CONQURE: A Co-Execution Environment for Quantum and Classical Resources

Atulya Mahesh[1], Frank Mueller[1]
[1]*North Carolina State University*, mueller@cs.ncsu.edu

***Abstract*—Quantum computing holds potential for accelerating complex kernels, but its integration faces significant hurdles, including fragmented hardware and software ecosystems, lack of standardized abstractions, and inefficient workload management. Proprietary systems often restrict flexibility, while open-source alternatives assume specific hardware, limiting broad adoption. Moreover, automating executions on experimental devices is challenging due to transient usage patterns and suboptimal scheduling in shared environments. These issues underscore the need for a private cloud queue that enables secure, scalable job submission, persistence, and tracking without relying on commercial providers, facilitating remote access across research labs while maintaining authentication and modularity.**

This work proposes CONQURE**, a co-execution environment for quantum and classical high-performance computing (HPC) resources.** CONQURE **is a fully open-source framework that introduces a novel modular scheduling system integrated with a private cloud queue. It allows users to offload quantum kernels as circuits, relay results back to classical contexts, and manage quantum resources efficiently via the** CONQURE **API, addressing key gaps in scalability and automation for near-term quantum applications.**

***Index Terms*—Quantum Computing, Quantum Cloud Portal, Job Management**

I. Introduction

Quantum computing has the potential to disrupt classical computing for a number of fields including clinical research [1], optimization [2], high-energy physics [3], finance [4] and logistics [5], where a potential for significant speedups over its classical algorithms exists. A number of device technologies ranging from superconducting devices (e.g., IBM Q, Rigetti, QCI, OQC, Google, Amazon) to ion traps (IonQ, Quantinuum) to neutral atoms (QuEra, Pasqal) have become available through cloud access, e.g., via Amazon Braket [6], Azure Quantum [7], qBraid [8] or directly from the vendors. Existing commercial solutions feature quantum computing ecosystems on the basis of Python packages to facilitate software development over a number of hardware devices. Today's leading quantum computing hardware technologies include Superconducting qubits (IBM, Google, Amazon, Alice & Bob), Ion-Traps (IONQ, Quantinuum) and Neutral Atoms (QuEra Computing Inc.) and utilize software libraries like Qiskit [9], Cirq [10], Tket [11], Pennylane [12] and CUDA-Q [13]. Lower-level control of the hardware is enabled by interfacing through proprietary software layers, or research infrastructure such as DAX [14], ARTIQ [15], Qubic [16], [17], and QICK [18]. This ecosystem of hardware and software technologies supports a growing interest in quantum computing and is evolving rapidly.

Despite its potential, integrating quantum computing tasks in practical systems is held back by the variability in hardware resources and their supported software packages and sub-optimal job scheduling in shared-resource environments. Furthermore, automating workload execution on smaller-scale devices is challenging due to their experimental setting with specialized equipment and transient usage patterns.

To address the issues of efficient scheduling, automation and scattered tool integration, we introduce CONQURE, a novel fully featured modular quantum stack with Private Cloud integration. The contributions of this paper are as follows:

- We create an open-source software framework for the management of quantum resources and workloads.
- We evaluate our integrated software stack on simulated workloads.
- We demonstrate functionality of CONQURE on an experimental ion-trap device.

II. Background

The integration of quantum computing workloads into HPC requires interoperability between various resources, both classical and quantum, across the hardware and software stack. The fragmented landscape of frameworks supporting quantum resources, in particular, necessitates a middleware solution that can seamlessly integrate frameworks at different layers in the software stack, and specifically HPC software stacks while still providing Python compatibility. CONQURE's architecture is designed to facilitate the automation of workloads on quantum devices today, eliminating concerns of software interoperability, development of supported QIR and hardware-specific toolchains.

Proprietary systems mandate the use of certain software libraries, which support specific hardware resources. While open-source solutions exist, they too come with assumptions about the hardware they are meant to be run on. While a number of well established frameworks exist and pipelines

optimized for different kinds of devices have been developed, the average user is forced to make a decision which specific software library to use, based on the hardware they want to execute on. In contrast, our design allows the use of different hardware and software libraries via our custom translation layer.

Other middleware architectures, such as UQP [19], only support sequential operations as defined by their QIR. This limits their usability on near-term devices, which benefit from parallelization as a vital mechanism to mitigate the effect of qubit decoherence on overall noise.

III. Design

CONQURE is designed to provide an easy-to-deploy system that enables users to efficiently schedule workloads destined for quantum devices. This is done while also providing abstractions that simplify the submission of jobs and subsequent retrieval of results. While QisDAX [20] established a translation from Qiskit to DAX, CONQURE extends this by integrating a cloud-queuing mechanism and job persistence via an integrated database. CONQURE focuses on providing a robust infrastructure for handling job submissions, improving resource utilization, and ensuring scalability.

A. CONQURE *Stack*

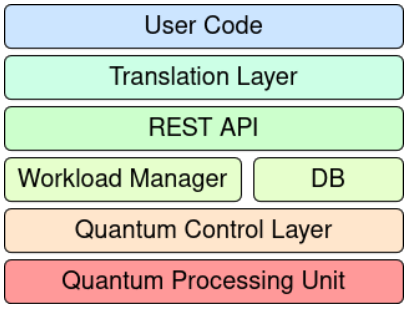

Fig. 1: CONQURE Hardware-Software Stack

CONQURE employs a layered architecture designed for maximum flexibility across quantum hardware and software platforms. As shown in Fig. 1, the system is designed with five modular layers that can be adapted for various architectures:

- **User Interface Layer:** Provides programming frontends (Qiskit, Circ, Tket, etc.) for quantum kernel specification and classical-quantum workflow orchestration.
- **Translation Layer:** Converts platform-agnostic quantum operations into hardware-specific instructions through interchangeable adapters. This layer abstracts vendor-specific compilation and optimization routines. This layer can be bypassed for use cases where the user has already created a workload with hardware specific routines.
- **Workload Manager:** Combines cloud queuing, job scheduling, and resource management subsystems. Implements priority-based execution policies and hybrid workflow coordination between classical and quantum resources.
- **DB:** Maintains job metadata, quantum circuit definitions, and execution results.
- **Quantum Control Layer:** Device-agnostic interface for pulse-level control systems, designed to support diverse qubit technologies through pluggable drivers.

The architecture enables the replacement of components at any layer, i.e., users can substitute the quantum control layer to one designed for ion traps while designing circuits in Qiskit, or they can replace the scheduling subsystem without affecting higher-level APIs. This modularity ensures compatibility with emerging hardware technologies and evolving HPC software ecosystems.

CONQURE *Cloud Queue API*

Compatibility with AWS Cloud Queue for Quantum Devices: CONQURE is designed as an interoperable alternative for AWS' cloud queue for quantum devices [21], ensuring that researchers using AWS do not need to change their existing applications, but also providing missing components for an independent open-source stack for quantum researchers as opposed to commercial quantum service providers. The API calls for `get_results()` and `create_work()`, depicted in Listing 1, are compliant with AWS' Cloud Queue specification. Our API serves as a bridge to connect the translated user written code with the private cloud where the workload is run on either integrated QPUs or simulators. The red dashed box in Fig 2 signifies where the AWS-compliant interface CONQURE is situated. However, in contrast to AWS, CONQURE is designed to provide services in a private cloud, i.e., an environment with authentication requirements that enables remote quantum kernel execution across research labs. Here, authentication needs to be maintained a priori to spawn CONQURE services.

Scalability and Performance: Given that CONQURE is intended for use by researchers across labs, scalability was a key consideration in the API design. The system must be able to handle high volumes of requests without compromising performance. Fig. 2 outlines the flow of data across different parts of the architecture. Our API supports asynchronous job submissions, allowing users to submit multiple jobs without needing to wait for their completion. This is particularly important here because job execution times are expected to vary significantly. Job queuing via SLURM [22] ensures efficient resource utilization that can be tailored for each hardware device.

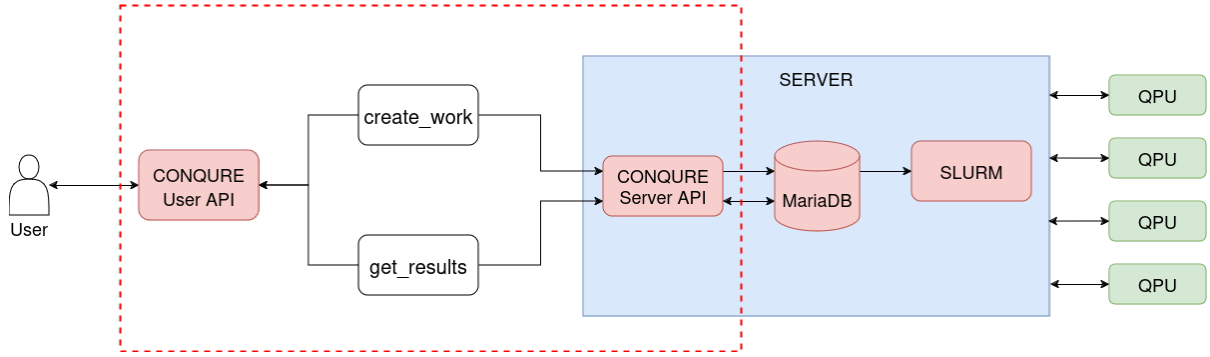

Fig. 2: CONQURE Cloud Queue Architecture depicting API calls (red dashed box) and interactions between modules

Job Persistency and Tracking: It is important that the end-users are able to manage and track their workloads after submission. Persistent job storage allows users to retrieve information about submitted jobs, including workload information and target devices. Users can also query the status of jobs at any time to track their completion, or to retrieve historical results from prior executions.

Modularity and Adaptability: CONQURE is designed as an intermediate layer between the software translation layers and the hardware control layer. It is crucial that it is able to integrate with different implementations of said software and hardware layers. Our API abstracts out this information and allows the user to send as a workload only those data items that are required by their downstream hardware control layer of choice.

Listing 1: CONQURE User Code example

```
from qiskit import QuantumCircuit, execute
from qiskit.providers.dax import DAX
import conqure

# GHZ Circuit definition in Qiskit
num_qubits = 4
ghz_circuit = QuantumCircuit(num_qubits)
ghz_circuit.h(0)

for i in range(num_qubits - 1):
    ghz_circuit.cx(i, i + 1)
ghz_circuit.measure_all()

# QisDAX Translation Layer
dax = DAX.get_provider()
backend_name = 'dax_artiq_device'
backend = dax.get_backend(backend_name)
backend.load_config("resources.toml")
dax_job = execute(ghz_circuit, backend,
                  shots=30)
workload = dax_job.get_dax()

# CONQURE UserClient
client = conqure.UserClient()
work_id = client.create_work(
    workload=workload,
    device_id=backend_name,
    priority="LOW"
)
client.wait_until_done(work_id)
results = client.get_results(work_id)

# QisDAX Translation Layer
qiskit_result =
↪  dax_job.get_result_obj(results)
```

IV. Results

CONQURE was deployed and tested using simulators and an ion-trap quantum device at the Duke Quantum Center [23], which we have access to. The CONQURE Cloud Queue framework was tested by running circuits from the Supermarq suite of benchmarks [24] on an experimental ion-trap device at Duke Quantum Center with 6 qubits and via simulation (local and remote/hosted). In addition, we a parallelized a VQE implementation, similar to the one described in [25] in an experiment around a max-cut problem on a graph with 7 vertices. Fig. 3 shows the convergence of various QPU runs with randomized initial parameters, i.e., each colored curve corresponds to a different set of ansatz parameters resulting in decreasing cost (y-axis). We schedule these VQE runs (a) serially and (b) in parallel on simulators, in the latter case to show the potential of parallelization over QPUs as a concept to optimize convergence time, i.e., the different runs themselves are parallelized resulting in a 3× speedup of execution times for the 6 parallel runs, which reflects cloud overhead vs. circuit lengths in simulation. For longer circuits on different QPU hardware devices, higher speedups (close to linear) can be expected.

V. Related Work

Research into the integration of quantum computing into HPC environments is a critical endeavor, driven by the objective to realize quantum advantage on computationally complex kernels in classical terms, which can be solved more efficiently on quantum devices.

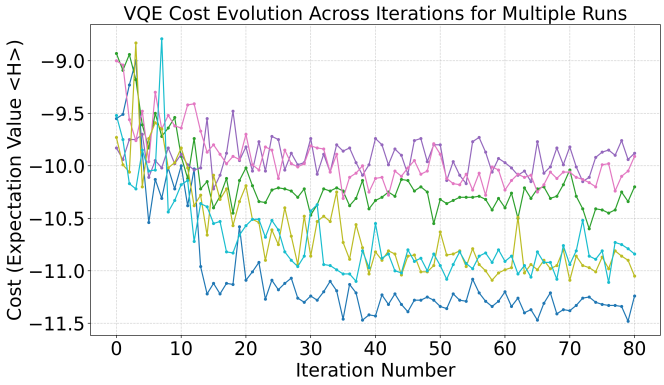

Fig. 3: Convergence of 6 different VQE runs for a Max-Cut Problem on a graph with 7 vertices

Saurabh et al. proposed a conceptual middleware solution [26] to build a foundation for the future of HPC middleware systems. Mantha et al. built upon this foundation with a middleware solution designed to manage classical and quantum resources at the application level [27]. UQP [19] introduced a platform designed for the integration of HPC environments with quantum accelerators. It contributed a novel ISA understood by UQP, which is translated from a Quantum Intermediate Representation (QIR). CONQURE's modularity and private cloud complement these designs while focusing on providing a comprehensive framework for managing quantum workloads across hardware platforms.

NordIQuEst [28] provides federated, regional access to quantum resources through unified interfaces and SLURM-based workload management, supporting multiple software frameworks for hybrid quantum-classical workflows. QDMI [29] provides a standardized and technology-agnostic C interface for managing quantum devices within HPC environments. It enables real-time, dynamic communication between software tools and hardware, allowing software to query device characteristics. CONQURE also aims to integrate quantum computers into HPC environments but focuses on persistent job orchestration, the integration of frameworks with a private cloud, and workflow automation in hybrid research environments.

The scheduling and management of resources and workloads is critical in the practical deployment of these tools. Qoncord [30] tackled the problem of workload scheduling in cloud environments and proposed a novel scheduling framework that capitalizes on the differences in approximation errors over in different phases of VQA. It splits VQA runs into distinct exploratory and fine-tuning phases, and identified that higher noise was acceptable in the exploratory phase and exploited this to schedule it with lower priority. Unlike Qoncord's focus on noise resilience, CONQURE focuses on scalability, job persistence and tracking.

VI. Conclusion

This work proposes CONQURE, a novel co-execution framework that addresses these gaps through a modular five-layer architecture. It includes user interfaces, a translation layer, workload management, job scheduling, database persistence, and a quantum control layer, enabling compatibility across diverse software and hardware platforms. By integrating a private cloud queue, CONQURE facilitates efficient resource utilization and automation for near-term quantum applications, paving the way for scalable quantum-classical co-execution. As CONQURE open-source software, it democratizes quantum research and offers an unprecedented HPC integration.

Acknowledgment

This work was supported in part by NSF awards MPS-2410675, PHY-1818914, PHY-2325080, OMA-2120757, and CISE-2316201. We would like to thank Andrew Van Horn from Duke Quantum Center for his support in the testing of CONQURE on real devices.

References

[1] D. Solenov, J. Brieler, and J. F. Scherrer, "The potential of quantum computing and machine learning to advance clinical research and change the practice of medicine," *Mo. Med.*, vol. 115, no. 5, pp. 463–467, Sep. 2018.

[2] A. Abbas, A. Ambainis, B. Augustino, A. Bärtschi, H. Buhrman, C. Coffrin, G. Cortiana, V. Dunjko, D. J. Egger, B. G. Elmegreen, N. Franco, F. Fratini, B. Fuller, J. Gacon, C. Gonciulea, S. Gribling, S. Gupta, S. Hadfield, R. Heese, G. Kircher, T. Kleinert, T. Koch, G. Korpas, S. Lenk, J. Marecek, V. Markov, G. Mazzola, S. Mensa, N. Mohseni, G. Nannicini, C. O'Meara, E. P. Tapia, S. Pokutta, M. Proissl, P. Rebentrost, E. Sahin, B. C. B. Symons, S. Tornow, V. Valls, S. Woerner, M. L. Wolf-Bauwens, J. Yard, S. Yarkoni, D. Zechiel, S. Zhuk, and C. Zoufal, "Challenges and opportunities in quantum optimization," *Nature Reviews Physics*, vol. 6, no. 12, pp. 718–735, Dec. 2024.

[3] A. Di Meglio, K. Jansen, I. Tavernelli, C. Alexandrou, S. Arunachalam, C. W. Bauer, K. Borras, S. Carrazza, A. Crippa, V. Croft, R. de Putter, A. Delgado, V. Dunjko, D. J. Egger, E. Fernández-Combarro, E. Fuchs, L. Funcke, D. González-Cuadra, M. Grossi, J. C. Halimeh, Z. Holmes, S. Kühn, D. Lacroix, R. Lewis, D. Lucchesi, M. L. Martinez, F. Meloni, A. Mezzacapo, S. Montangero, L. Nagano, V. R. Pascuzzi, V. Radescu, E. R. Ortega, A. Roggero, J. Schuhmacher, J. Seixas, P. Silvi, P. Spentzouris, F. Tacchino, K. Temme, K. Terashi, J. Tura, C. Tüysüz, S. Vallecorsa, U.-J. Wiese, S. Yoo, and J. Zhang, "Quantum computing for high-energy physics: State of the art and challenges," *PRX Quantum*, vol. 5, p. 037001, Aug 2024. [Online]. Available: https://link.aps.org/doi/10.1103/PRXQuantum.5.037001

[4] D. Herman, C. Googin, X. Liu, Y. Sun, A. Galda, I. Safro, M. Pistoia, and Y. Alexeev, "Quantum computing for finance," *Nature Reviews Physics*, vol. 5, no. 8, p. 450–465, Jul. 2023. [Online]. Available: http://dx.doi.org/10.1038/s42254-023-00603-1

[5] F. Phillipson, "Quantum computing in logistics and supply chain management an overview," 2025. [Online]. Available: https://arxiv.org/abs/2402.17520

[6] "AWS. Amazon braket." [Online]. Available: https://aws.amazon.com/braket/

[7] Microsoft, "Azure quantum development kit." [Online]. Available: https://github.com/microsoft/qsharp

[8] R. J. Hill, R. Jain, H. Gupta, T. Jun Liang, M. M. Louamri, R. Young, E. Weis, K. Tsuoka, G. Jacobson, C. McIrvin, P. Weinberg, S. Purohit, J. Necaise, E. A. Vara, P. Chakraborty, J. Liu, A. W. Coladangelo, P. Kakhandiki, H. Makhanov, P. Sharma, A. Arulandu, A. Cosentino, and K. Setia, "qBraid-SDK: Platform-agnostic quantum runtime framework." Mar. 2025. [Online]. Available: https://github.com/qBraid/qBraid

[9] A. Javadi-Abhari, M. Treinish, K. Krsulich, C. J. Wood, J. Lishman, J. Gacon, S. Martiel, P. D. Nation, L. S. Bishop, A. W. Cross, B. R. Johnson, and J. M. Gambetta, "Quantum computing with qiskit," 2024. [Online]. Available: https://arxiv.org/abs/2405.08810

[10] C. Developers, *Cirq*. Zenodo, Apr. 2025. [Online]. Available: https://zenodo.org/doi/10.5281/zenodo.4062499

[11] S. Sivarajah, S. Dilkes, A. Cowtan, W. Simmons, A. Edgington, and R. Duncan, "t—ket⟩: a retargetable compiler for nisq devices," *Quantum Science and Technology*, vol. 6, no. 1, p. 014003, nov 2020. [Online]. Available: https://dx.doi.org/10.1088/2058-9565/ab8e92

[12] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, S. Ahmed, V. Ajith, M. S. Alam, G. Alonso-Linaje, B. AkashNarayanan, A. Asadi, J. M. Arrazola, U. Azad, S. Banning, C. Blank, T. R. Bromley, B. A. Cordier, J. Ceroni, A. Delgado, O. D. Matteo, A. Dusko, T. Garg, D. Guala, A. Hayes, R. Hill, A. Ijaz, T. Isacsson, D. Ittah, S. Jahangiri, P. Jain, E. Jiang, A. Khandelwal, K. Kottmann, R. A. Lang, C. Lee, T. Loke, A. Lowe, K. McKiernan, J. J. Meyer, J. A. Montañez-Barrera, R. Moyard, Z. Niu, L. J. O'Riordan, S. Oud, A. Panigrahi, C.-Y. Park, D. Polatajko, N. Quesada, C. Roberts, N. Sá, I. Schoch, B. Shi, S. Shu, S. Sim, A. Singh, I. Strandberg, J. Soni, A. Száva, S. Thabet, R. A. Vargas-Hernández, T. Vincent, N. Vitucci, M. Weber, D. Wierichs, R. Wiersema, M. Willmann, V. Wong, S. Zhang, and N. Killoran, "Pennylane: Automatic differentiation of hybrid quantum-classical computations," 2022. [Online]. Available: https://arxiv.org/abs/1811.04968

[13] NVIDIA, "Cuda-q," 2025. [Online]. Available: https://nvidia.github.io/cuda-quantum/

[14] L. Riesebos, B. Bondurant, J. Whitlow, J. Kim, M. Kuzyk, T. Chen, S. Phiri, Y. Wang, C. Fang, A. V. Horn, J. Kim, and K. R. Brown, "Modular software for real-time quantum control systems," in *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE, Sep. 2022, p. 545–555. [Online]. Available: http://dx.doi.org/10.1109/QCE53715.2022.00077

[15] S. Bourdeauducq, R. Jördens, P. Zotov, J. Britton, D. Slichter, D. Leibrandt, D. Allcock, A. Hankin, F. Kermarrec, Y. Sionneau, R. Srinivas, T. R. Tan, and J. Bohnet, "Artiq 1.0," May 2016. [Online]. Available: https://doi.org/10.5281/zenodo.51303

[16] Y. Xu, G. Huang, J. Balewski, R. Naik, A. Morvan, B. Mitchell, K. Nowrouzi, D. I. Santiago, and I. Siddiqi, "Qubic: An open-source fpga-based control and measurement system for superconducting quantum information processors," *IEEE Transactions on Quantum Engineering*, vol. 2, p. 1–11, 2021. [Online]. Available: http://dx.doi.org/10.1109/TQE.2021.3116540

[17] Y. Xu, G. Huang, N. Fruitwala, A. Rajagopala, R. K. Naik, K. Nowrouzi, D. I. Santiago, and I. Siddiqi, "Qubic 2.0: An extensible open-source qubit control system capable of mid-circuit measurement and feed-forward," 2023. [Online]. Available: https://arxiv.org/abs/2309.10333

[18] L. Stefanazzi, K. Treptow, N. Wilcer, C. Stoughton, C. Bradford, S. Uemura, S. Zorzetti, S. Montella, G. Cancelo, S. Sussman, A. Houck, S. Saxena, H. Arnaldi, A. Agrawal, H. Zhang, C. Ding, and D. I. Schuster, "The QICK (quantum instrumentation control kit): Readout and control for qubits and detectors," *Rev. Sci. Instrum.*, vol. 93, no. 4, p. 044709, Apr. 2022.

[19] A. Elsharkawy, X. Guo, and M. Schulz, "Integration of quantum accelerators into hpc: Toward a unified quantum platform," 2024. [Online]. Available: https://arxiv.org/abs/2407.18527

[20] K. Badrike, A. S. Dalvi, F. Mazurek, M. D'Onofrio, J. Whitlow, T. Chen, S. Phiri, L. Riesebos, K. R. Brown, and F. Mueller, "Qisdax: An open source bridge from qiskit to trapped-ion quantum devices," *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 01, pp. 825–836, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:260477619

[21] "AWS Cloud Queue for Quantum Devices." [Online]. Available: https://aws.amazon.com/blogs/quantum-computing/new-open-source-tool-expands-access-to-lab-based-quantum-prototypes-cloud-queue-for-quantum-devices/

[22] A. B. Yoo, M. A. Jette, and M. Grondona, "Slurm: Simple linux utility for resource management," in *Job Scheduling Strategies for Parallel Processing*, D. Feitelson, L. Rudolph, and U. Schwiegelshohn, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 44–60.

[23] D. Q. Center, "Homepage." [Online]. Available: https://quantum.duke.edu/

[24] T. Tomesh, P. Gokhale, V. Omole, G. S. Ravi, K. N. Smith, J. Viszlai, X.-C. Wu, N. Hardavellas, M. R. Martonosi, and F. T. Chong, "Supermarq: A scalable quantum benchmark suite," in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2022, pp. 587–603.

[25] D. Tsukayama, J.-i. Shirakashi, T. Shibuya, and H. Imai, "Enhancing computational accuracy with parallel parameter optimization in variational quantum eigensolver," *AIP Advances*, vol. 15, no. 1, p. 015226, 01 2025. [Online]. Available: https://doi.org/10.1063/5.0236028

[26] N. Saurabh, S. Jha, and A. Luckow, "A conceptual architecture for a quantum-hpc middleware," in *2023 IEEE international conference on quantum software (QSW)*. IEEE, 2023, pp. 116–127.

[27] P. Mantha, F. J. Kiwit, N. Saurabh, S. Jha, and A. Luckow, "Pilot-quantum: A quantum-hpc middleware for resource, workload and task management," 2024. [Online]. Available: https://arxiv.org/abs/2412.18519

[28] C. Carugno, J. Muff, M. P. Johansson, S. Karlsson, and A. Lanzanova, "Nordiquest: the nordic-estonian quantum computing e-infrastructure quest," 2024. [Online]. Available: https://arxiv.org/abs/2406.02216

[29] R. Wille, L. Schmid, Y. Stade, J. Echavarria, M. Schulz, L. Schulz, and L. Burgholzer, "QDMI – Quantum Device Management Interface: A Standardized Interface for Quantum Computing Platforms," in *IEEE International Conference on Quantum Computing and Engineering (QCE)*.

[30] M. Wang, P. Das, and P. J. Nair, "Qoncord: A multi-device job scheduling framework for variational quantum algorithms," in *2024 57th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, Nov. 2024, p. 735–749. [Online]. Available: http://dx.doi.org/10.1109/MICRO61859.2024.00060