Hewlett Packard Enterprise

Full Stack Framework for High Performance Quantum-Classical Computing

Xin Zhan, K. Grace Johnson, Masoud Mohseni,
Soumitra Chatterjee, Aniello Esposito, Barbara Chapman,
Kirk Bresniker, Marco Fiorentino, Ray Beausoleil

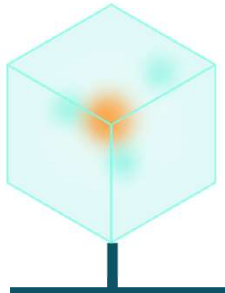
Hewlett Packard Enterprise

July 25, 2025

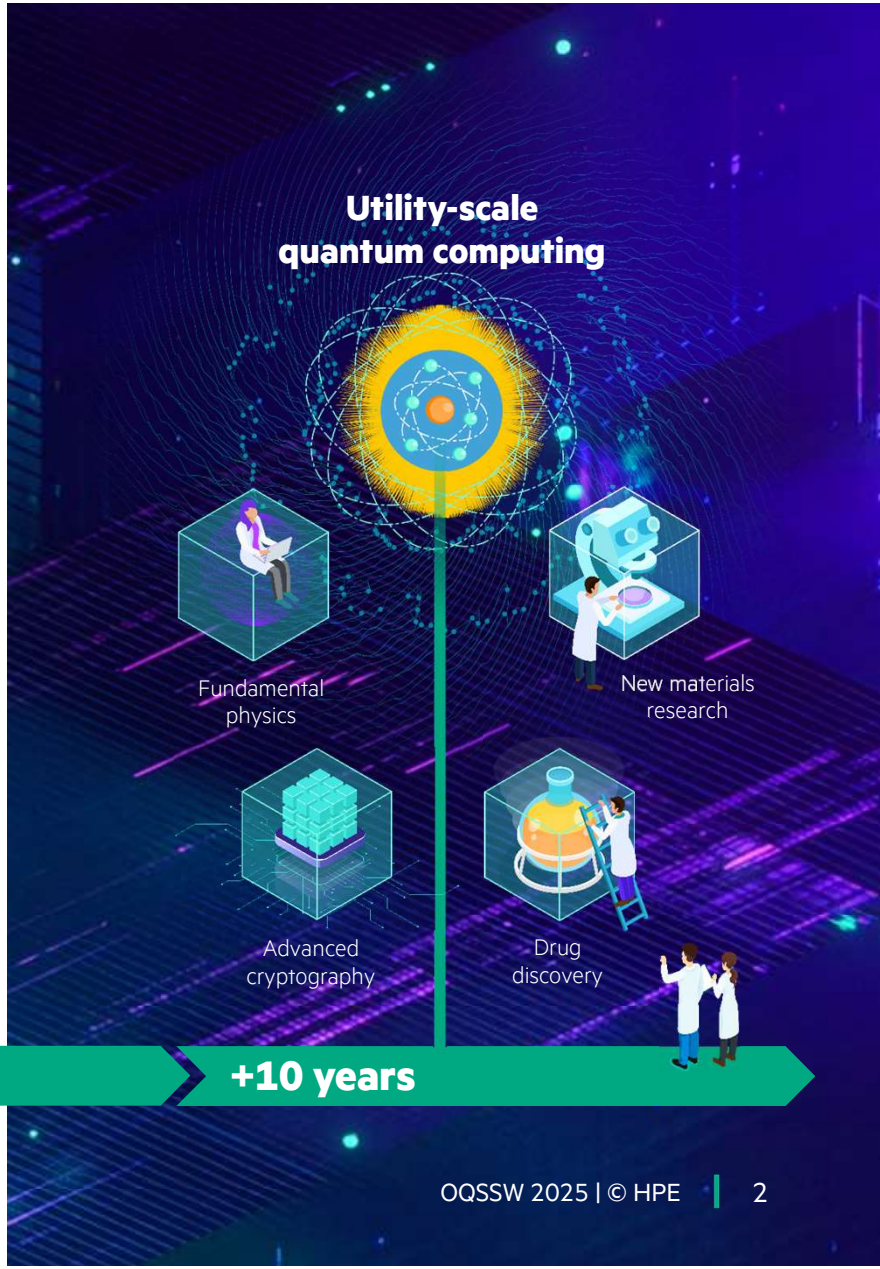
Toward utility-scale quantum computing

- Quantum Computing as a tool to tackle some of our most challenging problems and targeting applications beyond the limit of classical HPC.

- Today: limited to ~100 noisy physical qubits (1 logical qubit)
  - ~10M physical qubits needed for utility-scale application
  - ~100k physical qubits on a single quantum processors (QPUs)
- **Need to create a network of ~100 QPUs**



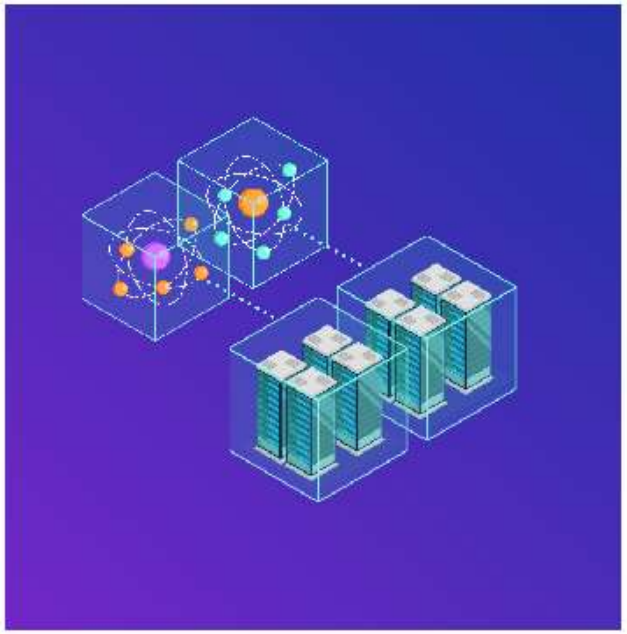
We need QC-HPC integration for scaling



Today | **+5 years** | **+10 years**

Challenges of building a utility-scale quantum computer

Software and HPC Integration

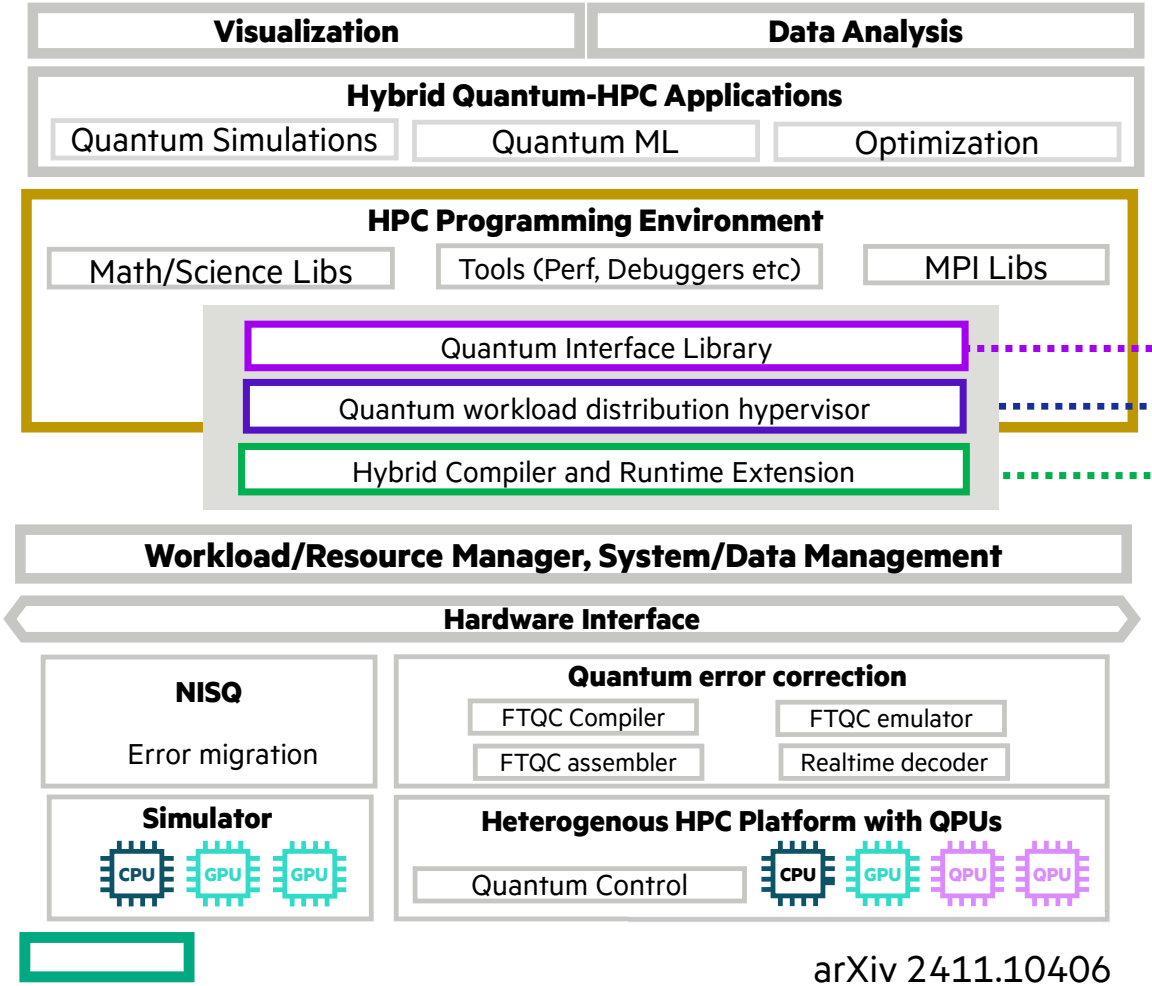


Quantum processors (QPUs) will be finite in size and likely smaller than the one million qubits necessary for utility

Error correction and hybrid quantum-classical algorithms require classical HPC and low latency to QPUs

The quantum computer must be easily programmable by the HPC application end user

Quantum-HPC Full Stack Framework: QPU and Quantum SDK agnostic



A. Quantum interface library

- Replace circuit simulation with API
- Connect and experiment with multiple commercial quantum SDKs
- Enable C/C++/Fortran HPC applications to invoke quantum kernels from vendor-specific quantum SDKs

B. Adaptive circuit knitting

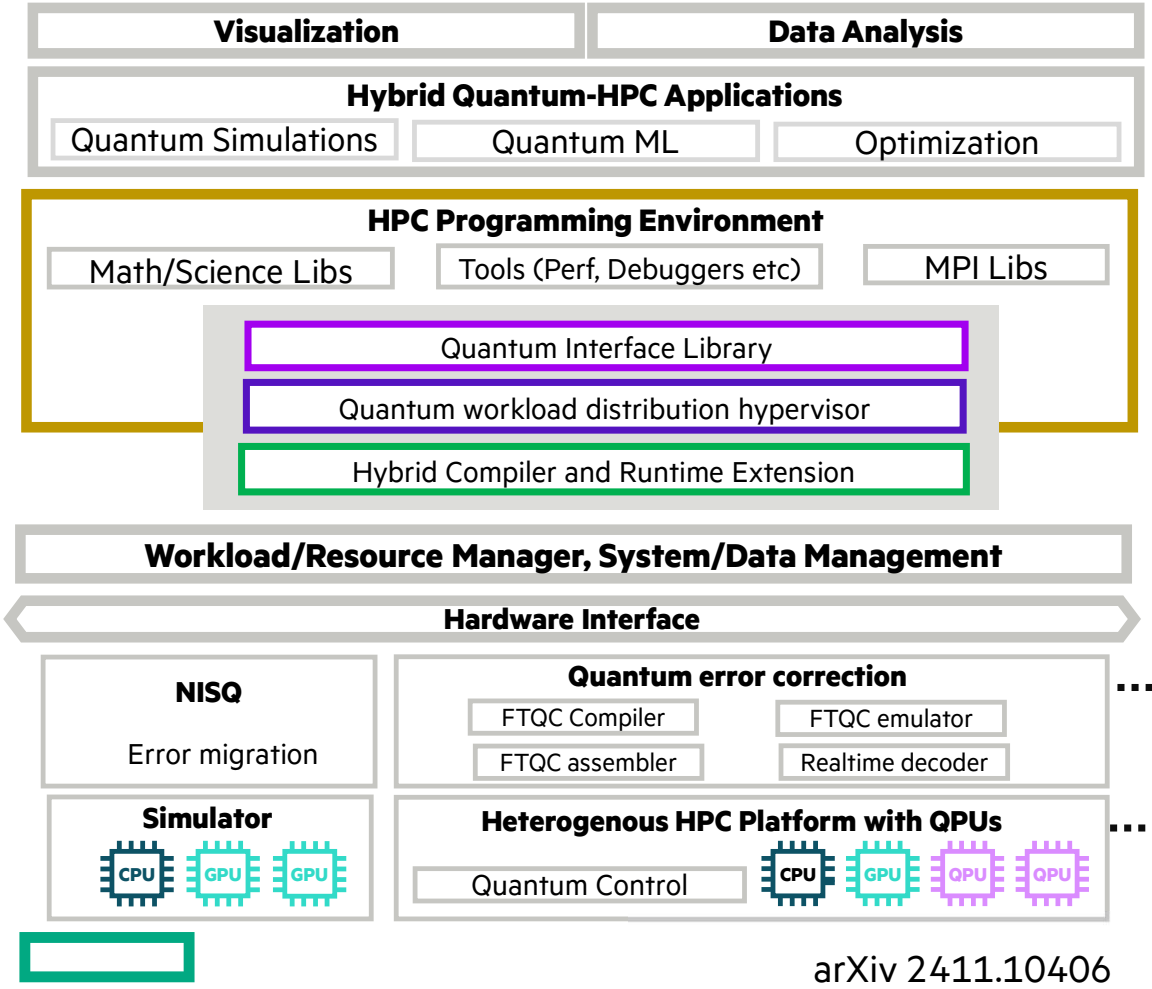
- Efficiently partition and distribute quantum circuits using classical communication
- Overcome exponential classical post-processing of standard circuit knitting techniques

C. Cray Programming Environment Extension

- Hybrid quantum compilation and runtime extensions to CPE
- Designed for compatibility, performance, and scalability
- Support full range of heterogeneous HPC platforms and hardware architectures
- Compatibility with multiple quantum compilers

arXiv 2411.10406

Quantum-HPC Full Stack Framework: QPU and Quantum SDK agnostic



Quantum operating system

- FTQC compiler, emulator, and assembler use hardware noise profiles to execute FTQC programs using TopQAD
- Compiler synthesizes optimized circuits consist of multi-qubit lattice surgeries
- Emulator uses hardware noise models to infer logical error rates
- Assembler performs zoning and memory allocation and schedules lattice surgeries
- Real-time decoder on DGX Quantum

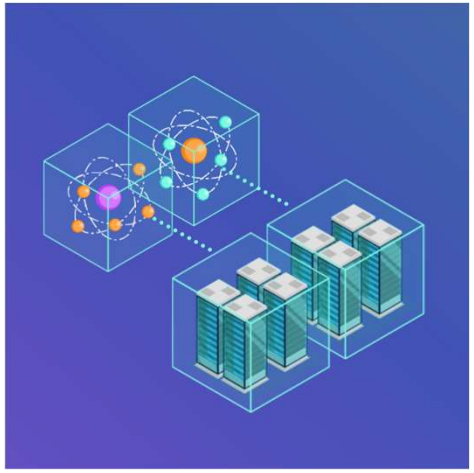
Heterogenous hardware platform

- Coprocessors: CPU/GPU, Quantum Processing Unit (QPU), Probabilistic Processing Unit (PPU) including FPGA and custom-design ASIC with HPC interconnects

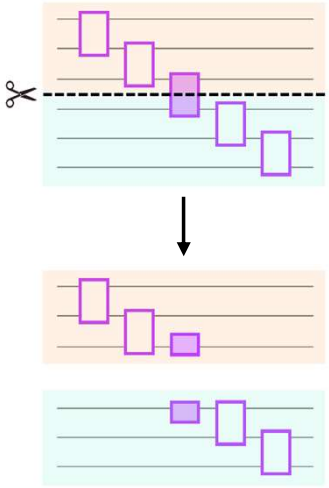
arXiv 2411.10406

Requirement for Distributed Quantum Computation

For hybrid quantum-classical computing to scale, we need efficient methods for **partitioning** and **distributing quantum workloads**.

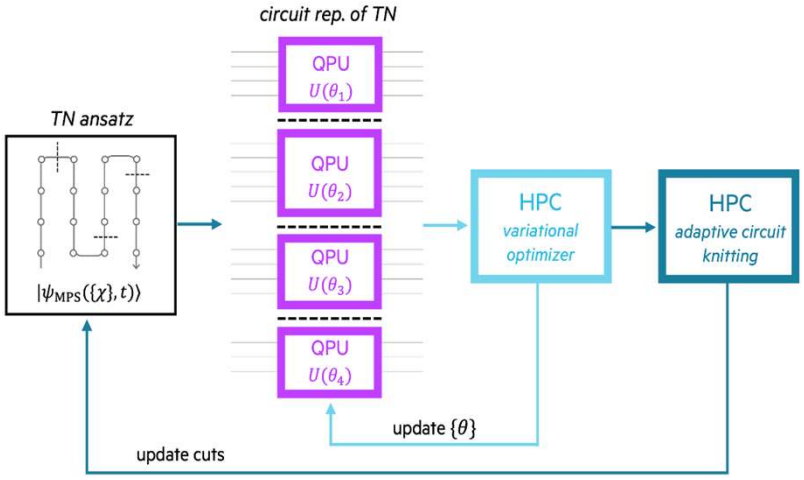


Circuit knitting is a promising method for partitioning quantum circuits, but requires a classical overhead that **scales exponentially** with the number of cuts.



We are developing **adaptive circuit knitting algorithms** that can significantly reduce this overhead:

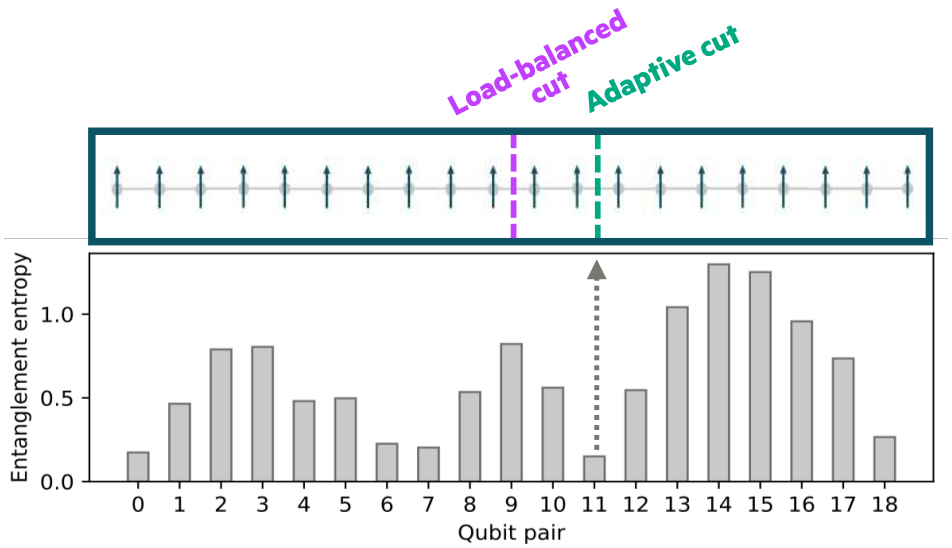
- Based on tensor network techniques
- Use entanglement measures to determine best cuts as circuit evolves



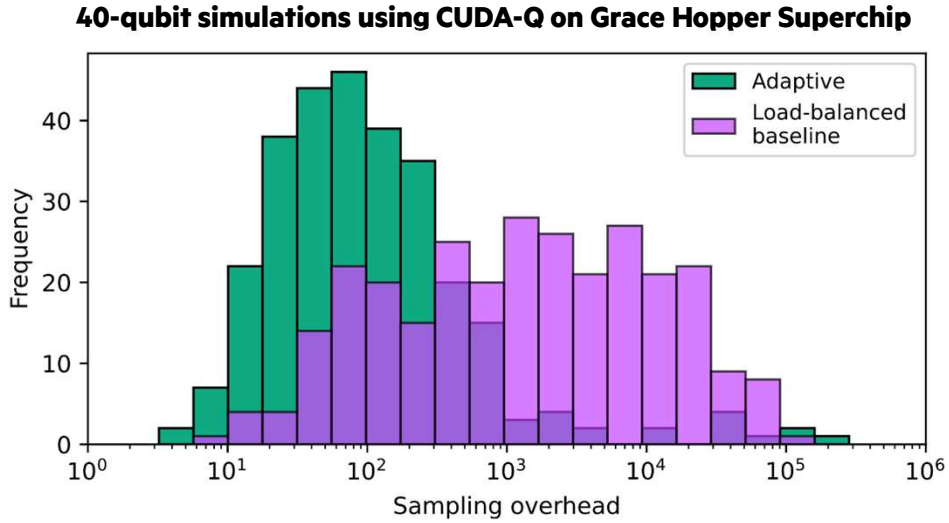
Quantum Workload Distribution

Simulating quantum spin systems

Efficient partitions
found via entropy measures



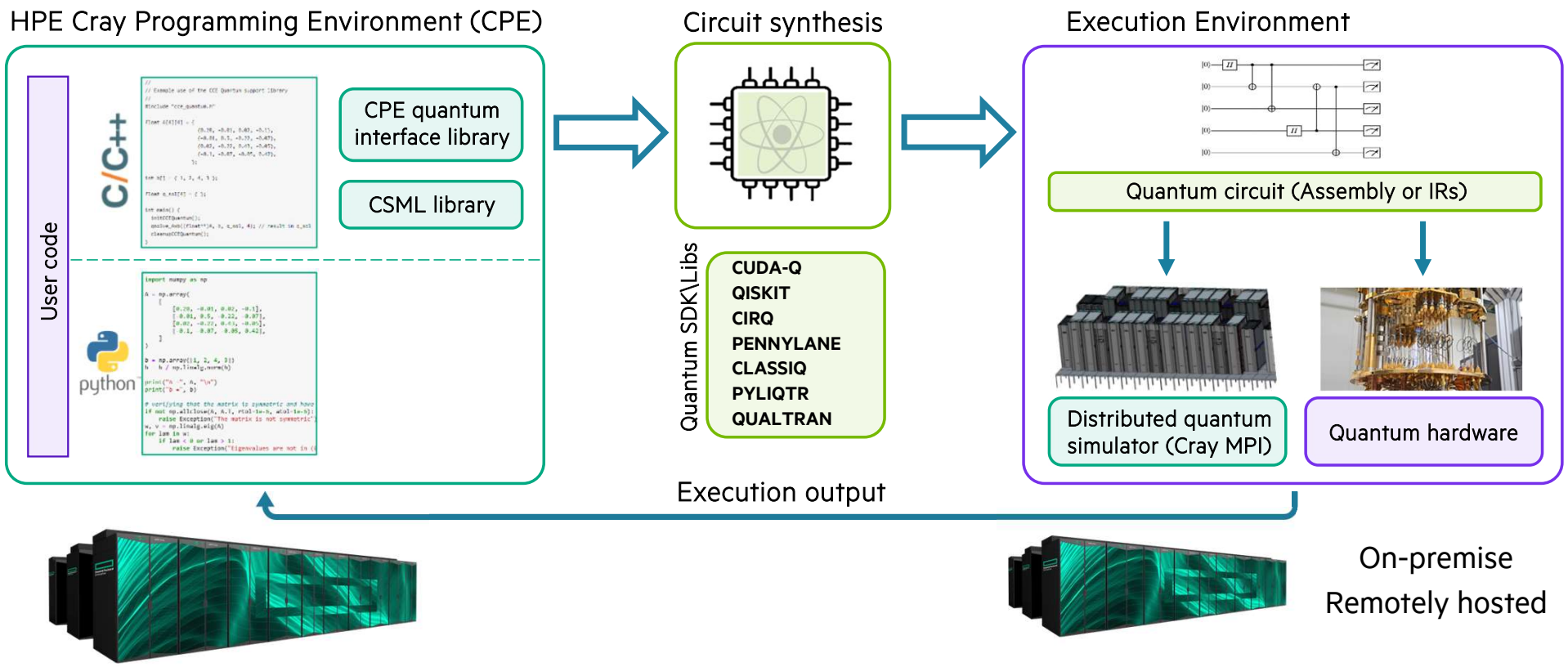
1-3 orders of magnitude improvement
in classical overhead



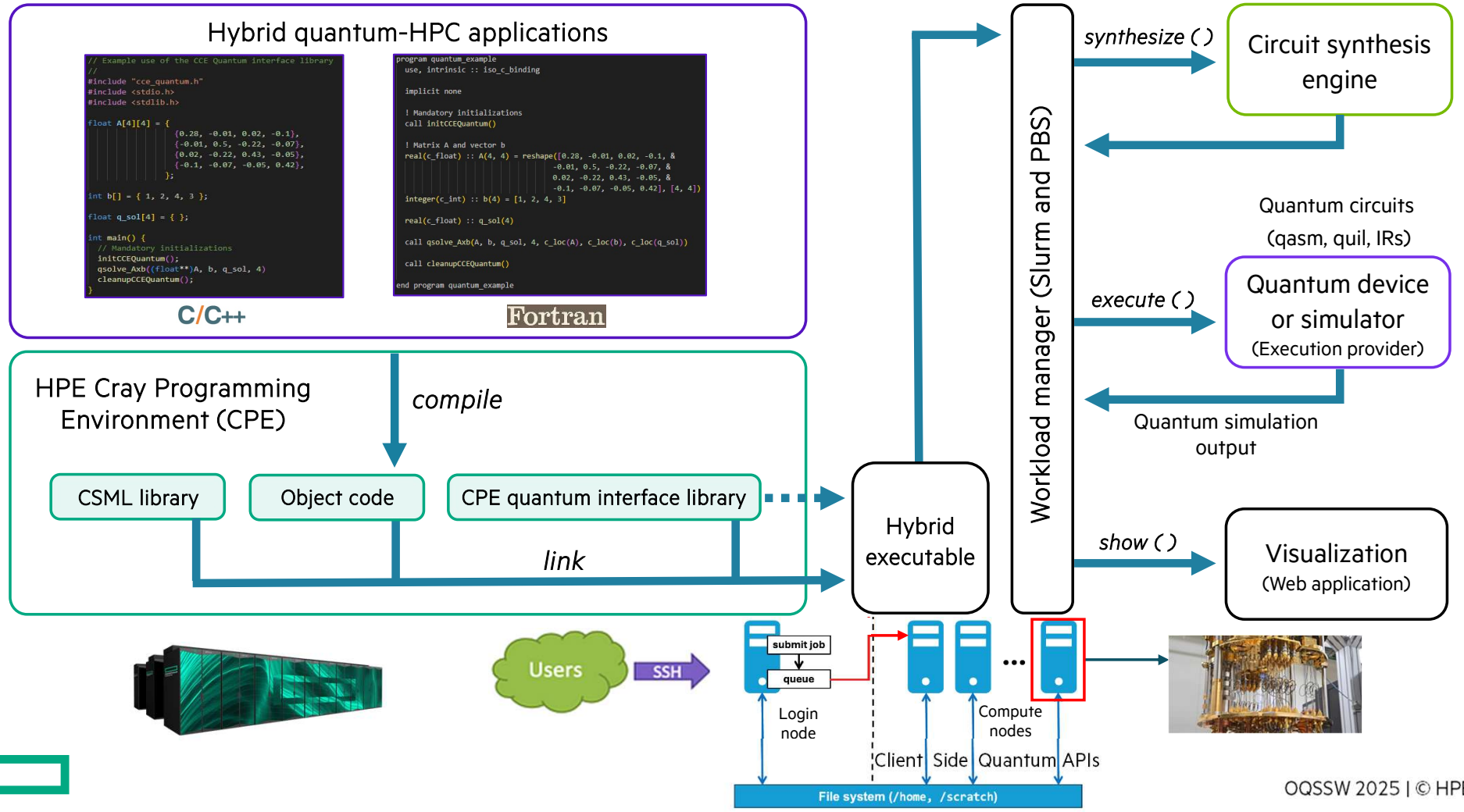
Joint work with NVIDIA

https://www.nvidia.com/en-us/on-demand/session/gtc25-dd73669/

Hybrid Quantum-HPC Workflow



Hybrid Quantum-HPC Programming and Execution Process



HPE Quantum-Classical Full Stack: Extensions and applications

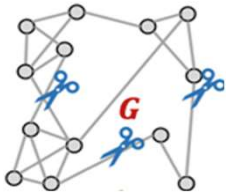
Programming Interface
Python and C/C++

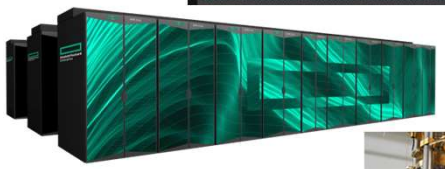
- HPE Cray EX with HPCM
- Two AMD® EPYC 7763 (Milan) CPU /node ~ 128 core /node
- Slingshot 11 network fabric
- HPE Cray MPI and Cray Programming Environment (NVIDIA® CUDA, AMD ROCm, etc.)
- Workload Manager and Containerization (Slurm®, PBS, PMIX)

Linear system of equations

```
Classical-Quantum Hybrid Parallel Workload
Cray C/C++ Compilier // Cray MPICH // Cray Math Library-BLAS
############################################################
Solving a linear system of equations:
Synthesized the full circuit.
Sent circuit and metadata to simulator.
Received circuit and metadata from classical application
Broadcasted circuit and metadata to all simulator ranks.
Sent solution to classical application.
Received solution from simulator.

A = [[ 0.28 -0.01  0.02 -0.1 ]
 [-0.01  0.5  -0.22 -0.07]
 [ 0.02 -0.22  0.43 -0.05]
 [-0.1  -0.07 -0.05  0.42]]

b = [0.18257419 0.36514837 0.54772256 0.73029674]

Pauli strings list:

II :  (0.408+0j)
IZ :  (-0.052+0j)
IX :  (-0.03+0j)
ZI :  (-0.017+0j)
ZZ :  (-0.057+0j)
ZX :  (0.02+0j)
XI :  (-0.025+0j)
XZ :  (0.045+0j)
XX :  (-0.16+0j)
YY :  (-0.06+0j)

Number of qubits for matrix representation = 2

Comparing solutions:
classical:   [1.55071576 2.36243885 2.73915645 2.82784967]
HHL:         [1.60021777 2.40844196 2.77379842 2.8671254 ]
relative distance:  1.8 %

############################################################
```

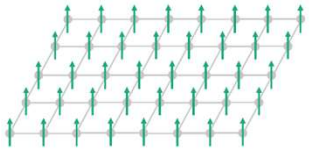
Distributed Max Cut





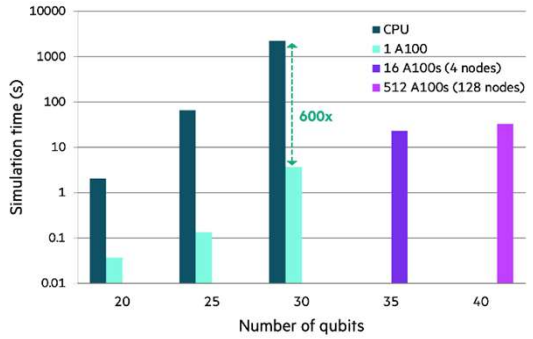
Performance and scalability

Transverse field Ising model simulations



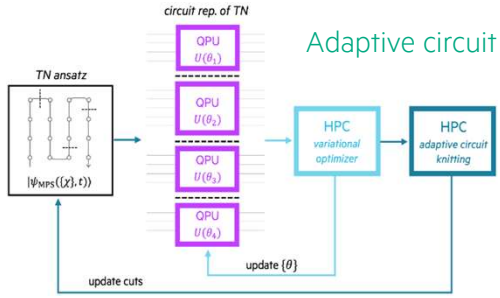
2D spin lattice (40 qubits)

AMD CPU + 4 Nvidia A100 GPU (40GB and 80GB)/node



Hypervisor

Adaptive circuit knitting



Hewlett Packard
Enterprise

Thank you

Questions

xin.zhan@hpe.com