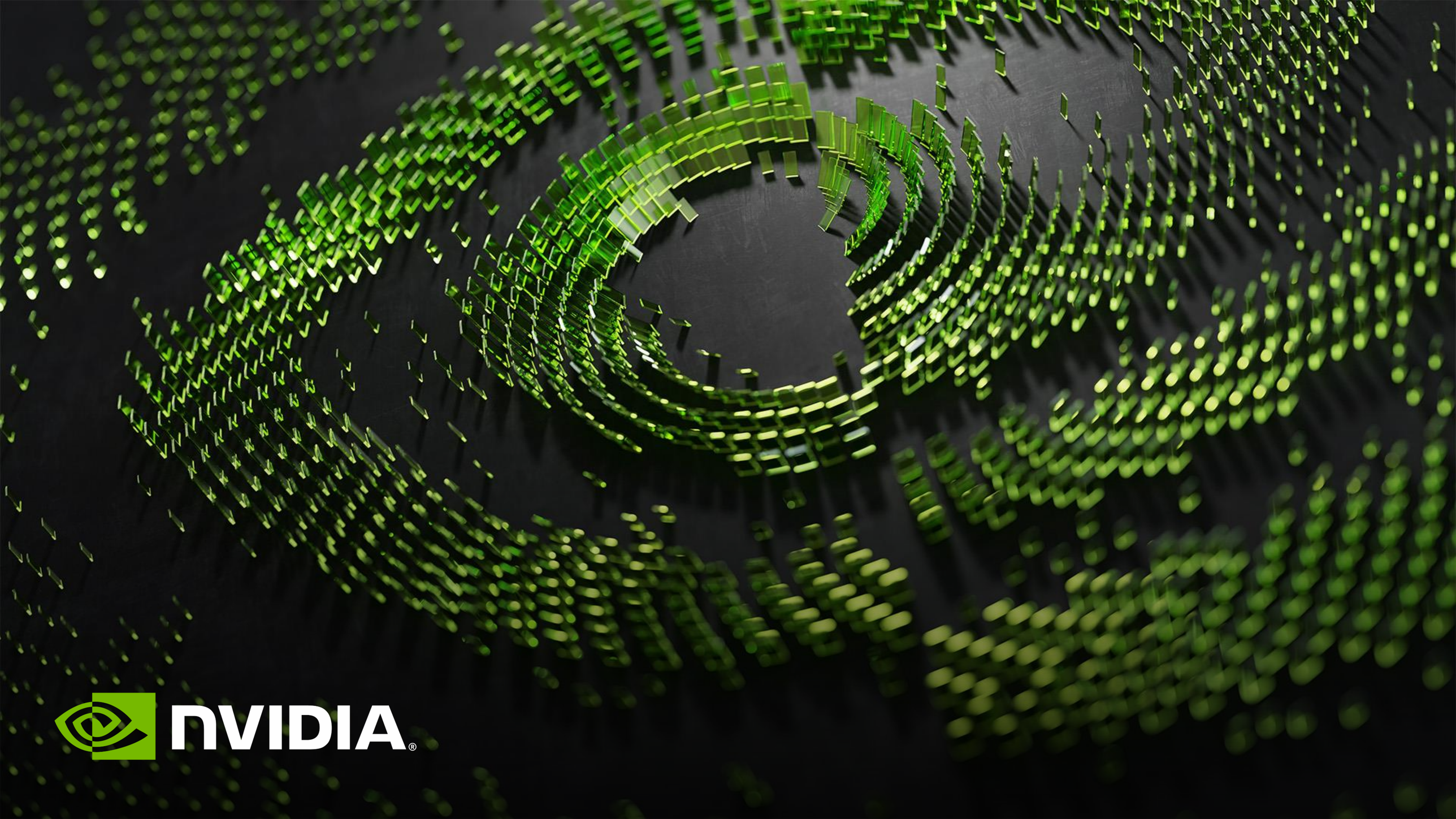
cuQuantum

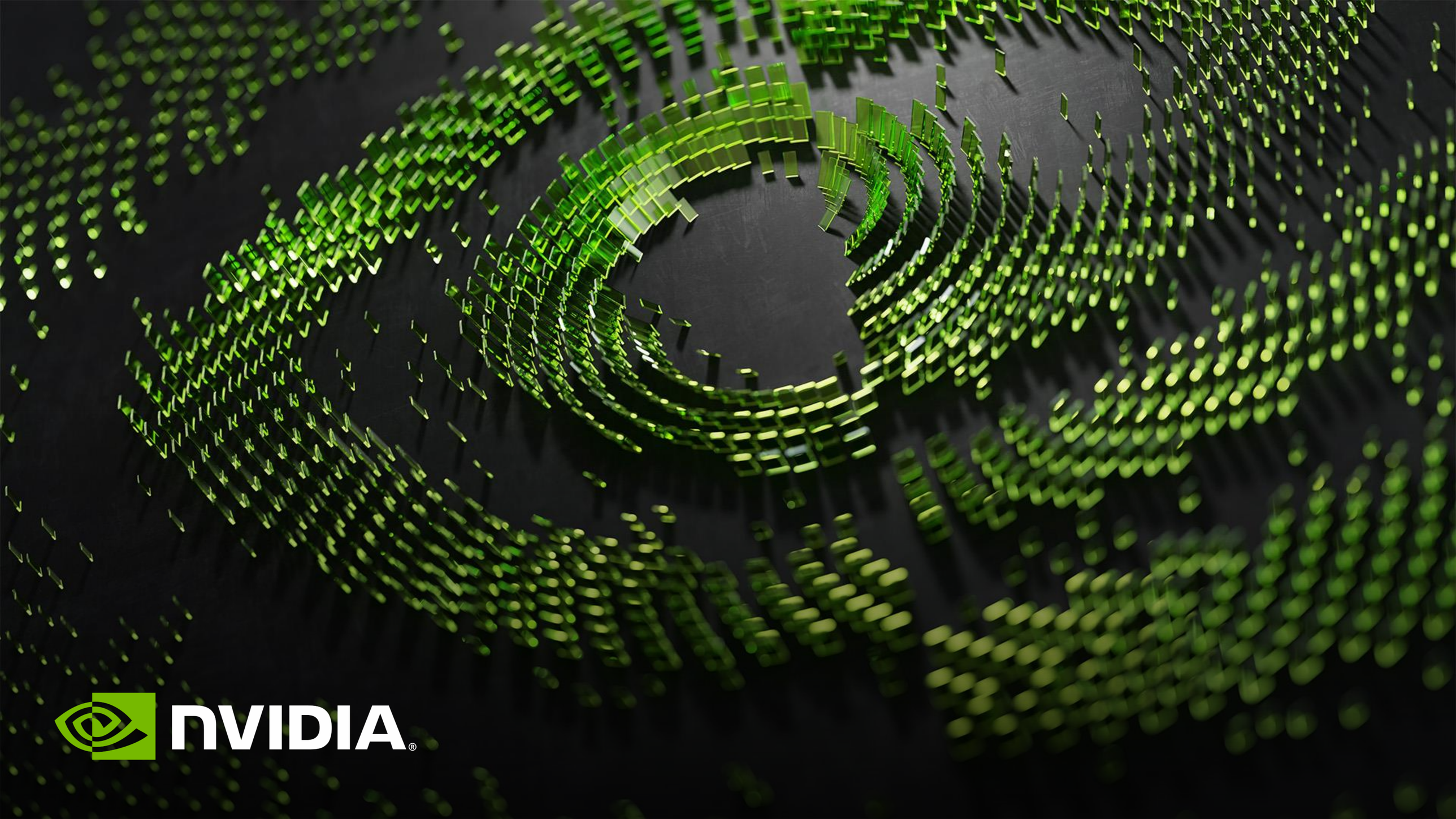
- **SDK for Quantum Circuit Simulation**
  - Accelerate Quantum Circuit Simulators on GPUs
  - Simulate ideal or noisy qubits
  - Enable algorithms research with scale and performance not possible on quantum hardware, or on simulators today
- **cuQuantum available now**
  - Integrated into leading quantum computing frameworks Cirq, Qiskit, and Pennylane
  - C and Python APIs
  - Available today at developer.nvidia.com/cuquantum



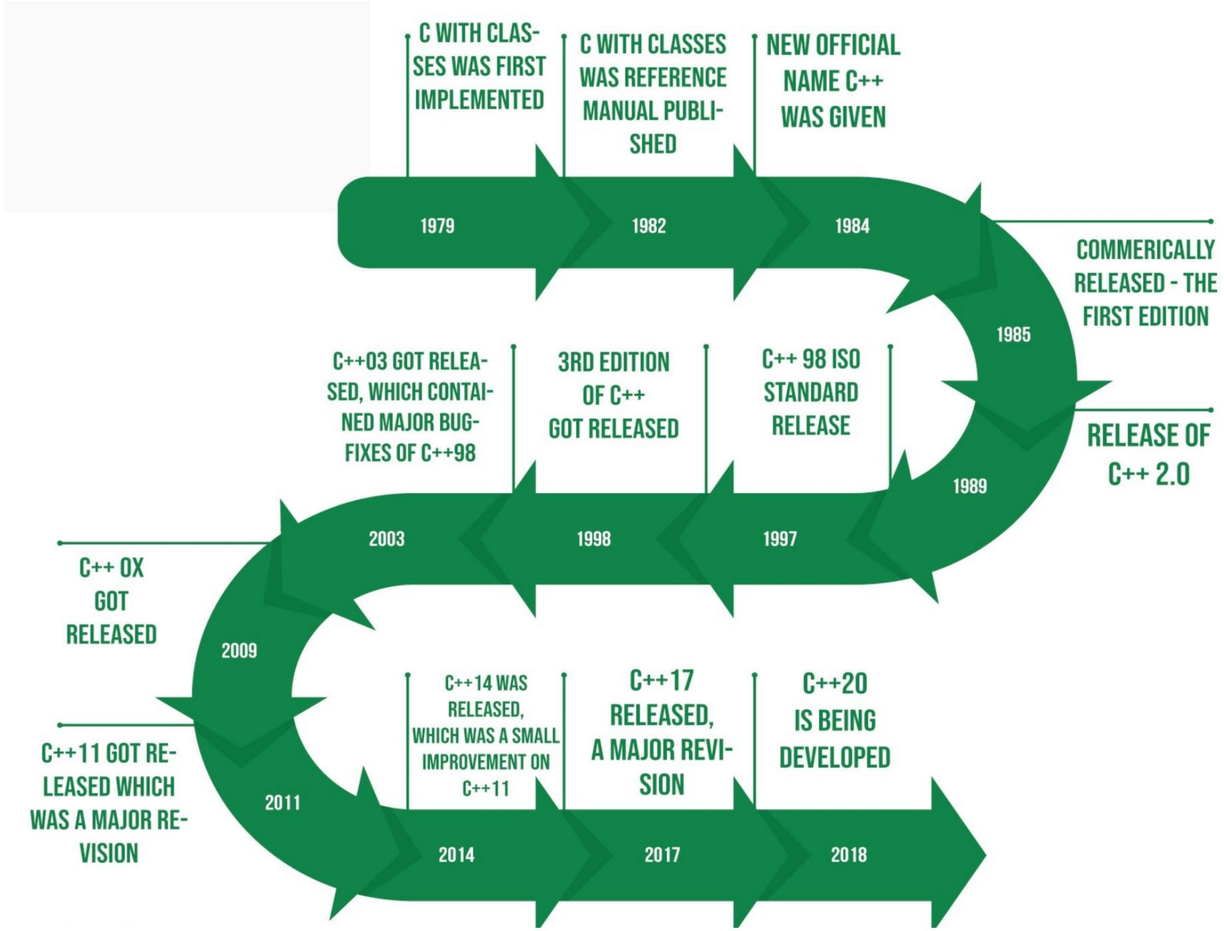
Cirq Qiskit PENNYLANE

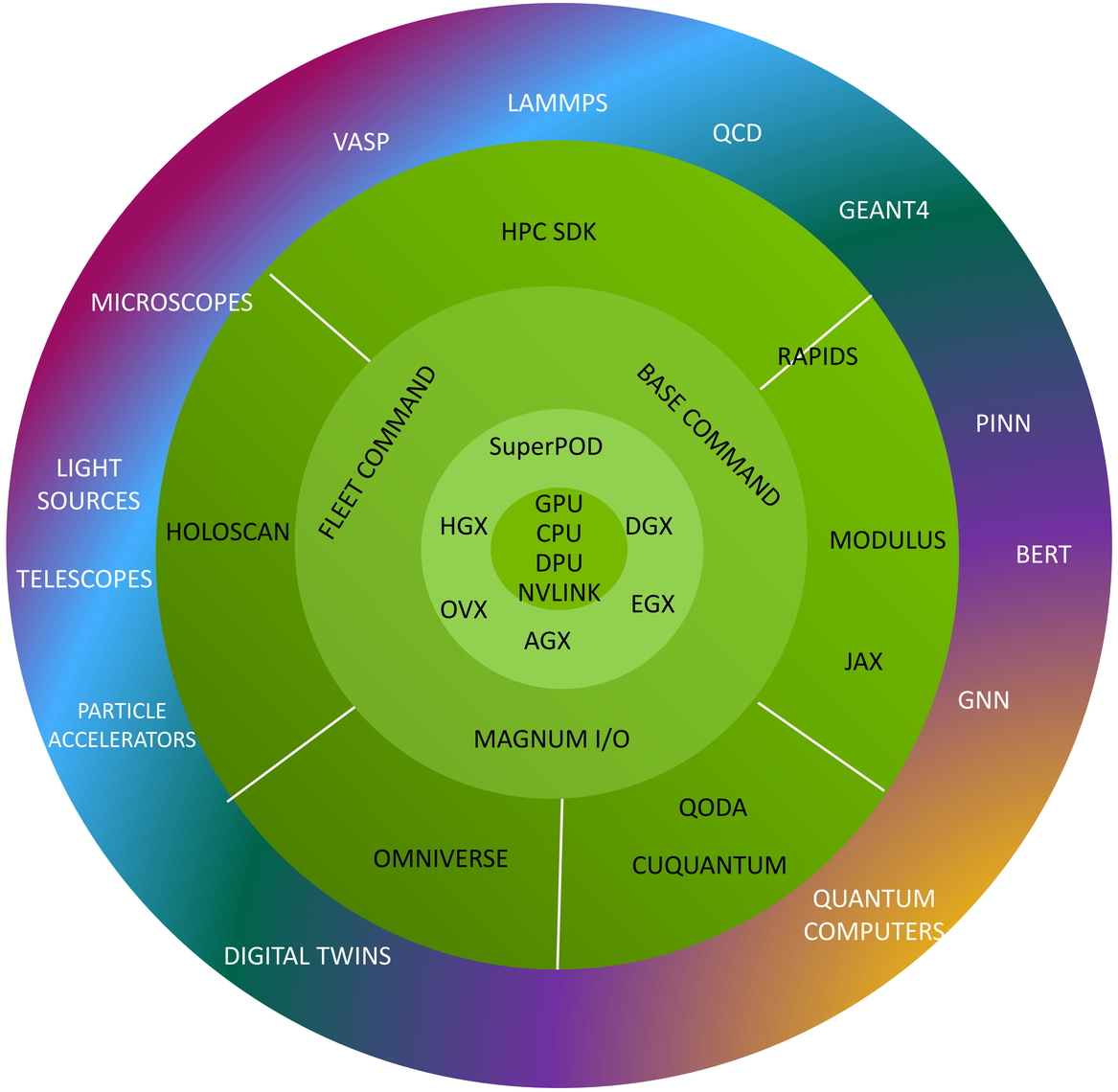
XACC QUANTUM FRAMEWORK Orquestra® QIBO


NVIDIA


NVIDIA

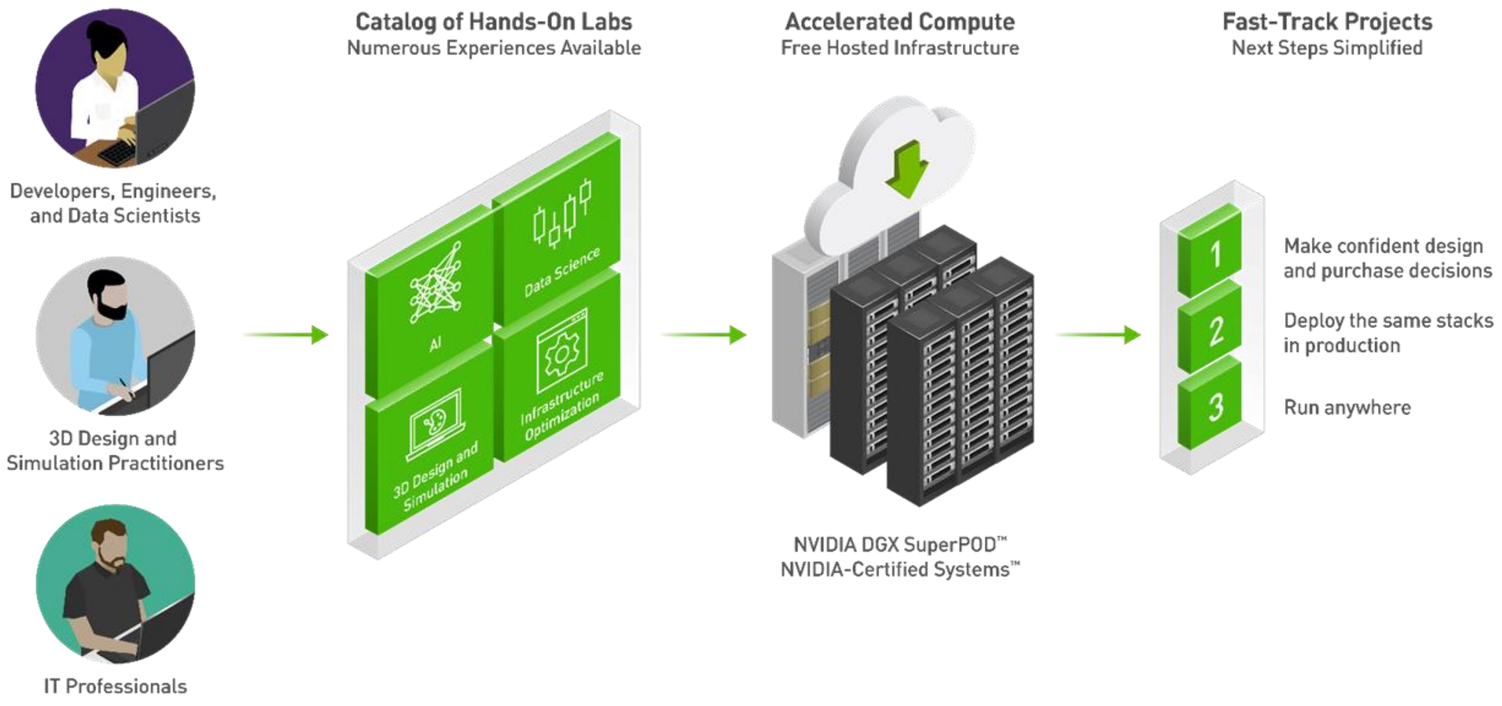
History of C++


C WITH CLASSES WAS FIRST IMPLEMENTED
1979
C WITH CLASSES WAS REFERENCE MANUAL PUBLISHED
1982
NEW OFFICIAL NAME C++ WAS GIVEN
1984
COMMERICALLY RELEASED - THE FIRST EDITION
1985
RELEASE OF C++ 2.0
1989
C++ 98 ISO STANDARD RELEASE
1997
3RD EDITION OF C++ GOT RELEASED
1998
C++03 GOT RELEASED, WHICH CONTAINED MAJOR BUGFIXES OF C++98
2003
C++ 0X GOT RELEASED
2009
C++11 GOT RELEASED WHICH WAS A MAJOR REVISION
2011
C++14 WAS RELEASED, WHICH WAS A SMALL IMPROVEMENT ON C++11
2014
C++17 RELEASED, A MAJOR REVISION
2017
C++20 IS BEING DEVELOPED
2018


LAMMPS
VASP
QCD
GEANT4
HPC SDK
MICROSCOPES
RAPIDS
PINN
FLEET COMMAND
BASE COMMAND
SuperPOD
LIGHT SOURCES
GPU
CPU
DPU
NVLINK
HGX
DGX
HOLOSCAN
MODULUS
BERT
TELESCOPES
OVX
EGX
AGX
JAX
GNN
PARTICLE ACCELERATORS
MAGNUM I/O
QODA
OMNIVERSE
CUQUANTUM
QUANTUM COMPUTERS
DIGITAL TWINS

NVIDIA LAUNCHPAD

Instantly experience end-to-end workflows for AI, data science, 3D design collaboration, and more



Get Started at nvidia.com/launchpad