Interfacing Quantum Computing Systems with High-Performance Computing Systems: An overview

KONSTANTINOS RALLIS, National Centre for Scientific Research "Demokritos", Greece and Democritus University of Thrace, Greece

IOANNIS LILIOPOULOS, National Centre for Scientific Research "Demokritos", Greece and Democritus University of Thrace, Greece

GEORGIOS D. VARSAMIS, National Centre for Scientific Research "Demokritos", Greece and Democritus University of Thrace, Greece

EVANGELOS TSIPAS, National Centre for Scientific Research "Demokritos", Greece and Democritus University of Thrace, Greece

IOANNIS G. KARAFYLLIDIS, Democritus University of Thrace, Greece

GEORGIOS CH. SIRAKOULIS, Democritus University of Thrace, Greece

PANAGIOTIS DIMITRAKIS, National Centre for Scientific Research "Demokritos", Greece

The connection and eventual integration of High-Performance Computing (HPC) with Quantum Computing (QC) represents a transformative advancement in computational technology, promising significant enhancements in solving complex, previously intractable problems. This manuscript provides a comprehensive overview of the current state of HPC-QC interfacing, detailing architectural methodologies, software stack developments, middleware functionalities, and hardware integration strategies. It critically assesses existing hardware-level integration models, ranging from standalone and loosely-coupled architectures to tightly-integrated and on-node systems. The software ecosystem is analyzed, highlighting prominent frameworks such as Qiskit, PennyLane, CUDA-Q, and middleware solutions like Pilot-Quantum, essential for seamless hybrid computing environments. Furthermore, the manuscript discusses practical applications in optimization, machine learning, and many-body dynamics, where hybrid HPC-QC systems can offer substantial advantages. It also describes existing challenges, including hardware limitations (coherence, scalability, connectivity), software maturity, communication overhead, resource management complexities, and cost factors. Finally, future directions towards tighter hardware and software integration are discussed, emphasizing ongoing research developments and emerging trends that promise to expand the capabilities and accessibility of hybrid HPC-QC systems.

Additional Key Words and Phrases: Quantum Software and Middleware, Quantum Programming Frameworks, Distributed Systems Architecture, High-Performance Computing (HPC), Hybrid Quantum-Classical Computing

1 Introduction

As computing technology continues to evolve, it is increasingly applied to a broader range of problems in both research and industry, as well as in everyday life. This technological expansion is inevitably accompanied by a growing demand for computational resources to address increasingly complex and data-intensive large-scale problems. To cover the growing demands for computational resources and computation power, High-Performance Computing (HPC) systems

Authors' Contact Information: Konstantinos Rallis, National Centre for Scientific Research "Demokritos", Athens, Greece and Democritus University of Thrace, Xanthi, Greece, krallis@ee.duth.gr; Ioannis Liliopoulos, National Centre for Scientific Research "Demokritos", Athens, Greece and Democritus University of Thrace, Xanthi, Greece, ililiopo@ee.duth.gr; Georgios D. Varsamis, National Centre for Scientific Research "Demokritos", Athens, Greece and Democritus University of Thrace, Xanthi, Greece, gevarsam@ee.duth.gr; Evangelos Tsipas, National Centre for Scientific Research "Demokritos", Athens, Greece and Democritus University of Thrace, Xanthi, Greece, etsipas@ee.duth.gr; Ioannis G. Karafyllidis, Democritus University of Thrace, Xanthi, Greece, ykar@ee.duth.gr; Georgios Ch. Sirakoulis, Democritus University of Thrace, Xanthi, Greece, gsirak@ee.duth.gr; Panagiotis Dimitrakis, National Centre for Scientific Research "Demokritos", Athens, Greece, p.dimitrakis@qi.demokritos.gr.

have been employed, and so far, they are the dominant approach for dealing with large-scale computations with enhanced performance. Such systems are now extensively used not only by research centers and organizations, but also by industry sectors. Enterprises, governmental bodies, and international collaborations have driven the development, deployment, and enhancement of HPC infrastructures by funding and operating their own systems. At the same time, significant research efforts have focused on optimizing performance, improving energy efficiency, enhancing scalability, and increasing accessibility of these systems.

In parallel, apart from the progress and advancements in High-Performance Computing, novel computing systems have emerged and have already attracted significant research and implementation efforts: Quantum Computers. This technology promises to provide a significant boost to the available computational power and to significantly enhance the problem solving capabilities of the existing systems. By exploiting the fundamental quantum mechanical phenomena that govern the operation of quantum computers, these systems are expected to tackle computationally intensive problems, such as a set of specific NP-hard and NP-complete problems, more efficiently [1].

However, due to current hardware limitations, the NISQ (Noisy Intermediate-Scale Quantum) era quantum computers cannot yet be used for large-scale problems. They are currently still requiring specific installation standards and infrastructure with demanding maintenance. As of today, several big tech companies provide their own quantum computers for use, and with the evolution of the quantum hardware technology, more and more quantum resources with increased computational capabilities (increased number of qubits, the unit of quantum computation) are becoming available to the public [2].

Running a program and solving a problem on a quantum computer while also harvesting its special capabilities and achieving actual performance gains is not completely straightforward. It requires special handling and encoding of data, as well as an appropriate formulation of the problem in order to be solved by a quantum computer [3]. Thus far, there is no single standardized compilation process that automatically formulates and maps every problem to a QC-compatible form, making this an active area of ongoing research.

As previously mentioned and evident from the above, today's NISQ-era quantum computers cannot tackle large-scale problems and require an additional classical system to handle essential computations. As of today, the most commonly accepted way of using a QC system is to have it interfaced and in combination with a classical computing system, mainly an HPC system. In this approach, quantum computers and their computing part, the Quantum Processing Unit (QPU), are treated by the HPC system as another computing unit (resource), just like CPU and GPU nodes in heterogeneous distributed systems [4]. Such an approach, even though promising, introduces a set of hard-to-solve challenges. The HPC system is responsible for formulating, encoding and mapping the problem for the QPU, assign jobs to QPUs and time-manage their operation while also preparing and fetching the required data and instructions to them. This integration of QPUs to HPC computing systems and the effective HPC-QC interfacing are crucial steps for the exploitation of current QC systems, their use in actual problems where they can provide actual advantages in terms of performance, as well as for the development of the QC technology as a whole.

In this work, we present a comprehensive review of state-of-the-art methodologies and emerging trends in the integration of High-Performance Computing (HPC) with Quantum Computing (QC), structured as follows: In Section 2, we provide essential background on HPC and QC technologies, as well as on hybrid HPC-QC systems, highlighting their characteristics and capabilities. Section 3 delves into the software ecosystem, detailing key quantum programming frameworks and middleware solutions, including main representatives such as Qiskit, PennyLane, CUDA-Q, Pilot-Quantum and others. In Section 4, we discuss hardware-level integration strategies, covering physical interconnect technologies and quantum networking. Section 5 illustrates practical application domains where hybrid HPC-QC

systems have shown significant promise, specifically optimization problems, quantum machine learning, and quantum simulations. Then in Section 6, we critically examine existing technical and operational challenges that currently limit broader adoption, and finally summarize and conclude in Section 7, commenting on possible future research directions.

2 Background

2.1 High-Performance Computing Systems

High-Performance Computing (HPC), is a technology that goes far beyond the classical personal computing systems as we know them, not only in terms of their form factor, but also in terms of their operation use. At their core, HPC systems are based on the same operational principles as conventional computing systems. They rely on digital logic circuits and execute Boolean algebra, whose basic computational unit is the binary digit bit, and their basic computing cell unit is the transistor. However, HPC is more than that. It is a technology that uses a set of computing resources, clusters of processing units. Those processing units can be up to several thousands in a single HPC system, and they can also be of a heterogeneous nature. This means they do not only consist of CPUs, but they can also contain other types of specialized accelerators, such as GPUs, especially suitable for parallel processing workloads, FPGAs, TPUs, Neuromorphic Processors and other units, which can be used to handle several diverse tasks [5–7].

Such systems are mainly developed by big tech companies and even countries that are able to handle the costs of manufacturing and maintenance. In order to cover the increasing needs of computing resources in an affordable manner, HPC resources are also provided to the public through the cloud, in the form of service [8].

Nowadays, in the era of Artificial Intelligence (AI), HPC systems are becoming more and more significant and required by a greater pool of users of diverse backgrounds, as they find application in a broader set of fields with increased requirements in computational power and resources. They are the driving force of research in AI, as they can handle through their resources the computing-intensive tasks of training of large scale AI models and preprocessing and manipulation of Big Data [9, 10]. HPC systems are also highly used in the Automotive Industry, where intensive simulation is required for the design and parameter estimation of new products, among others Computational Fluid Dynamic (CFD) simulations for Aerodynamics [11]. Related to that, HPC systems are also used in the field of Energy prediction and storage, including weather forecasting, especially nowadays where renewable energy sources are dominant and are heavily based on external environmental factors (solar irradiance, wind etc.) [12]. They are an indispensable tool in the hands of governments as means for civil protection, used to predict possible natural disasters (i.e. seismic activity, tidal waves, tsunamis etc.) [13]. HPC systems can also be used for tasks related to banking and financing [14, 15] or even, through their heterogeneous resources, in the form of render farms for 3D Visuals [16]. All this is just a representative subset of the full application spectrum of HPC systems.

2.2 Quantum Computing Systems

Quantum computing and quantum computers represent a different approach to computing compared to what has been reported above. A quantum computer is practically a machine that performs computations by leveraging quantum mechanical phenomena, such as superposition and quantum entanglement. The basic unit of quantum computing is the qubit (quantum bit), which, unlike classical bits that can represent information as binary values, is capable of encapsulating not only two values, 0 and 1, but also infinite values in between, in a form of coherent superposition between those two "basis" states. Moreover, when two or more qubits become entangled, their quantum states become

related and interdependent, enabling complex quantum operations that can lead to actual computational advantage [17].

In terms of representing those states and thus physically fabricating the qubits in hardware, there is no single standardized approach dominating the field yet. Several alternative quantum computing hardware technologies have already been proposed and are in active development and investigation. The most common of them include superconducting qubits [18], trapped-ion qubits [19], photonic (light-based) qubits [20], neutral atoms [21], and spin qubits in silicon [22]. Several technology companies and research groups around the world are investigating and investing in one or more of those technologies, each one providing its own characteristics and having its own advantages and disadvantages as shown in Tab. 1, targeting to achieve scalable and fault-tolerant quantum computation.

Quantum computing promises to substantially accelerate solutions to specific classes of problems that even classical HPCs would not be able to solve in feasible time. In fact, what quantum computers are able to do, is to solve specific types of computationally intensive problems, many of which are lying in the NP-complete and NP-hard categories, more efficiently, in some cases even in polynomial time [1]. Such problems, on which quantum computing can be beneficial, are present in a wide variety of fields. Healthcare and more specifically drug discovery is one of the most common, referring mainly to the speedup of protein folding simulations and the investigation of drug target interactions [23–25]. Beyond protein folding, quantum algorithms find application in genomics-based diagnostics, personalized medicine and pathogen detection, through tasks such as DNA sequence alignment and genome assembly [26, 27]. Adding to that, QC can significantly aid Climate Simulations, as climate and weather are highly complex many-dimensional systems and simultaneously help to the highly related task of Energy Grid optimization [28]. Another field of application for QC is Cryptography and Security, mainly based on its capabilities to deliver fast solutions to the factorization problem through the well-known Shor's algorithm [29]. QCs are practically creating a new whole segment in Cryptography, called Post-Quantum Cryptography. Machine Learning and AI are additional areas where the application of QCs is tested, mainly through Quantum Machine Learning, especially for tasks with large feature spaces. QCs can also be exploited for their ability to simulate quantum systems (Quantum Simulation), mainly targeting Material Science, Chemistry and many more [30]. It also offers significant boost to combinatorial problems through enhanced combinatorial optimization [31] and excels in searches in unsorted databases mainly through Grover's Algorithm [32].

But besides its prospects and seemingly wide spectrum of application, QC is not a panacea. The speedup that they offer tends to be problem-specific, fitting specific classes of problems that can harness their special properties, like those that have been mentioned above [33]. Being at a technology readiness level (TRL) of 3-4, and still in the NISQ era, QC technology has yet several significant challenges to overcome, such as error correction, qubit coherence, scalability and many more. Overcoming these obstacles is essential for quantum computing to transition from experimental demonstrations to practical, large-scale, and fault-tolerant computational platforms.

2.3 Interfacing quantum computers with high-performance computers

High-performance computing (HPC) and quantum computing (QC) as presented previously, are two distinct yet complementary technological domains. Interfacing them into quantum-classical Hybrid Systems and leveraging their enhanced combined capabilities provides a great potential to revolutionize various fields, spanning from scientific research to complex industrial applications [34].

Quantum computers can perform certain computations exponentially faster than classical computers, like combinatorial optimization and factorization, unstructured search, and others, making them particularly well-suited for solving complex problems that are intractable for traditional high-performance computing systems. Integrating quantum

Table 1. Comparison of different qubit technologies.

Category	Superconducting Qubits	Trapped-Ion Qubits	Photonic Qubits	Neutral-atom Qubits	Spin Qubits in Silicon	Topological Qubits
Operation	Zero-resistance superconductors at millikelvin temperatures (Josephson junctions).	Ions trapped by electromagnetic fields; quantum states encoded in energy levels.	Quantum information carried by single photons, manipulated/detected optically.	Neutral atoms in optical tweezers/lattices; Rydberg excitation for strong coupling.	Single-electron spins in silicon quantum dots; quantum control similar to gate-controlled transistors.	Quasi-particles (e.g., Majorana fermions).
Pros	Mature fabrication, relatively fast gates.	Long coherence times. High-fidelity gate operations. All qubits are identical by nature.	Fast transmission, less prone to decoherence (weak interaction with the environment).	Highly scalable arrays, long coherence.	Uses mature silicon tech, potential for classical/quantum integration.	Potential intrinsic fault tolerance, reduced overhead for error correction.
Challenges	Requires very low temperatures (deep cryogenics).	Slower gates, difficult scaling and multi-zone control.	Hard to make two-qubit gates, efficient and scalable on-chip photon sources and detectors, and optical loss, are still challenging.	Precise large-array control, high-gate fidelity stability and uniformity.	Demands extreme spin control, decoherence from nuclear spins.	Highly experimental, complex materials, fabrication and scalability unclear.
Key Players	IBM, Google, Rigetti, academic/industrial consortia.	IonQ, Quantinuum (Honeywell), Alpine Quantum Technologies (AQT), various universities.	Xanadu, PsiQuantum, photonics research groups.	QuEra, Pasqal, Infleqtion (ColdQuanta).	Intel, Quantum Motion, UNSW (Australia).	Microsoft.

computers with high-performance computing environments allows for a powerful hybrid approach, where classical systems are used to manage the overall workflow and offload to the QPUs specific tasks that are better suited for quantum processing [35]. At an abstract level, such a workflow will include initially the breaking down -by the HPC- of a complex problem to individual subtasks. The classical system will determine which one of those suits better to be executed by the HPC and which one can benefit from execution on a Quantum Processor. Then the classical system will again proceed with the whole orchestration process, the job scheduling and assignment, as well as the data management. This involves the encoding of data and mapping of the selected tasks to a form that a Quantum Processor is able to understand and process. After the execution of the each subtask by both the classical and quantum nodes, the classical system is again responsible for the handling, decoding and combination of the output data.

As quantum computers and thus quantum processors continue to evolve by increasing the number of fault tolerant qubits, the need to interface them with high-performance computing infrastructures has become increasingly important and promising for the development of the technology [36–38]. As mentioned in [34], a promising approach to address the challenges that arise with the design of large-scale quantum processors is to mimic modern high-performance computing systems, where thousands of heterogeneous processors in various forms and storage units are interconnected

via a communication network, and the computational problems are solved by adopting a distributed computing approach [34]. This will allow quantum processors to be seamlessly integrated into a high-performance computing infrastructure, in the form of additional distributed computing resources, similar to conventional processors, enabling the efficient and optimized distribution and parallelization of workloads between classical and quantum components.

Another key aspect of interfacing quantum computers with high-performance computing systems, is the development of sophisticated software tools and frameworks that can facilitate the integration and smooth operation of these two distinct computing paradigms. This step is crucial for valid mapping of computing problems in a form suitable for Quantum Computers, as well as for appropriate job scheduling and resource allocation, in order to avoid significant overheads and delays that will ultimately reduce the total performance of the Hybrid System.

Recent research has particularly focused on developing programming models and execution frameworks that abstract the complexity of quantum hardware, providing unified interfaces for developers to design and deploy hybrid quantum-classical applications. As described in [4], integration can occur at multiple layers: at the application layer, classical and quantum algorithmic modules are combined via hybrid programming languages or workflow systems; at the compilation layer, hybrid source code is compiled into appropriate instructions; and at the hardware layer, HPC binaries and QPU pulse-level instructions are scheduled for execution on their target devices. Similarly, [39] presents an approach for integrating HPC and QC software environments into a unified workflow, emphasizing the importance of bridging existing software stacks for seamless operation. Architectural solutions like the shared buffer memory space in [40] facilitate direct interaction between host CPUs and quantum processors, managed by accelerator systems and shared memory buffers. Other approaches, such as the unified HPC-QC toolchain discussed in [41], allow for high-level algorithm compilation, resource scheduling, and runtime orchestration across hybrid computational nodes. Finally, cloud-based integration models, reported by [42], showcase the paradigm where clients submit quantum jobs to vendor-managed clouds, with jobs managed, queued, and executed remotely. These architectural approaches illustrate the diverse methods being explored to enable efficient and practical HPC-QC integration.

Several software-based approaches and frameworks have already been developed specifically to address the challenges of integrating quantum computing systems with high-performance computing infrastructures. Those software stacks cover the whole spectrum of topics that are required for an efficient co-existence of HPCs and QCs in Hybrid Systems, from programming and problem mapping to transpiling, scheduling, resource allocating and managing of input and output data. The aforementioned existing architectures for hybrid quantum-classical systems as well as the existing techniques and the software stack for HPC-QC interconnection, will be further analyzed and compared.

3 Software Stack and Middleware

3.1 Programming Models

Over the last few years there has been an extensive effort in order to design better and less noise-prone QCs and more efficient HPC systems [43–46]. Furthermore, there is a drift among the scientific community towards the interconnection of those two systems via a low-latency physical link [47]. Although this research field has drawn the attention of the majority of the scientific community, emphasis shall also be given in designing the appropriate software stack that enables anyone to exploit the computational power originating from the seamless integration of the aforementioned components [39].

To achieve seamless integration between HPCs and QCs, a viable software stack must be able to combine and employ the benefits from two types of systems which are entirely different from one another. On one side, Quantum

Table 2. Most popular commercially available QC programming frameworks.

Name	Vendor	Programming Language	Target Qubit Technology
Qiskit [49]	IBM	Python	Superconducting
Classiq [50]	Classiq Technologies	Python/Qmod [50]	Superconducting, Trapped-Ion, Spin Qubits, Photonic
Cirq [51]	Google	Python	Trapped-Ion, Neutral-Atom
PennyLane [52]	Xanadu	Python	Superconducting, Trapped-Ion, Neutral-Atom, Photonic etc.
CUDA-Q [53]	NVIDIA	C++/Python	Superconducting, Trapped-Ion, Neutral-Atom, Photonic
Braket [54]	Amazon	Python/Julia	Superconducting, Trapped-Ion, Neutral-Atom
Azure Quantum [55]	Microsoft	Python/Q# [56]	Superconducting, Trapped-Ion, Neutral-Atom, Topological (experimental)
Forest [57]	Rigetti	Python/Quil [57]	Superconducting
Qadence [58]	Pasqal	Python	Neutral-Atom
qBraid SDK [59]	qBraid	Python	Superconducting, Trapped-Ion, Neutral-Atom

Computers exploit quantum phenomena like the Quantum Superposition and Quantum Entanglement in order to tackle hard-to-solve computational problems. Current state-of-the-art quantum computing frameworks provide a quite friendly end-user environment and allow users to program Quantum computers, typically in the form of constructing quantum circuits, by using high-level programming languages like Python or C++. These quantum circuits are then transpiled to be made compatible with the topology of a specific quantum device and optimized for execution on present day noisy quantum systems, a process known as routing [48]. Of course, these tasks are almost impossible to be manually performed by users and thus all commercially available frameworks provide automatic tools to handle them. The most popular commercially available QC programming frameworks are summarized in Tab. 2.

On the other side, HPCs comprise a vast number of computational nodes like CPUs, GPUs and FPGAs, which are working in parallel, in order to address many complex scientific challenges, like large-scale simulations and big data analytics etc. [60]. Similarly to QCs, all programs sent for execution to HPCs must be precompiled. Furthermore, when programming large HPCs the concepts of job scheduling, node communication and synchronization must be taken into consideration. Job scheduling refers to the process of resource allocation and job distribution among the available computational nodes, aiming for optimal system performance. Additionally, due to the distributed "nature" of the HPCs, information must be exchanged among system's nodes and all nodes must be perfectly synchronized, under the auspices of a central node, to ensure that all data dependencies are satisfied. The computational overhead added to the total execution time, due to the aforementioned processes, is proportional to the volume of the data shared and the inter-node communication frequency. Current HPC programming frameworks contain components for code compilation and runtime execution monitoring along with a broad range of libraries and interfaces for efficient application development and parallel programming, like OpenMP [61] and MPI [62]. The most dominant high-level languages for HPC programming are C, C++ and Fortran, since the source code written in them is compiled before its execution.

Understandably, any software stack designed to link HPCs and QCs has to comprise all the aforementioned components to ensure a seamless integration between the two systems. A representative figure that illustrates the workflow of such a software stack is presented in Fig. 1.

Current state-of-the-art software stacks imitate one another regarding their abstraction layers. Most of them comprise three discrete layers, namely the 'application', 'compilation' and 'hardware' layer [63]. The latter two are usually merged in a single layer under the name 'Middleware' or 'Orchestration' layer. Its components and role will be analyzed in the following section. The 'application' layer usually supports a user-friendly front-end ecosystem that contains all the appropriate tools and functions for efficient code design (compilers, libraries etc.) [64]. In most cases this front-end is Python-driven and it comes with the appropriate bindings to other high-level languages (e.g. C/C++ or FORTRAN). At

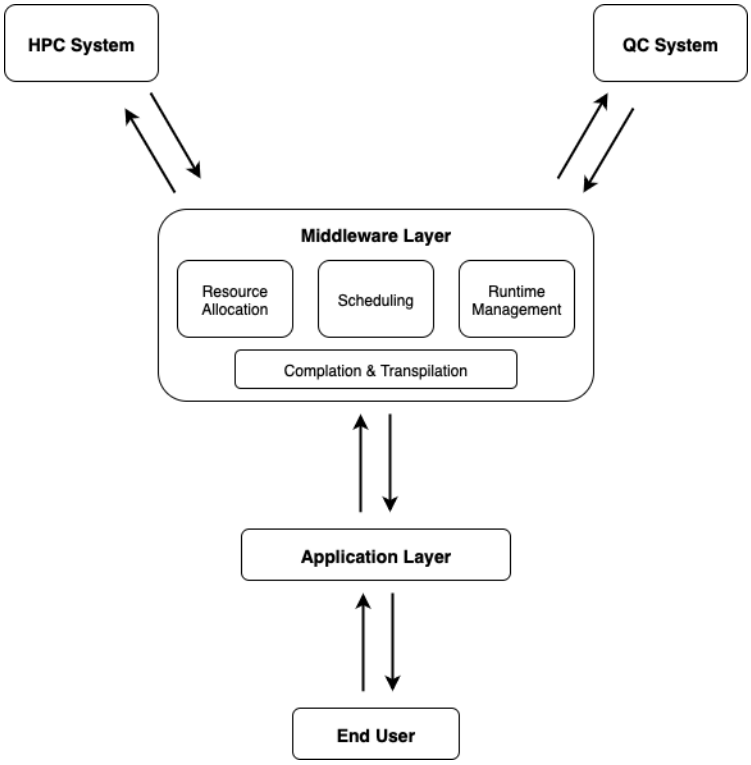

Fig. 1. High-level architecture of a hybrid HPC-QC computing workflow. The application layer interacts with the end user and communicates tasks to the middleware layer, which is responsible for resource allocation, scheduling, compilation, and runtime management across both the high-performance computing (HPC) system and the quantum computing (QC) system. The middleware coordinates the seamless integration of classical and quantum resources to execute hybrid applications.

this level of abstraction, the software must support a plethora of QC programming frameworks [65] and thus must be QC programming-agnostic.

3.2 Middleware/Orchestration Frameworks

The 'Middleware' layer or just 'Middleware' is the most important component in an HPC-QC software stack since it is responsible for the compilation of the hybrid workload, the proper job scheduling, the monitoring of the heterogenous (classical and quantum) tasks and the information interchange between the HPC nodes and the QCs. As mentioned above, the 'Middleware' consists of two main layers, the 'compilation' and the 'hardware' layer. From a higher-level abstraction the 'Middleware' architecture is demonstrated in Fig. 2.

When in the 'compilation' layer the hybrid code undergoes a series of error inspections (lexical, semantic etc.) and then is separated in its classical and quantum counterpart. Then, the classical (i.e. the source code to be executed in HPCs) code is optimized, during the synthesis stage, and combined alongside the appropriate library files to form the binary file. The quantum circuits (i.e. the quantum part) are also optimized and transpiled in terms of the target QC. After that, the binary file(s) and the transpiled quantum circuit(s) are forwarded to the 'hardware' layer to be executed in the corresponding nodes.

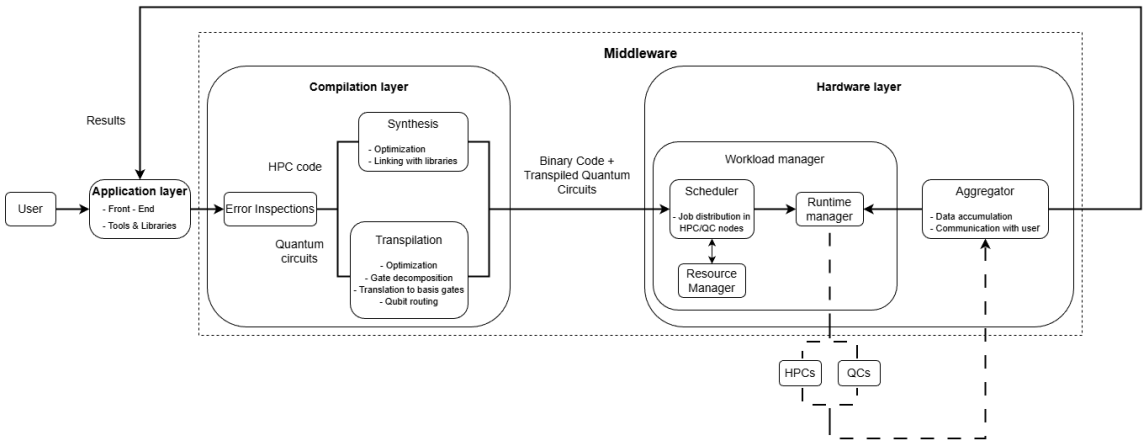

Fig. 2. A higher-level abstraction of the 'Middleware' architecture.

The 'hardware' layer is in charge of the workload execution lifecycle. It consists of two modules, namely the 'workload manager' and the 'aggregator'. The 'workload manager', itself, consists of three submodules, namely the 'scheduler', 'resource manager' and 'runtime manager'. Initially, the combined workload from the 'compilation' layer is transferred to the 'scheduler', which distributes it to the appropriate HPC and QC nodes for execution, via the 'runtime manager'. To achieve that, schedulers are in constant communication with 'resource managers', which are aware for the current status of each node. The process of job scheduling is quite trivial, as many parameters must be taken into account, such as each node's idle time, each process' execution time and possible data dependencies between jobs that may lead to further communication overhead [66], to name a few. The 'runtime manager', as the name suggests, is responsible for the program execution on both classical and quantum hardware and for the inter-node communication and synchronization. Some of the most commonly-used workload managers are SLURM [67] and Agnostiq's Covalent [68]. Last but not least, the 'aggregator' module collects the execution results from both classical and quantum hardware. Then it forwards them to both the 'runtime manager' so that any inter-node data dependencies to be satisfied and all nodes to be synchronized and to the application layer for demonstration purposes.

3.3 Existing Platforms and Frameworks

In this section, we will discuss and refer to both existing QC frameworks that support the appropriate tools for HPC integration and state-of-the-art HPC-QC platforms and frameworks designed in a software-agnostic manner to support multiple QC hardware. The most well-known QC frameworks that support HPC integration tools are the following.

Qiskit [49]: Qiskit is a Python-based open-source framework, designed by IBM. It is one of the most-favorable QC programming and algorithm design frameworks, due to its simplicity and vast number of tools available. It comprises many simulators, both ideal and with noise, which enable users to test their quantum algorithms behavior under NISQ devices. Furthermore, it includes tools which allow users to execute their quantum circuits on real IBM's quantum computers and optionally to improve their results by applying error mitigation and suppression techniques [69–71]. Additionally, it provides some interfaces, like the Qiskit Serverless, for continuous HPC-QC integration and resource management, through cloud-based services in the IBM Quantum Platform. As of April 2025, Qiskit (v2.x) also supports limited C-based QC programming.

PennyLane [52]: PennyLane is a Python-based open-source framework, designed by Xanadu. Like Qiskit, it is one of the most used QC programming frameworks, especially in terms of Quantum Machine Learning, due to its great documentation and plethora of available tools. It supports many simulators for quantum algorithms testing and also includes the necessary functions to support quantum circuit execution to a variety of quantum hardware ranging from superconducting qubits QCs (IBM, Rigetti) to trapped-ion QCs (IonQ, AQT) etc. Xanadu developers have also created an experimental package, under the name Catalyst, that enables just-in-time (JIT) compilation of PennyLane programs [72]. Furthermore, this package is equipped with advanced control flow routines that support both quantum and classical instructions, hence offering the opportunity for HPC-QC integration.

Cirq [51]: Cirq is a Python-based open-source library, designed by Google. It provides numerous abstractions for constructing and optimizing quantum circuits with respect to today's NISQ era quantum hardware and simulators for accurate quantum algorithm validation. It comes with built-in functions which allow users to run their experiments on Google's quantum processors and also with the necessary interface that enables users to execute their quantum circuits in other vendor's (AQT, IonQ, Pasqal) quantum hardware. Furthermore, due to its complete compatibility with TensorFlow Quantum [73], a library designed for building hybrid quantum-classical models with a focus on quantum data, it is feasible to integrate QCs with HPCs by exploiting Cirq.

Braket [54]: AWS Braket is a cloud-based service developed by Amazon. It is fully interconnected with the AWS service, thus helping users create quantum circuits, deploy and execute them in different types of quantum simulators and real hardware provided by IonQ, IQM and Rigetti. Furthermore, this dependency from AWS services provides Braket users the opportunity to develop hybrid quantum-classical algorithms and deploy them into hybrid HPC-QC systems without any struggle. The main reason for that is that AWS comprises a complete tool kit for HPC programming and resource management, like the ParallelCluster and the Parallel Computing Services, that integrate flawlessly to the AWS core.

Azure Quantum [55]: Azure Quantum is a cloud-based service developed by Microsoft. Azure Quantum is part of Microsoft's Azure ecosystem and comprises of a wide spectrum of frameworks and platforms, like the Quantum compute platform. By exploiting these platforms users are able to construct and execute their quantum circuits to simulators and real hardware devices provided by Microsoft [55], by either using Python or Q # as their language of preference. Q # is a high-level, open-source programming language developed by Microsoft for writing quantum programs and it is included in the Microsoft's Quantum Development Kit [55]. Like Amazon Braket, Azure Quantum users have the opportunity to utilize the Azure platform and its services in order to create hybrid HPC-QC workloads and deploy them into state-of-the-art systems.

Rigetti QCS [74]: Rigetti QCS, is a Quantum Cloud-based service provided by Rigetti. It provides the necessary tools for users to construct and execute their quantum circuits on Rigetti's quantum simulators and superconducting quantum hardware, via the Forest SDK [57], by using either Quil or Python. Quil, which stands for Quantum Instruction Language, is a programming language for developing quantum programs, via the Quil SDK, and execute them in Rigetti's QPUs. Additionally, Rigetti's developers have created pyQuil, a python library that allows users to generate Quil programs from quantum gates and classical operations written in Python. Also, it comprises the necessary abstractions, like low-latency access, security and scalability of both classical and quantum resources, in order to offer users a reliable framework for HPC-QC hybrid programming.

Apart from the aforementioned QC frameworks that, nowadays, include the necessary routines for HPC-QC integration, some specifically-designed HPC-QC platforms have been developed also.

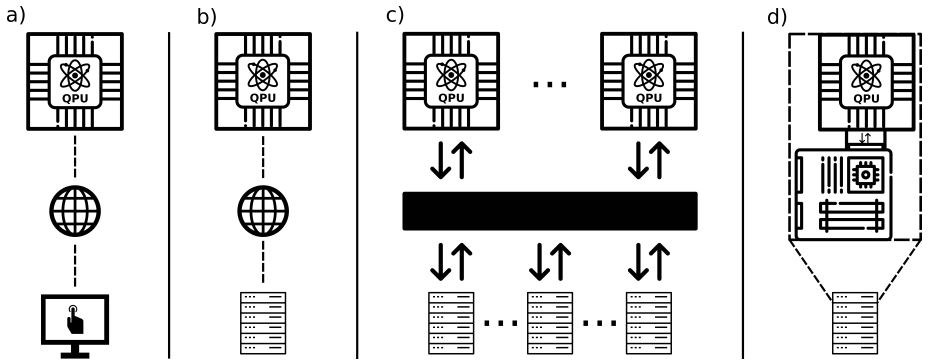

Fig. 3. a) Standalone QPU - Loose integration: User interacts with a single QC through a web interface. b) Co-location - Loose Integration: The HPC interacts with a single QPU. They are two distinct infrastructures which are physically near and communicate through common local network or even apart from each other and communicate via cloud services c) Co-location - Tight Integration: The HPC system interacts with multiple QPUs. They are physically near but still separate hardware infrastructures which communicate via common hardware high-speed interconnects d) On-node - Tight Integration: The QPU is now embedded inside the classical computing node in the form of an external co-processor like a GPU.

XACC [75]: XACC, which stands for eXtreme-scale Accelerator programming framework, is a system-level software framework designed by A.J. McCaskey et. al. This framework is built upon a service-oriented architecture to assist users program, compile and run their hybrid workloads into HPC-QC systems. It is structured in a hardware-agnostic manner in order to provide support for both current and future HPC-QC integrated systems. XACC is implemented as a C++ framework, mainly due to C++ proven efficiency and performance, but is supports programming in other high-level languages, like Python, Julia, etc. through language binding libraries.

CUDA-Q [53]: CUDA-Q is an open-source platform for quantum development, designed by NVIDIA. Alike the aforementioned QC frameworks, CUDA-Q is developed mostly to support hybrid classical-quantum large-scale workloads. The platform's architecture allows computation on both classical devices, like CPUs and GPUs and on quantum hardware resources. Like XACC, it is constructed in a QPU-agnostic manner, hence integrating with all qubit technologies. Additionally, it offers a variety of tools for large-scale systems error-correction and also it supports GPU-accelerated simulations when the corresponding quantum resources are not available.

Pilot-Quantum [76]: Pilot-Quantum is an open-source middleware for monitoring and controlling resources and workloads across HPC-QC systems, designed by P. Mantha et. al. Its implementation is based on the P* model [77], which is a minimal, but complete model for Pilot abstractions [78], introduced by A. Luckow et. al. Pilot-Quantum comprises routines for effective resource and workload management along with scheduling in the application layer. Moreover, it was designed so that it could incorporate seamlessly with most available quantum programming frameworks, like Qiskit, Cirq, Pennylane etc.

4 Hardware Level Integration

4.1 Architectural Approaches

Towards the creation of Hybrid HPC-QC systems, the approach that will be followed on the hardware-level integration of the corresponding components is crucial, as it will affect several aspects of the whole system, its total complexity and

performance, and will also define the requirements and the needed components of the software stack for software-level interfacing.

The integration of HPC and QC systems can be clustered, among others, into three (3) main distinct categories, based on the followed microarchitecture and thus the relative location between the two physical subsystems and their interaction. Those three (3) categories are: 1) Standalone, 2) Co-located, and 3) On-node integration [4, 79].

1) Standalone: This 1^{st} category involves a scenario where the QPU functions independently, without any automated connection to a HPC system, aligning with the principles of **loose integration model**. A traditional computing system is still required to enable user interaction with the QPU, typically through a web-based interface, as illustrated in Fig. 3(a). In such setups, users are responsible for manually preparing their computational tasks and oversee the whole QC workflow. This includes formulating the problem, selecting or designing an appropriate algorithm [80], and translating the quantum algorithm into a high-level quantum-compatible representation, most often a quantum circuit, using high-level programming languages. The quantum circuit may then be optimized and transpiled into hardware-specific instructions for execution on a designated QPU, which returns the output for further processing [49]. This model lacks any automated HPC-QC hybrid workflow support and offers minimal system integration. Nevertheless, it is particularly well-suited for quantum algorithm development, testing, and evaluation, as well as for exploring the properties and performance characteristics of various QPUs and qubit technologies [81].

2) Co-location: The 2^{nd} category describes systems in which HPC and QC hardware are placed close to each other. Co-location typically implies that both resources reside in the same physical environment or communicate over a shared network infrastructure. In some cases, they may also be housed separately but interact through cloud-based connections. This category includes two (2) primary configurations: one where HPC and QC systems are physically co-located, and another where they are network-connected but not necessarily in the same physical location.

The first configuration, co-location via a shared communication network, adheres to the principles of the **loose integration model**. In this setup, the HPC system, utilizing its classical computing nodes, can interact with a single QPU, as illustrated in Fig. 3(b), enabling the execution of hybrid quantum-classical workloads. The general workflow mirrors that of the standalone approach: it begins with preparing the quantum algorithm and ends with post-processing the results. However, a notable advancement in this configuration is the automation of several steps that previously required manual intervention [65]. Additionally, this type of interface supports the execution of hybrid iterative algorithms which rely on frequent data exchanges between classical and quantum components. Despite these enhancements, achieving efficient automated communication demands a more advanced and intricate software stack to coordinate system resources effectively. Given the challenges associated with deploying and maintaining QPUs locally, cloud-based communication is increasingly common. Many QPUs are now provided through Quantum-as-a-Service (QaaS) models [82], allowing remote access rather than local integration. However, this approach must contend with network-induced latency, which can hinder overall system performance and create bottlenecks. Moreover, if communication occurs over public or cloud-based networks, data security becomes a significant concern [83].

The second configuration involves the physical co-existence of resources and aligns with the principles of the **tight model of integration**. In this arrangement, multiple QPUs are physically situated alongside classical computing nodes, as depicted in Fig. 3(c). These QPUs—potentially sourced from different vendors—are interconnected and communicate both with each other and with classical components through specialized, low-latency hardware interfaces. This physical proximity allows for the expansion of available quantum computational capacity and facilitates efficient data exchange, fostering deeper collaboration between classical and quantum processes while significantly minimizing latency. By incorporating a greater number of QPUs, this architecture supports more effective task distribution and the execution

of complex quantum algorithms across multiple units, in a fashion akin to classical parallel computing [84]. However, due to its tightly integrated and heterogeneous nature, this setup demands a more advanced and robust software stack. The system must be capable of managing diverse QPU technologies, coordinating resource scheduling, and ensuring smooth, high-throughput data communication to support seamless operation.

3) On-node Integration: The 3^{rd} category refers to the direct integration of QPUs within HPC nodes themselves, as illustrated in Fig. 3d), which is practically the definition of **tight integration**. This approach mirrors the integration of other hardware accelerators like GPUs or TPUs and follows the principles of tight coupling. It represents the most advanced vision for hybrid HPC-QC architectures, encompassing implementations that range from QPUs directly mounted on motherboards to more sophisticated and futuristic designs utilizing chiplets that scale-up QPUs and combine classical and quantum components into a unified system [47, 85]. In this configuration, QPUs work in close conjunction with classical processors, even as low as at the instruction level, enabling seamless execution of hybrid algorithms where classical and quantum operations interact automatically. This effectively transforms QPUs into powerful, application-specific accelerators optimized for particular problem domains that can leverage their unique capabilities [86]. The physical proximity within this setup allows for real-time quantum-classical computation, significantly reducing latency issues that arise in other models where communication occurs over external networks. Moreover, it inherently enhances data security by eliminating the need for public or cloud-based data transmission. Despite its advantages, on-node integration is the most technically demanding model. It requires highly complex hardware and software frameworks to manage resource coordination and ensure smooth communication. The integration challenge is further intensified by the technological intricacies of qubit fabrication and their dependence on ultra-low temperature operation, especially when QPUs are to be packaged alongside classical components.

A summarized version of those four (4) different setups that lay in the three (3) described categories, can be viewed in Tab. 3.

Table 3. Comparison of HPC–Quantum Integration Architectures

Integration Type	Advantages	Limitations	Typical Interconnects	Use Cases
Standalone Integration (Loose – Standalone)	Simple setup. Easy access through cloud providers. Good for development and education.	Very high latency. Manual workflows. No real-time hybrid execution.	WAN / Internet, Web APIs.	Quantum algorithm prototyping, educational purposes.
Loose Co-Located Integration (Loose – Co-located)	Lower latency vs. Standalone. Local control of QPU. Basic hybrid workflows are possible.	Network latency persists. Requires on-site QC hardware. Separate resource management.	Ethernet, InfiniBand, Internet.	Hybrid VQE/QAOA algorithms, on-premises hybrid experiments.
Tight Co-Located Integration (Tight – Co-located)	Low latency. Supports multi-QPU setups. Unified resource scheduling.	Complex integration. Heterogeneous QPU vendor issues. Advanced orchestration is required.	PCIe Gen 4/5, CXL, InfiniBand, Experimental Quantum Networking.	Quantum chemistry, combinatorial optimization, multi-QPU hybrid HPC.
On-Node Integration (Tight – On-node)	Near-zero latency. Real-time classical-quantum interaction. Enables adaptive hybrid workloads.	Extreme hardware complexity. Cryogenic challenges. Still in research stage, no products yet.	Direct PCIe, CXL, Cryo-CMOS controllers, Chiplet integration (future).	Real-time hybrid HPC-QC, adaptive quantum error correction, next-gen HPC accelerators.

4.2 Physical interconnects between QPUs and HPC systems

Classical Peripheral Interfaces: In many hybrid quantum-classical computing architectures, the quantum component, mainly the QPU and its control unit, is treated as a peripheral accelerator, much like a GPU. Following this approach, the connection between the HPC system and the QPU can be established using conventional high-speed interfaces such as PCI Express (PCIe). A notable example of this is the NVIDIA DGX Quantum system, which utilizes PCIe Gen 5 for QPU integration [87]. Beyond PCIe, another viable option for HPC-QPU interfacing is the Compute Express Link (CXL), which shares the same physical and electrical foundation as PCIe but offers enhanced capabilities. CXL supports high bandwidth and low latency communication while also enabling features like cache coherence and shared memory access, which are particularly beneficial for tightly coupled systems [60]. These interfaces frequently include FPGA-based controllers that play a crucial role in generating the precise control pulses and readout signals needed for quantum operations [88, 89]. FPGAs offer highly configurable, fine-grained control with accurate timing, allowing for reprogrammable and flexible coordination between classical and quantum units. However, to further improve performance, especially in terms of efficiency and integration density—Application-Specific Integrated Circuit (ASIC)-based quantum controllers are also being explored as an alternative to FPGAs [90].

High-Speed Networks (InfiniBand, Ethernet): As discussed earlier, most existing Hybrid HPC-QC systems currently adhere to the loose integration model. In such configurations, quantum computing components are often located remotely and typically housed within separate cryogenic infrastructure, and must communicate with HPC systems over a network. This interaction is commonly facilitated by high-speed networking technologies such as InfiniBand or advanced Ethernet connections. These networks handle the transmission of computational tasks and results between the HPC host and the QPU's control unit [60]. InfiniBand, in particular, is widely used due to its support for Remote Direct Memory Access (RDMA), offering exceptionally low latency and high throughput [91]. This network-based approach is especially relevant in setups where quantum and classical components are not physically co-located but are instead interconnected through a local high-speed network.

Cryogenic Wiring and Microwave Links: Qubits can be fabricated using a variety of technologies. However, due to their extreme sensitivity to noise, most qubit implementations require ultra-low temperature environments, typically in the millikelvin range. For this reason, both the qubits and their associated control electronics are usually housed within cryostat. To accommodate this, various cryo-CMOS controller architectures have been proposed and studied, specifically designed to operate in cryogenic conditions near the qubits. These controllers are capable of generating control signals, amplifying signals when needed, especially during readout, and performing initial layers of local signal processing. A typical qubit readout employs a cryogenic low-noise amplifier, operating at very low temperatures, for signal boosting, apart from other isolating and filtering layers. Their proximity to the qubits helps streamline the physical interconnects, reduce latency and noise, enhance overall performance, and support the scalability of quantum systems [92–95].

To bridge the gap between room-temperature HPC electronics and cryogenically cooled qubits and their control units, two (2) primary types of physical interconnects are commonly employed: coaxial cables and waveguides. Coaxial cables are flexible and capable of transmitting both DC bias signals and high-frequency microwave pulses. In contrast, waveguides are rigid structures engineered specifically for the transmission of microwave signals at very high frequencies, offering advantages such as lower signal loss and reduced thermal conduction. These interconnects are essential for delivering microwave pulses, applying DC biases, and retrieving readout signals. Functionally, they form the physical layer connecting PCIe or FPGA-based controllers within the HPC system to the QPU. Typically, the control of a single

qubit requires at least two dedicated input lines, along with additional input/output lines for readout operations [96]. As quantum processors scale up, the number of required physical lines increases substantially, creating a significant bottleneck. To address this challenge, a variety of solutions have been proposed. These include the deployment of cryogenic RF switches and crossbar interconnect architectures [97–99], which aim to reduce the number of required connections. Additionally, frequency multiplexing is being explored to boost channel density—allowing multiple signals to share the same transmission medium. Promising developments include cryogenic electronic multiplexers [100] and photonic fiber links enhanced by the incorporation of wavelength division multiplexing (WDM) [101], which, by using particular optical modulation schemes to encode various signals, may be able to combine numerous coaxial cables into a single optical fiber [102].

Quantum Networking: As discussed in Section 4.1, hybrid quantum-classical systems can incorporate multiple QPUs, potentially from different vendors. To enable tighter integration and harness the combined computational power of these QPUs, especially for solving more complex problems that demand a larger number of qubits, direct communication between QPUs becomes essential. This form of inter-QPU communication, occurring without intervention by classical HPC components, is known as Quantum Networking. Quantum networking facilitates the transfer of quantum states between QPUs through entangled links, and the type of interconnection used depends on both the integration approach and the underlying qubit technology [103]. Several promising technologies are under development to support this. For instance, photonic-based links using waveguides or optical fibers are suitable for transmitting quantum states across long distances [104]. In scenarios where chips are physically close, inter-chip interconnects are used to directly link qubits across different chips. These can take the form of coaxial cables, superconducting transmission lines, or capacitive couplings via resonators, depending on the specific qubit implementation [103]. Quantum teleportation offers a more advanced method, using entangled photon pairs to transfer quantum information remotely between qubits without physical transmission of the particles themselves [105]. Another approach, ion shuttling, is used in ion-trap systems and involves the physical movement of ions within the cryogenic setup to bring qubits into close enough proximity for interaction [106]. For the realization of quantum networks, the development of quantum transducers is critical. This technology allows for coherent conversion between different quantum information carriers, enabling cross-platform entanglement and integrating diverse QPUs into a unified quantum-classical computing environment [107]. These interconnection techniques are still in the experimental phase, but they lay the foundation for the next generation of quantum computing—where information will be exchanged through qubits rather than classical bits. This will enable distributed or parallel quantum computation and potentially lead to the formation of quantum local area networks (QLANs), where multiple entangled QPUs are orchestrated by HPC infrastructure [108, 109].

The use of those physical interconnects in HPC-QC integrated systems of different architectures, can be viewed in Tab. 3.

5 Applications and Use Cases of HPC-QC interconnected systems

While there is a rapid development of quantum computers technology, as highlighted in the Introduction section, we are currently in the so called NISQ era [110]. Despite the fact that current hardware reached a level of handling problems that require beyond 100 qubits, their mapping yields deep circuits that due to noise lead to state decomposition and loss of information. On the other hand, classical HPC systems face significant challenges when tackling NP-hard and NP-complete problems, while it has been showcased that quantum computers can offer a significant advantage in solving such problems [111]. As the size of these problems increases, the time required to solve them scales exponentially, demanding substantial computational resources. This rapid growth in execution time often renders solutions impractical

for larger input dimensions. NP-hard and NP-complete problems are not just theoretical curiosities—they frequently arise in both research and industrial contexts, making their efficient resolution a critical concern across many domains [112]. To solve these kind of problems, hybrid quantum-classical approaches emerged [112, 113] with the more prominent being the Variational Quantum Eigensolver (VQE) [113, 114] and the Quantum Approximate Optimization Algorithm (QAOA) [115]. The core idea behind these types of algorithms is to let quantum computers handle the tasks that are hard to solve on classical computers, and use classical ones for tasks like parameters adjustment, evaluation and pre/post processing, to name a few. In the following sections, we consider areas that HPC-QC interconnected systems present prominent applications.

5.1 Optimization

Finding the global minimum or maximum of an objective function describing a problem, is a major issue of traditional optimization techniques like Bayesian optimization [116, 117] and genetic algorithms [118]. Classical approaches frequently getting trapped in local minima, especially when the problem size increases. As stated previously, quantum computers with the emergnce of variational algorithms, QAOA and quantum annealing [119, 120], promise a compelling potential in solving complex combinatorial optimization problems. Quantum algorithms can accelerate the optimization process in large and complex areas like scheduling [121], logistics [122], transportation [123] and finance [124, 125] to name a few. Typically, quantum optimization protocols, utilize the hybrid quantum-classical interconnection. The problem is expressed as an Ising Hamiltonian [126], the quantum algorithm prepares and evolves the states that corresponds to a solution, while the classical part is related to the estimation of the expected value and the parameter update.

5.2 Machine Learning

Quantum Machine Learning (QML) integrates quantum computing with Machine Learning (ML) to enhance various aspects of ML workflows. QML applications are often grouped by the type of input data, classical or quantum, and by the learning paradigm, i.e., supervised [127, 128], unsupervised [129], and reinforcement learning [130, 131]. In the current NISQ era, workflows where classical data are encoded into quantum circuits, utilizing various feature encoding techniques, gain increasing attention. By increasing the datasets scale these models can be highly benefited by the HPC integration, where data are distributed across several nodes. This hybrid integration can be further extended from the sole purpose of data handling and can be incorporated to the models themselves. Several approaches have been proposed, where a mixture of classical and quantum components synergize to create a seamless workflow [132–138]. Starting from the Multi-Layer Perceptron (MLP) quantum circuits can be integrated into the learning and prediction process by replacing whole layers of the model, or even specific neurons [139, 140]. Generative Adversarial Networks (GANs) [141, 142] utilize a generator and a discriminator to create synthetic data that resemble the original ones, in cases where there is a shortage of data. Both of the generator and discriminator components can be replaced by Parameterized Quantum Circuits (PQCs), individually, both of them and as it is the case of MLPs, some parts of them. Another straightforward application for the hybrid approach is the reservoir computing [143–145] paradigm. The straightforward approach in this case is that the reservoir output layer to be enhanced by quantum components. Yet, the greater advantage comes by replacing the reservoir itself with a quantum circuit that follows the systems properties and connectivity and thus correspond to a natural reservoir architecture [146]. Generally, for many ML workflows and models the hybrid architecture not only by replacing/enhancing classical components with quantum circuits but also by taking advantage of the effectiveness of quantum computers in tasks evolving linear algebra [147].

5.3 Many-Body dynamics

Feynman originally envisioned general-purpose quantum computers as powerful simulators for natural systems that are inherently quantum. With the advent of algorithms VQE and Quantum Monte Carlo adapted for quantum computers [148–150], hybrid HPC-QC workflows have emerged, leveraging the quantum devices' efficiency in solving many-body problems alongside the scalability of classical high-performance computing. These workflows are particularly promising in quantum chemistry, where classical methods are nearing their computational limits. Even in the current NISQ era, growing research demonstrates the potential value quantum computers offer for challenging tasks such as electronic structure determination and ground-state energy calculations [151–157]. Robledo-Moreno et al. recently showcased that QCs, supported by large-scale classical resources, can achieve approximate but useful results in quantum chemistry problems that are beyond the reach of classical exact methods or standalone quantum processors [158]. Drug discovery is another area where classical computing limitations appear. Again, the HPC-QC integration can offer advantages in crucial steps of the workflow such as the protein folding problem [159] and molecular docking [160]. Thus, the goal of such an HPC-QC framework is to enable the integration of quantum algorithms into simulations, where a particular part of the simulation can be offloaded to the quantum computer.

6 Challenges and Limitations

Even though highly promising and under intensive investigation, the efficient and highly performant integration and operation of HPC-QC systems is affected by a significant and diverse challenges that need to be overcome. Those challenges are related with all the layers of HPC-QC system integration (application layer, middleware layer, hardware layer) and include among others, hardware maturity, data communication restrictions, software development, resource management, security, as well as cost and accessibility issues.

6.1 Quantum Hardware Maturity

Quantum hardware is still continuously progressing and several different technologies are being tested and evaluated, targeting on providing more and of higher quality qubits. However, the existing quantum hardware technology in its whole, is still in the NISQ era, is accompanied with considerable limitations.

A distinctive challenges that is present in current hardware technology is qubit quality. For the time being, the provided qubits are extremely sensitive to external pertrubations from the environment, which causes their quantum states to decay overtime. This is the well-known problem of decoherence. The existing quantum devices of the NISQ era suffer from low coherence times, reducing the available duration of the quantum computation to the order of a few hundreds of microseconds and thus limiting the depth of the quantum circuits that can be executed [161].

Additionally, a significant source of noise, and thus errors is the application of quantum gates. Every quantum operation (gate) executed by a qubit introduces a small amount of noise which is accumulated and propagated through the quantum circuits. This is mainly attributed to the existing gate fidelities, which can ultimately lead to a cascade of errors [162]. The readout process and circuitry can also be responsible for the introduction of errors. Even if the quantum computation itself was performed within the required range of accuracy, the measurement process itself can lead to a wrong output [163].

As the qubit density increases, and qubits are naturally getting closer with one another, the crosstalk phenomenon becomes evident. This phenomenon introduces correlated errors, leading to corrupted quantum states that significantly affect the integrity of the quantum computation [162].

Additionally, the scalability of the QPUs which mainly refers to the increase of the qubit count in a single quantum chip, is a very common and significant problem related to quantum hardware technology. At the current technology level, QPUs may include up to a few tens of hundreds of physical qubits, a number which is slowly but constantly increasing. The increase of this number is challenging across all qubit technologies, and with such small scale of QPUs, the current capabilities of solving large scale problems are restricted [164]. A significant limiting factor on the scalability of QPUs is the wiring complexity. Connecting control and readout lines from room-temperature classical electronics to each qubit inside a cryostat presents a significant wiring challenge that scales unfavorably with the number of qubits [101].

Another significant problem related to quantum hardware is the connectivity. The intercommunication of qubits with one another is many times restricted, as each qubit can only interact with a small part of its neighbours. This can be problematic for quantum algorithms where the physical correlation is required between qubits that are distant. The use of SWAP gates, can provide a solution to this problem by bringing qubit states physically closer, however they add significant overhead which ultimately degrades the overall performance. For this reason, optimization is investigated towards the development of compilation techniques that minimize SWAP overhead [165].

6.2 Data Management and Communication

Apart from the quality and the scalability of the hardware itself, the communication between the QPU and HPC parts, plays a significant role on the overall performance and thus the significance and usability of hybrid systems, and has its own challenges and bottlenecks.

As mentioned above, in Section 4, the latency and bandwidth can be significant problems in the communication between the HPC and QC systems. It is a crucial topic, taking under consideration the large volumes of data that needs to be exchanged between the two components, as well as the pre- and post- processing that is demanded, every time data is transferred from one subsystem to the other. A significant amount of latency especially in hybrid algorithms that contain feedback loops, can possibly negate all the speedup gains provided by the QC. In such cases, the data exchange between a CPU and a QPU can become an intensive and time-consuming task, and its manipulation with efficient I/O strategies and techniques targeting to the minimization of overhead is a challenging but crucial step towards high-performance HPC-QC hybrid systems [166].

6.3 Ecosystem Maturity

From a software point of view, there are still significant problems to be solved and challenges to be addressed, as the ecosystem for HPC-QC hybrid systems is far less mature compared to that of classical HPC systems.

At the current state, and in contrast with the existing HPC solutions, the software stack related to quantum computers and quantum programming is far less developed and established. This is reflected on the exploited programming tools, as quantum programming often utilizes programming languages that target ease of use and flexibility, such as Python or Julia, incorporating libraries and frameworks specific for Quantum Computing, while the majority of HPC applications employ programming languages like C and C++, or programming models and frameworks such as OpenMP and MPI, which target mainly on performance. Bridging this gap requires significant effort in developing robust, performance-oriented quantum software tools [64]. A significant problem that also affects the QC software stack and toolchain is the absence of standardization, which spans from as low as the hardware communication protocols, up to the software APIs and intermediate representations. This creates a fragmented ecosystem where each user provides its own heuristic solution, with no universal compatibility across multiple systems of different technologies, adding increased overhead

and significantly reducing the re-usability of various software components. Some initial steps towards standardization have already been attempted through the realization of Quantum intermediate representations which are technology agnostic [167]. At the higher level of the toolchain, another significant problem is the lack of a standardized and automatic methodology for mapping and translating classical problems into quantum formats. Breaking down a problem into parts, identifying which of those parts can benefit from quantum processing and how to map them onto quantum algorithms, as well as selecting which is the best quantum algorithm for a specific case, remains a highly challenging and active area of research [168].

6.4 Resource Management and Scheduling

In the area of middleware, the resource management and job scheduling in hybrid systems that contain not only heterogeneous computational resources, but go even beyond conventional by including also multiple QPUs from different vendors, are complex and challenging tasks.

In their current form, QPUs provided by vendors are being accessed through web APIs with proprietary queuing systems, which makes it really challenging to be integrated with the existing sophisticated schedulers used by the conventional HPC systems, such as SLURM. Thus, the need for development of specific interfaces or plugins for existing HPC schedulers that will allow the seamless integration and coordination of diverse QPU resources with classical resources is evident. In such efforts, another significant problem that needs to be taken under consideration and requires specific handling, is the possible workload imbalance which is introduced with the use of QPUs. Different QPU technologies can have very different operation times, which makes it hard for traditional HPC schedulers to operate effectively, as they often assume more predictable job durations. On the other hand, simple co-scheduling strategies, like the exclusive allocation of a QPU to an HPC job for its whole duration, can be proven extremely inefficient by leading to underutilization of either the classical or the quantum node [169, 170]. Thus, the development of scheduling policies which are able to cover dependencies between heterogeneous HPC and QC resources and also target optimization for overall performance and ensure fairness among users is an active and challenging area of research

6.5 Cost and Accessibility

Finally, among others, a major challenge that is simultaneously a significant obstacle in the broad development and adaptation of QCs, and consequently to actual realization of high-performance hybrid HPC-QC systems, is the increased financial investment required for both owning and accessing them.

Both HPC and QC components are made of high-cost technologies. Especially QC components are inherently expensive technologies, as they are relatively novel, require complex fabrication processes, expensive specialized hardware and demand environmental controls in order to operate, such as cryogenics for a significant amount of QPU types. All this, increases the cost of operation and also the cost of maintenance. For this reason cloud computing is currently dominating the area of QCs and also HPC-QC systems, where access is provided through cloud platforms by companies, significantly lowering access costs and making those technologies available to a broad audience [83].

However, apart from the use of physical QPUs in QC or hybrid HPC-QC systems, the simulation of such workflows and quantum tasks can also be considered expensive. Simulating quantum computations on classical computing systems can be extremely resource intensive even for small scale problems, a fact that significantly slows the development and investigation of new quantum algorithms [171]. The area of simulation of quantum operations with classical computing systems is a challenging area of research where also application specific hardware that emulates quantum operations is explored and tested [172].

7 Discussion and Conclusions

The integration of High-Performance Computing (HPC) with Quantum Computing (QC) represents a transformative shift in computational systems, combining the strengths of classical processing power with quantum-related advantages. Throughout this manuscript, several aspects of HPC-QC interfacing have been analyzed, including system architecture models, the evolving software and middleware ecosystem, hardware integration strategies, and representative application domains. These hybrid systems combine and leverage the complementary strengths of HPC and QC, establishing a framework for solving problems that neither modality could solve efficiently on its own.

Architectural approaches on HPC-QC integration span from loosely to tightly coupled designs. At one end, the loosely-coupled configurations treat QPUs as external systems, accelerators accessed through high-speed networks or cloud services, offering simplicity in implementation, at the cost of increased communication latency. On the other hand, co-located models bring quantum and classical resources in closer physical proximity, to reduce latency and improve throughput. At the extreme case, on-node integration envisions the incorporation of a QPU directly into an HPC node, enabling near-zero latency and real-time quantum-classical interactions. Each model entails its own distinct trade-offs in terms of latency and complexity: While loosely integrated systems are currently the most practical, the on-node integration paradigm is viewed as the long-term ideal in terms of performance, despite the existing engineering challenges.

In terms of software, the ecosystem that supports hybrid HPC-QC systems has matured significantly in recent years. High-level programming frameworks are constantly evolving and have already expanded the toolkit for developing hybrid algorithms, providing to the developers easily accessible, tunable, and standardized interfaces. Simultaneously, several software platforms, such as NVIDIA's CUDA Quantum and XACC frameworks have emerged, working towards the effective unification of classical and quantum computing workflows, allowing quantum accelerators to be included within traditional HPC applications. Adding to that, at the lower, middleware layer, dedicated solutions have already appeared and are constantly developing, coping with the significant challenges of resource management orchestration, job scheduling, and task execution across both HPC and QC resources. Those middleware tools showcase the significance of solid combined workload management in achieving a seamless integration. Together, the advances in frameworks and middleware are steering the community toward a more unified software stack for hybrid HPC-QC computing, although further improvements in interoperability, developer tools, and programming abstractions are still needed to reach the level of maturity seen in established classical HPC software.

Nonetheless, significant technical and operational challenges remain before HPC-QC systems can reach their full potential. Existing quantum computing hardware is still at the NISQ era, characterized by limited qubit counts, short coherence times, high error rates, and restricted qubit connectivity. These limitations significantly reduce the size and complexity of quantum subroutines that can be reliably executed on current hardware, and showcase the need for more efficient error correction algorithms and less noise-prone future quantum devices. The software ecosystem is also still at an early stage of development, far behind the maturity level of that of conventional HPC systems. Even though constantly evolving, it still lacks universal standards and full interoperability between platforms. From an operational point of view, integrating quantum processors into HPC infrastructures also presents non-trivial challenges. Issues such as the need for cryogenic environments and specialized control electronics for many QPU technologies, the complexity of co-scheduling quantum jobs alongside classical tasks, and the increased cost and expertise required to maintain quantum hardware, all pose practical obstacles. These challenges emphasize the necessity for continued

research, engineering innovation, and investment in both hardware and software to overcome current bottlenecks and ensure that hybrid HPC-QC systems can be scaled up and used reliably.

Looking forward, ongoing and future developments promise to further tighten the integration between HPC and quantum platforms. On the hardware side and mainly on the part of peripheral circuits and systems that are responsible for connecting and controlling QC hardware, new technologies are constantly being explored to enable tighter interconnection with lower latency in hybrid systems. In addition, quantum networking technologies are expected to play a significant role in the longer term by allowing multiple QPUs to be linked into a unified computational fabric, extending the reach of hybrid architectures beyond a single QPU. Simultaneously, the software and middleware layers are about to evolve, following the well-established paradigms of conventional supercomputing. In addition, more sophisticated job schedulers and workload managers tailored for quantum-enhanced HPC clusters, improvements in real-time execution of hybrid quantum-classical workflows, and standardized application programming interfaces (APIs) are expected to emerge in the near future. These developments will improve the portability, usability, and efficiency of HPC-QC systems, making them more accessible. Moreover, as quantum device fabrication and gate fidelity improve, leading to lower hardware error rates, QC technology is expected to migrate from the NISQ, to the FTQC (Fault-Tolerant Quantum Computing) era in the forthcoming years. This change toward fault-tolerant QPUs, can ultimately lead to large-scale quantum computations, enabling far deeper quantum circuits and more complex algorithms to be executed reliably as part of HPC workflows.

Hybrid HPC-QC systems are set to become necessary tools for a range of high-impact applications as they evolve. Practically, combining HPC with quantum accelerators holds great promise for solving large-scale optimization problems (e.g. in logistics or portfolio optimization) and enhancing machine learning tasks, as well as for enabling more accurate simulations of many-body quantum systems in chemistry and materials science that exceed the practical bounds of conventional computing. Early demonstrations in these domains have already shown that quantum co-processors, when used together with classical HPC resources, can possibly provide computational speedups or increased result accuracy for certain problem categories. As hybrid architectures mature, we can expect their impact to broaden across scientific research and industry, offering new capabilities for tackling challenges that were previously considered intractable. Ultimately, the bridging of high-performance classical computing with increasingly powerful quantum computing is likely to redefine the boundaries of computational capability. By harnessing the unique strengths of both paradigms in a unified system, future HPC-QC platforms promise to push far beyond today's computational frontiers.

Acknowledgments

This research has been supported by the project "A catalyst for EuropeaN ClOUd Services in the era of data spaces, high-performance and edge computing (NOUS)", Grant Agreement Number 101135927. Funded by the European Union's HORIZON-CL4-2023-DATA-01 call, views and opinions expressed are, however, those of the authors only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

References

[1] Rhonda Au-Yeung, Nicholas Chancellor, and Pascal Halffmann. Np-hard but no longer hard to solve? using quantum computing to tackle optimization problems. *Frontiers in Quantum Science and Technology*, 2:1128576, 2023.

[2] Jaime Alvarado-Valiente, Javier Romero-Álvarez, Enrique Moguel, José García-Alonso, and Juan M Murillo. Technological diversity of quantum computing providers: a comparative study and a proposal for api gateway integration. *Software Quality Journal*, 32(1):53–73, 2024.

[3] Chenghong Zhu, Xian Wu, Zhaohui Yang, Jingbo Wang, Anbang Wu, Shenggen Zheng, and Xin Wang. Quantum compiler design for qubit mapping and routing: A cross-architectural survey of superconducting, trapped-ion, and neutral atom systems. *arXiv preprint arXiv:2505.16891*, 2025.

[4] Amr Elsharkawy, Xiao-Ting Michelle To, Philipp Seitz, Yanbin Chen, Yannick Stade, Manuel Geiger, Qunsheng Huang, Xiaorang Guo, Muhammad Arslan Ansari, Christian B Mendl, et al. Integration of quantum accelerators with high performance computing-a review of quantum programming tools. *ACM Transactions on Quantum Computing*, 2023.

[5] Dejan Milojicic, Paolo Faraboschi, Nicolas Dube, and Duncan Roweth. Future of hpc: Diversifying heterogeneity. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 276–281. IEEE, 2021.

[6] Antonio Macia-Lillo, Higinio Mora, Antonio Jimeno-Morenilla, and Tamai Ramirez. Towards abstraction of heterogeneous accelerators for hpc/ai tasks in the cloud. In *2024 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 151–159. IEEE, 2024.

[7] Vikram Rajan, Niwaran Chandra Kumar, Nihar Ranjan, and Amgothu Murali Krishna. Design challenges in hpc for ai/ml applications. In *2024 IEEE 17th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*, pages 11–15. IEEE, 2024.

[8] Torsten Hoefler, Marcin Copik, Pete Beckman, Andrew Jones, Ian Foster, Manish Parashar, Daniel Reed, Matthias Troyer, Thomas Schulthess, Daniel Ernst, et al. Xaas: Acceleration as a service to enable productive high-performance cloud computing. *Computing in Science & Engineering*, 26(3):40–51, 2024.

[9] Fei Yin and Feng Shi. A comparative survey of big data computing and hpc: From a parallel programming model to a cluster architecture. *International Journal of Parallel Programming*, 50(1):27–64, 2022.

[10] Imane Ettifouri, Mostapha Zbakh, and Claude Tadonki. The need for hpc in ai solutions. In *International Conference of Cloud Computing Technologies and Applications*, pages 137–159. Springer, 2024.

[11] Flavio CC Galeazzo, Marta Garcia-Gasulla, Elisabetta Boella, Josep Pocurull, Sergey Lesnik, Henrik Rusche, Simone Bnà, Matteo Cerminara, Federico Brogi, Filippo Marchetti, et al. Performance comparison of cfd microbenchmarks on diverse hpc architectures. *Computers*, 13(5):115, 2024.

[12] Tosiyuki Nakaegawa. High-performance computing in meteorology under a context of an era of graphical processing units. *Computers*, 11(7):114, 2022.

[13] Valeria Gribova and Dmitry Kharitonov. Information and computing ecosystem's architecture for monitoring and forecasting natural disasters. *Computers*, 13(12):334, 2024.

[14] Xiao-Yang Liu, Jie Zhang, Guoxuan Wang, Weiqing Tong, and Anwar Walid. Fingpt-hpc: Efficient pretraining and finetuning large language models for financial applications with high-performance computing. *arXiv preprint arXiv:2402.13533*, 2024.

[15] Chenchen Song. Design and application of financial market option pricing system based on high-performance computing and deep reinforcement learning. *Scientific Programming*, 2022(1):8525361, 2022.

[16] Branimir Kolarek, Ljubo Gamulin, and Davor Davidović. Cultural heritage on hpc-creating high resolution 3d models using photogrammetry. In *2024 47th MIPRO ICT and Electronics Convention (MIPRO)*, pages 1121–1127. IEEE, 2024.

[17] Roman Rietsche, Christian Dremel, Samuel Bosch, Léa Steinacker, Miriam Meckel, and Jan-Marco Leimeister. Quantum computing. *Electronic Markets*, 32(4):2525–2536, 2022.

[18] Sergey Bravyi, Oliver Dial, Jay M Gambetta, Darío Gil, and Zaira Nazario. The future of quantum computing with superconducting qubits. *Journal of Applied Physics*, 132(16), 2022.

[19] Kenneth R Brown, John Chiaverini, Jeremy M Sage, and Hartmut Häffner. Materials challenges for trapped-ion quantum computers. *Nature Reviews Materials*, 6(10):892–905, 2021.

[20] Pieter Kok, William J Munro, Kae Nemoto, Timothy C Ralph, Jonathan P Dowling, and Gerard J Milburn. Linear optical quantum computing with photonic qubits. *Reviews of modern physics*, 79(1):135–174, 2007.

[21] Sam R Cohen and Jeff D Thompson. Quantum computing with circular rydberg atoms. *PRX Quantum*, 2(3):030322, 2021.

[22] Guido Burkard, Thaddeus D Ladd, Andrew Pan, John M Nichol, and Jason R Petta. Semiconductor spin qubits. *Reviews of Modern Physics*, 95(2): 025003, 2023.

[23] Pei-Hua Wang, Jen-Hao Chen, Yu-Yuan Yang, Chien Lee, and Yufeng Jane Tseng. Recent advances in quantum computing for drug discovery and development. *IEEE Nanotechnology Magazine*, 17(2):26–30, 2023.

[24] Georgios D Varsamis and Ioannis G Karafyllidis. A quantum walks assisted algorithm for peptide and protein folding prediction. *Biosystems*, 223: 104822, 2023.

[25] Ioannis Liliopoulos, Georgios D Varsamis, Theodora Karamanidou, Christos Papalitsas, Grigorios Koulouras, Vassilios Pantazopoulos, Thanos G Stavropoulos, and Ioannis G Karafyllidis. Quantum algorithm for protein-ligand docking sites identification in the interaction space. *Journal of Computer-Aided Molecular Design*, 39(1):1–15, 2025.

[26] Georgios D Varsamis, Ioannis G Karafyllidis, KM Gilkes, U Arranz, R Martin-Cuevas, G Calleja, J Wong, HC Jessen, Panagiotis Dimitrakis, P Kolovos, et al. Quantum algorithm for de novo dna sequence assembly based on quantum walks on graphs. *BioSystems*, 233:105037, 2023.

[27] Georgios D Varsamis, Ioannis G Karafyllidis, KM Gilkes, U Arranz, R Martin-Cuevas, G Calleja, Panagiotis Dimitrakis, Petros Kolovos, Raphael Sandaltzopoulos, HC Jessen, et al. Quantum gate algorithm for reference-guided dna sequence alignment. *Computational biology and chemistry*, 107:107959, 2023.

[28] Jonas Blenninger, David Bucher, Giorgio Cortiana, Kumar Ghosh, Naeimeh Mohseni, Jonas Nüßlein, Corey O'Meara, Daniel Porawski, and Benedikt Wimmer. Q-grid: Quantum optimization for the future energy grid. *KI-Künstliche Intelligenz*, pages 1–11, 2024.

[29] CH Ugwuishiwu, UE Orji, CI Ugwu, and CN Asogwa. An overview of quantum cryptography and shor's algorithm. *Int. J. Adv. Trends Comput. Sci. Eng*, 9(5), 2020.

[30] Andrew J Daley, Immanuel Bloch, Christian Kokail, Stuart Flannigan, Natalie Pearson, Matthias Troyer, and Peter Zoller. Practical quantum advantage in quantum simulation. *Nature*, 607(7920):667–676, 2022.

[31] Francisco Chicano, Gabiel Luque, Zakaria Abdelmoiz Dahi, and Rodrigo Gil-Merino. Combinatorial optimization with quantum computers. *Engineering Optimization*, pages 1–26, 2025.

[32] Daowen Qiu, Le Luo, and Ligang Xiao. Distributed grover's algorithm. *Theoretical Computer Science*, 993:114461, 2024.

[33] Sukhpal Singh Gill, Adarsh Kumar, Harvinder Singh, Manmeet Singh, Kamalpreet Kaur, Muhammad Usman, and Rajkumar Buyya. Quantum computing: A taxonomy, systematic review and future directions. *Software: Practice and Experience*, 52(1):66–114, 2022.

[34] Travis S Humble and Keith A Britt. Software systems for high-performance quantum computing. In *2016 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–8. IEEE, 2016.

[35] Mahdi Chehimi and Walid Saad. Physics-informed quantum communication networks: A vision toward the quantum internet. *IEEE network*, 36(5): 32–38, 2022.

[36] Matthias Möller and Cornelis Vuik. On the impact of quantum computing technology on future developments in high-performance scientific computing. *Ethics and information technology*, 19:253–269, 2017.

[37] Keith A Britt and Travis S Humble. High-performance computing with quantum processing units. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 13(3):1–13, 2017.

[38] Haryono Soeparno and Anzaludin Samsinga Perbangsa. Cloud quantum computing concept and development: A systematic literature review. *Procedia Computer Science*, 179:944–954, 2021.

[39] Martin Schulz, Martin Ruefenacht, Dieter Kranzlmüller, and Laura Brandon Schulz. Accelerating hpc with quantum computing: It is a software challenge too. *Computing in Science & Engineering*, 24(4):60–64, 2022. doi: 10.1109/MCSE.2022.3221845.

[40] Alexander J McCaskey, Eugene F Dumitrescu, Dmitry Liakh, Mengsu Chen, Wu-chun Feng, and Travis S Humble. A language and hardware independent approach to quantum–classical computing. *SoftwareX*, 7:245–254, 2018.

[41] Philipp Seitz, Amr Elsharkawy, Xiao-Ting Michelle To, and Martin Schulz. Toward a unified hybrid hpcqc toolchain. In *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, volume 2, pages 96–102. IEEE, 2023.

[42] Gokul Subramanian Ravi, Kaitlin N Smith, Pranav Gokhale, and Frederic T Chong. Quantum computing in the cloud: Analyzing job and machine characteristics. In *2021 IEEE International Symposium on Workload Characterization (IISWC)*, pages 39–50. IEEE, 2021.

[43] Daniel Reed, Dennis Gannon, and Jack Dongarra. Reinventing high performance computing: challenges and opportunities. *arXiv preprint arXiv:2203.02544*, 2022.

[44] Timothy Proctor, Kevin Young, Andrew D Baczewski, and Robin Blume-Kohout. Benchmarking quantum computers. *Nature Reviews Physics*, pages 1–14, 2025.

[45] Evan T Hockings, Andrew C Doherty, and Robin Harper. Scalable noise characterization of syndrome-extraction circuits with averaged circuit eigenvalue sampling. *PRX quantum*, 6(1):010334, 2025.

[46] Erik Gustafson. Noise improvements in quantum simulations of sqed using qutrits. *arXiv preprint arXiv:2201.04546*, 2022.

[47] Martin Ruefenacht, Bruno G Taketani, PASI Lähteenmäki, VILLE Bergholm, DIETER Kranzlmüller, LAURA Schulz, and MARTIN Schulz. Bringing quantum acceleration to supercomputers. *IQM/LRZ Technical Report, https://www. quantu m. lrz. de/fileadmin/QIC/Downloads/IQM HPC-QC-Integration-White paper. pdf*, 2022.

[48] Alexander Cowtan, Silas Dilkes, Ross Duncan, Alexandre Krajenbrink, Will Simmons, and Seyon Sivarajah. On the qubit routing problem. *arXiv preprint arXiv:1902.08091*, 2019.

[49] Ali Javadi-Abhari, Matthew Treinish, Kevin Krsulich, Christopher J Wood, Jake Lishman, Julien Gacon, Simon Martiel, Paul D Nation, Lev S Bishop, Andrew W Cross, et al. Quantum computing with qiskit. *arXiv preprint arXiv:2405.08810*, 2024.

[50] Tomer Goldfriend, Israel Reichental, Amir Naveh, Lior Gazit, Nadav Yoran, Ravid Alon, Shmuel Ur, Shahak Lahav, Eyal Cornfeld, Avi Elazari, et al. Design and synthesis of scalable quantum programs. *arXiv preprint arXiv:2412.07372*, 2024.

[51] Cirq Developers. Cirq, April 2025. URL https://doi.org/10.5281/zenodo.15191735.

[52] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, Shahnawaz Ahmed, Vishnu Ajith, M Sohaib Alam, Guillermo Alonso-Linaje, B AkashNarayanan, Ali Asadi, et al. Pennylane: Automatic differentiation of hybrid quantum-classical computations. *arXiv preprint arXiv:1811.04968*, 2018.

[53] The CUDA-Q development team. Cuda-q, May 2025. URL https://doi.org/10.5281/zenodo.15407754.

[54] Amazon Web Services. Amazon Braket. https://aws.amazon.com/braket/. Accessed: 2025-05-27.

[55] Microsoft. Azure Quantum Development Kit. URL https://github.com/microsoft/qsharp.

[56] Krysta Svore, Alan Geller, Matthias Troyer, John Azariah, Christopher Granade, Bettina Heim, Vadym Kliuchnikov, Mariia Mykhailova, Andres Paz, and Martin Roetteler. Q# enabling scalable quantum computing and development with a high-level dsl. In *Proceedings of the real world domain specific languages workshop 2018*, pages 1–10, 2018.

[57] Robert S Smith, Michael J Curtis, and William J Zeng. A practical quantum instruction set architecture. *arXiv preprint arXiv:1608.03355*, 2016.

[58] Dominik Seitz, Niklas Heim, João P Moutinho, Roland Guichard, Vytautas Abramavicius, Aleksander Wennersteen, Gert-Jan Both, Anton Quelle, Caroline de Groot, Gergana V Velikova, et al. Qadence: a differentiable interface for digital and analog programs. *IEEE Software*, 2025.

[59] Ryan Hill, Ricky Young, and Kanav Setia. qbraid-sdk: Platform-agnostic quantum runtime framework, July 2024. URL https://doi.org/10.5281/zenodo.12627597.

[60] Thomas Beck, Alessandro Baroni, Ryan Bennink, Gilles Buchs, Eduardo Antonio Coello Pérez, Markus Eisenbach, Rafael Ferreira da Silva, Muralikrishnan Gopalakrishnan Meena, Kalyan Gottiparthi, Peter Groszkowski, et al. Integrating quantum computing resources into scientific hpc ecosystems. *Future Generation Computer Systems*, 161:11–25, 2024.

[61] OpenMP Architecture Review Board. OpenMP Application Program Interface Version 6.0. https://www.openmp.org/wp-content/uploads/OpenMP-API-Specification-6-0.pdf, 2024. Accessed: 2025-05-27.

[62] Message Passing Interface Forum. MPI: A Message-Passing Interface Standard Version 4.1. https://www.mpi-forum.org/docs/mpi-4.1/mpi41-report.pdf, 2023. Accessed: 2025-05-27.

[63] Amr Elsharkawy, Xiaorang Guo, and Martin Schulz. Integration of quantum accelerators into hpc: Toward a unified quantum platform. In *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*, volume 1, pages 774–783. IEEE, 2024.

[64] Mateusz Meller, Vendel Szeremi, and Oliver Thomson Brown. Programming tools for analogue quantum computing in the high-performance computing context–a review. *arXiv preprint arXiv:2501.16943*, 2025.

[65] Travis S Humble, Alexander McCaskey, Dmitry I Lyakh, Meenambika Gowrishankar, Albert Frisch, and Thomas Monz. Quantum computers for high-performance computing. *IEEE Micro*, 41(5):15–23, 2021.

[66] Yuping Fan. Job scheduling in high performance computing. *arXiv preprint arXiv:2109.09269*, 2021.

[67] Andy B Yoo, Morris A Jette, and Mark Grondona. Slurm: Simple linux utility for resource management. In *Workshop on job scheduling strategies for parallel processing*, pages 44–60. Springer, 2003.

[68] Will Cunningham, Casey Jao, Sankalp Sanand, Alejandro Esquivel, Faiyaz Hasan, Venkat Bala, Prasanna Venkatesh, Andrew S. Rosen, Madhur Tandon, Okechukwu Emmanuel Ochia, Dave Welsch, Ara Ghukasyan, jkanem, Aravind, HaimHorowitzAgnostiq, Ruihao Li, Scott Wyman Neagle, valkostadinov, FilipBolt, WingCode, Sayandip Dutta, Poojith U Rao, Anna Hughes, ArunPsiog, RaviPsiog, Santosh kumar, mpvgithub, and Udayan. Agnostiqhq/covalent: v0.240.0, May 2025. URL https://doi.org/10.5281/zenodo.15400489.

[69] Nic Ezzell, Bibek Pokharel, Lina Tewala, Gregory Quiroz, and Daniel A Lidar. Dynamical decoupling for superconducting qubits: A performance survey. *Physical Review Applied*, 20(6):064027, 2023.

[70] Ewout Van Den Berg, Zlatko K Minev, and Kristan Temme. Model-free readout-error mitigation for quantum expectation values. *Physical Review A*, 105(3):032620, 2022.

[71] Joel J Wallman and Joseph Emerson. Noise tailoring for scalable quantum computation via randomized compiling. *Physical Review A*, 94(5):052325, 2016.

[72] David Ittah, Ali Asadi, Erick Ochoa Lopez, Sergei Mironov, Samuel Banning, Romain Moyard, Mai Jacob Peng, and Josh Izaac. Catalyst: a python jit compiler for auto-differentiable hybrid quantum programs. *Journal of Open Source Software*, 9(99):6720, 2024.

[73] Michael Broughton, Guillaume Verdon, Trevor McCourt, Antonio J Martinez, Jae Hyeon Yoo, Sergei V Isakov, Philip Massey, Ramin Halavati, Murphy Yuezhen Niu, Alexander Zlokapa, et al. Tensorflow quantum: A software framework for quantum machine learning. *arXiv preprint arXiv:2003.02989*, 2020.

[74] Peter J Karalekas, Nikolas A Tezak, Eric C Peterson, Colm A Ryan, Marcus P Da Silva, and Robert S Smith. A quantum-classical cloud platform optimized for variational hybrid algorithms. *Quantum Science and Technology*, 5(2):024003, 2020.

[75] Alexander J McCaskey, Dmitry I Lyakh, Eugene F Dumitrescu, Sarah S Powers, and Travis S Humble. Xacc: a system-level software infrastructure for heterogeneous quantum–classical computing. *Quantum Science and Technology*, 5(2):024002, 2020.

[76] Pradeep Mantha, Florian J Kiwit, Nishant Saurabh, Shantenu Jha, and Andre Luckow. Pilot-quantum: A quantum-hpc middleware for resource, workload and task management. *arXiv preprint arXiv:2412.18519*, 2024.

[77] Andre Luckow, Mark Santcroos, Andre Merzky, Ole Weidner, Pradeep Mantha, and Shantenu Jha. P*: a model of pilot-abstractions. In *2012 IEEE 8th International Conference on E-Science*, pages 1–10. IEEE, 2012.

[78] Andre Luckow, Mark Santcroos, Ole Weidner, Andre Merzky, Sharath Maddineni, and Shantenu Jha. Towards a common model for pilot-jobs. In *Proceedings of the 21st international symposium on High-Performance Parallel and Distributed Computing*, pages 123–124, 2012.

[79] Konstantinos Rallis, Ioannis Liliopoulos, Evangelos Tsipas, Georgios D Varsamis, Nikolaos Melissourgos, Ioannis G Karafyllidis, Georgios Ch Sirakoulis, and Panagiotis Dimitrakis. Hardware-level interfaces for hybrid quantum-classical computing systems. *arXiv preprint arXiv:2503.18868*, 2025.

[80] Tobias Rohe, Simon Grätz, Michael Kölle, Sebastian Zielinski, Jonas Stein, and Claudia Linnhoff-Popien. From problem to solution: A general pipeline to solve optimisation problems on quantum hardware. *arXiv preprint arXiv:2406.19876*, 2024.

[81] Benjamin Weder, Uwe Breitenbücher, Frank Leymann, and Karoline Wild. Integrating quantum computing into workflow modeling and execution. In *2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC)*, pages 279–291. IEEE, 2020.

[82] Aakash Ahmad, Ahmed B Altamimi, and Jamal Aqib. A reference architecture for quantum computing as a service. *Journal of King Saud University-Computer and Information Sciences*, 36(6):102094, 2024.

[83] Muhammed Golec, Emir Sahin Hatay, Mustafa Golec, Murat Uyar, Merve Golec, and Sukhpal Singh Gill. Quantum cloud computing: Trends and challenges. *Journal of Economy and Technology*, 2:190–199, 2024.

[84] Arnau Pastor, Pau Escofet, Sahar Ben Rached, Eduard Alarcón, Pere Barlet-Ros, and Sergi Abadal. Circuit partitioning for multi-core quantum architectures with deep reinforcement learning. In *2024 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, 2024.

[85] Kaitlin N. Smith, Gokul Subramanian Ravi, Jonathan M. Baker, and Frederic T. Chong. Scaling superconducting quantum computers with chiplet architectures. In *Proceedings of the 55th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '22, page 1092–1109. IEEE Press, 2023. ISBN 9781665462723. doi: 10.1109/MICRO56248.2022.00078. URL https://doi.org/10.1109/MICRO56248.2022.00078.

[86] Kuan-Cheng Chen, Xiaoren Li, Xiaotian Xu, Yun-Yuan Wang, and Chen-Yu Liu. Multi-gpu-enabled hybrid quantum-classical workflow in quantum-hpc middleware: Applications in quantum simulations. *arXiv preprint arXiv:2403.05828*, 2024.

[87] NVIDIA Corporation. NVIDIA DGX Quantum. https://www.nvidia.com/en-eu/data-center/dgx-quantum/. Accessed: 2025-03-17.

[88] Leandro Stefanazzi, Kenneth Treptow, Neal Wilcer, Chris Stoughton, Collin Bradford, Sho Uemura, Silvia Zorzetti, Salvatore Montella, Gustavo Cancelo, Sara Sussman, et al. The qick (quantum instrumentation control kit): Readout and control for qubits and detectors. *Review of Scientific Instruments*, 93(4), 2022.

[89] Yilun Xu, Gang Huang, Neelay Fruitwala, Abhi Rajagopala, Ravi K Naik, Kasra Nowrouzi, David I Santiago, and Irfan Siddiqi. Qubic 2.0: An extensible open-source qubit control system capable of mid-circuit measurement and feed-forward. *arXiv preprint arXiv:2309.10333*, 2023.

[90] Fatemeh Nikbakhtnasrabadi and Martin Weides. Quantum control architecture and circuit blocks for solid-state microwave qubits. *Authorea Preprints*, 2025.

[91] Yuval Shpigelman, Gilad Shainer, Richard Graham, Yong Qin, Gerardo Cisneros-Stoianowski, and Craig Stunkel. Nvidia's quantum infiniband network congestion control technology and its impact on application performance. In *International Conference on High Performance Computing*, pages 26–43. Springer, 2022.

[92] Robert Bogdan Staszewski, Imran Bashir, Elena Blokhina, and Dirk Leipold. Cryo-cmos for quantum system on-chip integration: Quantum computing as the development driver. *IEEE Solid-State Circuits Magazine*, 13(2):46–53, 2021.

[93] Hiroshi Oka. Cryo-cmos device technology for quantum computers. *JSAP Review*, 2022:220305, 2022.

[94] JM Hornibrook, JI Colless, ID Conway Lamb, SJ Pauka, H Lu, AC Gossard, JD Watson, GC Gardner, S Fallahi, MJ Manfra, et al. Cryogenic control architecture for large-scale quantum computing. *Physical Review Applied*, 3(2):024010, 2015.

[95] Fabio Sebastiano, JPG Van Dijk, B Patra, J Van Staveren, X Xue, CG Almudever, G Scappucci, M Veldhorst, LMK Vandersypen, A Vladimirescu, et al. Cryo-cmos interfaces for large-scale quantum computers. In *2020 IEEE International Electron Devices Meeting (IEDM)*, pages 25–2. IEEE, 2020.

[96] Abhijit Das, Maurizio Palesi, John Kim, and Partha Pratim Pande. Chip and package-scale interconnects for general-purpose, domain-specific and quantum computing systems-overview, challenges and opportunities. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2024.

[97] Anton Potočnik, Steven Brebels, Jeroen Verjauw, Rohith Acharya, Alexander Grill, Danny Wan, Massimo Mongillo, Ruoyu Li, Tsvetan Ivanov, Steven Van Winckel, et al. Millikelvin temperature cryo-cmos multiplexer for scalable quantum device characterisation. *Quantum Science and Technology*, 7(1):015004, 2021.

[98] Bogdan-Călin Ciobanu, Luca Perju Verzotti, and Pantelimon George Popescu. Optimal and scalable entanglement distribution over crossbar quantum networks. *Scientific Reports*, 14(1):11714, 2024.

[99] Ruoyu Li, Luca Petit, David P Franke, Juan Pablo Dehollain, Jonas Helsen, Mark Steudtner, Nicole K Thomas, Zachary R Yoscovits, Kanwal J Singh, Stephanie Wehner, et al. A crossbar network for silicon quantum dot qubits. *Science advances*, 4(7):eaar3960, 2018.

[100] Rohith Acharya, Steven Brebels, Alexander Grill, Jeroen Verjauw, Ts Ivanov, D Perez Lozano, Danny Wan, Jacques Van Damme, AM Vadiraj, Massimo Mongillo, et al. Multiplexed superconducting qubit control at millikelvin temperatures with a low-power cryo-cmos multiplexer. *Nature Electronics*, 6(11):900–909, 2023.

[101] Sanskriti Joshi and Sajjad Moazeni. Scaling up superconducting quantum computers with cryogenic rf-photonics. *Journal of Lightwave Technology*, 42(1):166–175, 2023.

[102] Florent Lecocq, Franklyn Quinlan, Katarina Cicak, Jose Aumentado, SA Diddams, and JD Teufel. Control and readout of a superconducting qubit using a photonic link. *Nature*, 591(7851):575–579, 2021.

[103] Pau Escofet, Sahar Ben Rached, Santiago Rodrigo, Carmen G Almudever, Eduard Alarcón, and Sergi Abadal. Interconnect fabrics for multi-core quantum processors: A context analysis. In *Proceedings of the 16th International Workshop on Network on Chip Architectures*, pages 34–39, 2023.

[104] Brian Marinelli, Jie Luo, Hengjiang Ren, Bethany M Niedzielski, David K Kim, Rabindra Das, Mollie Schwartz, David I Santiago, and Irfan Siddiqi. Dynamically reconfigurable photon exchange in a superconducting quantum processor. *arXiv preprint arXiv:2303.03507*, 2023.

[105] Xiao-Min Hu, Yu Guo, Bi-Heng Liu, Chuan-Feng Li, and Guang-Can Guo. Progress in quantum teleportation. *Nature Reviews Physics*, 5(6):339–353, 2023.

[106] Daniel Schoenberger, Stefan Hillmich, Matthias Brandl, and Robert Wille. Shuttling for scalable trapped-ion quantum computers. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2024.

[107] Krishna C Balram and Kartik Srinivasan. Piezoelectric optomechanical approaches for efficient quantum microwave-to-optical signal transduction: the need for co-design. *Advanced Quantum Technologies*, 5(3):2100095, 2022.

[108] Koji Azuma, Sophia E Economou, David Elkouss, Paul Hilaire, Liang Jiang, Hoi-Kwong Lo, and Ilan Tzitrin. Quantum repeaters: From quantum networks to the quantum internet. *Reviews of Modern Physics*, 95(4):045006, 2023.

[109] David Barral, F Javier Cardama, Guillermo Díaz, Daniel Faílde, Iago F Llovo, Mariamo Mussa Juane, Jorge Vázquez-Pérez, Juan Villasuso, César Piñeiro, Natalia Costas, et al. Review of distributed quantum computing. from single qpu to high performance quantum computing. *arXiv preprint arXiv:2404.01265*, 2024.

[110] John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018.

[111] Yagnik Chatterjee, Eric Bourreau, and Marko J Rančić. Solving various np-hard problems using exponentially fewer qubits on a quantum computer. *Physical Review A*, 109(5):052441, 2024.

[112] Adam Callison and Nicholas Chancellor. Hybrid quantum-classical algorithms in the noisy intermediate-scale quantum era and beyond. *Physical Review A*, 106(1):010101, 2022.

[113] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L O'brien. A variational eigenvalue solver on a photonic quantum processor. *Nature communications*, 5(1):4213, 2014.

[114] Jules Tilly, Hongxiang Chen, Shuxiang Cao, Dario Picozzi, Kanav Setia, Ying Li, Edward Grant, Leonard Wossnig, Ivan Rungger, George H Booth, et al. The variational quantum eigensolver: a review of methods and best practices. *Physics Reports*, 986:1–128, 2022.

[115] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

[116] Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast bayesian optimization of machine learning hyperparameters on large datasets. In *Artificial intelligence and statistics*, pages 528–536. PMLR, 2017.

[117] Jonas Močkus. On bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference Novosibirsk, July 1–7, 1974 6*, pages 400–404. Springer, 1975.

[118] Firas Gerges, Germain Zouein, and Danielle Azar. Genetic algorithms with local optima handling to solve sudoku puzzles. In *Proceedings of the 2018 international conference on computing and artificial intelligence*, pages 19–22, 2018.

[119] Aleta Berk Finnila, Maria A Gomez, C Sebenik, Catherine Stenson, and Jimmie D Doll. Quantum annealing: A new method for minimizing multidimensional functions. *Chemical physics letters*, 219(5-6):343–348, 1994.

[120] Tadashi Kadowaki and Hidetoshi Nishimori. Quantum annealing in the transverse ising model. *Physical Review E*, 58(5):5355, 1998.

[121] Jaeho Choi, Seunghyeok Oh, and Joongheon Kim. Quantum approximation for wireless scheduling. *Applied Sciences*, 10(20):7116, 2020.

[122] Abhishek Awasthi, Francesco Bär, Joseph Doetsch, Hans Ehm, Marvin Erdmann, Maximilian Hess, Johannes Klepsch, Peter A. Limacher, Andre Luckow, Christoph Niedermeier, Lilly Palackal, Ruben Pfeiffer, Philipp Ross, Hila Safi, Janik Schönmeier-Kromer, Oliver von Sicard, Yannick Wenger, Karen Wintersperger, and Sheir Yarkoni. Quantum computing techniques for multi-knapsack problems. In Kohei Arai, editor, *Intelligent Computing*, pages 264–284, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-37963-5.

[123] Venkat Padmasola, Zhaotong Li, Rupak Chatterjee, and Wesley Dyk. Solving the traveling salesman problem via different quantum computing architectures. *arXiv preprint arXiv:2502.17725*, 2025.

[124] Giuseppe Buonaiuto, Francesco Gargiulo, Giuseppe De Pietro, Massimo Esposito, and Marco Pota. Best practices for portfolio optimization by quantum computing, experimented on real quantum devices. *Scientific Reports*, 13(1):19434, 2023.

[125] Kamila Zaman, Alberto Marchisio, Muhammad Kashif, and Muhammad Shafique. Po-qa: A framework for portfolio optimization using quantum algorithms. In *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*, volume 1, pages 1397–1403. IEEE, 2024.

[126] Andrew Lucas. Ising formulations of many np problems. *Frontiers in physics*, 2:5, 2014.

[127] Maria Schuld and Francesco Petruccione. Supervised learning with quantum computers. *Quantum science and technology (Springer, 2018)*, 2018.

[128] Vojtěch Havlíček, Antonio D Córcoles, Kristan Temme, Aram W Harrow, Abhinav Kandala, Jerry M Chow, and Jay M Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747):209–212, 2019.

[129] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum algorithms for supervised and unsupervised machine learning. *arXiv preprint arXiv:1307.0411*, 2013.

[130] Nico Meyer, Christian Ufrecht, Maniraman Periyasamy, Daniel D Scherer, Axel Plinge, and Christopher Mutschler. A survey on quantum reinforcement learning. *arXiv preprint arXiv:2211.03464*, 2022.

[131] Valeria Saggio, Beate E Asenbeck, Arne Hamann, Teodor Strömberg, Peter Schiansky, Vedran Dunjko, Nicolai Friis, Nicholas C Harris, Michael Hochberg, Dirk Englund, et al. Experimental quantum speed-up in reinforcement learning agents. *Nature*, 591(7849):229–233, 2021.

[132] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671): 195–202, 2017.

[133] Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. 2008.

[134] N Schetakis, D Aghamalyan, P Griffin, and M Boguslavsky. Review of some existing qml frameworks and novel hybrid classical–quantum neural networks realising binary classification for the noisy datasets. *Scientific reports*, 12(1):11927, 2022.

[135] E Ghasemian and MK Tavassoly. Hybrid classical-quantum machine learning based on dissipative two-qubit channels. *Scientific Reports*, 12(1): 20440, 2022.

[136] Iris Cong, Soonwon Choi, and Mikhail D Lukin. Quantum convolutional neural networks. *Nature Physics*, 15(12):1273–1278, 2019.

[137] AI Gircha, AS Boev, K Avchaciov, PO Fedichev, and AK Fedorov. Hybrid quantum-classical machine learning for generative chemistry and drug design. *Scientific Reports*, 13(1):8250, 2023.

[138] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum support vector machine for big data classification. *Physical review letters*, 113(13): 130503, 2014.

[139] Kerstin Beer, Dmytro Bondarenko, Terry Farrelly, Tobias J Osborne, Robert Salzmann, Daniel Scheiermann, and Ramona Wolf. Training deep quantum neural networks. *Nature communications*, 11(1):808, 2020.

[140] Ioannis Liliopoulos, Georgios D Varsamis, Kristin Milchanowski, Rafael Martin-Cuevas, Konstantina Safouri, Panagiotis Dimitrakis, and Ioannis G Karafyllidis. Hybrid classical-quantum multilayer neural networks for monitoring agricultural activities using remote sensing data. *Quantum Machine Intelligence*, 7(1):4, 2025.

[141] He-Liang Huang, Yuxuan Du, Ming Gong, Youwei Zhao, Yulin Wu, Chaoyue Wang, Shaowei Li, Futian Liang, Jin Lin, Yu Xu, et al. Experimental quantum generative adversarial networks for image generation. *Physical Review Applied*, 16(2):024051, 2021.

[142] Christa Zoufal, Aurélien Lucchi, and Stefan Woerner. Quantum generative adversarial networks for learning and loading random distributions. *npj Quantum Information*, 5(1):103, 2019.

[143] Julien Dudas, Baptiste Carles, Erwan Plouet, Frank Alice Mizrahi, Julie Grollier, and Danijela Marković. Quantum reservoir computing implementation on coherently coupled quantum oscillators. *npj Quantum Information*, 9(1):64, 2023.

[144] Milan Kornjača, Hong-Ye Hu, Chen Zhao, Jonathan Wurtz, Phillip Weinberg, Majd Hamdan, Andrii Zhdanov, Sergio H Cantu, Hengyun Zhou, Rodrigo Araiza Bravo, et al. Large-scale quantum reservoir learning with an analog quantum computer. *arXiv preprint arXiv:2407.02553*, 2024.

[145] Chuanzhou Zhu, Peter J Ehlers, Hendra I Nurdin, and Daniel Soh. Practical and scalable quantum reservoir computing. *arXiv preprint arXiv:2405.04799*, 2024.

[146] Yudai Suzuki, Qi Gao, Ken C Pradel, Kenji Yasuoka, and Naoki Yamamoto. Natural quantum reservoir computing for temporal information processing. *Scientific reports*, 12(1):1353, 2022.

[147] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15):150502, 2009.

[148] William J Huggins, Bryan A O'Gorman, Nicholas C Rubin, David R Reichman, Ryan Babbush, and Joonho Lee. Unbiasing fermionic quantum monte carlo with a quantum computer. *Nature*, 603(7901):416–420, 2022.

[149] Shu Kanno, Hajime Nakamura, Takao Kobayashi, Shigeki Gocho, Miho Hatanaka, Naoki Yamamoto, and Qi Gao. Quantum computing quantum monte carlo with hybrid tensor network for electronic structure calculations. *npj Quantum Information*, 10(1):56, 2024.

[150] Yukun Zhang, Yifei Huang, Jinzhao Sun, Dingshun Lv, and Xiao Yuan. Quantum computing quantum monte carlo. *arXiv preprint arXiv:2206.10431*, 2022.

[151] Yudong Cao, Jonathan Romero, Jonathan P Olson, Matthias Degroote, Peter D Johnson, Mária Kieferová, Ian D Kivlichan, Tim Menke, Borja Peropadre, Nicolas PD Sawaya, et al. Quantum chemistry in the age of quantum computing. *Chemical reviews*, 119(19):10856–10915, 2019.

[152] Sam McArdle, Suguru Endo, Alán Aspuru-Guzik, Simon C Benjamin, and Xiao Yuan. Quantum computational chemistry. *Reviews of Modern Physics*, 92(1):015003, 2020.

[153] Bela Bauer, Sergey Bravyi, Mario Motta, and Garnet Kin-Lic Chan. Quantum algorithms for quantum chemistry and quantum materials science. *Chemical reviews*, 120(22):12685–12717, 2020.

[154] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M Chow, and Jay M Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *nature*, 549(7671):242–246, 2017.

[155] Google AI Quantum, Collaborators*†, Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Sergio Boixo, Michael Broughton, Bob B Buckley, et al. Hartree-fock on a superconducting qubit quantum computer. *Science*, 369(6507):1084–1089, 2020.

[156] Mario Motta, Gavin O Jones, Julia E Rice, Tanvi P Gujarati, Rei Sakuma, Ieva Liepuoniute, Jeannette M Garcia, and Yu-ya Ohnishi. Quantum chemistry simulation of ground-and excited-state properties of the sulfonium cation on a superconducting quantum processor. *Chemical Science*, 14(11):2915–2927, 2023.

[157] Alán Aspuru-Guzik, Anthony D Dutoi, Peter J Love, and Martin Head-Gordon. Simulated quantum computation of molecular energies. *Science*, 309(5741):1704–1707, 2005.

[158] Javier Robledo-Moreno, Mario Motta, Holger Haas, Ali Javadi-Abhari, Petar Jurcevic, William Kirby, Simon Martiel, Kunal Sharma, Sandeep Sharma, Tomonori Shirakawa, et al. Chemistry beyond exact solutions on a quantum-centric supercomputer. *arXiv preprint arXiv:2405.05068*, 2024.

[159] Anton Robert, Panagiotis Kl Barkoutsos, Stefan Woerner, and Ivano Tavernelli. Resource-efficient quantum algorithm for protein folding. *npj Quantum Information*, 7(1):38, 2021.

[160] Christos Papalitsas, Yanfei Guan, Shreyas Waghe, Athanasios Liakos, Ioannis Balatsos, and Vassilios Pantazopoulos. Quantum approximate optimization algorithms for molecular docking. *arXiv preprint arXiv:2503.04239*, 2025.

[161] Andrea Delgado and Prasanna Date. Defining quantum-ready primitives for hybrid hpc-qc supercomputing: a case study in hamiltonian simulation. *Frontiers in Computer Science*, 7:1528985, 2025.

[162] Irina Heinz and Guido Burkard. Crosstalk analysis for single-qubit and two-qubit gates in spin qubit arrays. *Physical Review B*, 104(4):045420, 2021.

[163] Krishnageetha Karuppasamy, Varun Puram, Stevens Johnson, and Johnson P Thomas. A comprehensive review of quantum circuit optimization: Current trends and future directions. *Quantum Reports*, 7(1):2, 2025.

[164] Hila Safi, Karen Wintersperger, and Wolfgang Mauerer. Influence of hw-sw-co-design on quantum computing scalability. In *2023 IEEE International Conference on Quantum Software (QSW)*, pages 104–115. IEEE, 2023.

[165] Chenghong Zhu, Xian Wu, Jingbo Wang, and Xin Wang. S-sync: Shuttle and swap co-optimization in quantum charge-coupled devices. *arXiv preprint arXiv:2505.01316*, 2025.

[166] Thomas Lubinski, Cassandra Granade, Amos Anderson, Alan Geller, Martin Roetteler, Andrei Petrenko, and Bettina Heim. Advancing hybrid quantum–classical computation with real-time execution. *Frontiers in Physics*, 10:940293, 2022.

[167] F Javier Cardama, Jorge Vázquez-Pérez, César Piñeiro, Juan C Pichel, Tomás F Pena, and Andrés Gómez. Review of intermediate representations for quantum computing. *The Journal of Supercomputing*, 81(2):418, 2025.

[168] Deborah Volpe, Nils Quetschlich, Mariagrazia Graziano, Giovanna Turvani, and Robert Wille. A predictive approach for selecting the best quantum solver for an optimization problem. In *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*, volume 1,

pages 1014–1025. IEEE, 2024.

[169] Paolo Viviani, Roberto Rocco, Matteo Barbieri, Gabriella Bettonte, Elisabetta Boella, Marco Cipollini, Jonathan Frassineti, Fulvio Ganz, Sara Marzella, Daniele Ottaviani, et al. Assessing the elephant in the room in scheduling for current hybrid hpc-qc clusters. *arXiv preprint arXiv:2504.10520*, 2025.

[170] Emmanouil Giortamis, Francisco Romão, Nathaniel Tornow, and Pramod Bhatotia. {QOS}: Quantum operating system. In *19th USENIX Symposium on Operating Systems Design and Implementation (OSDI 25)*, pages 429–447, 2025.

[171] Yiqing Zhou, E Miles Stoudenmire, and Xavier Waintal. What limits the simulation of quantum computers? *Physical Review X*, 10(4):041038, 2020.

[172] Iosif-Angelos Fyrigos, Panagiotis Dimitrakis, and Georgios Ch. Sirakoulis. Quantum computing on memristor crossbars. In *Design and Applications of Emerging Computer Systems*, pages 623–647. Springer, 2023.