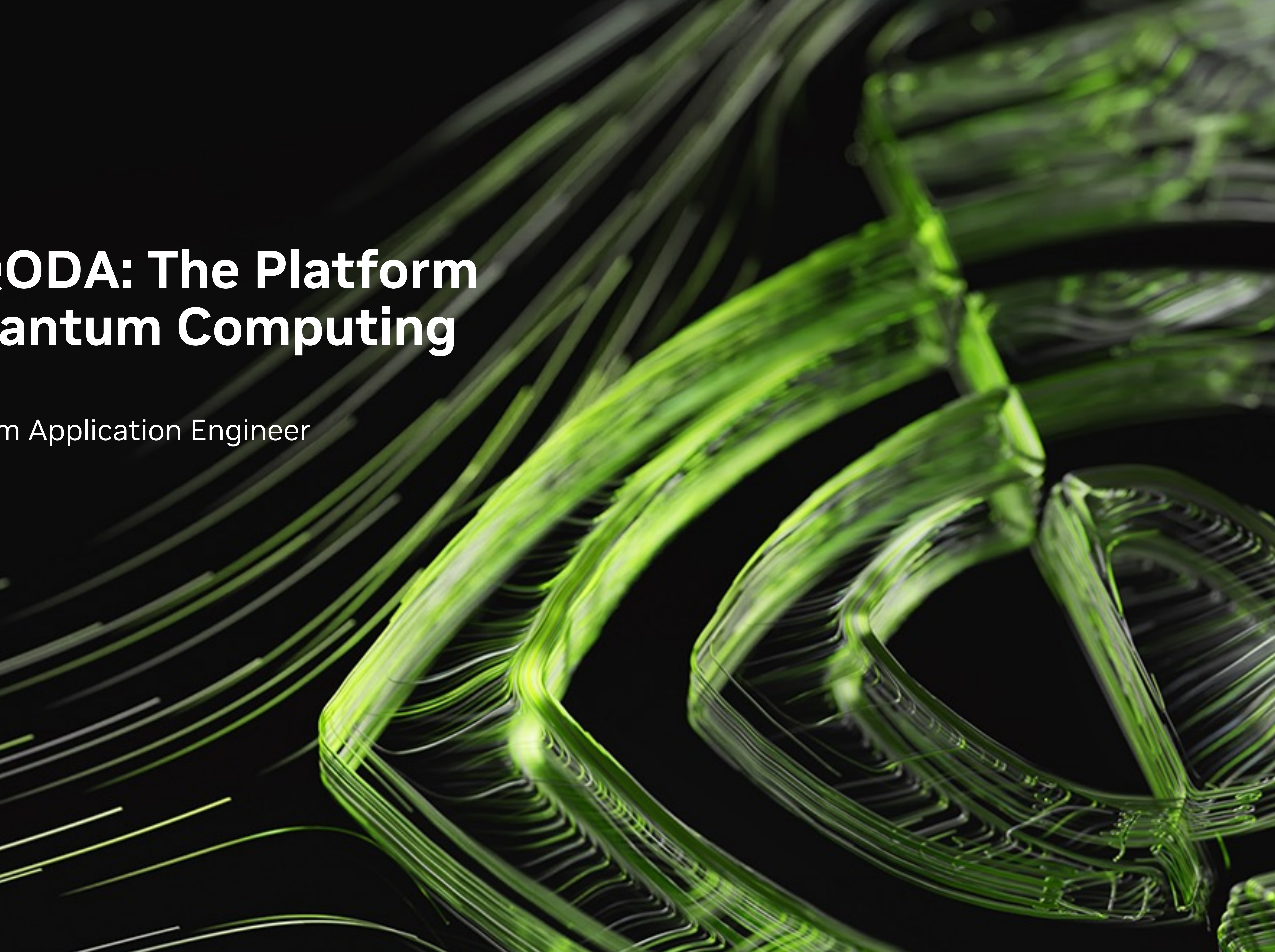
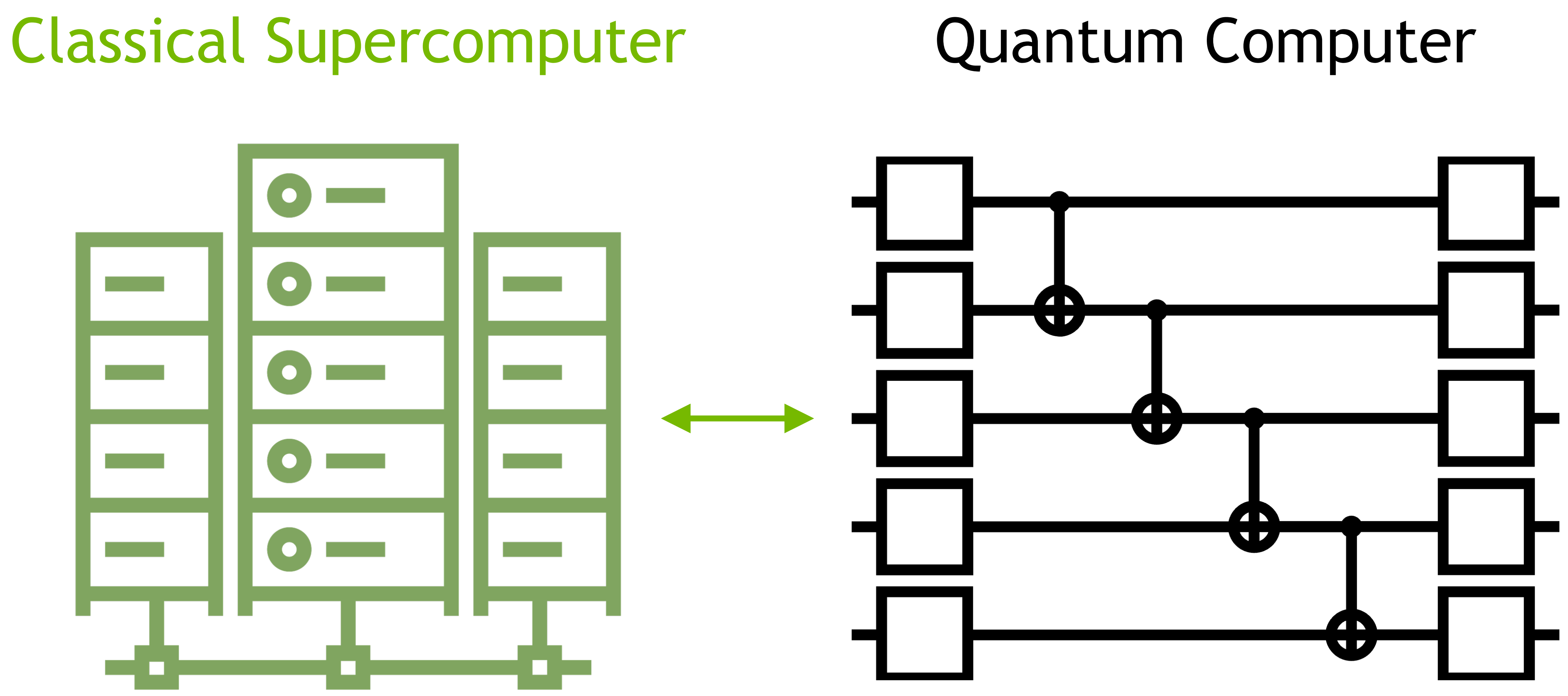
Introducing QODA: The Platform for Hybrid Quantum Computing

Zohim Chandani, Quantum Application Engineer



ECOSYSTEM GAPS LIMIT THE PROGRESS OF HYBRID QUANTUM APPLICATIONS



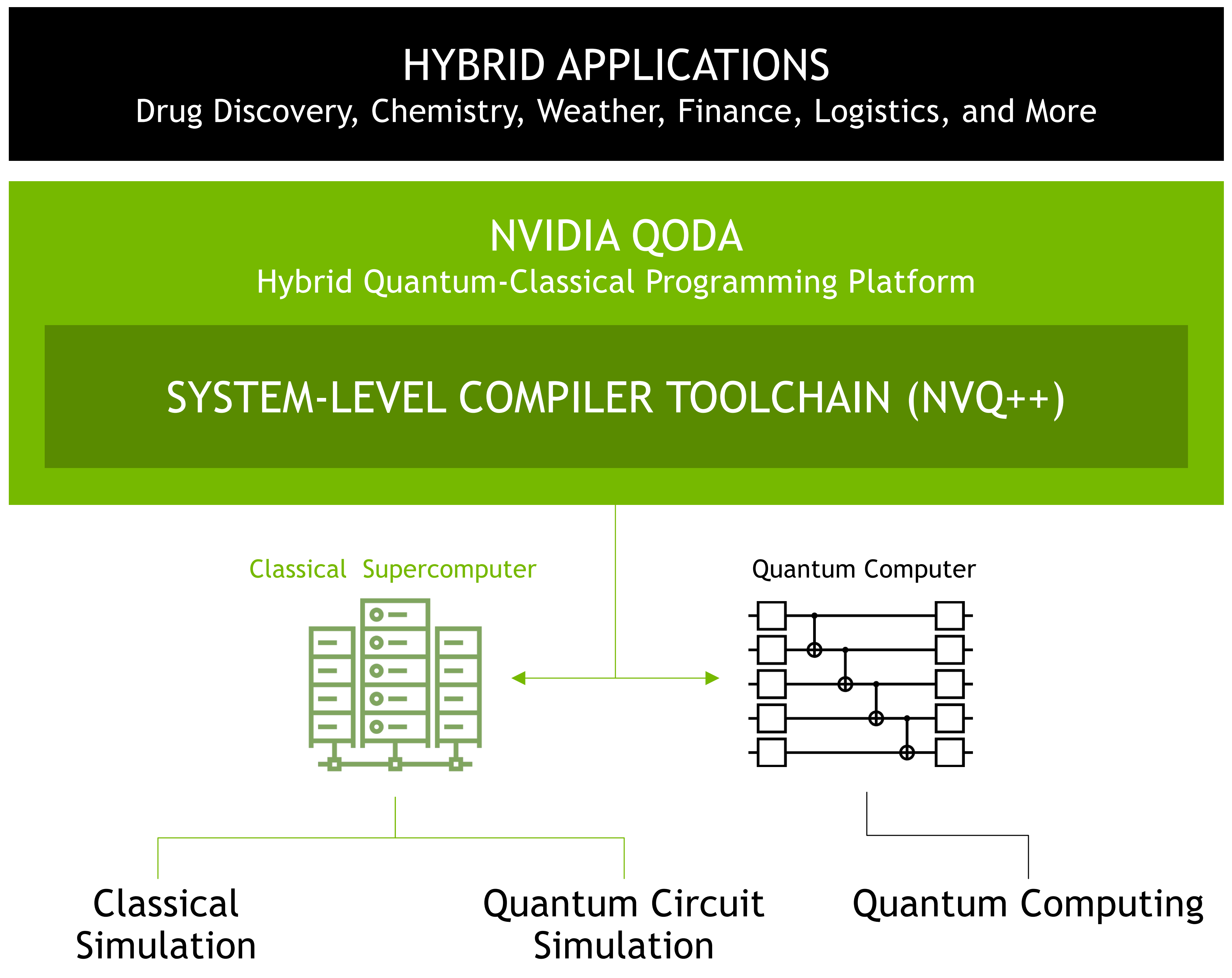
ECOSYSTEM CHALLENGES

- No Performant Software Stack
- Not Accessible To Domain Scientists
- Hybrid System Bottlenecks

Introducing NVIDIA QODA

A Platform For Hybrid Quantum-classical Computing

NVIDIA QODA PLATFORM



QODA FEATURES

Supports any kind of QPU, emulated or physical

Compiler for hybrid systems

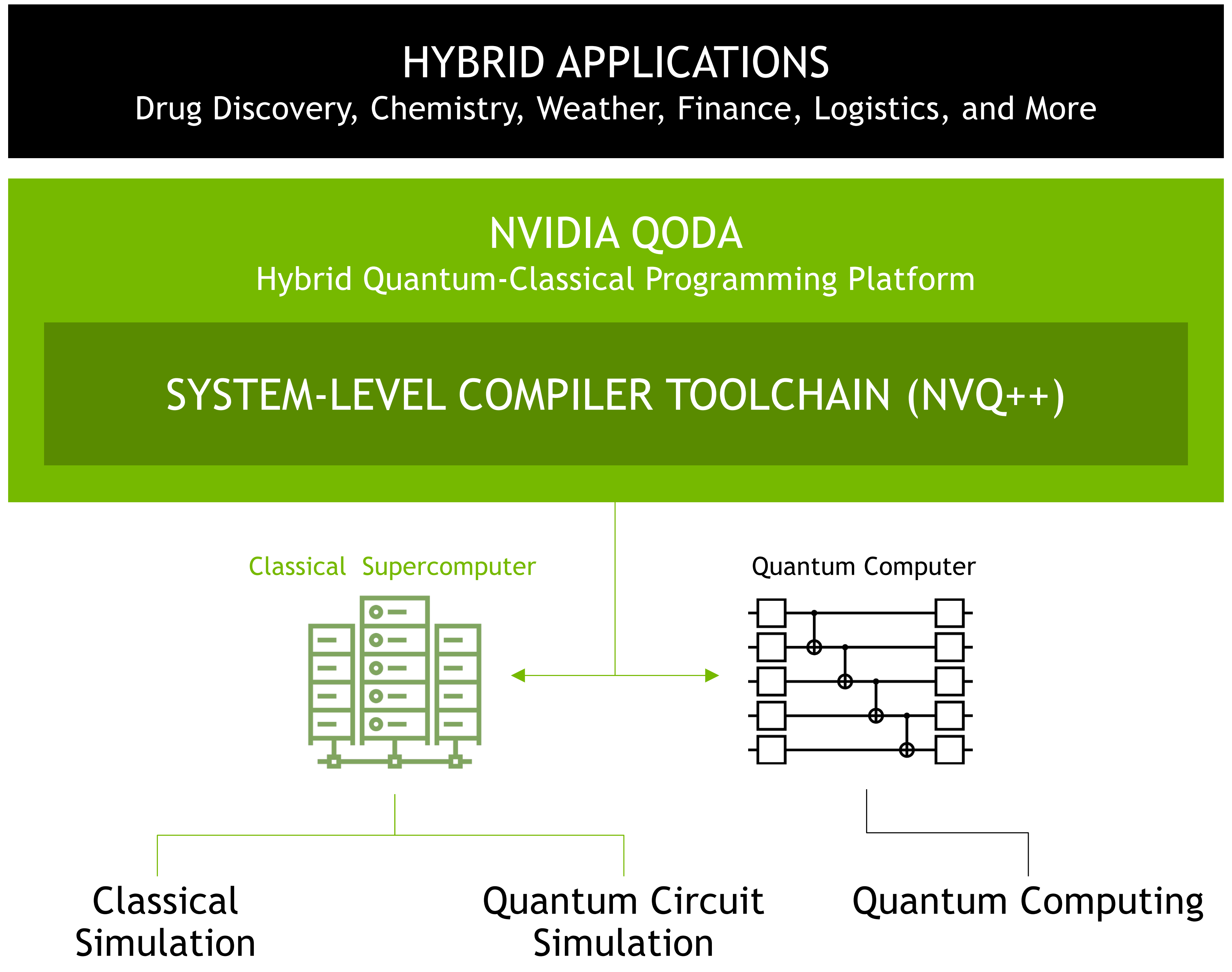
Open and interoperable with today’s applications

Single source C++ and Python programming model

Introducing NVIDIA QODA

Natively Hybrid And Interoperable With GPU Supercomputing

NVIDIA QODA PLATFORM



HYBRID APPLICATIONS

Drug Discovery, Chemistry, Weather, Finance, Logistics, and More

NVIDIA QODA

Hybrid Quantum-Classical Programming Platform

SYSTEM-LEVEL COMPILER TOOLCHAIN (NVQ++)

Interoperable with GPU Supercomputing

```
auto cnts = qoda::sample(q, …);
```

Standard Parallelism

```
std::sort(std::execution::par, …);
```

CUDA

```
kernel<<<…>>>(…);
cudaDeviceSynchronize();
```

OpenMP

```
#pragma omp target teams loop
for (…) …
```

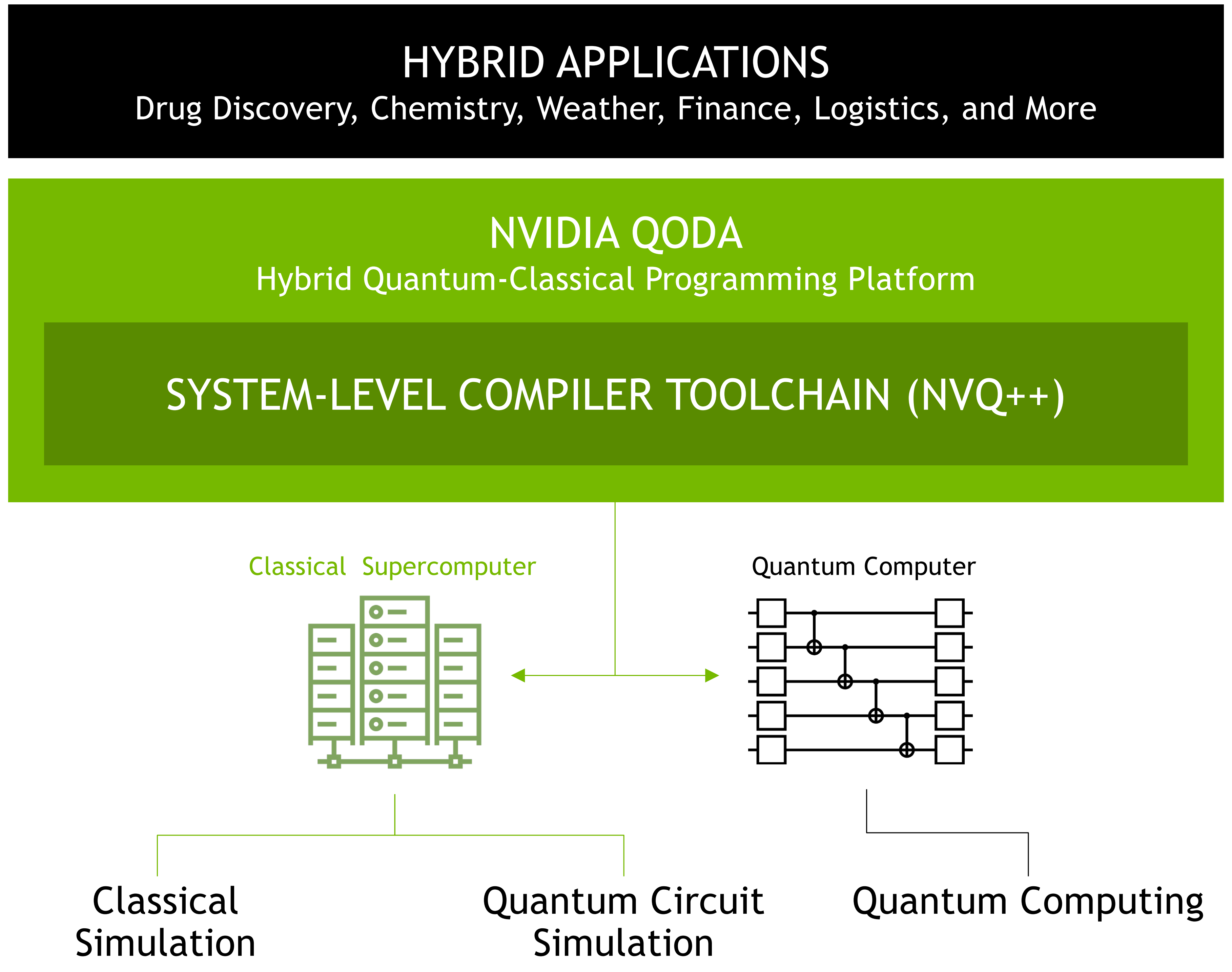
OpenACC

```
#pragma acc parallel loop
for (…) …
```

Introducing NVIDIA QODA

Adopted by Community's Global Leaders to Enable Quantum-Accelerated Applications

NVIDIA QODA PLATFORM



HYBRID APPLICATIONS

Drug Discovery, Chemistry, Weather, Finance, Logistics, and More

NVIDIA QODA

Hybrid Quantum-Classical Programming Platform

SYSTEM-LEVEL COMPILER TOOLCHAIN (NVQ++)

Classical Simulation

Quantum Circuit Simulation

Quantum Computing

QODA PLATFORM ECOSYSTEM

IQM

PASQAL

QUANTUM BRILLIANCE

QUANTINUUM

rigetti

XANADU

CLASSIQ

QCWARE

ZAPATA

JÜLICH Forschungszentrum

NERSC

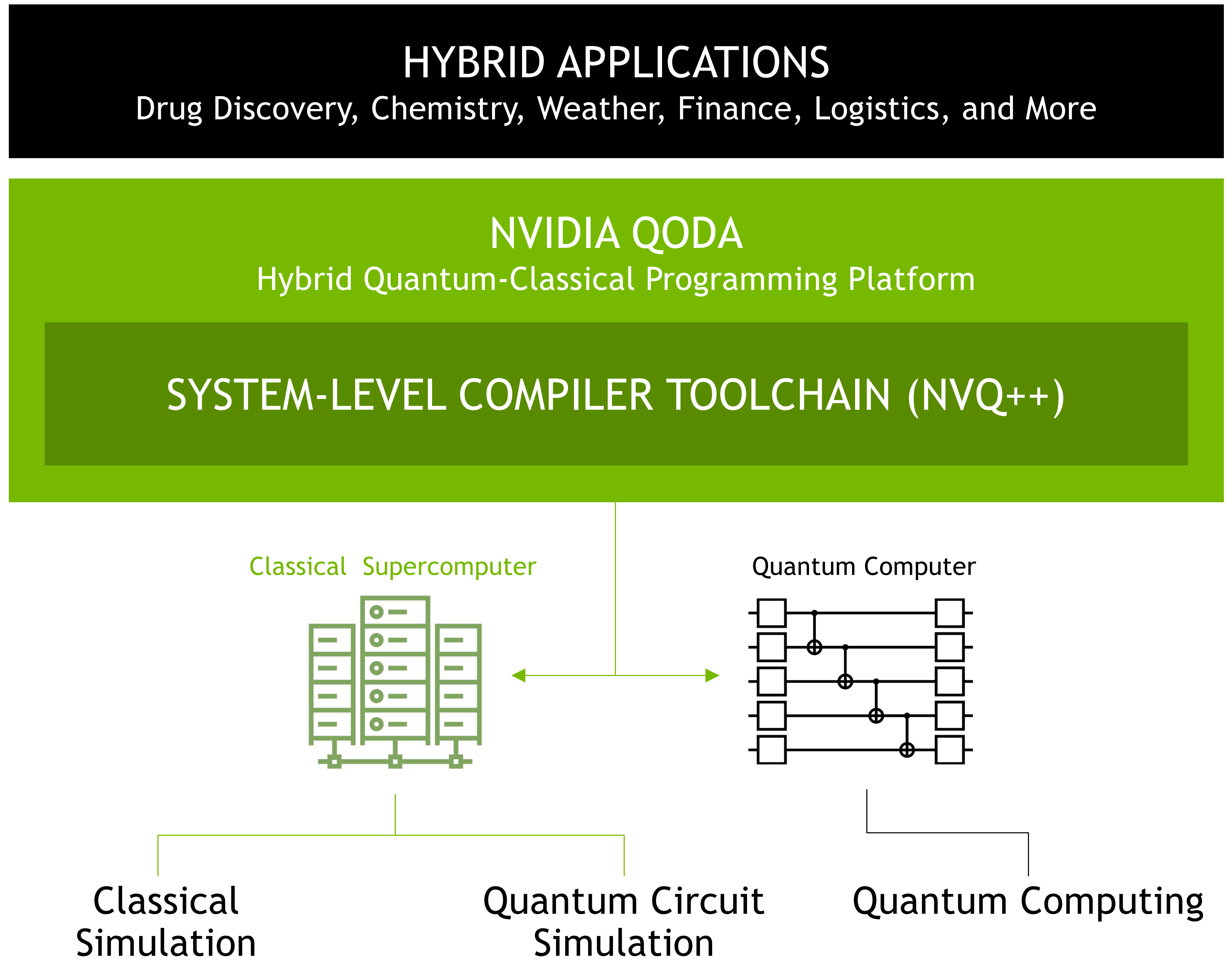
OAK RIDGE National Laboratory

RIKEN

Introducing NVIDIA QODA

Natively Hybrid And Interoperable With GPU Supercomputing

NVIDIA QODA PLATFORM



HYBRID APPLICATIONS

Drug Discovery, Chemistry, Weather, Finance, Logistics, and More

NVIDIA QODA

Hybrid Quantum-Classical Programming Platform

SYSTEM-LEVEL COMPILER TOOLCHAIN (NVQ++)

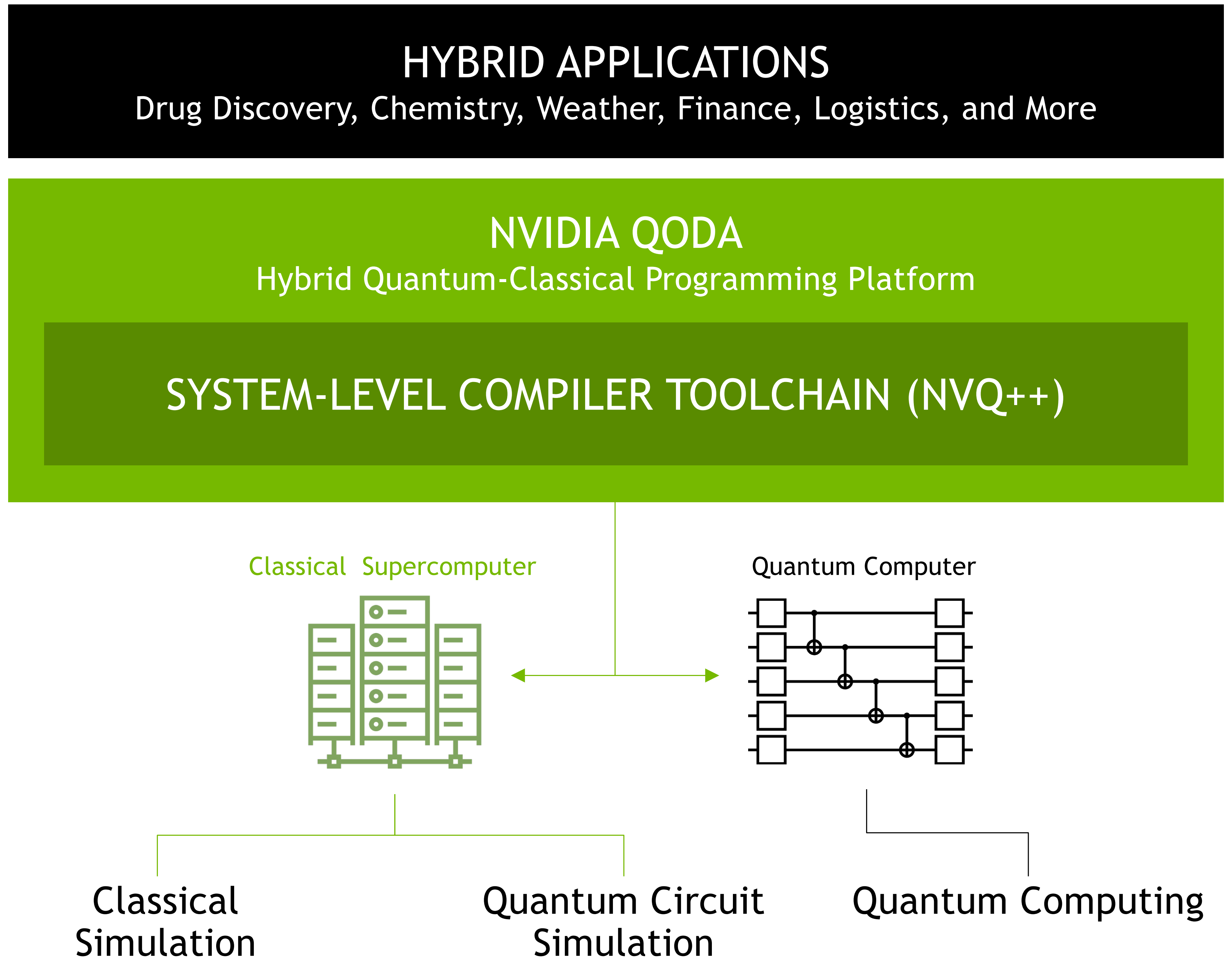
Familiar to Domain Scientists

```
// Define a QODA Quantum Kernel.
auto ansatz = [] (double theta) __qpu__ {
  qoda::qbit q, r;
  x(q);
  ry(theta, r);
  cnot(r, q);
};

// Define the Hamiltonian via Pauli tensor products.
qoda::spin_op H = 5.907 - 2.1433 * x(0) * x(1)
                - 2.1433 * y(0) * y(1)
                + .21829 * z(0) - 6.125 * z(1);

qoda::nlopt::cobyla opt

// Run Variational Quantum Eigensolver with 1 param.
auto [min_e, opt_p] = qoda::vqe(ansatz, H, opt, 1);
```

Introducing NVIDIA QODA

Delivering Unmatched Performance, Scalability, And Usability

NVIDIA QODA PLATFORM



HYBRID APPLICATIONS
Drug Discovery, Chemistry, Weather, Finance, Logistics, and More

NVIDIA QODA
Hybrid Quantum-Classical Programming Platform

SYSTEM-LEVEL COMPILER TOOLCHAIN (NVQ++)

Classical Supercomputer

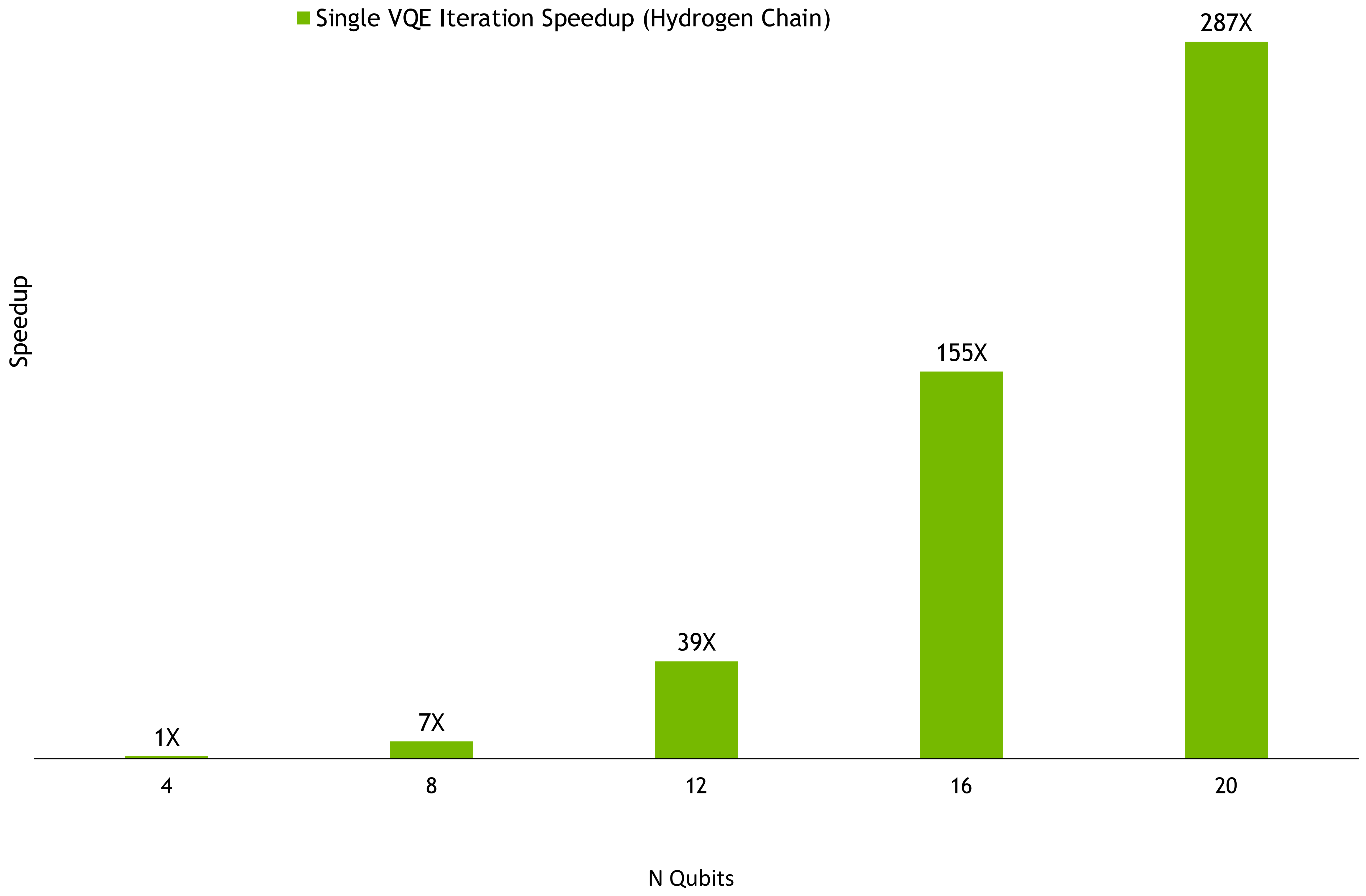
Quantum Computer

Classical Simulation

Quantum Circuit Simulation

Quantum Computing

ENGINEERED FOR PERFORMANCE AND SCALE

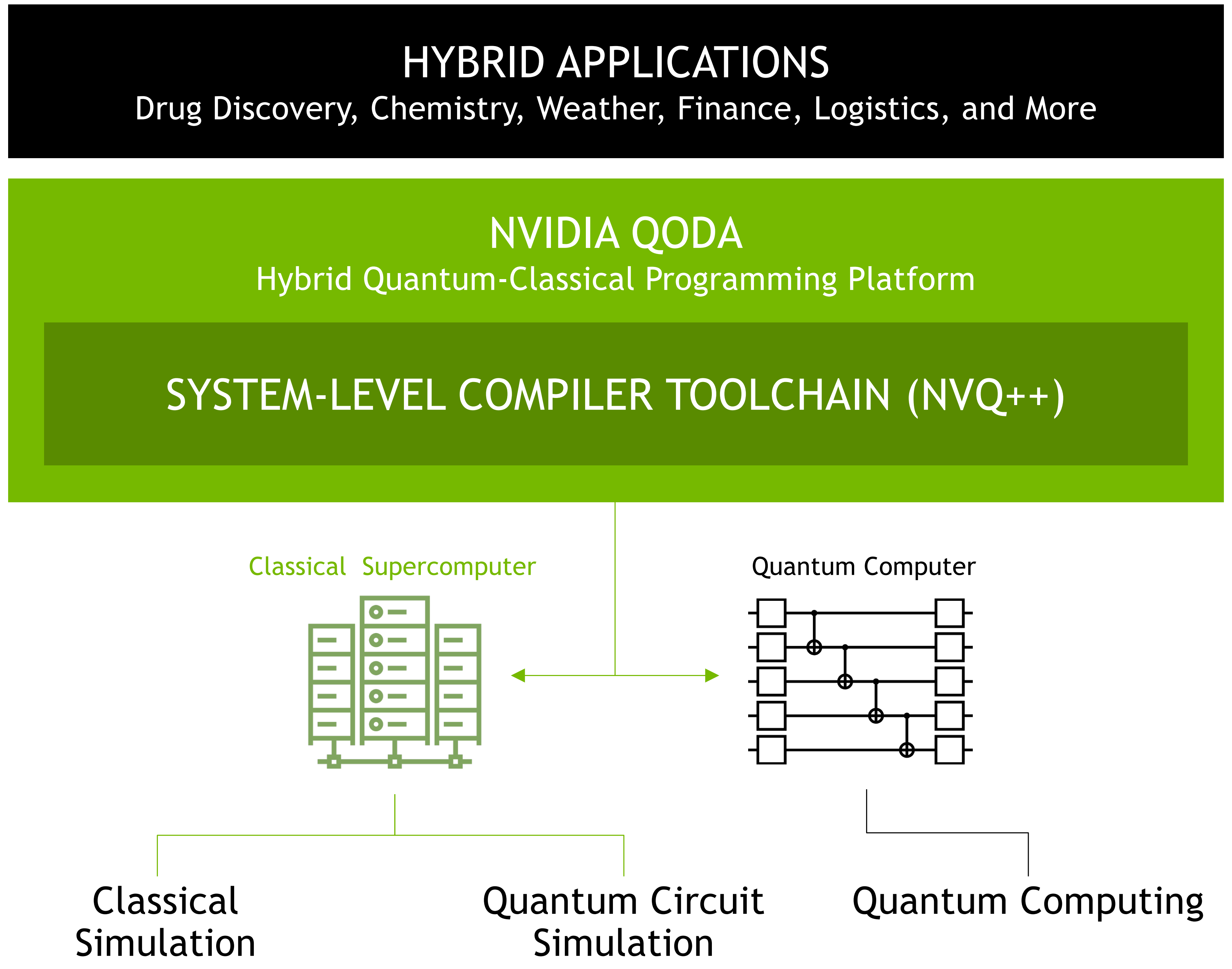


VQE: NVQ++/cuStateVec on A100 GPU vs Leading Pythonic Framework and SImulator with Thrust on A100 GPU

Introducing NVIDIA QODA

Natively Hybrid And Interoperable With GPU Supercomputing

NVIDIA QODA PLATFORM



HYBRID APPLICATIONS
Drug Discovery, Chemistry, Weather, Finance, Logistics, and More

NVIDIA QODA
Hybrid Quantum-Classical Programming Platform

SYSTEM-LEVEL COMPILER TOOLCHAIN (NVQ++)

Classical Simulation

Quantum Circuit Simulation

Quantum Computing

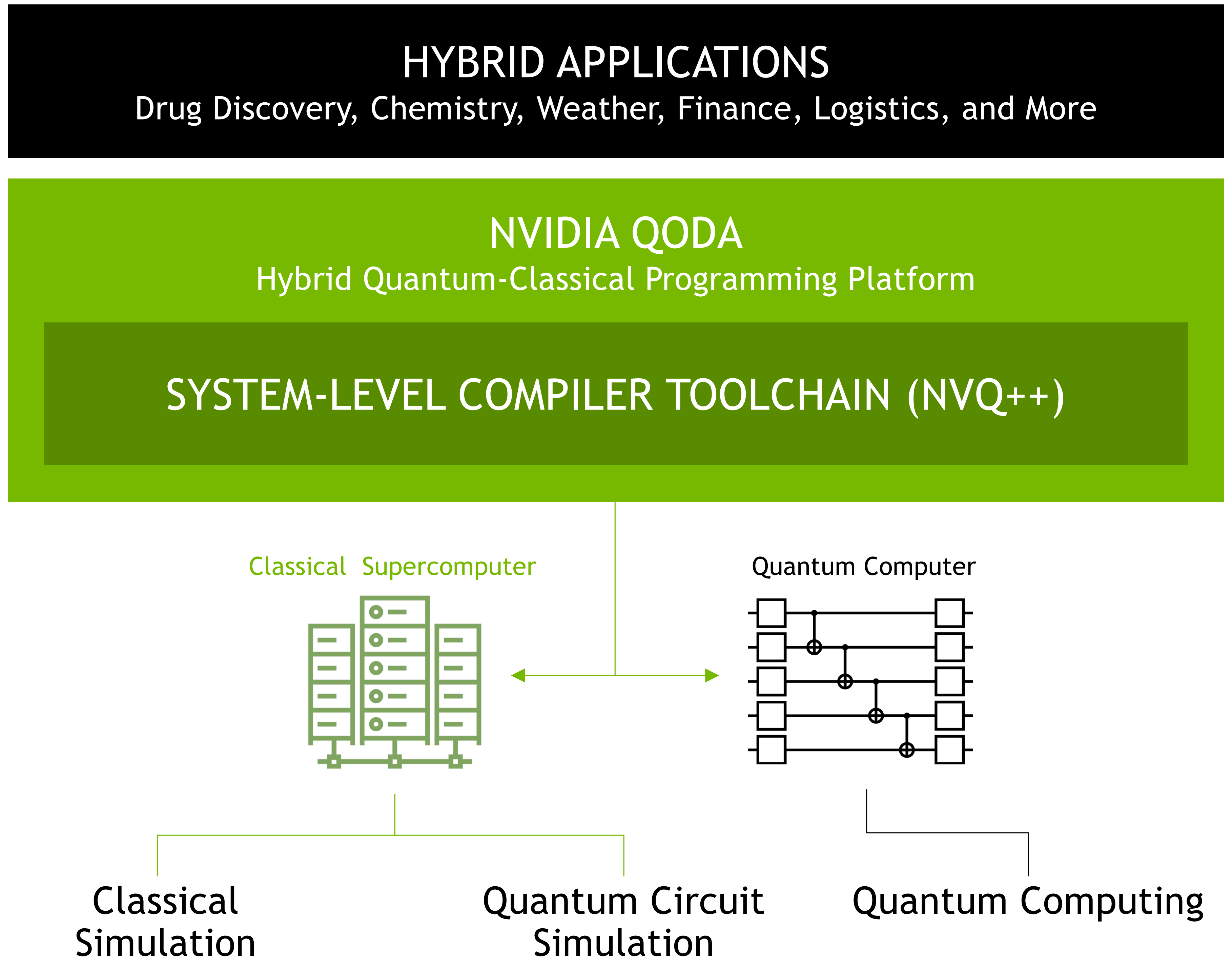
Interoperable with GPU Supercomputing

```
// Compute expectation values with QPU.
qoda::spin_op h = ...;
std::vector<double> sig_exps;
for (auto& pauli_op : generate_pauli_permutations(h.n_qubits()))
  sig_exps.push_back(qoda::observe(qite, pauli_op, h.n_qubits()));



...
// Compute LU Factorization of S_mat on the GPU.
auto dim = std::pow(2, h.n_qubits());
cusolverDnXgetrf(handle, params, dim, dim, CUDA_C_64F, S_mat,
                 lda, NULL, CUDA_C_64F, buffer_on_device,
                 bytes_on_device, buffer_on_host,
                 bytes_on_host, info);
```

Introducing NVIDIA QODA

Enabling Innovative Quantum Systems Research

NVIDIA QODA PLATFORM



HYBRID APPLICATIONS

Drug Discovery, Chemistry, Weather, Finance, Logistics, and More

NVIDIA QODA

Hybrid Quantum-Classical Programming Platform

SYSTEM-LEVEL COMPILER TOOLCHAIN (NVQ++)

Classical Supercomputer

Quantum Computer

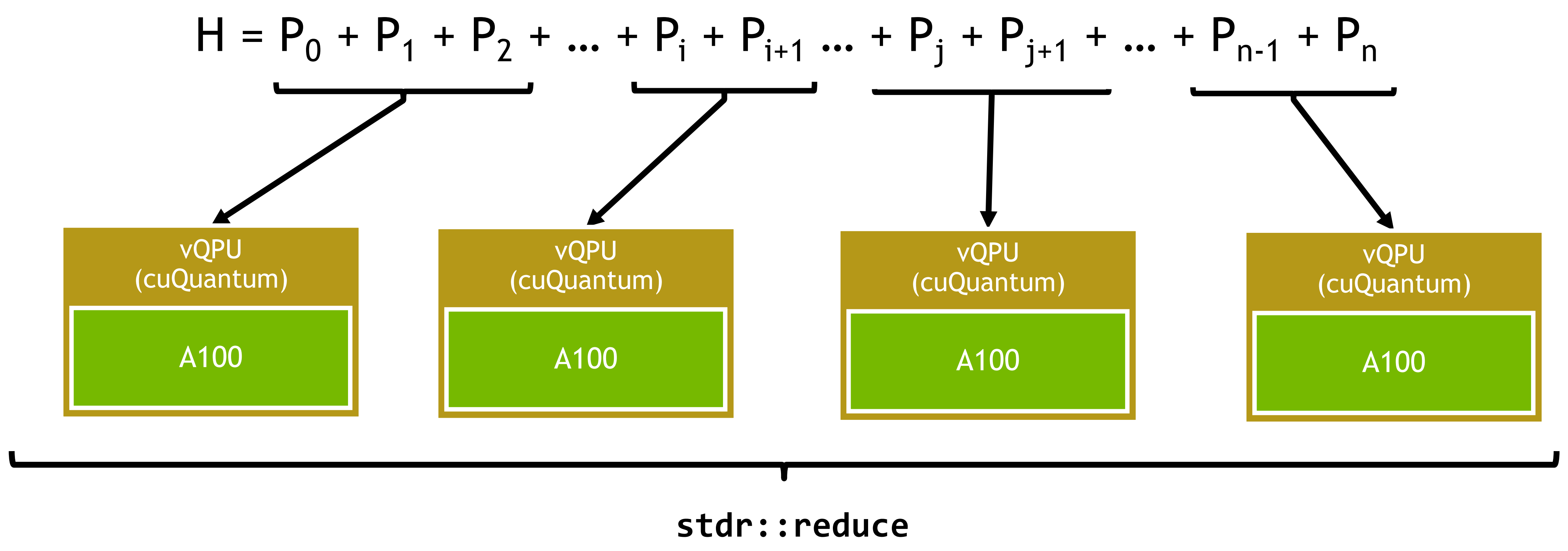
Classical Simulation

Quantum Circuit Simulation

Quantum Computing

Experiment on Future Quantum Systems

```
std::vector<qoda::observe_sender<double>> subs;
for (auto qpu : qoda::all_qpus()) {
 auto sub_H = H.subspan(qpu.idx() * terms_per_qpu, (qpu.idx() + 1) *
terms_per_qpu);
  subs.emplace_back(
    qoda::observe_async(qpu, sub_H, ansatz, ...));
}
auto sum = stdr::reduce(std::execution::par, qoda::when_all(subs), 0.0);
```



$$H = P_0 + P_1 + P_2 + \dots + P_i + P_{i+1} \dots + P_j + P_{j+1} + \dots + P_{n-1} + P_n$$

vQPU (cuQuantum) A100

vQPU (cuQuantum) A100

vQPU (cuQuantum) A100

vQPU (cuQuantum) A100

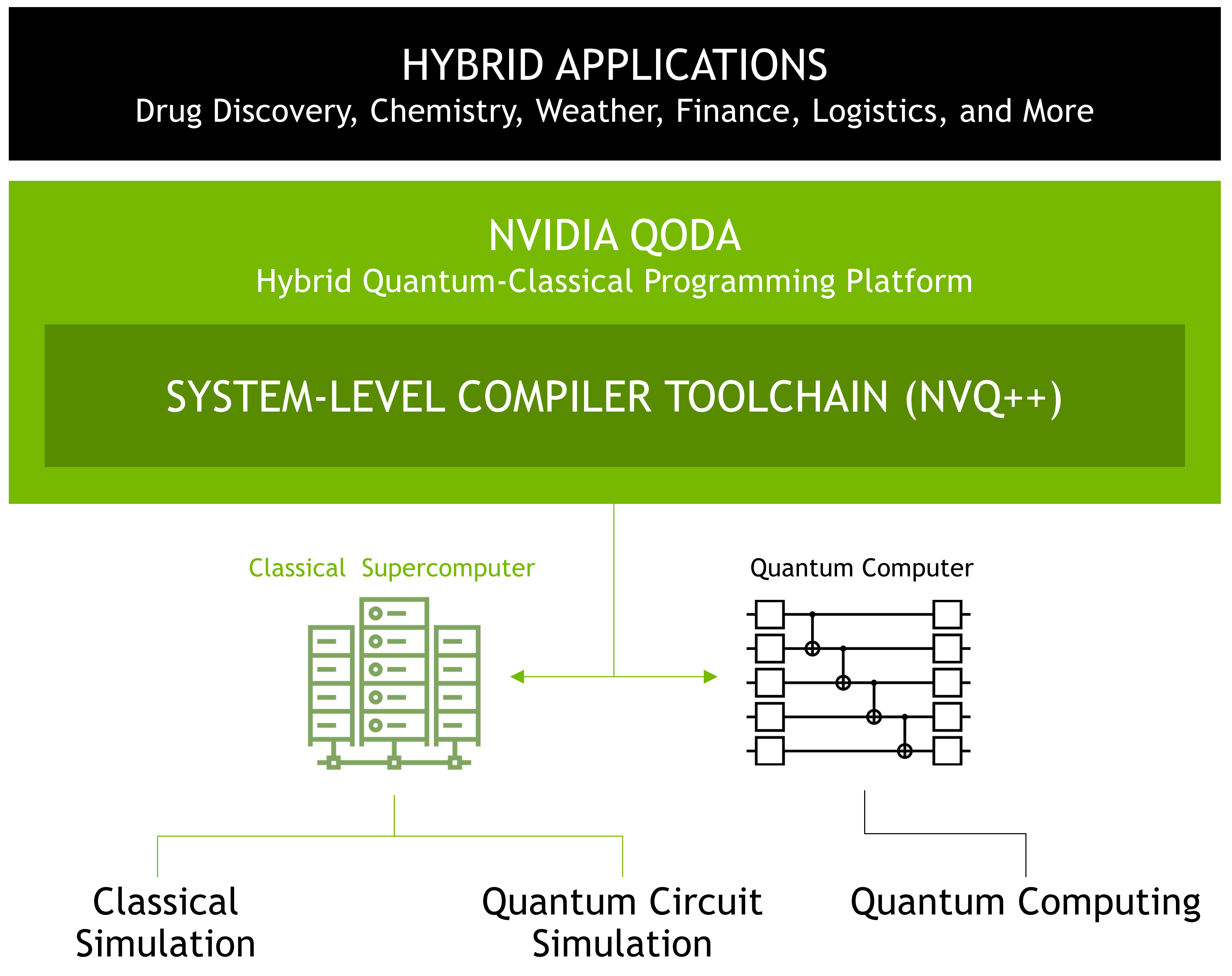
`stdr::reduce`

VQE: NVQ++/cuStateVec on A100 GPU vs Leading Pythonic Framework and SImulator with Thrust on A100 GPU

Introducing NVIDIA QODA

Enabling Innovative Quantum Systems Research

NVIDIA QODA PLATFORM

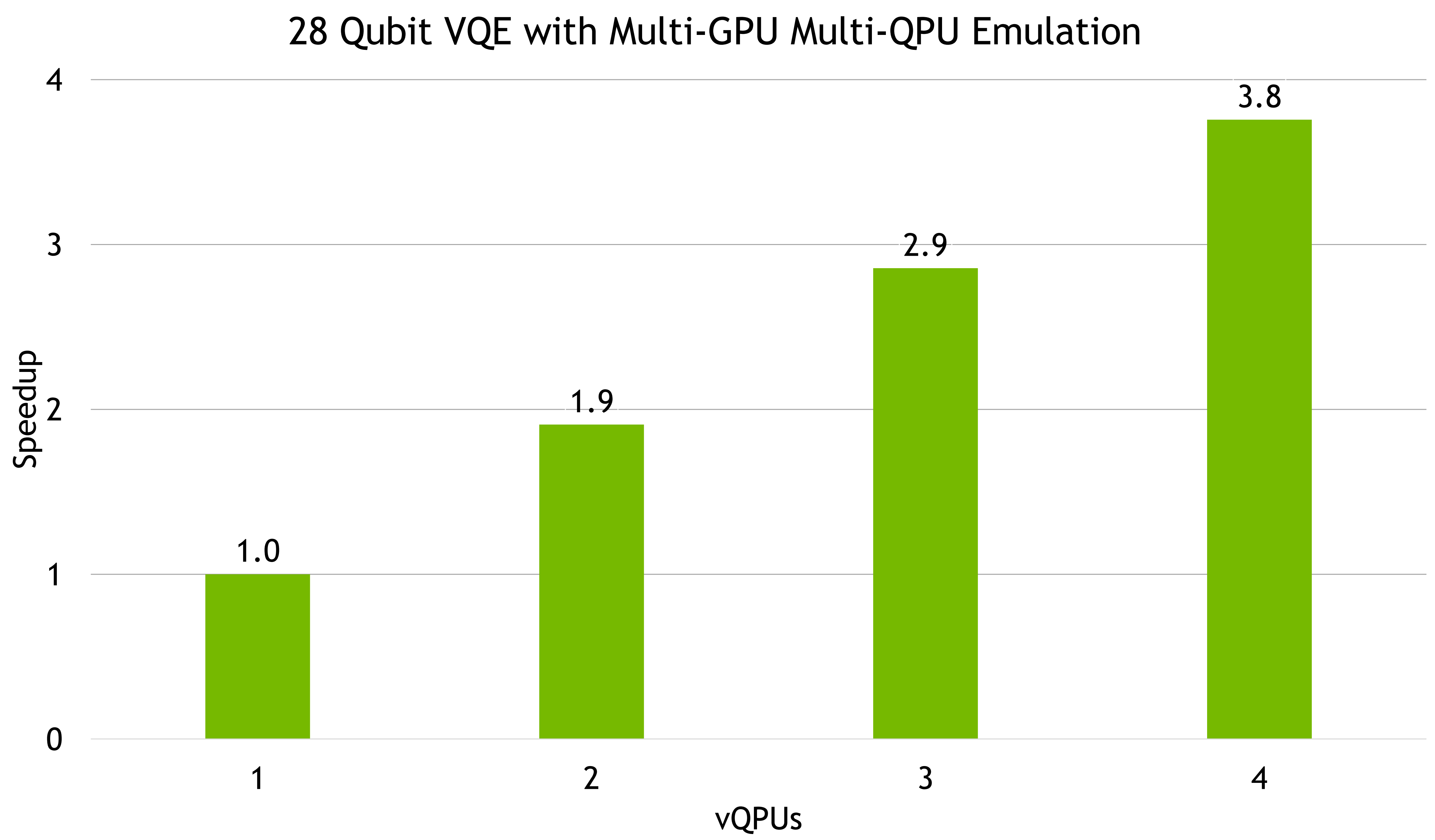


HYBRID APPLICATIONS
Drug Discovery, Chemistry, Weather, Finance, Logistics, and More

NVIDIA QODA
Hybrid Quantum-Classical Programming Platform

SYSTEM-LEVEL COMPILER TOOLCHAIN (NVQ++)

Experiment on Future Quantum Systems



28 Qubit VQE with Multi-GPU Multi-QPU Emulation

VQE: NVQ++/cuStateVec on A100 GPU vs Leading Pythonic Framework and SImulator with Thrust on A100 GPU

NVIDIA®