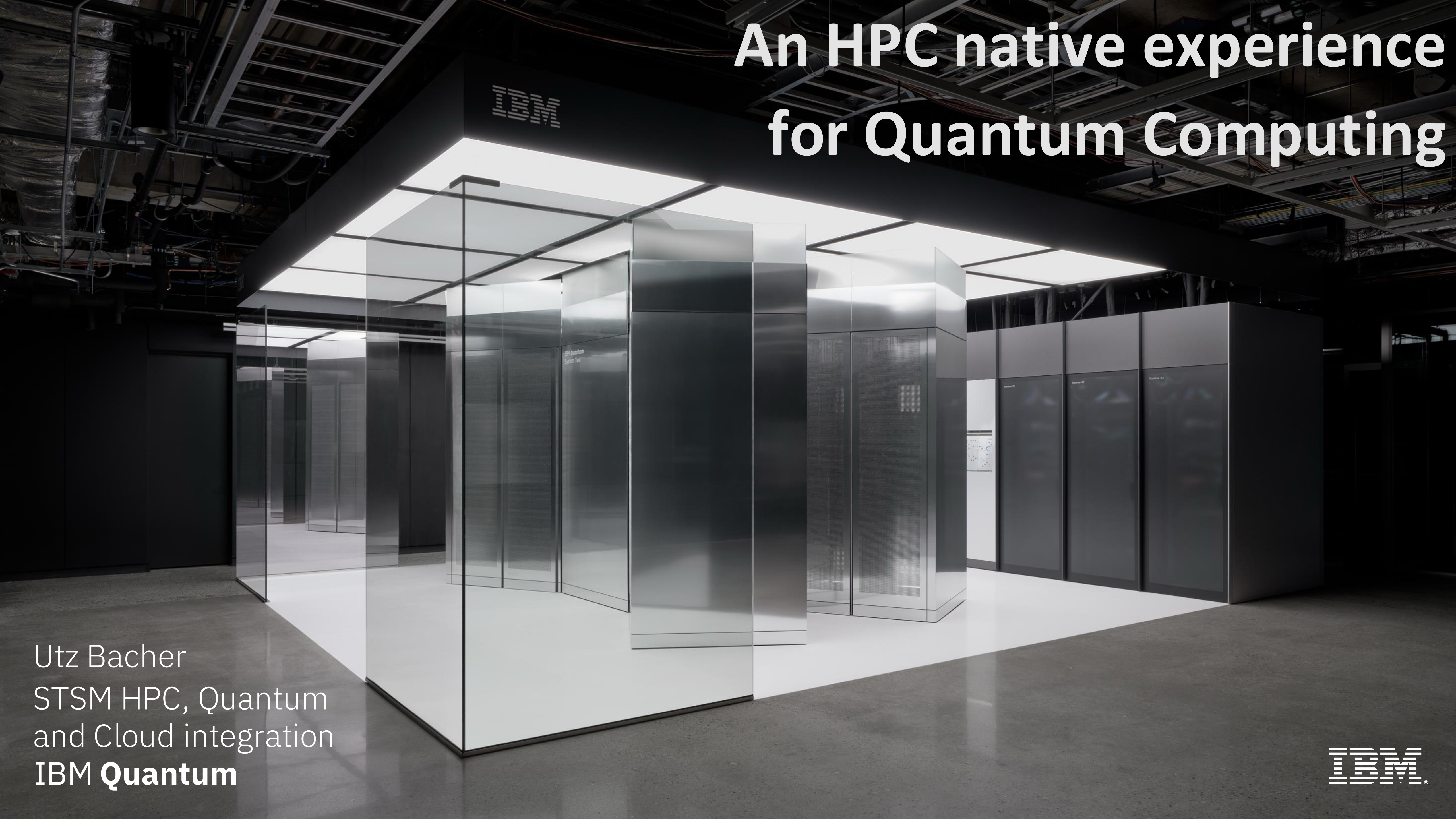
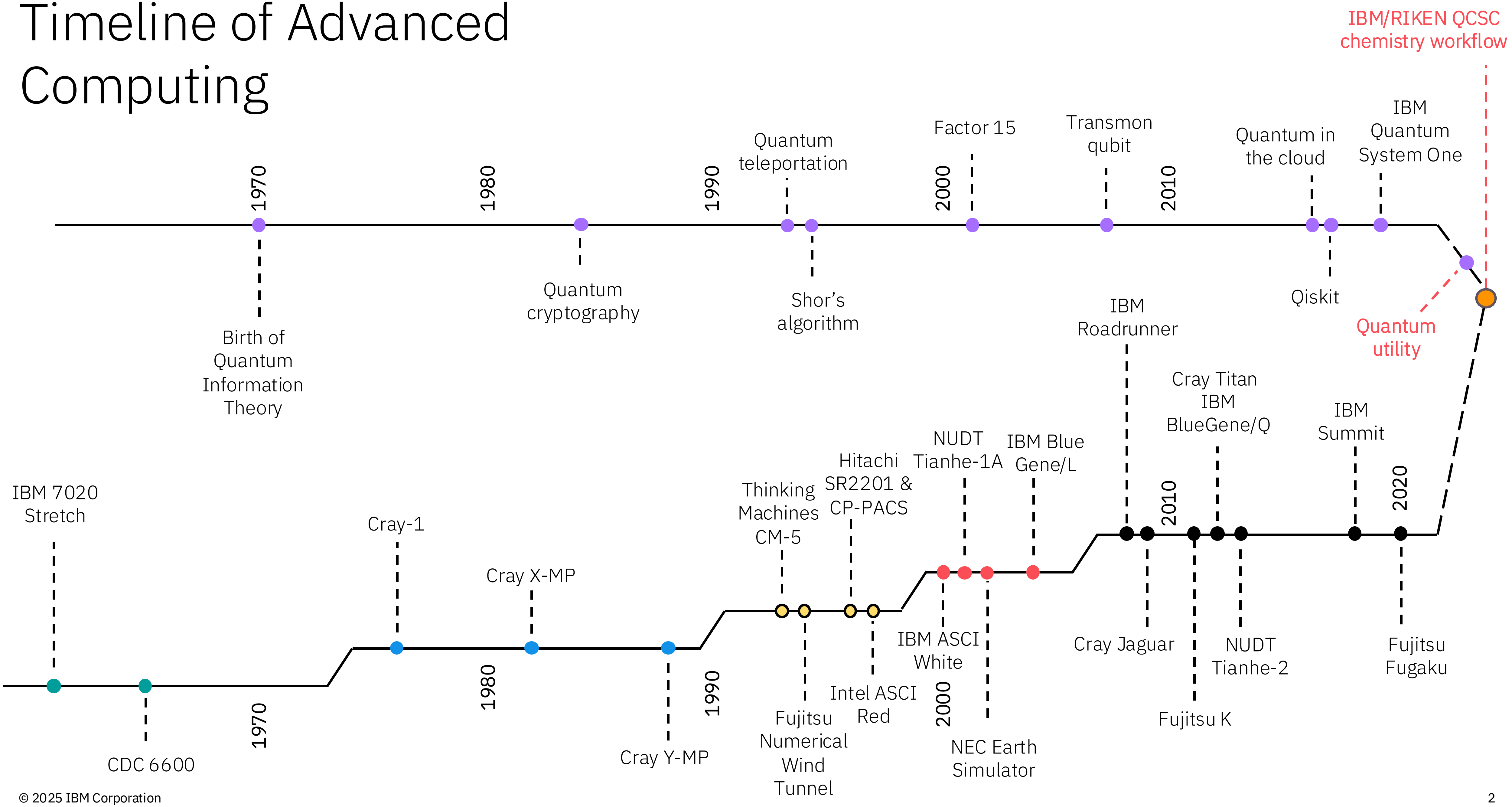
An HPC native experience for Quantum Computing



Utz Bacher
STSM HPC, Quantum and Cloud integration
IBM **Quantum**

IBM

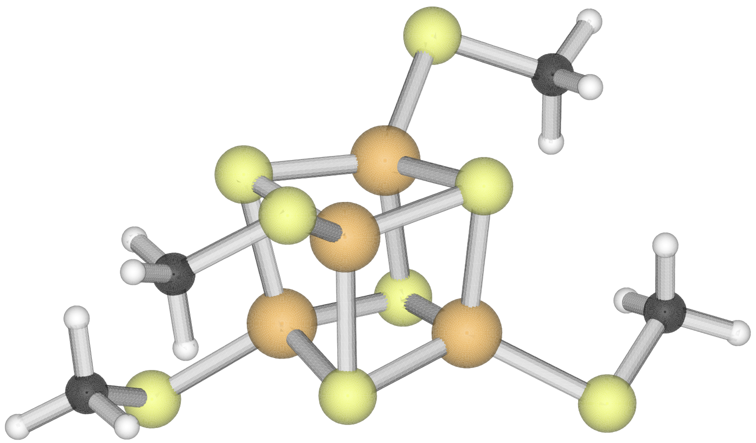
Timeline of Advanced Computing



(*) Adapted from Reed *et al.* *arXiv*: 2203.02544 (2022)

Quantum-centric supercomputing is unlocking new applications for near-term quantum hardware

Evaluating the ground-state energy for [4Fe-4S],
a molecule beyond exact diagonalization scale (26e,23o).



[4Fe-4S] on 77 qubits (TZP-DKH basis set): 6.7M Pauli operators

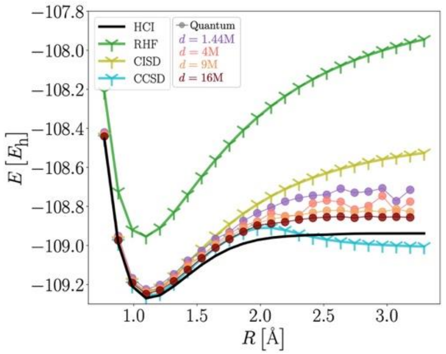
Conventional Quantum	VQE estimation at 10µs/circuit ~3M years
Quantum-centric supercomputing	Subspace estimation at 10µs/circuit ~2 hours
Fault tolerant	Phase estimation qubits: 4.53M 13 days runtime*

*Estimation done using surface code

Science Advances vol. 11, no. 25 (2025)

SQD for chemistry on IBM quantum processors

Chemistry beyond exact solutions…

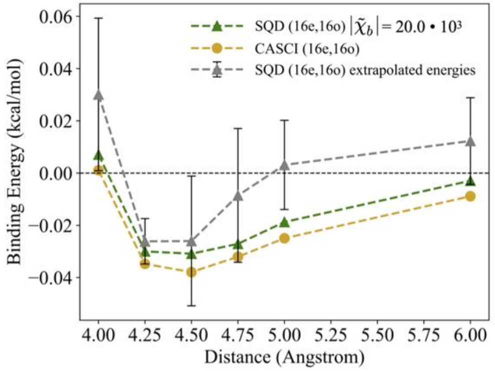


77 qubits
3500 2-qubit gates

RIKEN
IBM Quantum

arXiv:2405.05068

High-precision SQD applied to supramolecular interactions

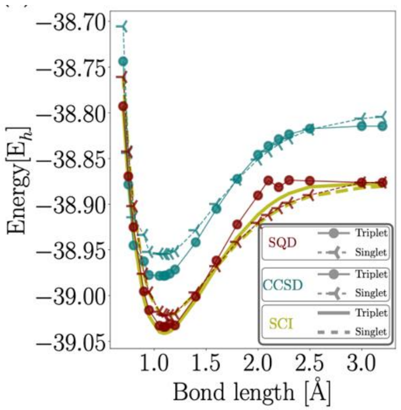


52 qubits
1600 2-qubit gates

Cleveland Clinic
IBM Quantum

arXiv:2410.09209

SQD applied to open-shell systems

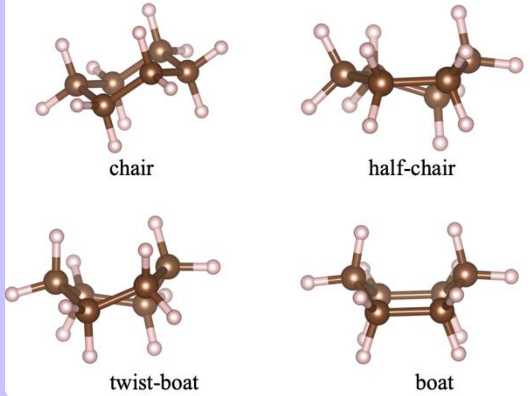


52 qubits
3500 2-qubit gates

Lockheed Martin
IBM Quantum

arXiv:2411.04827

DMET + SQD for organic chemistry

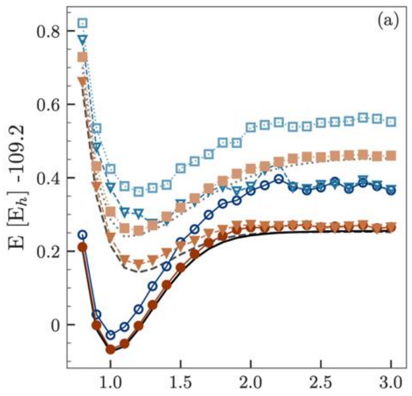


41 qubits
1500 2-qubit gates

Cleveland Clinic
IBM Quantum

arXiv:2411.09861

Ext-SQD: excited states

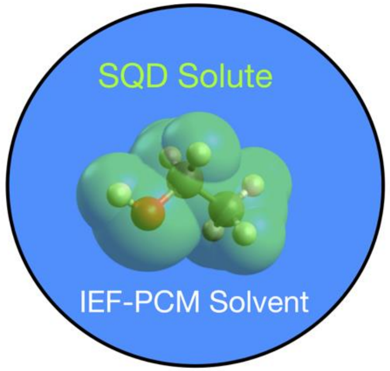


58 qubits
1792 2-qubit gates

IBM Quantum

arXiv:2411.00468

SQD + IEF-PCM for solute-solvent interactions

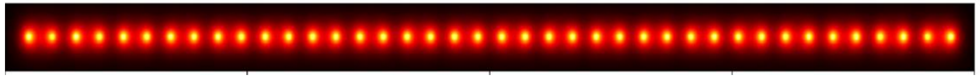


52 qubits
2422 2-qubit gates

Cleveland Clinic

arXiv:2502.10189

5k challenge on hydrogen chains



84 qubits, 5000 2-qubit gates

IBM Quantum

Unpublished

Ingredients for High Performance Computing with Quantum

Application Code

Qiskit addons & ecosystem

Qiskit SDK

Qiskit Slurm plugin

QPU, runtime stack executing primitives

Addons: Algorithms as Building Blocks to Speed Up Research

Application Code

Qiskit addons & ecosystem

Qiskit S

Qiskit S

QPU, ru

Faster research
Up to:

↑ 5x

1 year of research → **2.5 months** of research

"Chemistry Beyond Exact Solutions on a Quantum-Centric Supercomputer"
—
Science Advances vol. 11, no. 25 (2025)

RIKEN

"Accurate quantum-centric simulations of supramolecular interactions"
—
arXiv:2410:09209

Cleveland Clinic

"Quantum-Centric Study of Methylene Singlet and Triplet States"
—
arxiv.org/abs/2411.04827

Lockheed Martin

Qiskit SDK: Enabling Highest Performance

Application Code

Qiskit addons & ecosystem

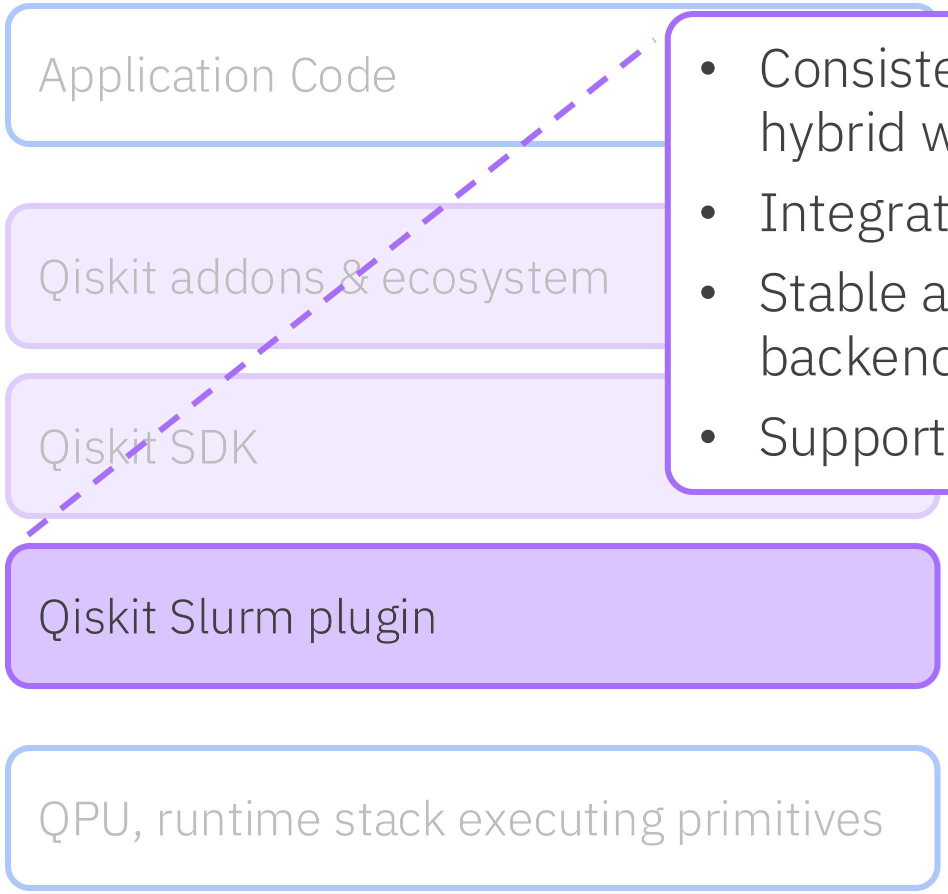
Qiskit SDK

Qiskit Slurm plugin

QPU, runtime stack executing

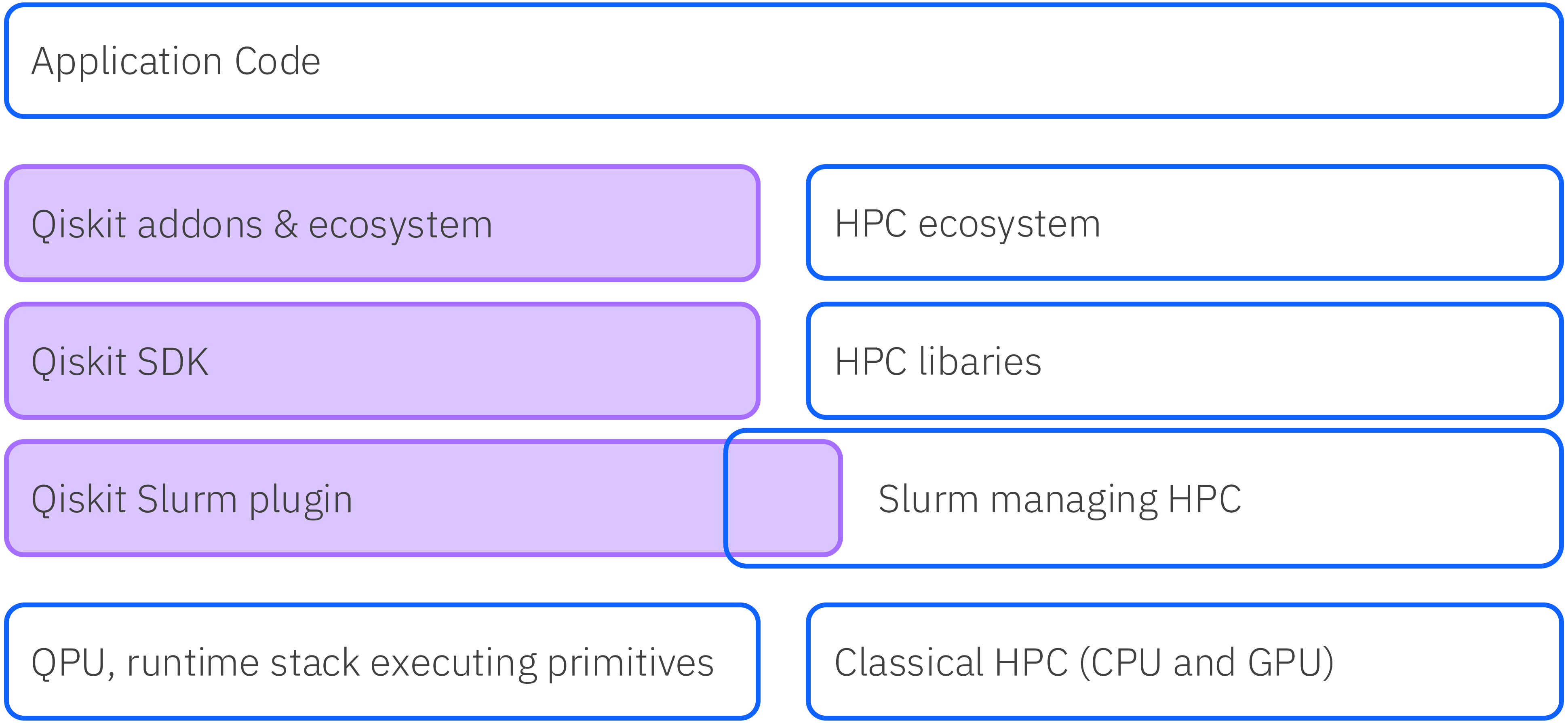
- C++ headers for seamless integration with C++
- C Foreign Function Interface to accommodate basically any compiled language
- Qiskit SDK core re-written in Rust for best SDK performance

Qiskit Slurm Plugin: Integrating Quantum and Classical Workload



- Consistent resource management and scheduling for hybrid workload through slurm
- Integrate quantum with common HPC workflows and tools
- Stable application source code through abstraction of backend and primitives
- Supports IBM and Pasqal today

High Performance Computing with Quantum



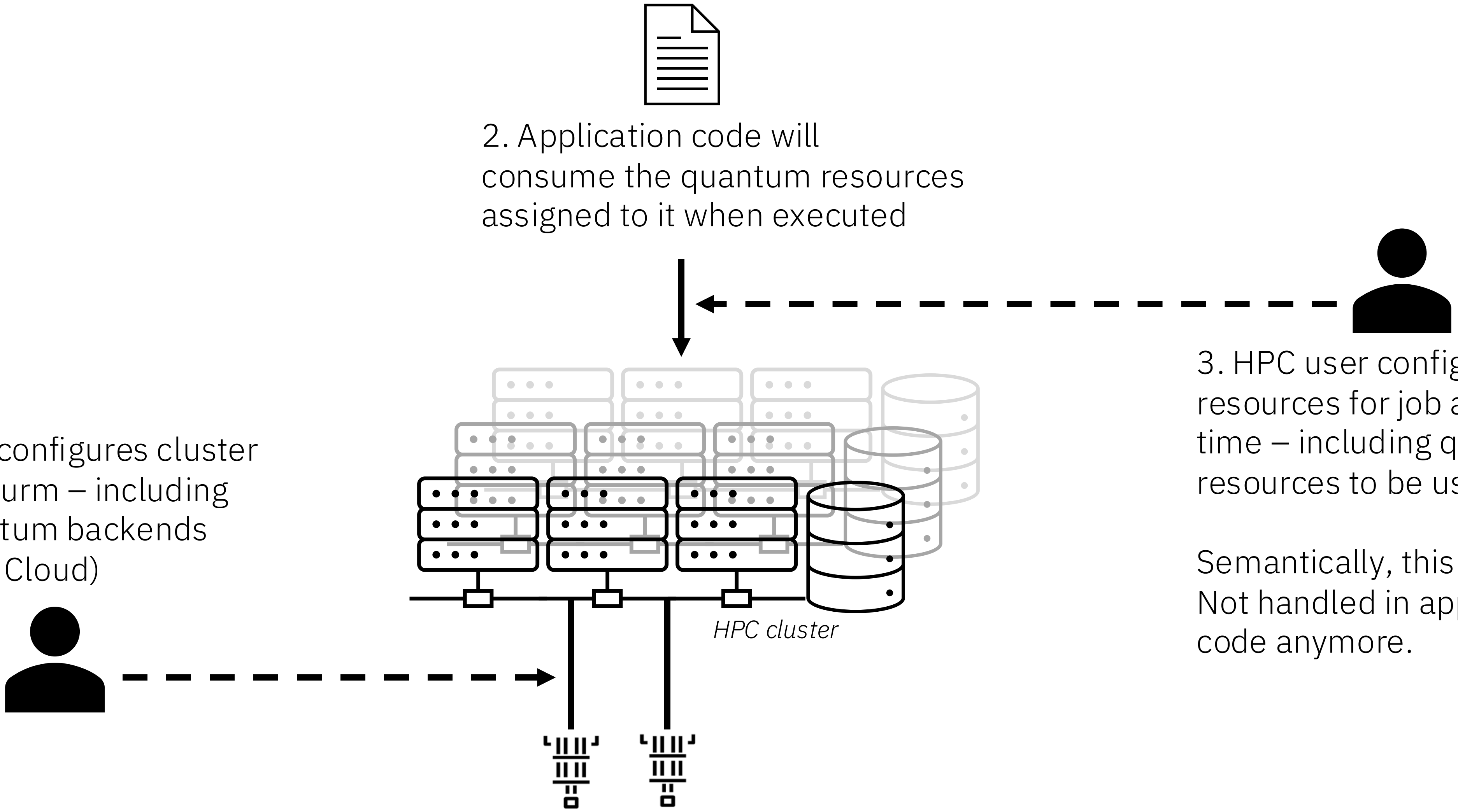
HPC-native User Experience for Quantum Resources with Slurm

2. Application code will consume the quantum resources assigned to it when executed

3. HPC user configures resources for job at submission time – including quantum resources to be used.

Semantically, this is configuration. Not handled in application source code anymore.

1. HPC admin configures cluster resources in slurm – including available quantum backends (local or in the Cloud)



See ux documentation on github for more details
https://github.com/qiskit-community/spank-plugins/blob/main/docs/ux.md

Components

1. Slurm

- Interface to user and admin
- Manage cluster resources (classical and quantum, Cloud and on-prem)
- Schedule and run classical jobs on nodes

2. Spank Plugin

written in C

- Gather and manage information for quantum usage
  - including secrets when needed (credentials for quantum backends)
- Prepare environment variables for job process to use QRMI

3. Job

uses qiskit

4. QRMI Library

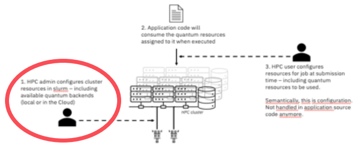
written in Rust

Quantum Resource Management Interface

- Library invoked as part of job execution
- Abstraction for different quantum vendors, access methods, and device details
- Interface with backend service
  - acquire (lock) backend (at submission time)
  - run and monitor job (at primitive invocation time)

5. Quantum Backend

Example: Slurm-Level Resource Definition of Usable Backends



HPC admin

```
{
  "resources": [
    {
      "name": "ibm_fez",
      "type": "qiskit-runtime-service",
      "environment": {
        "QRMI_IBM_QRS_ENDPOINT": "https://quantum.cloud.ibm.com/api/v1",
        "QRMI_IBM_QRS_IAM_ENDPOINT": "https://iam.cloud.ibm.com",
        "QRMI_IBM_QRS_IAM_APIKEY": "<CLOUD APIKEY>",
        "QRMI_IBM_QRS_SERVICE_CRN": "<IQP INSTANCE CRN>"
      }
    },
    {
      "name": "FRESNEL",
      "type": "pasqal-cloud",
      "environment": {
        "QRMI_PASQAL_CLOUD_PROJECT_ID": "<PROJECT_ID>",
        "QRMI_PASQAL_CLOUD_AUTH_TOKEN": "<AUTH TOKEN>"
      }
    }
  ]
}
```

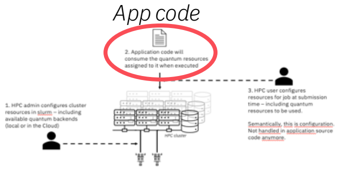
Endpoint and credentials for ibm_fez backend. Credentials can also be set by user

Configuration skeleton for Pasqal backend

Example: Application Source Code Uses Available Backends

Other imports here

Circuit construction here




```
# note these imports are not complete
from qiskit_qrmi_primitives import QRMIService
from qiskit_qrmi_primitives.ibm import SamplerV2

# Create QRMI
load_dotenv()
service = QRMIService()

resources = service.resources()
qrmi = resources[0] # could be several resources

# typically in a helper function, but showing here for transparency:
qrmi_target = qrmi.target()
qrmi_target = json.loads(qrmi_target.value)

backend_config = BackendConfiguration.from_dict(target["configuration"])
backend_props = BackendProperties.from_dict(target["properties"])
target = convert_to_target(backend_config, backend_props)

pm = generate_preset_pass_manager(
    optimization_level=3,
    target=target,
)
pm.post_optimization = PassManager(
    [ FoldRzzAngle(), Optimize1qGatesDecomposition(target=target),
        RemoveIdentityEquivalent(target=target), ]
)

isa_circuits = pm.run(circuits)

# Initialize QRMI Sampler
options = {
    "default_shots": 1024,
}
sampler = SamplerV2(qrmi, options=options)
job = sampler.run(isa_circuits)
```

All imports needed for QRMI

Pinpoint backend provided by slurm

Construct target to use in qiskit

Prepare and run transpilation

Run Sampler primitive on backend

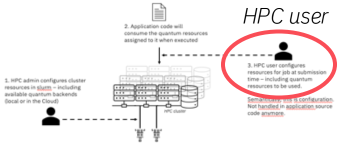
Example: User‘s Job Submission Configures Backend to Use

```
#!/bin/bash

#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --time=500
#SBATCH --output=<LOGS_PATH>

#SBATCH --gres=qpu:1
#SBATCH --qpu=ibm_fez

#SBATCH --... # other options

srun my_application
```

parameters for job

refer to QPU resource defined by slurm admin

application to run



Quantum-centric supercomputing integration at Rensselaer Polytechnic Institute (RPI)

AiMOS supercomputer

10 Gb/s ethernet

10 miles

IBM Quantum System One
127-qubit Eagle processor

IBM and RIKEN: Quantum system 2 and supercomputer Fugaku

"By combining Fugaku and the IBM Quantum System Two, RIKEN aims to lead Japan into a new era of high-performance computing.

Our mission is to develop and demonstrate practical quantum-HPC hybrid workflows that can be explored by both the scientific community and industry. The connection of these two systems enables us to take critical steps toward realizing this vision."

Dr. Mitsuhisa Sato,
Division Director of the Quantum-HPC Hybrid Platform Division,
RIKEN Center for Computational Science

Resources

Github repos

- SPANK plugin: https://github.com/qiskit-community/spank-plugins
- QRMI: https://github.com/qiskit-community/qrmi

arXiv

- Quantum resources in resource management systems: arXiv:2506.10052

Get started e.g. for free with the Open plan of IBM Quantum

- https://quantum.cloud.ibm.com/

IBM

Backup: Detailed Flow (from github repo)

https://github.com/qiskit-community/spank-plugins/blob/main/docs/images/high-level-plugin-flow.png