Accelerating Toward Quantum Advantage Together

Hiroshi Horii
Senior Technical Staff Member
Head of IBM Quantum Japan
Senior Manager of QCSC SW
IBM Quantum

IBM **Quantum**

Bring useful quantum computing to the world

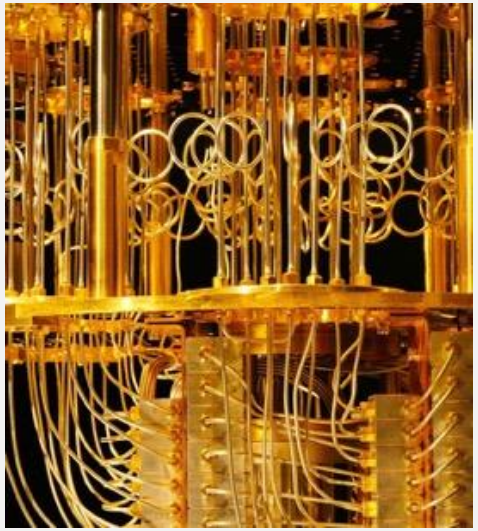
日本における量子コンピュータの研究開発

1. 業界ユースケース開拓

東京大学主導の「量子イノベーションイニシアティブ協議会（QII）」は**世界最大の産官学の研究ネットワーク**

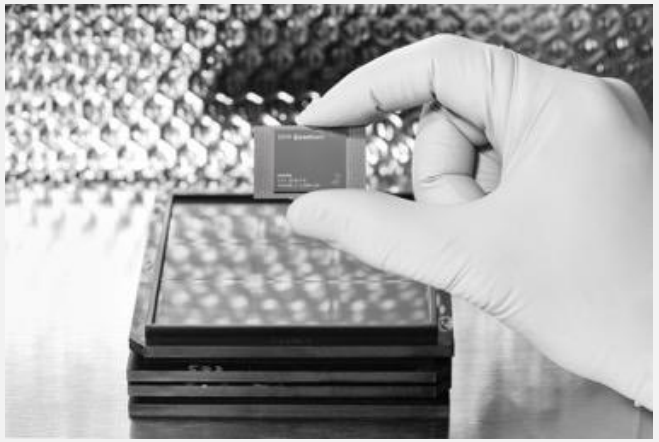
2. サプライチェーン参画

- 日本には**米国以外で唯一のIBMの量子ハードウェア開発環境（テストベッド）**を設置
- 必要とされる**重要な部品、材料の重要サプライヤー**となりうる潜在的な企業が多くある



3. 量子を中心としたスーパーコンピューター

- **理研がスーパーコンピューター「富岳」とIBMの次世代量子システムを連携**（2024/4）
- 一体として動作するシステムを開発



Key milestones in bringing useful quantum computing to the world

2023

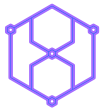
Establish quantum utility

2026

Demonstrate quantum advantage

2029

Deliver the first large-scale, fault-tolerant quantum computer



A noisy quantum computer produces accurate expectation values on 127 qubits and 2880 gates, outside of brute force classical computation.

Y. Kim, A. Eddins, et al, Nature. 618, 500–505 (2023)

Article

Evidence for the utility of quantum computing before fault tolerance

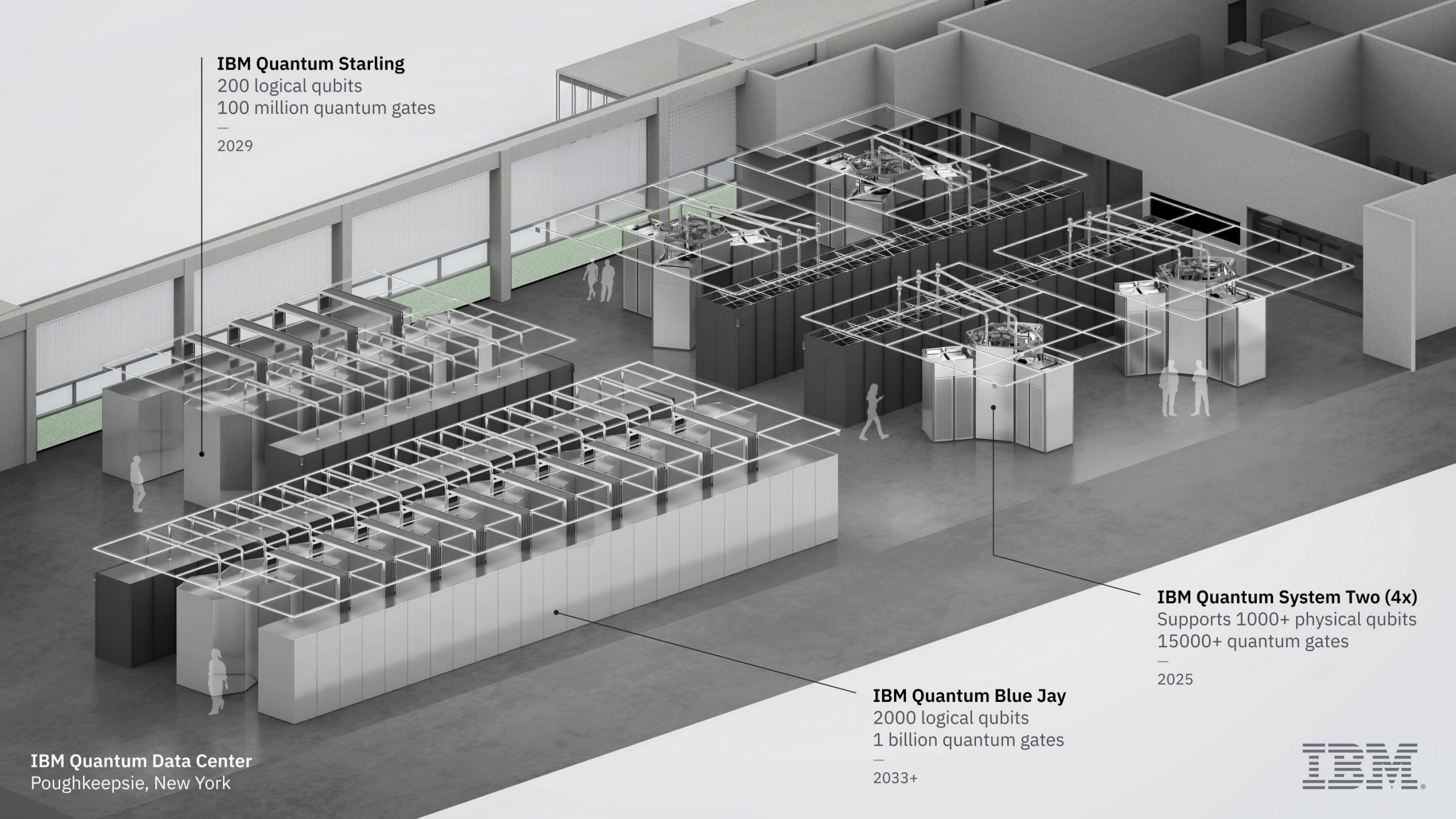
https://doi.org/10.1038/s41586-023-06096-3

Received: 24 February 2023

Accepted: 18 April 2023

Published online: 14 June 2023

Youngseok Kim[1,6], Andrew Eddins[2,6], Sajant Anand[3], Ken Xuan Wei[1], Ewout van den Berg[1], Sami Rosenblatt[1], Hasan Nayfeh[1], Yantao Wu[3,4], Michael Zaletel[3,5], Kristan Temme[1] & Abhinav Kandala[1]


IBM Quantum Starling
200 logical qubits
100 million quantum gates
—
2029
IBM Quantum System Two (4x)
Supports 1000+ physical qubits
15000+ quantum gates
—
2025
IBM Quantum Blue Jay
2000 logical qubits
1 billion quantum gates
—
2033+
IBM Quantum Data Center
Poughkeepsie, New York

Removing the roadblocks to fault tolerance at scale

Is there a code that scales efficiently?

Nature 627, 778–782 (2024)

Quantum memory using bivariate bicycle codes requires ~10x fewer physical qubits than the surface code

Does this code retain its advantage during computation?

Tour de gross: A modular quantum computer based on bivariate bicycle codes

Theodore J. Yoder[1], Eddie Schoute[1], Patrick Rall[1], Emily Pritchett[1], Jay M. Gambetta[1], Andrew W. Cross[1], Malcolm Carroll[1], and Michael E. Beverland*[1]

[1]IBM Quantum

Abstract

We present the *bicycle architecture*, a modular quantum computing framework based on high-rate, low-overhead quantum LDPC codes identified in prior work. For two specific bivariate bicycle codes with distances 12 and 18, we construct explicit fault-tolerant logical instruction sets and estimate the logical error rate of each instruction under circuit noise. We develop a compilation strategy adapted to the constraints of the bicycle architecture, enabling large-scale universal quantum circuit execution. Integrating these components, we perform end-to-end resource estimates demonstrating that an order of magnitude larger logical circuits can be implemented with a given number of physical qubits on a bicycle architecture than on surface code architectures. We anticipate further improvements through advances in code constructions, circuit designs, and compilation techniques.

arXiv:2506.03094 (2025)

End-to-end resource estimates demonstrating that bicycle architectures retain an order of magnitude qubit advantage over surface code architectures when implementing large logical circuits

Is real-time decoding possible?

Improved belief propagation is sufficient for real-time decoding of quantum memory

Tristan Müller,* Thomas Alexander, Michael E. Beverland,
Markus Bühler, Blake R. Johnson, Thilo Maurer, and Drew Vandeth
IBM Quantum
(Dated: June 3, 2025)

We introduce a new heuristic decoder, Relay-BP, targeting real-time quantum circuit decoding for large-scale quantum computers. Relay-BP achieves high accuracy across circuit-noise decoding problems: significantly outperforming BP+OSD+CS-10 for bivariate-bicycle codes and comparable to min-weight-matching for surface codes. As a lightweight message-passing decoder, Relay-BP is inherently parallel, enabling rapid low-footprint decoding with FPGA or ASIC real-time implementations, similar to standard BP. A core aspect of our decoder is its enhancement of the standard BP algorithm by incorporating disordered memory strengths. This dampens oscillations and breaks symmetries that trap traditional BP algorithms. By dynamically adjusting memory strengths in a relay approach, Relay-BP can consecutively encounter multiple valid corrections to improve decoding accuracy. We observe that a problem-dependent distribution of memory strengths that includes negative values is indispensable for good performance.

arXiv:2506.01779 (2025)

Demonstrating the feasibility of real-time decoding of qLDPC codes such as bivariate bicycle codes with a modified algorithm (Relay-BP) that eliminates the need for a second-stage decoder.

We predict that quantum advantage will be found by the end of 2026...

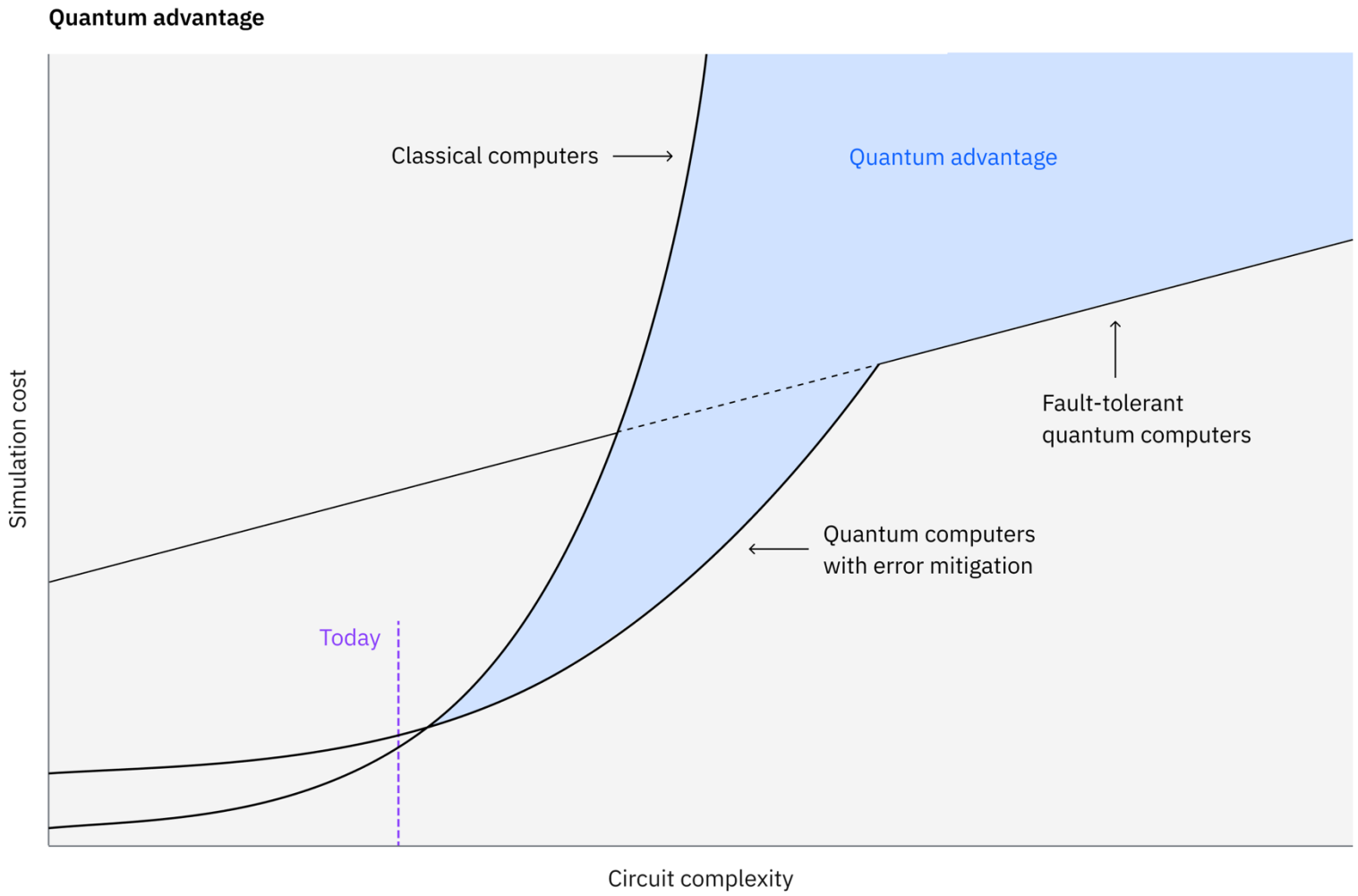
... but only if we work together.

What is quantum advantage?

The ability to execute an information processing task on quantum hardware in a way that satisfies two essential criteria:

1. This correctness of the output can be **rigorously validated**.

2. The calculation is performed with a **quantum separation** that offers superior efficiency, cost-effectiveness, or accuracy than what can be obtained with classical methods alone.



Read the white paper → https://ibm.biz/QA-whitepaper
Read the blog → https://ibm.biz/QA-blog

Advantage trackers

Track verifiable quantum advantage candidates across three pathways, with current status of classical and quantum computations, supporting evidence and contributing organizations.

Observable estimations | Variational problems | Classically verifiable problems

Observable estimations

Submissions in this tracker report expectation values for observables alongside rigorous error bars. Validation requires mathematically provable confidence intervals over the reported value.

View circuit options Open submission ticket

Quantum-centric supercomputing (QCSC):

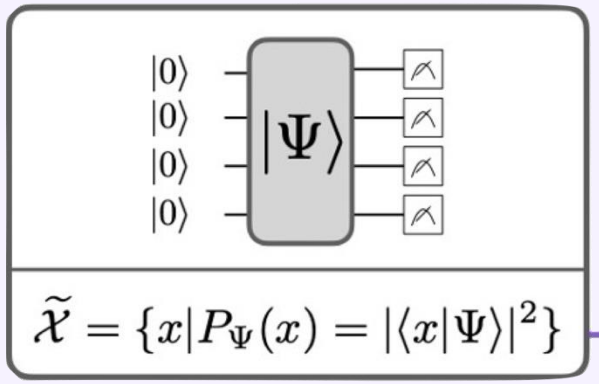
A hybrid computing architecture that leverages both classical and quantum computers.

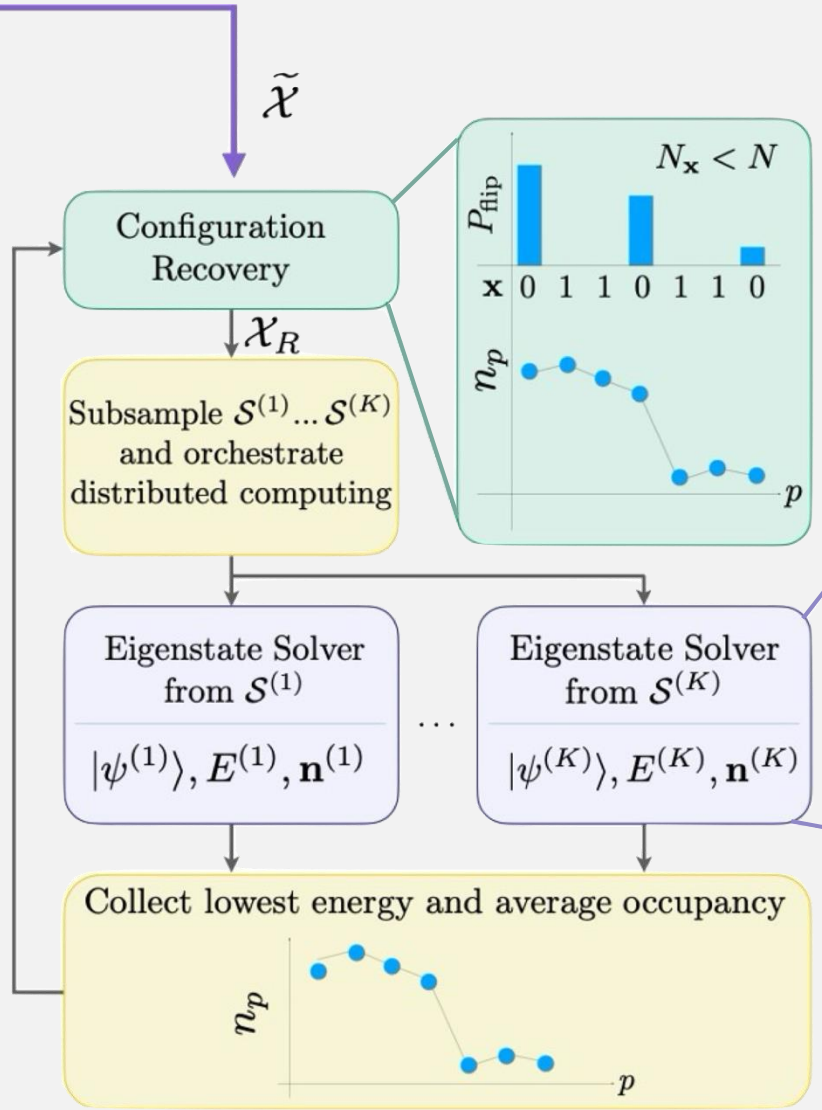
Sample-based quantum diagonalization (SQD)*

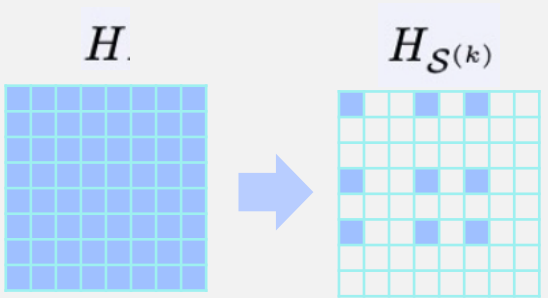
*) Robledo-Moreno, Javier, et al. *arXiv:2405.05068*
**) For QSCI - Kanno, Keita, et al. *arXiv:2302.11320*

Execute using Qiskit Runtime Primitives. Quantum compute generates samples from an exponentially large space

Sampler()







$$H_{\mathcal{S}^{(k)}} = P_{\mathcal{S}^{(k)}} H P_{\mathcal{S}^{(k)}}$$

$$\text{with } P_{\mathcal{S}^{(k)}} = \sum_{\mathbf{x} \in \mathcal{S}^{(k)}} |\mathbf{x}\rangle\langle\mathbf{x}|$$

$$H_{\mathcal{S}^{(k)}} \left|\psi^{(k)}\right\rangle = E^{(k)} \left|\psi^{(k)}\right\rangle$$

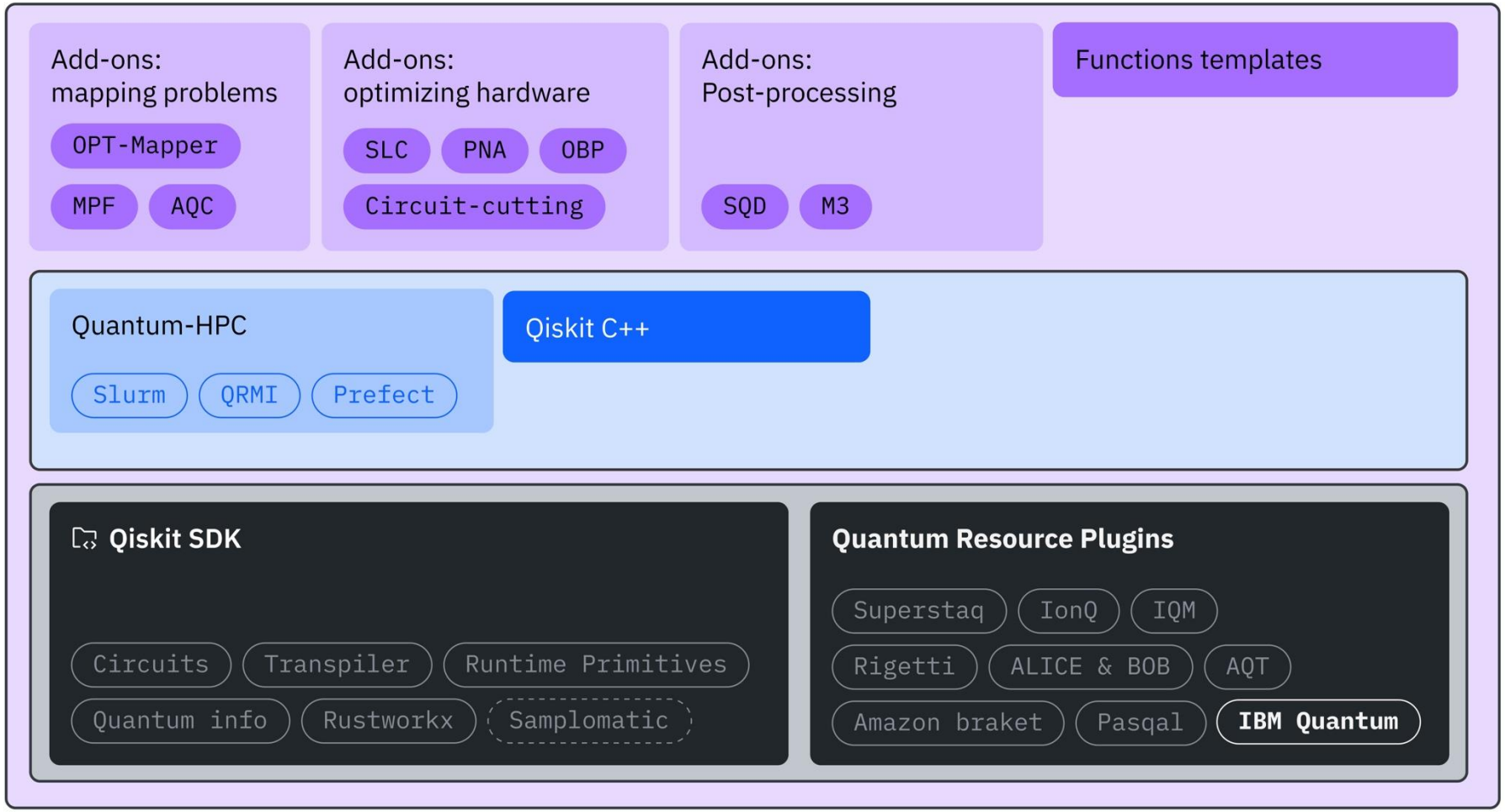
Fugaku

Qiskit –
Open ecosystem for quantum computing

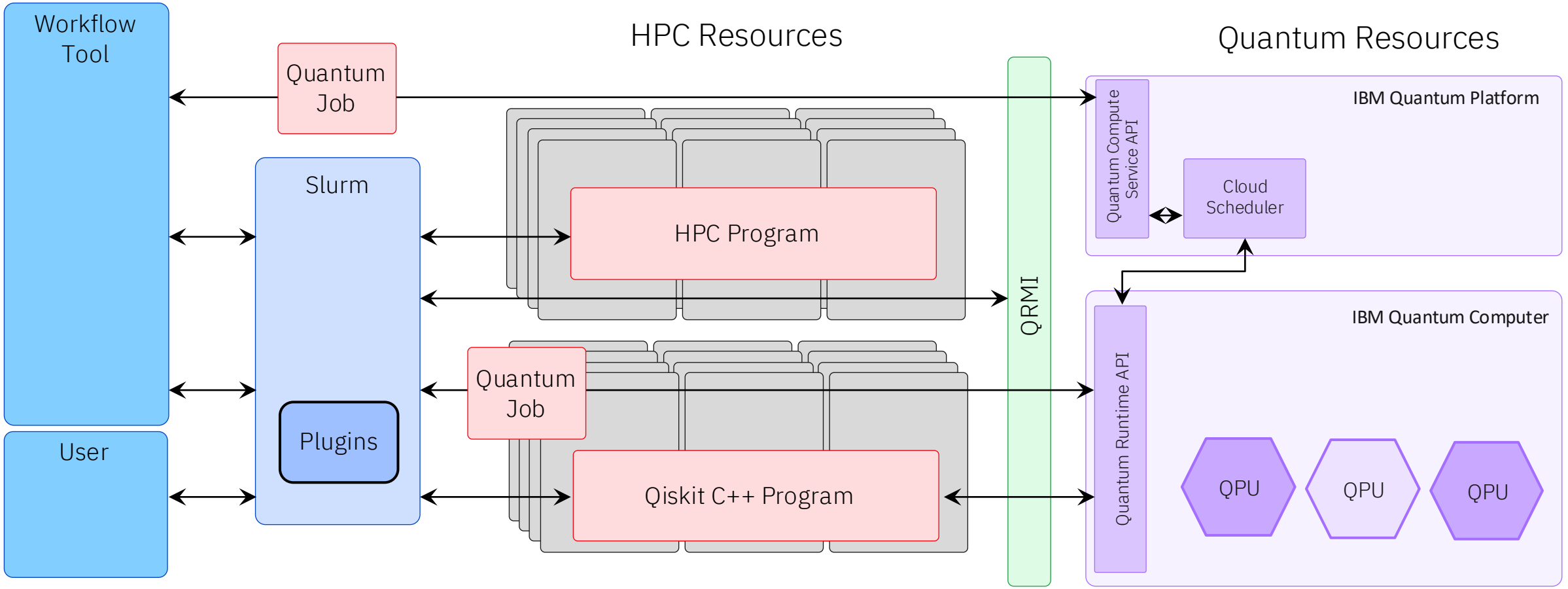
Qiskit for:
- Algorithm research

Qiskit for:
- High Performance Computing

Qiskit for:
- Quantum information science research



Workflow provides efficient resource utilization



Quantum resources in resource management system

Quantum resources in resource management systems

Utz Bacher,[1] Mark Birmingham,[2] Christopher D. Carothers,[3] Andrew Damin,[3] Carlos D. Gonzalez Calaza,[4,5] Ashwin Kumar Karnad,[4] Stefano Mensa,[2] Matthieu Moreau,[6] Aurelien Nober,[6] Munetaka Ohtani,[7] Max Rossmannek,[8] Philippa Rubin,[2] M. Emre Sahin,[2] Oscar Wallis,[2] Amir Shehata,[9] Iskandar Sitdikov,[10] and Aleksander Wennersteen[6]

[1] *IBM Deutschland Research & Development GmbH, Boeblingen, Germany*
[2] *The Hartree Centre, STFC, Sci-Tech Daresbury, Warrington, WA4 4AD, United Kingdom*
[3] *Center for Computational Innovation, Rensselaer Polytechnic Institute, Troy, NY USA 12180*
[4] *Jülich Supercomputing Centre, Forschungszentrum Jülich, 52425 Jülich, Germany*
[5] *RWTH Aachen University, 52056 Aachen, Germany*
[6] *PASQAL, 24 Av. Emile Baudot, 91120 Palaiseau, France*
[7] *IBM Quantum, IBM Research – Tokyo, Tokyo 103-8510, Japan*
[8] *IBM Quantum, IBM Research—Zurich, Säumerstrasse 4, 8803 Rüschlikon, Switzerland*
[9] *Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA*
[10] *IBM Quantum, IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA*

(Dated: Friday 14th November, 2025)

Quantum computing resources are increasingly being incorporated into high-performance computing (HPC) environments as co-processors for hybrid workloads. To support this paradigm, quantum devices must be treated as schedulable first-class resources within existing HPC infrastructure. This enables consistent workload management, unified resource visibility, and support for hybrid quantum-classical job execution models.

This paper presents a reference architecture and implementation for the integration of quantum computing resources, both on-premises and cloud-hosted into HPC centers via standard workload managers. We introduce a Slurm plugin designed to abstract and control quantum backends, enabling seamless resource scheduling, minimizing queue duplication, and supporting job co-scheduling with classical compute nodes. The architecture supports heterogeneous quantum resources and can be extended to any workload (and container) management systems.

I. INTRODUCTION

The emergence of quantum computing motivates the incorporation of quantum resources into established high-performance computing (HPC) environments as a way of unlocking previously intractable workflows. The integration of quantum computing into HPC environments is an active research area [1–4], where the development of appropriate software abstractions turns out to be a hurdle. Hybrid quantum-classical applications, such as Sample-Based Quantum Diagonalization (SQD) [5], are already being executed on large-scale HPC systems. Production HPC centers typically rely on a single workload management system to coordinate all resources, in order to avoid the operational overhead of multiple scheduling layers (including multiple queues), incosistent management, and degraded user workflows.

The differences between existing workload manager resource abstractions and the heterogeneous interfaces of quantum computers, whether locally hosted or accessed through cloud services, require an intermediate compatibility layer. Earlier approaches commonly used wrapper scripts, external submission mechanisms, or relied on specific software stacks [6]. A primary challenge in integrating quantum hardware is the diversity of platforms and their vendor-specific APIs. A solution tied to a single vendor's control system would be brittle and limit future flexibility. The Quantum Resource Management Interface (QRMI), which we present in this work, is a conceptual framework designed to solve this problem by providing a standardized, vendor-agnostic abstraction layer, without being coupled to any single software stack. The design follows a loosely coupled architecture, allowing adoption across different resource managers and quantum providers, and support for both on-prem and cloud-based resources.

As a reference implementation, we target Slurm [7], a widely deployed workload manager in HPC. Slurm's plugin framework provides the necessary hooks to extend its scheduling and resource management capabilities. We introduce a plugin that registers quantum resources as schedulable entities, enabling users to submit and manage hybrid jobs within the same Slurm-managed environment. The same architectural concepts can be generalized to other workload management systems.

Finally, we present integration architectures deployed in operational data centers that have adopted this work, demonstrating feasibility and evaluating system behavior in production environments.

II. BACKGROUND

A. Resource Management Systems and the SPANK Framework

Resource management systems in high-performance computing are multi-tenant platforms tasked with allocating compute resources such as nodes, CPU cores, and accelerators according to policies that govern fairness, priority, reservations, and quality of service. These systems manage job submission, queuing, scheduling, ex-

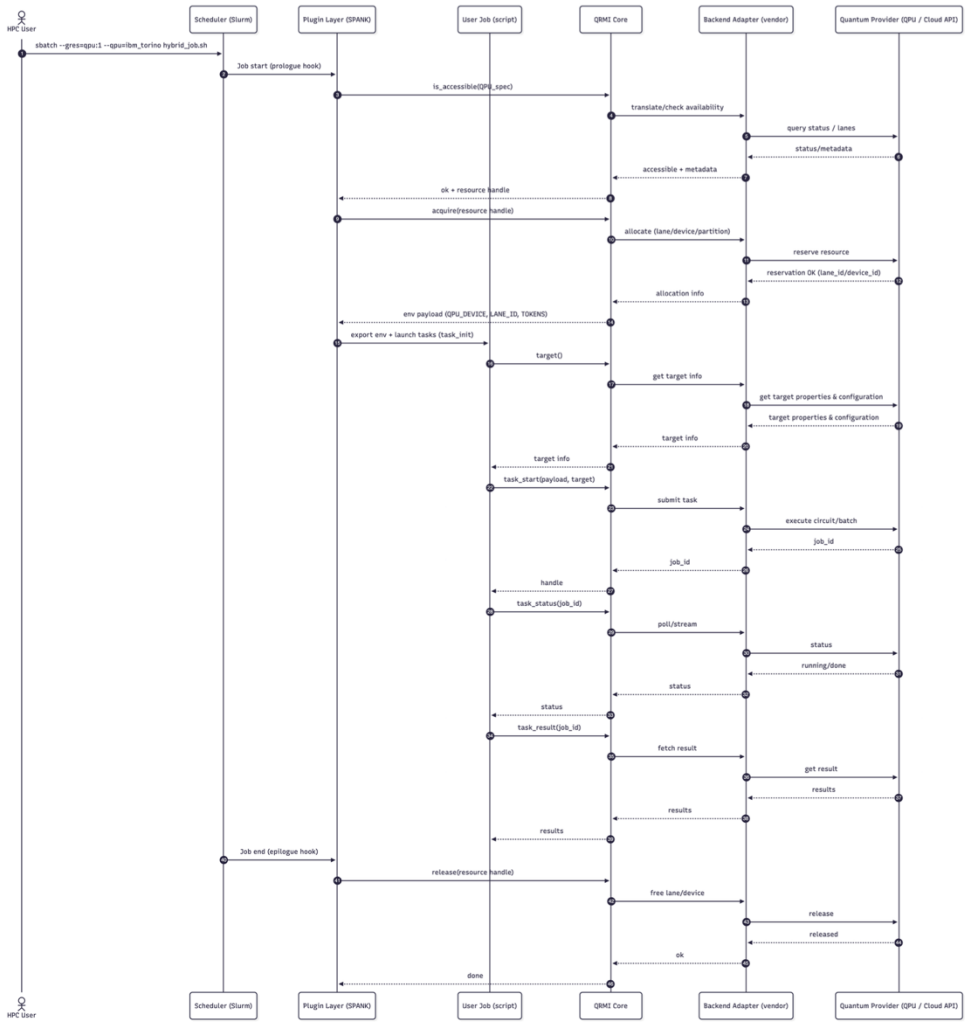


Figure 6: Sequence diagram of hybrid quantum-classical workflow within integrated environment.

Overlapping computation with Quantum and HPC resources

Closed-loop calculations of electronic structure on a quantum processor and a classical supercomputer at full scale

Tomonori Shirakawa
Center for Computational Science RIKEN
Kobe, Japan
0000-0001-7923-216X
t-shirakawa@riken.jp

Javier Robledo-Moreno
IBM Quantum
IBM T.J. Watson Research Center
Yorktown Heights, NY, USA
j.robledomoreno@ibm.com

Toshinari Itoko
IBM Quantum
IBM Research – Tokyo
Tokyo, Japan
0009-0006-2611-2301
itoko@jp.ibm.com

Vinay Tripathi
IBM Quantum
IBM T.J. Watson Research Center
Yorktown Heights, NY, USA
0000-0003-1418-9046
tripathi@ibm.com

Kento Ueda
IBM Quantum
IBM Research – Tokyo
Tokyo, Japan
Kento.Ueda@ibm.com

Yukio Kawashima
IBM Quantum
IBM Research – Tokyo
Tokyo, Japan
0000-0001-5918-1211
yukio.kawashima@ibm.com

Lukas Broers
Center for Computational Science RIKEN
Kobe, Japan
lukas.broers@riken.jp

William Kirby
IBM Quantum
IBM T.J. Watson Research Center
Yorktown Heights, NY, USA
William.Kirby@ibm.com

Himadri Pathak
Interdisciplinary Theoretical and Mathematical Sciences RIKEN
Wako, Japan
0000-0002-5919-8002
himadri.pathak@riken.jp

Hanhee Paik
IBM Quantum
IBM Research – Tokyo
Tokyo, Japan
hanhee.paik@us.ibm.com

Miwako Tsuji
Center for Computational Science RIKEN
Kobe, Japan
miwako.tsuji@riken.jp

Yuetsu Kodama
Center for Computational Science RIKEN
Kobe, Japan
yuetsu.kodama@riken.jp

Mitsuhisa Sato
Center for Computational Science RIKEN
Kobe, Japan
msato@riken.jp

Constantinos Evangelinos
IBM Quantum
IBM T.J. Watson Research Center
Yorktown Heights, NY, USA
0000-0003-1418-9046
cevange@us.ibm.com

Seetharami Seelam
IBM Quantum
IBM T.J. Watson Research Center
Yorktown Heights, NY, USA
sseelam@us.ibm.com

Robert Walkup
IBM Quantum
IBM T.J. Watson Research Center
Yorktown Heights, NY, USA
walkup@us.ibm.com

Seiji Yunoki
Pioneering Research Institute RIKEN
Wako, Japan
yunoki@riken.jp

Mario Motta
IBM Quantum
IBM T.J. Watson Research Center
Yorktown Heights, NY, USA
0000-0003-1647-9864
mario.motta@ibm.com

Petar Jurcevic
IBM Quantum
IBM T.J. Watson Research Center
Yorktown Heights, NY, USA
0000-0003-1234-6386
petar.jurcevic@ibm.com

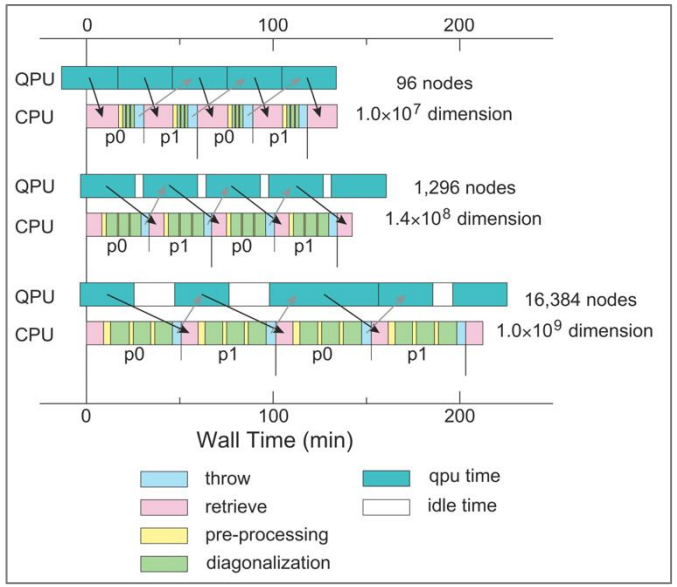
Hiroshi Horii
IBM Quantum
IBM Research – Tokyo
Tokyo, Japan
horii@jp.ibm.com

Antonio Mezzacapo
IBM Quantum
IBM T.J. Watson Research Center
Yorktown Heights, NY, USA
mezzacapo@ibm.com

***Abstract*—Quantum computers must operate in concert with classical computers to deliver on the promise of quantum advantage for practical problems. To achieve that, it is important to understand how quantum and classical computing can interact together, and how one can characterize the scalability and efficiency of hybrid quantum-classical workflows. So far, early**

Algorithm 1 Synchronous Quantum and Classical computations for Two Populations in Differential Evolution

```
async RunQuantum(itreration=0, population=0)
async RunQuantum(itreration=0, population=1)
for itr ← 0 to MaxItr − 1 do
    wait RunQuantum(itreration=itr, population=0)
    RunClassical(itreration=itr, population=0)
    if itr + 1 < MaxItr then
        RunQuantum(itreration=itr + 1, population=0)
    end if
    wait RunQuantum(itreration=itr, population=1)
    RunClassical(itreration=itr, population=1)
    if itr + 1 < MaxItr then
        RunQuantum(itreration=itr + 1, population=1)
    end if
end for
```



Prefect + Qiskit → Observability

Observability Architecture for Quantum-Centric Supercomputing Workflows

Naoki Kanazawa,* Hitomi Takahashi, Toshinari Itoko, Yukio Kawashima, and Hiroshi Horii†
IBM Quantum, IBM Research – Tokyo, Tokyo 103-8510, Japan

Yuto Morohoshi‡
IBM Quantum, IBM Research – Tokyo, Tokyo 103-8510, Japan and
Graduate School of Engineering Science, The University of Osaka,
1-3 Machikaneyama, Toyonaka, Osaka 560-8531, Japan
(Dated: November 14, 2025)

Quantum-centric supercomputing (QCSC) workflows often involve hybrid classical–quantum algorithms that are challenging to interpret due to their stochastic nature and limited runtime visibility. The high cost of quantum circuit execution and large-scale high-performance computing (HPC) infrastructure further restricts the number of feasible trials, making introspection critical for iterative development. We propose an observability architecture tailored for QCSC workflows, enabling persistent monitoring, structured telemetry collection, and post-hoc analysis. Applied to a representative workflow involving sample-based quantum diagonalization, our system reveals solver behavior across multiple iterations. This approach enhances transparency and reproducibility in QCSC environments, supporting infrastructure-aware algorithm design and systematic experimentation.

I. INTRODUCTION

Quantum-centric supercomputing (QCSC) is emerging as a promising paradigm for addressing computational problems that are intractable on classical systems alone. Recent studies have explored the architectural and algorithmic foundations of QCSC, highlighting its potential to accelerate scientific discovery through hybrid quantum–classical workflows [1–5]. These workflows embed quantum kernels within classical orchestration, typically executed on high-performance computing (HPC) infrastructure.

However, integrating quantum and classical components introduces new challenges in scalability, interpretability, and resource management [6–8]. A defining characteristic of QCSC workflows is their reliance on algorithms that combine stochastic quantum subroutines with classical control logic, particularly at HPC scale [9]. These algorithms are often sensitive to hyperparameter configurations, hardware noise, and execution timing, making their behavior difficult to predict or reproduce [10].

The complexity is further compounded by the high cost of execution, which includes not only quantum circuit sampling but also the overhead of HPC job scheduling and resource reservation. These constraints limit the number of feasible trials and underscore the need for introspection during algorithm development.

While observability has long been a focus in HPC systems, most existing efforts emphasize system-centric metrics such as CPU utilization, memory bandwidth, and thermal states [11–14]. These metrics are primarily consumed by system administrators for performance tuning and anomaly detection. In contrast, QCSC workflows demand application-centric observability—the ability to monitor solver behavior, track intermediate artifacts, and analyze scientific outcomes.

Previous work has explored hardware-level observability in quantum systems using sensor networks to infer quantum error characteristics [15], but such approaches are limited to low-level signals and do not capture the full complexity of hybrid algorithm execution.

In this study, we introduce an observability architecture that integrates with workflow management systems to support persistent monitoring and post-hoc analysis of QCSC workloads. The remainder of the paper is organized as follows. Section II presents a generic system architecture for QCSC workload execution. Section III describes a reference implementation on the Miyabi supercomputer and IBM Quantum. Section IV demonstrates the utility of the observability framework using a practical example involving the differential evolution algorithm applied to sample-based quantum diagonalization (SQD). We conclude with a discussion of future directions in Section V.

II. ARCHITECTURE

To structure our discussion, we introduce a workflow metrics pyramid that categorizes telemetry data into five levels, as shown in Fig. 1(a):

L0. Hardware-level: power, thermal, ...

L1. System-level: CPU, memory, I/O, network ...

L2. Job-level: accounting, node usage, ...

L3. Task-level: artifacts, wall-clock times, ...

L4. Domain-level: solver convergence, fidelity, ...

* Equal contributions; Now at another institution.
† horii@jp.ibm.com
‡ u542400c@ecs.osaka-u.ac.jp; Equal contributions

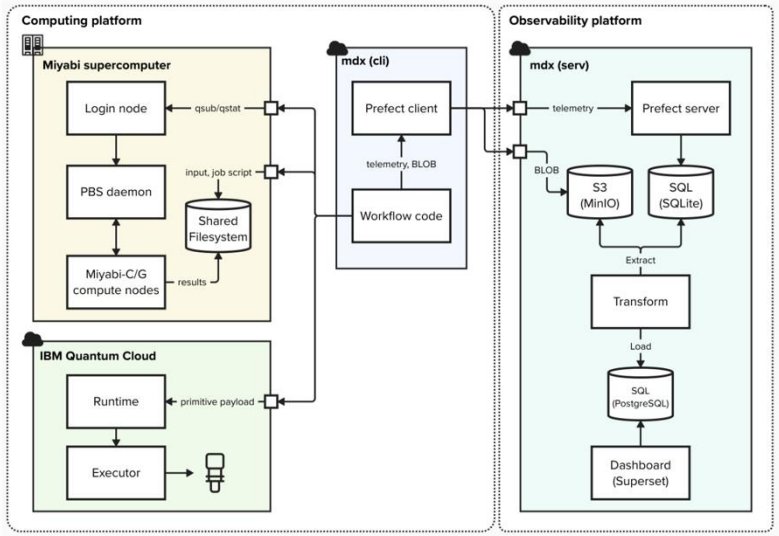
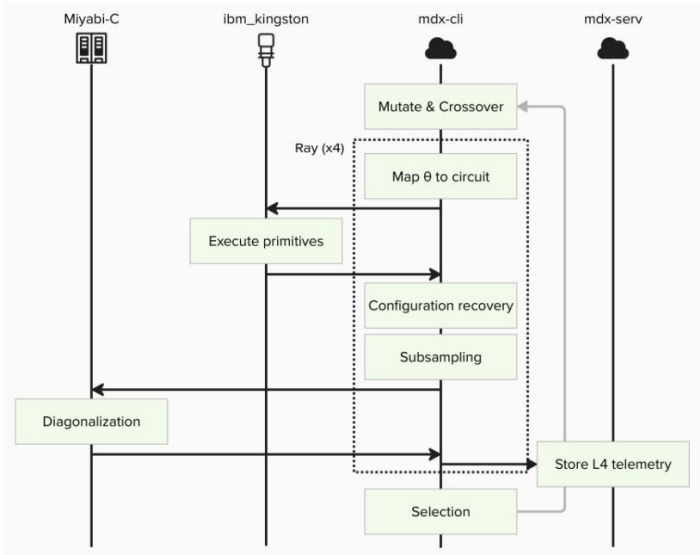


FIG. 2. Component diagram of QCSC observability architecture implemented on the Miyabi supercomputer and mdx platform.



GPU-accelearated Sample-based Quantum Diagonalization

GPU-Accelerated Selected Basis Diagonalization with Thrust for SQD-based Algorithms

Jun Doi
IBM Quantum
IBM Research - Tokyo
Tokyo, Japan
doichan@jp.ibm.com

Tomonori Shirakawa
Center for Computational Science
RIKEN
Kobe, Japan
t-shirakawa@riken.com

Yukio Kawashima
IBM Quantum
IBM Research - Tokyo
Tokyo Japan
yukio.kawashima@ibm.com

Seiji Yunoki
Center for Computational Science
RIKEN
Kobe, Japan
yunoki@riken.jp

Hiroshi Horii
IBM Quantum
IBM Research - Tokyo
Tokyo Japan
horii@jp.ibm.com

***Abstract*—Selected Basis Diagonalization (SBD) plays a central role in Sample-based Quantum Diagonalization (SQD), where iterative diagonalization of the Hamiltonian in selected configuration subspaces forms the dominant classical workload. We present a GPU-accelerated implementation of SBD using the Thrust library. By restructuring key components—including configuration processing, excitation generation, and matrix-vector operations—around fine-grained data-parallel primitives and flattened GPU-friendly data layouts, the proposed approach efficiently exploits modern GPU architectures. In our experiments, the Thrust-based SBD achieves up to ∼40× speedup over CPU execution and substantially reduces the total runtime of SQD iterations. These results demonstrate that GPU-native parallel primitives provide a simple, portable, and high-performance foundation for accelerating SQD-based quantum–classical workflows.**

***Index Terms*—Selected Basis Diagonalization, Sample-Based Quantum Diagonalization, GPU Computing, Thrust Library, Parallel Algorithms, Davidson Method, Quantum Chemistry Simulation, Quantum–Classical Hybrid Computing**

I. Introduction

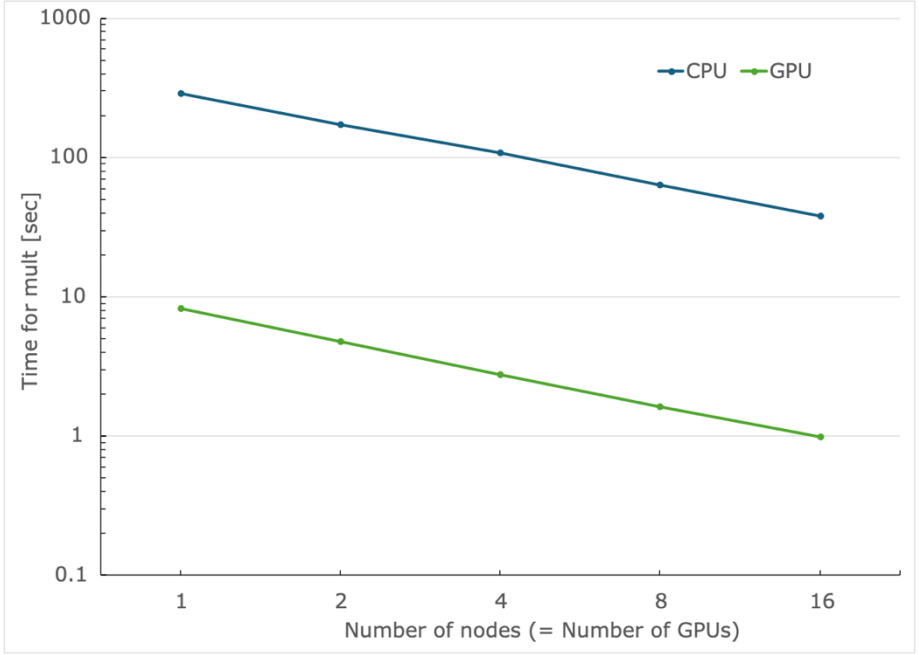
Hybrid quantum–classical algorithms have emerged as a powerful approach for studying strongly correlated quantum systems, in which quantum processors generate physically meaningful samples while classical high-performance computing (HPC) resources perform the demanding post-processing required for accurate energy estimation. Among these approaches, *Sample-based Quantum Diagonalization* (SQD) [6] has recently been proposed as a workflow that iteratively refines a subspace of many-electron configurations based on samples obtained from a quantum circuit. By encoding a molecular Hamiltonian into a parameterized quantum circuit and repeatedly sampling the resulting quantum state, SQD obtains electronic configurations whose probabilities reflect their importance for the target ground or excited states. These configurations are then filtered and ranked on classical hardware to construct a reduced Hilbert-space subspace on which a diagonalization procedure is performed.

Identify applicable funding agency here. If none, delete this.

A defining characteristic of SQD is that the *classical diagonalization step dominates the overall computational cost*, particularly for chemically accurate ground-state calculations. Each SQD iteration requires solving an eigenvalue problem on a selected configuration space that may contain up to 10^8–10^{10} determinants. Iterative eigensolvers such as the Davidson method therefore spend most of their runtime evaluating Hamiltonian matrix–vector products, whose cost scales with the size of the reduced basis. As a result, large-scale SQD calculations require substantial HPC resources even when quantum sampling is efficient.

Recent large-scale SQD studies [2], [6] have been performed on the Fugaku supercomputer using a highly optimized CPU-based implementation of Selected Basis Diagonalization (SBD) [1]. These studies demonstrated that SQD can scale to configuration spaces far beyond the reach of exact diagonalization when supported by massive CPU parallelism and large memory footprints. However, modern leadership-class supercomputers increasingly adopt GPU accelerators as their primary compute engines, as seen in systems such as Frontier, Aurora, and other exascale platforms. To take full advantage of these architectures, diagonalization methods must be redesigned to exploit fine-grained parallelism, high thread concurrency, and GPU-resident data structures.

Furthermore, as SQD evolves, the classical diagonalization backend must support a broader range of Hamiltonian evaluation strategies. While the tensor-based "half-bitstring" representation adopted in prior work is widely used and effective for certain structured systems, our implementation is designed to *generalize* this approach to also support full-bitstring representations, arbitrary spin symmetries, and more flexible operator forms. Although full-bitstring support is not yet enabled in the current release, the data layouts and excitation iterators have been structured to admit this extension with minimal changes. These requirements further motivate the development of GPU-native data layouts and parallel algorithms capable of sustaining high performance for both structured and sparse



Appendix: Using GPU Solvers and Custom Solver Blocks

This section explains how to:

1. Create a GPU-enabled SBD solver Block
2. Select which Solver to use at run time
3. Add custom Solver configurations without modifying the workflow code

This is an advanced usage of the SBD closed-loop workflow.

B. Creating a GPU Solver Block

You already created a CPU solver Block:

```
davidson-solver
```

Now, we will create a GPU-enabled version.

B.1 Create a new Block Instance

Run:

```
prefect block create sbd-solver-job
```

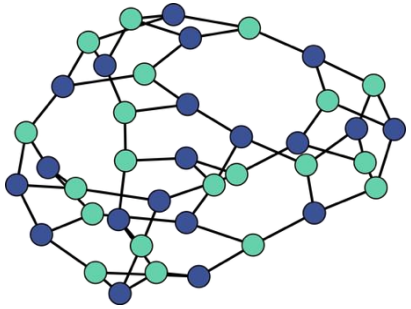
Open the displayed URL and fill in the fields.

Field	Value / Example
Block Name	`davidson-solver-gpu`
Root Directory	`/work/gz09/z12345/observability_demo_jobs` The absolute path to the work directory we created above.
Executable	`/work/gz00/z12345/sbd/diag` The absolute path to the `GPU version's diag` executable.

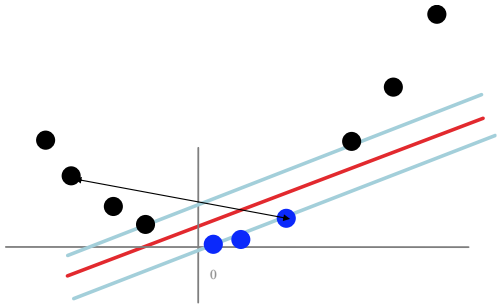
Key quantum computational areas

Hamiltonian simulation

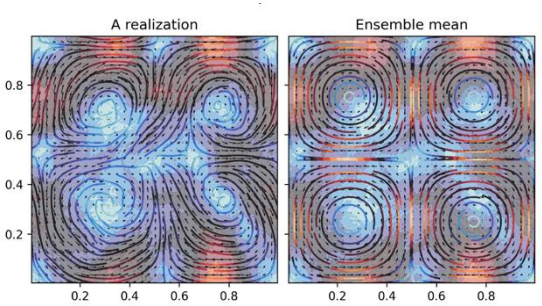
Optimization



Machine learning



Differential equations




IBM