Received: 26 November 2023 | Revised: 5 February 2024 | Accepted: 6 March 2024

DOI: 10.1002/spe.3331

RESEARCH ARTICLE

WILEY

iQuantum: A toolkit for modeling and simulation of quantum computing environments

Hoa T. Nguyen[1] | **Muhammad Usman**[2,3] | **Rajkumar Buyya**[1]

[1]The Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, The University of Melbourne, Parkville, Victoria, Australia

[2]School of Physics, The University of Melbourne, Parkville, Victoria, Australia

[3]Data61, CSIRO, Clayton, Victoria, Australia

Correspondence
Hoa T. Nguyen and Rajkumar Buyya, The Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, The University of Melbourne, Parkville, 3052, VIC, Australia.
Email: thanhhoan@student.unimelb.edu.au and rbuyya@unimelb.edu.au

Summary

Quantum computing resources are predominantly accessible through cloud services, with a potential future shift to edge networks. This paradigm and the increasing global interest in quantum computing have amplified the need for efficient, adaptable resource management strategies and service models for quantum systems. However, many limitations in the quantum resources' quantity, quality, availability, and cost pose significant challenges for conducting research in practical environments. To address these challenges, we proposed iQuantum, a holistic and lightweight discrete-event simulation toolkit uniquely tailored to model hybrid quantum computing environments. We also present a detailed system model for prototyping and problem formulation in quantum resource management. Through rigorous empirical validation and evaluations using large-scale quantum workload datasets, we demonstrate the flexibility and applicability of our toolkit in various use cases. iQuantum provides a versatile environment for designing and evaluating quantum resource management policies such as quantum task scheduling, backend selection, hybrid task offloading, and orchestration in the quantum cloud-edge continuum. Our work endeavors to create substantial contributions to quantum computing modeling and simulation, empowering the creation of future resource management strategies and quantum computing's broader applications.

KEYWORDS

quantum cloud-edge continuum, quantum computing environments, quantum resource management, quantum simulation, quantum system modeling, quantum task scheduling

1 | INTRODUCTION

Quantum computing holds enormous promise for solving computationally intractable problems, revolutionizing various domains such as drug discovery,[1] finance,[2] optimization,[3] and machine learning.[4–6] The emergence of the cloud-based quantum computing[7,8] and quantum computing-as-a-service (QCaaS)[9] models has enabled access to

Abbreviations: CLOPS, circuit operations per second; CNode, classical computation node; CTask, classical task; CSV, comma-separated values; CPU, central processing unit; NISQ, noisy intermediate-scale quantum; QASM, quantum assembly; QCaaS, quantum computing-as-a-service; QNode, quantum computation node; QoS, quality of service; QPU, quantum processing unit; QTask, quantum task; QV, quantum volume; VM, virtual machine.

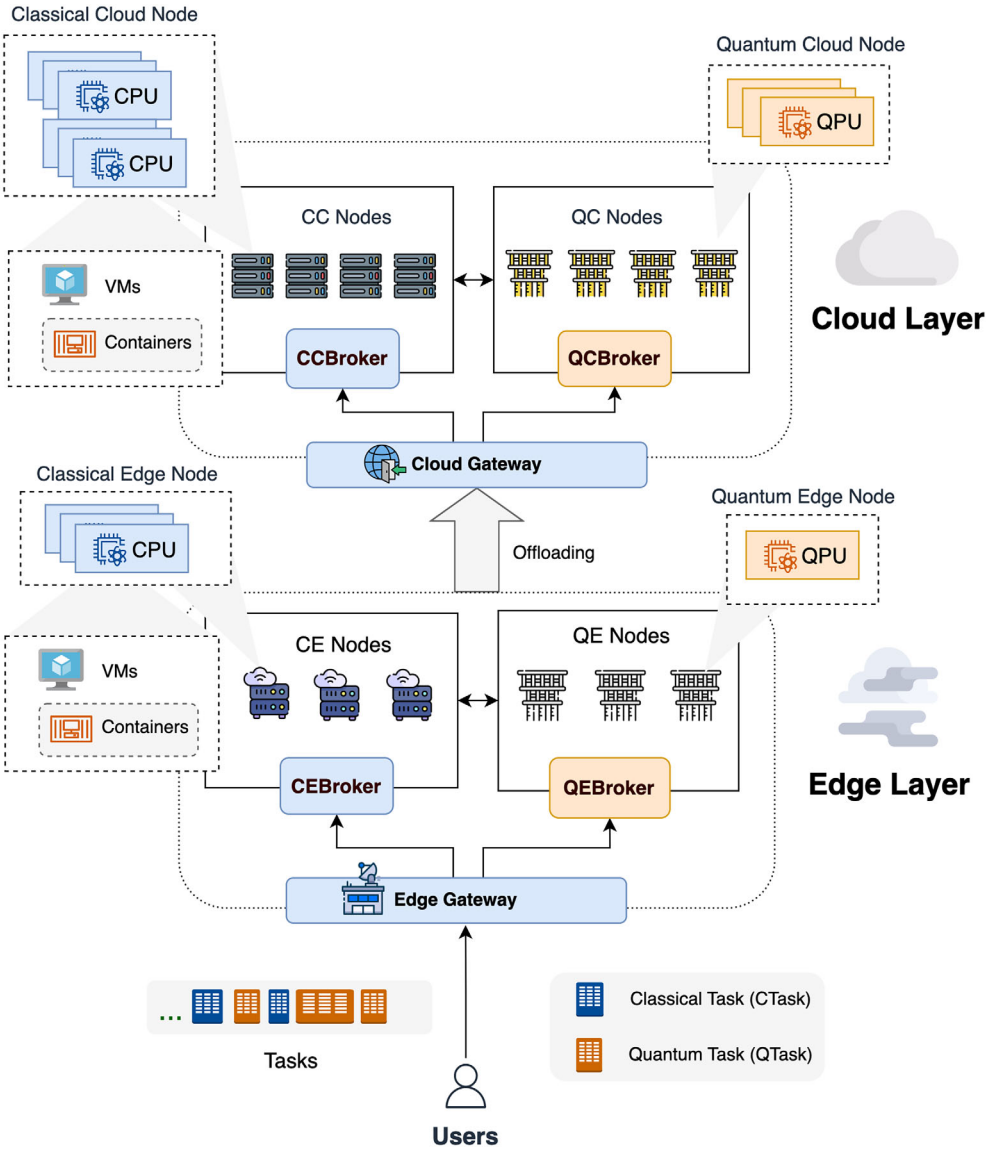

FIGURE 1 Overview of hybrid quantum computing paradigm envisions the seamless integration of quantum and classical computation resources across different cloud and emerging edge layers, with edge resources being geographically closer to end-users (data sources), albeit with computational limitations compared to their cloud counterparts.

quantum computation resources without a massive upfront investment in quantum hardware, leading to tremendous progress in quantum software and algorithm fronts.[10] Major cloud providers, such as Microsoft Azure,[11] AWS,[12] and IBM,[13] now offer cloud-based access to their quantum computing services. Moreover, quantum computation resources are predicted to be extended to the edge network[14,15] when quantum hardware becomes popular in the future, envisioning the emergence of the hybrid paradigm of quantum cloud-edge continuum,[16] whose main components are illustrated in Figure 1.

The future quantum computing paradigm is anticipated to incorporate heterogeneous quantum and classical computing entities situated in different layers, including the cloud and the fog/edge layer. The main differences between cloud-based resources and edge-based resources include the computation capacity, mobility, and geographical distance to the data source or users.[17] Each layer comprises different computation resources and intermediary components, such as gateways and brokers for resource management and orchestration. If edge computation resources are insufficient for executing incoming tasks, these tasks can be migrated or offloaded to the upper cloud layer with more powerful capacity.[18,19] It is important to highlight that this is the vision for the future expansion of quantum computing, whereas most available quantum resources are only accessible through the cloud due to the limitation in quantity, quality, and cost of current quantum hardware.[20]

As the demand for quantum computing services continues to rise rapidly, it triggers the inevitable requirement for efficient system design and resource management strategies to maximize the benefits of available quantum resources.[21] However, there are several challenges to designing and evaluating system and resource orchestration policies in practical environments.[22] First, access to physical quantum computers is limited and costly. Although vendors such as IBM Quantum offer free access to several quantum computers to the public, these devices are on a small scale, comprising only a few qubits. In addition, completing a cloud-based quantum task can take anywhere from seconds to hours because of the fair-share policy that is in place for sharing limited resources among a large number of users worldwide.[21] This means that some tasks may have to wait for others to finish before they can be executed in a real quantum computer. On the other hand, the pricing model for commercial quantum computing services is still expensive. This is because of the limitations and operating costs associated with the current quantum hardware available. For example, the IBM Quantum Pay-As-You-Go plan charges up to 1.6 USD for every second of quantum execution (as of August 2023). Furthermore, it's important to note that quantum hardware is still in the Noisy Intermediate-Scale Quantum (NISQ) era,[20] which implies the limitation in the quality and quantity of qubits inside the quantum chips. These challenges hinder large-scale evaluation and experimental validation of resource management strategies. As a result, it's crucial to have a simulation framework that can model hybrid quantum computing environments to aid in the design and evaluation of resource orchestration policies.

Over the last decade, simulation toolkits such as CloudSim[23] have gained popularity for modeling cloud environments and supporting resource management research. Moreover, several simulators have been proposed for hybrid cloud-edge and fog/edge environments, including EdgeCloudSim,[24] FogNetSim,[25] iFogSim,[26] EdgeSimPy.[27] These simulation toolkits play a significant role in the development of enormous resource management policies for cloud and edge computing environments. However, as far as we know, none of the existing cloud-edge simulators supports the modeling of quantum computing systems and workloads. Meanwhile, existing quantum simulators mainly focus on emulating quantum physical operations of quantum computers, and do not offer comprehensive support for modeling quantum cloud-edge computing environments. As these quantum simulators use classical resources to mimic the actual quantum execution, they can quickly reach the limitation of classical hardware and can usually support up to tens to hundreds of qubits.[28] Besides, several quantum simulators focus on quantum communications, which support modeling quantum network protocols.[29,30] While quantum networks represent a promising direction for the future of quantum computing,[31] it is important to highlight that quantum systems can still utilize classical drivers and networks to handle user requests and facilitate cooperation between quantum and classical computation. Ultimately, the lack of a modeling and simulation framework for quantum computing environments poses significant challenges for research in quantum system design and resource management, impeding researchers from effectively testing their system designs or task scheduling algorithms for hybrid quantum computing systems. Furthermore, it also complicates reproducing experimental results or comparing the performance of different algorithms or applications since no standard simulator is available.

To tackle these challenges, we propose *iQuantum*, a versatile and lightweight simulation framework designed to model quantum computing environments to facilitate quantum software and system research, focusing on resource management and orchestration. The main approach of our toolkit is simplifying and modeling the environments with resources that are quantum systems with key metrics such as number of qubits, quantum volume, quantum processor speed,[32] native gate sets, and qubit topology. Similarly, we extract the features of quantum circuits to model as workload entities in the environments. Then, we employ the discrete-event simulation method, which is a popular simulation technique for operations research.[33] We leverage the core engine and classical components of the latest version of CloudSim[23] to extend and adapt to the quantum computing environment, expanding from the cloud layer to edge layers with various potential use cases. Our toolkit can pave the way for the development of a quantum environment modeling and simulation, which empowers researchers to prototype, design, and evaluate their system design and policies in a simulated quantum computing environment, eliminating the need for costly access to practical quantum resources. Moreover, it enhances research and experimentation in quantum software and systems, enabling result comparison and experiment replication for more robust and impactful investigations aligned with the latest advances in quantum computing.

Our earlier study[14] introduced the initial ideas and proof-of-concept design for the creation of iQuantum, mainly focusing on cloud-based quantum environments. In this paper, we thoroughly extend the architecture design, system model, and implementation of iQuantum along with extensive empirical evaluation to demonstrate the effectiveness of iQuantum for modeling and simulation of quantum computing environments in the cloud-edge continuum. The major contributions of our extended study are as follows:

1. We present a comprehensive system model for quantum computing environments using key metrics and features of available quantum computers and quantum task execution. Besides, we also propose various models and simulation logic for different use cases in hybrid quantum resource management and orchestration. These models serve as theoretical references for prototyping and problem formulation in system design and resource management for hybrid quantum computing.
2. We design the architecture of iQuantum based on the discrete-event simulation approach of CloudSim and extensively extend the entire implementation of iQuantum to enhance the flexibility and support all proposed resource management use cases, including task scheduling, backend selection, hybrid task orchestration, and task offloading between edge and cloud layer.
3. We validate and evaluate iQuantum in different scenarios using trustworthy datasets, including IBM Quantum[13] calibration data for quantum systems and the MQT Bench dataset[34] for the workload. Our findings demonstrate that iQuantum is a versatile and efficient tool that holds great potential to support the development and evaluation of policies related to various resource management issues.
4. We discuss the lesson learned throughout the development of the iQuantum simulator, which can bring valuable insights for developing and extending quantum computing environment modeling and simulation frameworks in the future.

Besides, as an open-source toolkit, iQuantum is designed to enhance and collaborate with other tools in the quantum software ecosystem, specifically in the areas of quantum environment modeling and simulation. This area is inevitably essential for the advancement of quantum resource management policies, along with the rapid maturation of quantum hardware and software.

The rest of this paper is structured as follows: Section 2 reviews related work and identifies gaps in the literature. Section 3 outlines the system model for the quantum computing environment. Section 4 proposes the architecture design and implementation of iQuantum. Section 5 presents different models for various use cases of iQuantum in resource management and orchestration problems. Section 6 shows an explanatory example, sample workflow operation of iQuantum, and performance evaluations from various scenarios and workload datasets. Section 7 discusses the lessons learned from developing iQuantum. Finally, Section 8 concludes the study and suggests future directions.

2 | RELATED WORK

Table 1 summarizes the overall comparison of iQuantum and other related work in terms of modeling and simulation toolkit as well as quantum computing simulation.

TABLE 1 A feature comparison overview of related works with our iQuantum simulator.

Toolkits	Simulation focus	Environment modeling			Policies modeling	Datasets support	Event simulation	Integration possibility	Language
		Quantum	Cloud	Edge					
QuNetSim[29]	Network	E	×	×	Network protocols	—	—	✓	Python
NetSquid[30]	Network	E	×	×	Network protocols	—	✓	✓	Python
QuEST[43]	Circuit operation	E	×	×	—	—	—	✓	C++
PAS[44]	Circuit operation	E	×	×	—	—	—	—	—
QXTools[45]	Circuit operation	E	×	×	—	✓	—	—	Julia
iQuantum *(Our toolkit)*	System & workload	S	✓	✓	Resources management	✓	✓	✓	Java

Note: Classical cloud features derived from CloudSim.
Abbreviations: ✓, supported; ×, unsupported; —, N/A; E, emulation; S, simulation.

In the classical computing domain, modeling and simulation tools such as CloudSim[23] have become popular for their capability to facilitate the development and evaluation of resource management policies.

In the rapidly evolving era of the Internet of Things (IoT),[35] paradigms such as edge and fog computing are gaining prominence.[36,37] Although CloudSim, the predecessor of iQuantum, facilitates various simulation models for cloud computing-based use cases, it cannot be directly used for modeling edge/fog environments. As a result, new simulation toolkits are proposed to adapt to these kinds of computing paradigms. Mahmud et al.[26] proposed iFogSim2, which extends the first version of iFogSim[38] to support mobility, clustering, and microservices management policies in fog and edge computing. Similarly, Souza et al.[27] propose EdgeSimPy, which employs an agent-based modeling technique to represent each entity in the edge environment as an agent that has its own behavior, decision-making capabilities, and interactions with other agents and the environment. Additionally, IoT applications in emerging domains like blockchain IoT (B-IoT)[39] and medical applications[40] present a new frontier. Blockchain IoT integration offers enhanced security and efficiency in IoT scenarios. This integration is particularly significant in applications where blockchain's decentralized and immutable nature can strengthen data integrity and trust in distributed IoT networks. Simulators such as ChainFL[41] and xFogSim,[42] can be adapted to simulate such scenarios, providing valuable insights into the resource management challenges and opportunities inherent in B-IoT systems. Conversely, the key characteristic of the modeling toolkit is capturing the behaviors of the actual entities and modeling them as an object or agents, then simulating the actual interactions among different entities through events. However, there is no existing cloud/edge simulator support modeling quantum computing resources and workloads.

In terms of quantum computing simulation, it is important to highlight that there are numerous quantum simulators, but they are mostly focused on simulating the physical quantum operation, which mimics the real quantum systems and can be categorized as "*emulation*". However, none of the existing quantum simulation toolkits support modeling the environments and resource management problems, which can be categorized as "*simulation*", similar to other modeling toolkits in the classical realm.[23]

Industry-standard quantum simulators, such as IBM's quantum simulator integrated with Qiskit[46] and Google's quantum simulator employed with Cirq,[47] are renowned for their robustness in simulating quantum circuit operations. However, iQuantum distinguishes itself from these simulators in several key aspects. First, while IBM and Google's simulators primarily concentrate on the physical aspects of quantum computation, iQuantum mainly focuses on modeling quantum computing environments, facilitating research in resource management problems. This approach is similar to other well-known simulators in the classical domain, such as CloudSim,[23] iFogSim,[26] and EdgeSimPy.[27] Second, iQuantum is designed as a lightweight, discrete-event simulation framework, making it more accessible and easier to model and simulate the large-scale environment with heterogeneous quantum (and classical) instances. This contrasts with the more resource-intensive nature of several industry simulators, which may require more computational overhead[22] when employed to design and evaluate resource management algorithms. Additionally, iQuantum is designed with an interoperability approach, enabling it to leverage the modeling of circuit and quantum computation features extracted from these established simulators. Indeed, we can extract attributes of quantum circuits from benchmark datasets (such as MQT Bench[34]) using Qiskit and use IBM's benchmark data on quantum volume and circuit layer operation per second (CLOPS)[32] of quantum systems to model quantum computing environments in iQuantum (see Section 7).

Apart from the built-in quantum simulators of common quantum SDKs such as Qiskit, Cirq, Braket, and Q#, several works have proposed simulation frameworks for quantum operation and communications. Regarding quantum networks, Diadamo et al.[29] proposed QuNetSim as a framework for simulating different quantum network protocols, such as quantum key distribution and quantum routing. As QuNetSim relies on other qubit simulators (such as SimulaQron,[48] ProjectQ,[49] and QuTiP[50]), its main objective is to develop quantum network protocol simulation rather than distributed quantum system modeling. Similarly, NetSquid[30] is a discrete-event network simulator for simulating quantum network protocols and systems. Although quantum communications is certainly an important prospect of the future implication of quantum technology, it is still in its infancy and requires more research effort to develop standard protocols and methods. In practice, current quantum computing services can still leverage the classical driver counterpart and network communication for quantum execution. Therefore, our focus in iQuantum is to reflect the current nature of the quantum computing environment in consensus with classical resources.

Besides, several simulators for quantum operation have been proposed recently. Jones et al. proposed QuEST[43] to support simulating the behavior of quantum systems with high performance. Similarly, Bian et al.[44] proposed PAS as a lightweight quantum simulator, and the authors argued that PAS outperforms QuEST in terms of quantum operation simulation. QXTools[45] is another Julia-based toolkit for simulating large-scale quantum circuits using the tensor network approach, which can be executed on a distributed computation cluster. However, as the main focus of these simulators

is quantum operation, modeling and simulation toolkits for quantum systems and workloads are needed to facilitate the design of resource management policies.

In terms of modeling and simulation tools for quantum computing environments, especially for facilitating the design and evaluation of quantum resource management policies, iQuantum can be considered as one of the *first-of-its-kind* toolkits. As iQuantum is built on top of CloudSim,[23] which is one of the most widely-used simulators over the last decade for cloud computing environments, our toolkit can leverage the capabilities of CloudSim to expand its support to classical resource modeling in the cloud and extend to the edge network. Eventually, iQuantum serves as the unified toolkit for supporting the hybrid quantum-classical computing paradigm in the edge-cloud continuum, which enhances the seamlessly and simplicity of employing different frameworks and paves a new way for research in system design and resource management for various scenarios, such as quantum computing and hybrid quantum-classical computing environments. Although simulators can have their limitations compared to practical environments (such as the QFaaS framework[22] for practical quantum environments), by capturing the timing and interactions of events, modeling and simulation toolkits provide valuable insights into the performance resource utilization and scalability of the overall systems.

iQuantum also offers other advantages that distinguish it from existing toolkits. For instance, it provides support for external datasets, accommodating both OpenQASM files for quantum tasks and quantum computer calibration data for quantum systems. Our toolkit also facilitates data importing and exporting in a common CSV format, enabling further investigation and analysis. The modular architecture of iQuantum allows for easy extension and customization to support various use cases. Furthermore, iQuantum can be seamlessly integrated with common quantum SDKs like Qiskit,[46] enabling workload feature extraction and dataset generation. For users seeking more advanced resource management techniques, iQuantum's compatibility with Python-Java brokers, such as Py4J,[*] allows for straightforward integration of machine learning-based policies. Ultimately, these features collectively position iQuantum as a powerful and flexible toolkit for modeling and simulation in hybrid quantum computing environments.

3 | SYSTEM MODEL FOR QUANTUM ENVIRONMENTS

This section outlines the system model of key entities in the quantum computing environments, including quantum processing units (QPUs), quantum nodes, and quantum tasks. Figure 2 illustrates these entities and main attributes, which are described in detail below.

3.1 | QPUs and quantum computation nodes

3.1.1 | Quantum processing units

The computational unit of a physical quantum system (or quantum node–QNode) is a quantum chip (or quantum processing unit–QPU).

A QPU q_i of QNode Q can be defined as follows:

$$q_i = \{q^w, q^v, q^s, q^g, q^t, q^e\} \tag{1}$$

where:

1. q^w is the number of qubits (or width), which implies the scale of the QPU. The more number of qubits, the more quantum information a QPU can process.
2. q^v is the quantum volume (QV) of the QPU, which indicates the quality of qubits and the QPU's capability to execute a quantum circuit precisely. Cross et al.[51] proposed the measurement of QV as follows:

$$q^v = 2^{\min(d,m)} \tag{2}$$

where d and m are the depth and width of the largest square circuit that can be faithfully executed. The higher the value of QV, the higher the possibility of getting a precise execution result.

[*] https://www.py4j.org/.

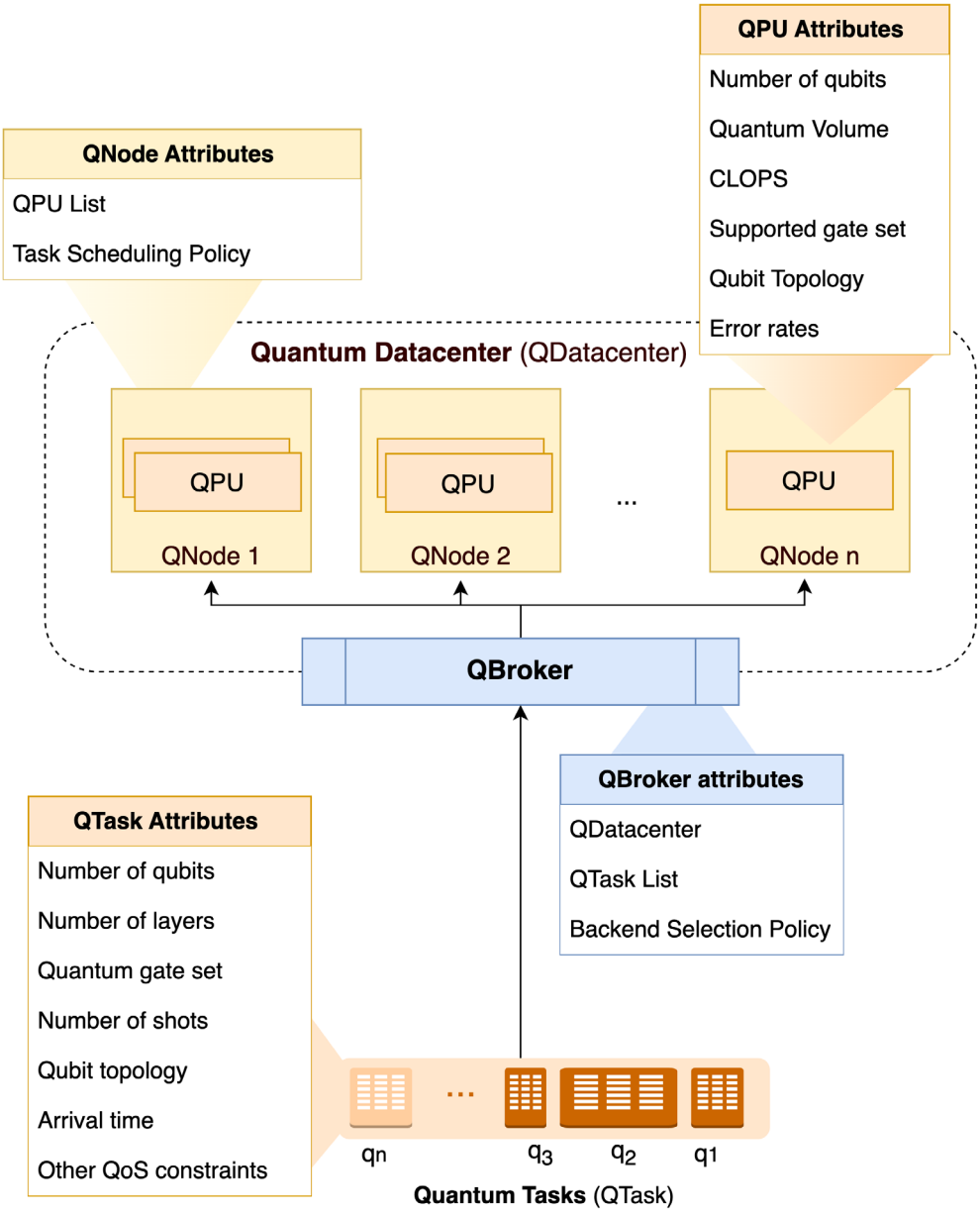

FIGURE 2 Overview of the system model and key attributes of entities in quantum computing environments.

3. q^s is the number of circuit layers that can be processed per second (CLOPS),[32] which indicates the speed of the QPU for processing quantum circuits. CLOPS can be empirically measured by

$$q^s = \frac{M \times K \times S \times D}{time\_taken} \tag{3}$$

where $M = 100, K = 10, S = 100, D = \log_2 q^v$, which stands for the number of evaluated templates, number of parameter updates, number of shots, and number of QV layers, respectively. The higher the CLOPS, the faster the QPU can operate.
4. q^g indicates a list of all supported single-qubit and multiple-qubit quantum gates of the QPU. For example, most available IBM Quantum chips from 5 to 65 qubits support five native gate sets, including CNOT, ID, RZ, SX, and X gate, where their most recent 127-qubit and 433-qubit chips (Eagle r3 and Osprey r1) replace the support of CNOT gate with ECR gate.

5. q^t represents the qubit topology (or connectivity) of all qubits in the QPU, which can be modeled by a graph $q^t = (\mathcal{V}, \mathcal{C})$, where $\mathcal{V} = \{v_i | 1 \leq i \leq |\mathcal{V}|\}, |\mathcal{V}| = q^w$ depicts the number of qubits and v_i denotes the i-th qubit; $\mathcal{C} = \{c_{i,j} | v_i, v_j \in \mathcal{V}, i \neq j\}$ denotes the connection of two qubits, where $c_{i,j}$ denotes the connection between qubit v_i and qubit v_j.
6. q^e represents a list of all error rates of the QPU, which can be defined as $q^e = (\mathcal{E}_v, \mathcal{E}_g)$, where $\mathcal{E}_v = \{e_i | 1 \leq i \leq |\mathcal{V}|\}$ denotes the list of all qubit error, and e_i^v depicts the readout assignment error of i-th qubit; $\mathcal{E}_g = \{e_k^g | 1 \leq k \leq |q^g|\}$ denotes the list of all quantum gate errors and e_k^g depicts the readout assignment error of k-th quantum gate. The errors of the two-qubit gate (CNOT and ECR gate) contain all errors between each pair of two qubits in the QPU. The error rates model can be useful in assisting the development of solutions for more complex resource management problems that take into account the quality of qubits and the quantum volume. However, it is important to note that supporting the evaluation of the quantum errors impacts on quantum execution results is not within the scope of our study and can be considered for future extension.

Besides, other calibration metrics of each qubit can be modeled to reflect the comprehensive properties of a QPU, such as T1 time, T2 time, frequency, and anharmonicity. It is worth noting that although we support modeling all of these features in the implementation of iQuantum, the complex resource management policies considering all error rates and all calibration data of each qubit in the QPU are out of scope in this study, which inspire the further contribution of other practitioner and researchers in the field of quantum computing systems and quantum error mitigation.

3.1.2 | Quantum nodes

A quantum computation node (QNode $\mathcal{Q}$) can be modeled as follows:

$$\mathcal{Q} = [\{q_i | 1 \leq i \leq n\}, \pi_s] \tag{4}$$

where n is the number of QPUs and π_s is the local task scheduling of the quantum node.

Presently, most available quantum computer from well-known vendors such as IBM Quantum, Rigetti, and IonQ only has single-chip quantum nodes. However, the proposal for multi-chip quantum nodes such as IBM Quantum System Two[†] are expected to be released in the near future. According to the current situation and potential development of quantum hardware, iQuantum supports both single-QPU ($n = 1$) and multi-QPU ($n \geq 1$) QNode models. Besides, users can design the scheduling policy π_s to determine the execution model of multiple incoming quantum tasks inside the quantum node. More details and examples of scheduling policies can be found in Section 5.1.

3.2 | Quantum datacenters and brokers

3.2.1 | Quantum datacenters

A cluster or centralized hub of multiple quantum nodes at the same location can be defined as a quantum datacenter ($\mathcal{D}^{\mathcal{Q}}$) as follows:

$$\mathcal{D}^{\mathcal{Q}} = [\{\mathcal{Q}_i | 1 \leq i \leq |\mathcal{Q}|\}, \xi, \chi] \tag{5}$$

where $|\mathcal{Q}|$ is the number of QNodes, ξ is the cost model of using quantum computation resources, and χ is the location of the quantum datacenter.

Different quantum cloud providers have different cost models for using their quantum computing services. For example, IBM Quantum offers a Pay-As-You-Go plan through IBM Cloud with a fixed price of using 27- to 127-qubit quantum computers at 1.6 USD per second of quantum runtime,[‡] whereas Amazon Braket[12] calculates the total cost ξ_i of executing i-th quantum task γ_i at quantum computer q by $\xi_i = \xi_q^t + \gamma_i^s \times \xi_q^s$, where ξ_q^t is the cost per task, ξ_q^s is cost per shots

[†] IBM Quantum System Two
[‡] https://www.ibm.com/quantum/access-plans.

of quantum node q, and γ_i^s is the number of shots that quantum task γ_i need to be executed. Additionally, different quantum computers offered by Amazon Braket have different per-task and per-shot prices. Therefore, users can customize the pricing strategy to have an appropriate model if they consider the cost of execution in the resource management problem.

The location χ of a quantum datacenter refers to either the cloud layer or edge layer, where the quantum datacenter is hosted. Besides, it also indicates the linked quantum broker, which interacts with the quantum data center for selecting and scheduling quantum tasks.

3.2.2 | Quantum brokers

The intermediary component in conjunction with a quantum datacenter to manage incoming quantum tasks and coordinates with the datacenter to determine the appropriate backend for task execution can be defined as a quantum broker (QBroker) $\mathcal{B}^{\mathcal{Q}}$ as follows:

$$\mathcal{B}^{\mathcal{Q}} = \{\mathcal{D}^{\mathcal{Q}}, \Gamma, \pi_b\} \tag{6}$$

where $\mathcal{D}^{\mathcal{Q}}$ is the linked quantum datacenter, Γ is a list of all incoming quantum tasks, and π_b is the backend selection policy to determine which available quantum node is suitable to place each incoming quantum task.

3.3 | Quantum tasks

A quantum task (QTask) γ is a fundamental unit of a quantum computation. Conceptually, a quantum application can include one or more quantum circuits that are being sent to be executed in a quantum node.[14] We model each single circuit as a QTask to simplify the simulation process, making it more tractable to model and analyze quantum computing environments, especially those involving complex scheduling and resource allocation scenarios. A complex quantum application that involves more than one quantum circuit can be modeled as multiple QTasks. The key features of a quantum task γ_i can be modeled as follows:

$$\gamma_i = \left\{\gamma^a, \gamma^g, \gamma^w, \gamma^d, \gamma^s, \gamma^e, \gamma^t\right\} \tag{7}$$

where:

1. γ^a is the arrival time of the quantum task.
2. γ^g is a list of all quantum gate sets used in the quantum circuit, which can contain different single- and multiple-qubit quantum gates.
3. γ^w is the quantum circuit width, which is measured by the number of qubits that need to be used.
4. γ^d is the number of circuit layers, which can be assumed to be directly related to the depth of the circuit.
5. γ^s is the number of shots (i.e., execution repetition) that circuit need to be executed.
6. γ^t is the connectivity of all qubits in the circuit.

Besides, a quantum task can be associated with other metrics related to Quality-of-Service (QoS), such as the acceptable error threshold for executing a quantum task. Quantum circuit features such as quantum gates, circuit width, circuit depth, and qubit connectivity extracted from SDKs such as Qiskit can be mapped to QTask automatically. For example, we extracted features of quantum circuits in QASM format from the MQT Bench dataset[34] and mapped them to corresponding QTasks to represent the workload needed to be processed within the quantum computing environment (see Section 7).

A task placement configuration for task $\gamma_i \in \Gamma$ can be represented as $\sigma_i = \{\gamma_i, q_k\} \in \Sigma$, where $q_k \in \mathcal{Q}$ and $1 \leq k \leq |\mathcal{Q}|$ denotes the index of the quantum computation node. More details about quantum task scheduling and use cases for quantum resource management strategies are discussed in Section 5.

4 | IQUANTUM ARCHITECTURE DESIGN AND IMPLEMENTATION

4.1 | Architecture and main components

We developed iQuantum using the *discrete-event* simulation approach, leveraging the core components of the CloudSim toolkit.[23] The initial design of iQuantum, which aimed to demonstrate the possibility of modeling quantum computing environments, was previously proposed in Reference 14. To support a hybrid quantum-classical environment and expand to modeling the potential use cases, we propose an improved layered design for the iQuantum toolkit, as shown in Figure 3. We have significantly enhanced the original design through extensive refactoring and adding new features to enhance the simulation process for both quantum and hybrid quantum-classical environments.

iQuantum is designed with a modular architecture that comprises five main layers: core simulation, physical, logical, middleware, and resource management. All elements in these layers are implemented to demonstrate the functionality and usefulness of our toolkit in modeling, simulating, and evaluating resource management strategies for hybrid quantum-classical computing in the cloud-edge continuum. Additionally, various utility toolkits have been developed to aid in the simulation process.

The following subsections describe the key components of each layer in the iQuantum framework.

4.1.1 | Core simulation layer

The Core Simulation layer in iQuantum is built on the discrete event management framework in CloudSim,[23] which is one of the most widely used simulation toolkits for classical cloud computing environments. The discrete event simulation technique is highly suitable for simulating and modeling dynamic events and their interactions over time in complex computation systems such as cloud-edge environments. This technique is commonly used in many other simulation

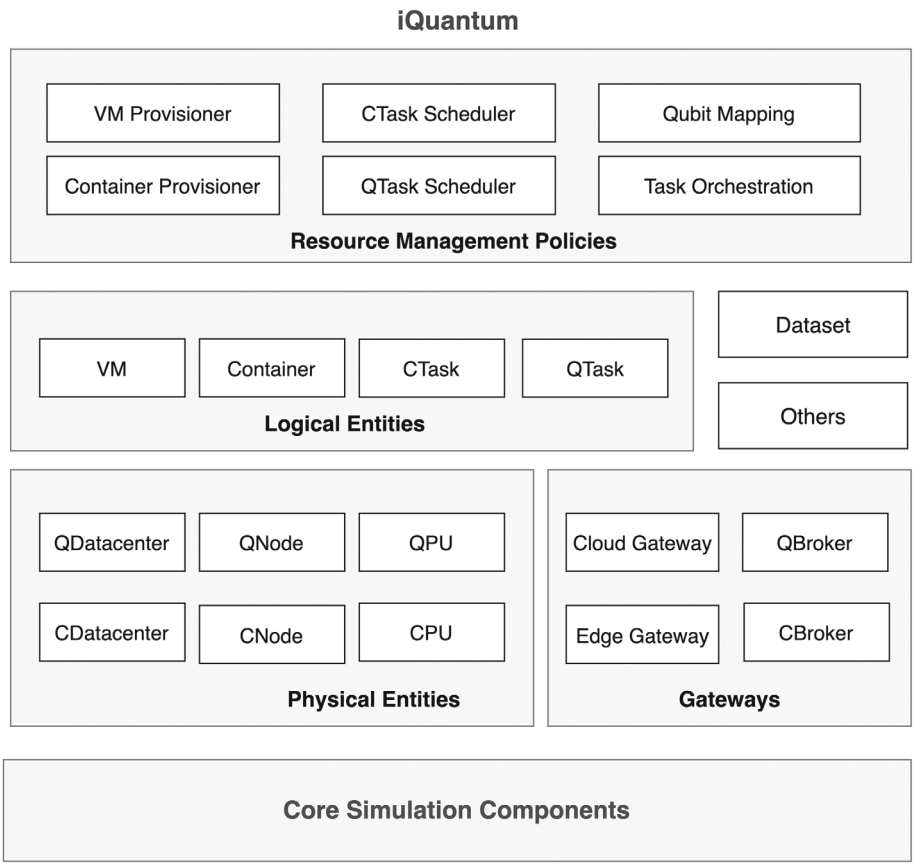

FIGURE 3 Architecture design of iQuantum incorporates five main layers, dataset support, and other auxiliary components.

toolkits, including iFogSim2,[26] ns-3,[52] and OMNET++.[53] We enhanced the core simulation framework of CloudSim to incorporate quantum features and improve clarity in the context of hybrid quantum-classical computing in cloud-edge environments. In iQuantum, the whole system is represented as a set of entities that interact with each other through events. After starting the simulation, the core simulation components initialize all entities based on the provided scenario and register all computation resources with the central Resource Information Service (RIS). The simulation moves forward by scheduling and processing all defined events in chronological order. Each event alters the state of the system and can possibly schedule new events for the future. The simulation continues until it reaches a predetermined termination condition, such as a specified time duration or a certain number of tasks.

4.1.2 | Physical layer

The Physical Layer encompasses various components, including QDatacenter, CDatacenter, QNode, and CNode, which collectively enable the simulation of a hybrid quantum-classical computing environment.

Processing Units in iQuantum comprise both (Classical) Central Processing Units (CPUs) and Quantum Processing Units (QPUs). CPUs handle classical computations and support the quantum task compilation and coordination of hybrid tasks in the system. Each CPU entity (formerly Pe in CloudSim) is modeled by Million Instructions Per Second (MIPS).[23] QPUs (or quantum chips) are responsible for executing quantum tasks. We leverage common metrics and properties of gate-based quantum devices to model a QPU in iQuantum, including the number of qubits (scale), Quantum Volume (QV-quality), Circuit Layer Operations Per Second (CLOPS-speed), qubit connectivity topology, supported quantum gates, qubit, and quantum gate error rates (see Figure 2). To simplify the terminologies, we used "node" to refer to the physical computation devices, where QNodes represent the quantum computer (or quantum system) and CNodes represent the classical server or host (formerly in CloudSim). Each node can contain one or more processing units, enabling parallel execution of tasks. In the classical domain, a CNode can be modeled with other capacities such as RAM and storage. As similar techniques for quantum computing have not yet been invented or are in the early stages of development, we only consider adding QPUs for QNodes in iQuantum at this stage. However, these metrics can be modeled in the future as the technology advances. Besides, it is important to note that current quantum computers mostly support a single quantum chip (QPU). However, a multi-QPU quantum computer is expected to be invented soon with planned roadmaps by major hardware vendors such as IBM. In addition to computing capacity, each node can be associated with different resource management policies and pricing models. For QNodes, users can model different cost models, as different quantum providers offer different pricing models. For instance, IBM Quantum offers a cost per second of the quantum execution model, while Amazon Braket offers a cost-per-execution and shots (iterations) model.

The datacenters in iQuantum, namely QDatacenter and CDatacenter, serve as the infrastructure for resource coordination. In general, a data center can be seen as a collection (or cluster) of multiple computing nodes (QNodes or CNodes), which can be coordinated to perform a common task. Each datacenter entity is associated with a corresponding class to model a list of all belonging computation nodes and other necessary characteristics.

4.1.3 | Logical layer

The Logical Layer in iQuantum consists of abstractions that represent the classical virtualized resources and computation tasks on the system. These abstractions are designed to provide a simplified and standardized interface for both classical and quantum counterparts. The classical part is mainly inherited from CloudSim entities, which includes resource and application abstractions such as Virtual Machines (VMs), containers, and classical tasks (CTasks). VMs are virtualized computing resources allocated within a CNode, while containers represent containerized computing resources allocated within a VM. They offer an isolated and flexible environment for deploying and running classical applications. Additionally, CTasks represent a classical task (formerly Cloudlet in CloudSim) that can be scheduled and executed on classical resources. The quantum part includes the model of quantum task (QTask) as the virtualization technique for quantum counterpart is not yet invented. A QTask represents a computation task that needs to be executed on the quantum computing resources. It encapsulates the quantum algorithm or quantum circuit, along with any necessary parameters and configurations, such as the number of qubits, number of circuit layers, quantum gates, and other execution constraints. By utilizing the QTask abstraction, users can simulate the execution of quantum algorithms and evaluate their performance in conjunction with the classical infrastructure.

4.1.4 | Gateway layer

The gateway layer comprises intermediary components, including gateways and brokers, which facilitate communication and task orchestration between cloud and edge computation components. In iQuantum, a gateway (cloud or edge gateway) is a communication interface between data center brokers and the layer below. It receives incoming tasks, classifies them, and dispatches them to the appropriate broker for further scheduling. The brokers, namely QCloudBroker, QEdgeBroker, CCloudBroker, and CEdgeBroker, then perform the task scheduling based on the resource management policies in the system.

4.1.5 | Resource management layer

This layer enables users to prototype and evaluate various management policies, including task scheduling, migration, orchestration, and resource provisioning.

1. *Task scheduling (or task placement)*: An efficient placement technique must be designed to optimize resource utilization, especially for limited and expensive resources such as quantum computing devices. It is also possible to consider other objectives, such as minimizing execution time or optimizing the precision of quantum execution results by scheduling tasks to quantum nodes with higher quantum volume.
2. *Task offloading*: Due to the dynamic nature of the cloud-edge environments, an adaptive task migration technique must be designed to migrate tasks to other computation nodes or offload tasks from the edge layer to the cloud layer when the edge resources are unavailable or insufficient to execute tasks.
3. *Resource provisioning*: iQuantum can prototype different resource allocation strategies, such as container or virtual machine provisioning and multi-processing unit (CPUs and QPUs) allocation to computation nodes (CNodes and QNodes). It is important to note that resource provisioning policies are mainly for classical resources, as quantum resource virtualization or partitioning is not yet mature. However, it can be extended in future releases to adapt to the current nature of quantum technology.

4.2 | Implementation of quantum components

The overview of the main class diagram for the quantum entities in iQuantum is shown in Figure 4. The core classes for discrete-event simulation are adapted from the latest version of CloudSim to support the seamless integration of iQuantum with classical component modeling.

1. iQuantum class is the core entity, which triggers the initialization of all entities, runs a clock tick during the simulation to record all events, manages the simulation, and triggers all entities to shut down at the end of the simulation.
2. SimEntity and SimEvent: Each core entity in the quantum environment, such as quantum data centers and brokers, is an extension of the SimEntity class, which can generate events for interaction with other entities. Each simulation event is associated with a unique ID (defined in iQuantumTags) and is represented using SimEvent class, which includes properties such as two entities, timestamps, and all exchanging information in the event.
3. ResourceInformationService (RIS): This class handles the resource registration of QDatacenter entities when they are initialized.
4. Gateway is the abstract class that represents the intermediary component that receives all incoming tasks from users or below the layer, then classifies and dispatches to corresponding brokers for further scheduling. Two implementations of this class, CloudGateway and Edgegateway, represent the gateway of cloud and edge layers, respectively.
5. QBroker class models a quantum broker that interacts with the Gateway and QDatacenter to schedule incoming QTask to a suitable QNode in the QDatacenter. Two extended classes, namely QCloudBroker and QEdgeBroker, handle the brokering job at the cloud layer and edge layer accordingly.
6. QDatacenter is the generalized class model of the datacenter (or a cluster) of multiple quantum nodes located in the same areas. There are two specific classes that extend this class for modeling a cloud-based quantum datacenter (QCloudDatacenter) and an edge-based quantum node cluster (QEdgeDatacenter). A list of all associated QNode and other configurations of QDatacenter are defined in QDatacenterCharacteristics class.

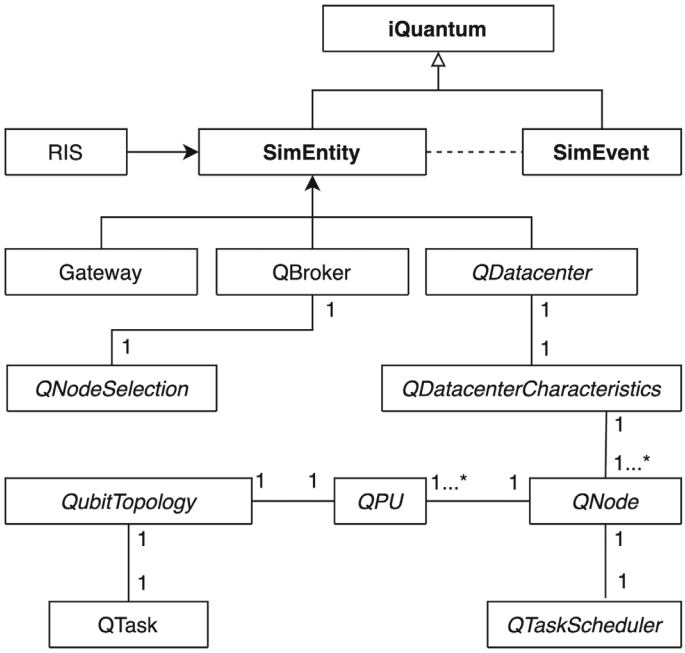

FIGURE 4 Overview of class diagram for quantum components.

7. QNode class represents the gate-based quantum computer, which is used for executing QTask. Each QNode can consist of one or more QPU, cost of execution, and a QTask scheduling policy. Each QPU can be modeled in QPU class. Each QPU class represents different metrics, such as the number of qubits, quantum volume, CLOPS, list of all supported gates, and qubit topology. In the QubitTopologyExtended class, we model the connectivity of all qubits along with the error rates of each qubit and quantum gate. Users can also import calibration data from vendors, such as IBM Quantum, to quickly model the quantum nodes.
8. QTask class represents all features of a gate-based quantum task (or quantum circuit). Users can extract the circuit features of a quantum task (from a QASM file or CSV dataset) and import them into iQuantum automatically. The connectivity of qubits in a QTask can also be modeled using the QubitTopology class.
9. QTaskScheduler is an abstract class to model the scheduling policy to distribute resources of each QNode among multiple incoming QTasks. We implemented the QTaskSchedulerSpaceShared by default for QNode, which is described in detail in Section 5.1.

In the context of our simulation framework, it is important to outline the scope of iQuantum regarding quantum physical operation. iQuantum's design does not include the direct simulation of quantum physical phenomena such as superposition and entanglement. This decision aligns with our framework's objective to provide a high-level simulation environment primarily concerned with the operational aspects of quantum computing environments. However, iQuantum is capable of modeling quantum circuit features from external quantum SDKs such as Qiskit. For instance, quantum circuit features can be obtained from these SDKs and subsequently incorporated into iQuantum's simulations to represent a QTask. This integration allows iQuantum to model the outcome and performance implications of quantum computations based on the empirical benchmark data, albeit at an abstract level. In this way, iQuantum can simulate the broader operational environment of quantum computing, such as the scheduling of quantum tasks into an appropriate quantum system to optimize resource management.

4.3 | Quantum nodes and workload datasets

Initially, simulation scenarios in iQuantum can be defined manually,[14] similar to CloudSim.[23] This approach allows users to customize the definition and attributes of all entities in the system. However, this task can be trivial for explanatory or small experiments but impractical for large-scale experiments with a vast number of heterogeneous devices and tasks.

To improve the flexibility of dataset processing in iQuantum, we support quick prototyping by using external datasets. Datasets in iQuantum are expected to be formatted as a CSV file, which is commonly used in machine learning and software domains.

1. **Quantum nodes**: iQuantum supports the use of a calibration datasheet format, adopted from IBM Quantum,[13] to model all attributes of a QNode, including detailed qubit topologies and error rates.
2. **Quantum tasks**: The quantum workload dataset in iQuantum can be generated by extracting all features from the OpenQASM file to a CSV file. You can import this dataset directly into iQuantum to automatically generate QTasks for evaluating resource management policies. We derived several sets of quantum workload datasets for iQuantum based on QASM files from the MQT Bench.[34]

5 | USE CASES OF IQUANTUM FOR QUANTUM RESOURCE MANAGEMENT SIMULATION

In this section, we demonstrate the usefulness and effectiveness of iQuantum in modeling quantum computing resource management problems, such as quantum task scheduling and hybrid quantum-classical task orchestration in the cloud-edge continuum. In order to make our explanation accessible to a wide range of readers, we explain how iQuantum can be used with general resource management policies and discuss the possibility of tackling more complex problems. It is important to note that creating new resource management policies is not within the scope of this paper.

5.1 | Quantum task scheduling

Effective task scheduling (or task placement) is a crucial aspect of distributed system research, which also applies to the quantum computing domain. The goal is to enhance the management of computational resources by optimizing resource utilization, reducing the overall time taken to complete tasks, and minimizing the cost of resource usage.

The overall simulation logic of QTask scheduling problems in iQuantum is described in Algorithm 1. We divide the quantum task scheduling process into two main stages: quantum node selection (or backend selection) and task allocation at QNode. After initializing the environment and the simulation clock, all tasks are submitted to the closest gateway (Cloud Gateway or Edge Gateway). Then, the gateway classifies and dispatches quantum and classical tasks to their corresponding brokers. For quantum tasks, the key steps of each stage in the scheduling process are as follows:

5.1.1 | Backend selection

QBroker communicates with the quantum data center to gather information about the current environment, combining with all features of the incoming tasks to determine which QNode is the most appropriate backend for executing each QTask. This procedure comprises two main steps:

1. ***Pre-selection***: First, QBroker performs this process to select all potential QNodes $\mathcal{Q}_{\mathcal{T}}$, which satisfies the strict requirements of executing tasks to reduce the overhead for the backend selection. Several prerequisites to place a QTask γ_i to a potential QNode $q_j \in \mathcal{Q}_{\mathcal{T}}$ are considered, including:
   a. The qubit number of a potential QNode (q_j^w) must be equal to or higher than the required qubit number (or circuit width) of QTask (γ_i^w), denoted as $q_j^w \geq \gamma_i^w$. At this stage, we assume that a quantum circuit cannot be split and executed on different QNodes. It is important to acknowledge that techniques such as circuit cutting,[54] are in the early stages of development and can be considered in the future.
   b. All quantum gates used in QTask (γ_i^g) need to be supported by the gate sets of the QNode (q_T^g), denoted as $\gamma_i^g \subseteq q_j^g$. Otherwise, the transpiler can be used to convert unsupported gates[55] to the native gate set of QNode.
   c. It is possible to find a mapping (ω_{ij}) of the QTask's qubit connectivity (γ_i^t) and the qubit topology of the QNode (q_j^t). iQuantum also allows users to model and evaluate their qubit mapping strategy or use the default backtracking-based qubit mapping policy.

Algorithm 1. QTask scheduling simulation logic

Input : All incoming QTasks: $\Gamma = [\gamma_1, \gamma_2, ..., \gamma_n]$
All available QNodes: $\mathcal{Q} = [q_1, q_2, ..., q_m]$
Backend Selection Policy: π_b
Output: List of placements: $\Sigma = [\sigma_1, \sigma_2, ..., \sigma_n]$

Initialize simulation clock $t \leftarrow 0$;
$\Sigma \leftarrow []$;
for $i \leftarrow 1$ *to* n **do**
  Update t & all QNodes processing time;
  $\gamma_i \leftarrow \Gamma[i]$;
  $q_T \leftarrow$ null;
  $\mathcal{Q}_\mathcal{T} \leftarrow$ null;
  for $j \leftarrow 1$ *to* m **do**
    $q_j \leftarrow \mathcal{Q}[j]$;
    if $q_j^w \geq \gamma_i^w$ *and* $\gamma_i^g \subseteq q_j^g$ **then**
      $\omega_{ij} \leftarrow$ qubitMapping(γ_i^t, q_j^t);
      if $\omega_{ij} \neq$ *null* **then**
        Update QTask γ_i;
        Append q_j to $\mathcal{Q}_\mathcal{T}$;
      end if
    end if
  end for
  if $\mathcal{Q}_\mathcal{T} \neq$ *null* **then**
    $q_T \leftarrow \pi_b(\mathcal{Q}_\mathcal{T}, \gamma_i)$;
  end if
  $\sigma_i \leftarrow \{\gamma_i, q_T\}$;
  Append σ_i to Σ;
end for
return Σ;
Distribute QTasks according to Σ;
Perform QNode local scheduling process;

Other constraints, such as cost, quantum volume, and quality of services (QoS), can also be considered. If all prerequisites are satisfied, the QNode will be added to the potential QNodes $\mathcal{Q}_\mathcal{T}$ for the backend selection.

2. ***Selection***: QBroker can then apply the backend selection policy (π_b) to select the most suitable QNode from the pre-selected list in the previous step. This policy is driven by different objectives and constraints defined by users.

5.1.2 | Task allocation

After the backend selection procedure, QDatacenter will place each QTask to the selected QNode for execution. All arrived tasks are place in the queue at QNode and will be executed based on the QNode's task allocation policy. Different strategies for task allocation can be designed to schedule multiple incoming quantum tasks in the appropriate order to minimize the execution time or achieve other resource utilization objectives.

For the classical computing domain, there are two main policies for allocating multiple tasks for execution in classical computation resources, including the Space-shared and Time-shared policies.[23] Examples of comparable strategies for quantum tasks are illustrated in Figure 5. We assume there is a 7-qubit QNode and 5 QTasks (arrived in order from q1 to q5) in the waiting queue. The width and height of each QTask represent its number of qubits and number of circuit layers, respectively.

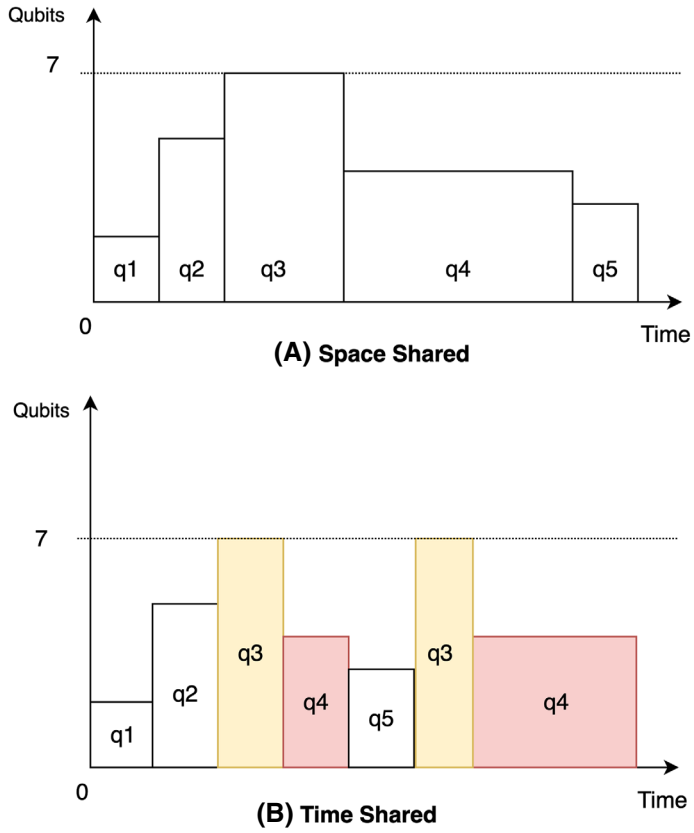

FIGURE 5 Example of two allocation strategies for five different quantum tasks (sequenced as q1 to q5) within a seven-qubit quantum node. (A) The Space Shared policy dedicates separate quantum resources to each task. (B) Time Shared policy allocates tasks in even time slots using a round-robin approach, resulting in pausing and resuming of tasks q3 and q4, rendering it impractical for quantum computing.

1. In space-shared policy, quantum resources (qubits) are allocated exclusively to each task for the entire duration of its execution without any time-sharing. The remaining tasks must wait for available resources if other tasks occupy current resources.
2. In the time-shared policy, the available resources can be divided into time slots and allocated to tasks or users in a round-robin fashion. Each task is assigned resources for a fixed time interval, after which the resources are reassigned to the next task in the queue.

It is important to highlight that the time-share approach is impractical for quantum task execution at the moment. It is mainly because a quantum task cannot be suspended and then resume its previous quantum state to continue executing in the future due to the no-cloning theorem of quantum computing. Therefore, the space-shared policy is suggested to be used as default in iQuantum for task allocation at QNode. In the future, we anticipate that multiple quantum tasks can be parallel executed in different areas of the qubit topology in a quantum computer. However, it is challenging to realize this approach in the current NISQ era[20] as it can be affected by noise and complicated qubit reset control.

5.2 | Hybrid quantum task orchestration

Quantum computing is predicted to enhance classical computing rather than replace it entirely. It can handle tasks that are best suited to its distinctive features. In this way, a hybrid quantum-classical computing system can be a potential paradigm to maximize the capabilities of both quantum and classical systems during the NISQ era.[56] This approach utilizes the advantages of classical computing for tasks such as data pre-processing and post-processing while assigning more computationally demanding tasks to quantum resources.[57]

Algorithm 2. Hybrid task orchestration logic

Input : All incoming tasks: $\Lambda = [\lambda_1, \lambda_2, \dots, \lambda_n]$
QNodes: $\mathcal{Q} = [q_1^c, q_2^c, \dots, q_m^c, q_1^e, q_2^e, \dots, q_k^e]$
CNodes: $\mathcal{C} = [c_1^c, c_2^c, \dots, c_p^c, c_1^e, c_2^e, \dots, c_l^e]$
Backend Selection Policies: π_Q, π_C
All VM/Container Provision Policies
Output: List of placements: $\Sigma = [\sigma_1, \sigma_2, \dots, \sigma_n]$

1 Initialize simulation clock $t \leftarrow 0$;
2 $C_V^e, C_V^c \leftarrow$ Apply VM/Container Provision Policies;
3 Submit all tasks to Edge gateway;
4 $\Sigma \leftarrow []$;
5 **for** $i \leftarrow 1$ *to* n **do**
6 Update t & all processing time;
7 $\lambda_i \leftarrow \Lambda[i]$;
8 $b_T \leftarrow$ null;
9 **if** $\lambda_i.type = QTask$ **then**
10 Send λ_i to QEBroker;
11 Apply QTask scheduling policy *(Algo. 1)*
12 $b_T \leftarrow \pi_Q(\lambda_i, \mathcal{Q}^e)$;
13 **else**
14 Send λ_i to CEBroker;
15 Apply CTask scheduling policy;
16 $b_T \leftarrow \pi_C(\lambda_i, C_V^e)$;
17 **end if**
18 $\sigma_i \leftarrow \{\lambda_i, b_T\}$;
19 **if** $\sigma_i \leftarrow$ ***false*** **then**
20 Offload λ_i to Cloud Gateway;
21 $\sigma_i \leftarrow$ Repeat 8–18 on Cloud resources;
22 **end if**
23 Append σ_i to Σ;
24 **end for**
25 **return** Σ;
26 Distribute tasks according to Σ;
27 Perform local scheduling process at nodes;

iQuantum leverages the classical entities in CloudSim to support modeling classical resources and its local resource management policies within the classical data center. Additionally, user can design their policy for orchestrating quantum and classical tasks from the Gateway to the appropriate broker. The hybrid task orchestration logic is depicted in Algorithm 2. It is important to note that different resource provisioning can be applied to classical servers (CNodes) to allocate logical resource units such as virtual machines (VMs) and containers. However, the equivalent technique is not yet available for quantum resources as QTasks are executed directly in quantum nodes.

After the environment setup, all tasks can be submitted directly to the closest Edge gateway. Then, Edge Gateway classifies and dispatches each task to Quantum Edge Broker (QEBroker) or Classical Edge Broker (CEBroker) for the scheduling process. Users can design different scheduling policies for classical and quantum tasks to achieve their objectives. For quantum tasks, the complete scheduling process can be done as described in Algorithm 1 (Section 5.1). For classical tasks, more discussion about task scheduling and resource management policies can be found in Reference 23.

Additionally, the task offloading and migration problems can be modeled in iQuantum in case the computation resources at the Edge layer are insufficient for executing the incoming tasks. For example, a quantum node at the Edge layer does not have enough qubits for executing quantum tasks as edge nodes have limited capacity compared to the cloud nodes. In this case, the corresponding broker will offload these failed tasks from the edge to the cloud gateway.

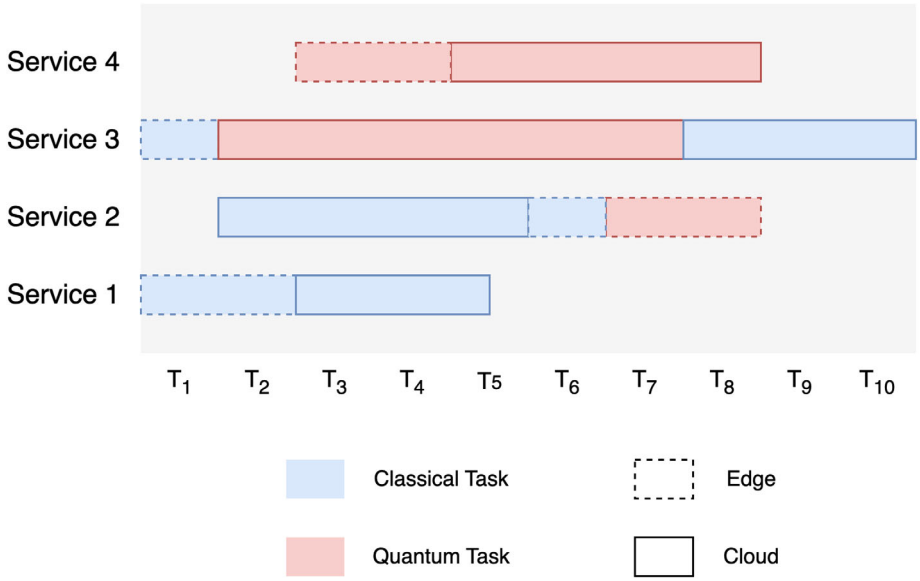

FIGURE 6 An example of hybrid task orchestration in the Cloud-Edge continuum. Four services incorporate different quantum and classical tasks, varying in execution time and resource requirements. Each task can be executed on edge computing resources or offloaded to the cloud if edge resources are insufficient for execution.

Accordingly, a similar orchestration process can be performed at the cloud layer to distribute all arrived tasks to the most suitable computation node.

An example of hybrid task orchestration in the cloud-edge continuum is illustrated in Figure 6. We assume four hybrid services (or applications) that need to be executed. Each service (or application) can contain multiple quantum and classical tasks that require different computation resources. Services 1 and 4 contain only classical tasks or quantum tasks, in which the smaller task can be sent to executed using edge resources, and the larger one can be offloaded to the cloud. Services 2 and 3 contain both quantum and classical tasks, which can be sent to their most appropriate computation node for execution. If the resource at the edge layer is sufficient for the incoming task, it can be used to place the task to reduce the communication latency and overhead for the cloud. More complex scenarios can be modeled within our toolkit. When iQuantum is used with CloudSim/iFogSim, one can model and simulate a hybrid quantum-classical computing environment and evaluate new hybrid task orchestration algorithms.

6 | EXAMPLE OF SIMULATION WORKFLOW

Figure 7 depicts an overview of the simulation workflow in iQuantum, which consists of four main stages.

First, the user needs to define the scenario manually or automatically using the dataset generator and importer. Then, when the simulation is initialized, it will first set up the environment with all pre-defined entities. Next, it executes all resource management policies, such as provisioning, scheduling, migration, and orchestration of all quantum and classical tasks. Finally, when the simulation stop condition is met, the simulation will be terminated, and the outcome will be generated in CSV format for further analysis.

To demonstrate the main steps of setting up a simulation in iQuantum, we present an explanatory example as Figure 8. This example is mainly derived from our earlier proposal[14] with improvement in the declaration. A quantum datacenter with a single 7-qubit QNode follows the configuration of *ibmq_oslo* node from IBM Quantum (QV 32, CLOPS 2,600). This QNode receives two quantum tasks, QTask 1 and QTask 2, with 4 qubits and 3 qubits, respectively. Both QTasks employed 3 basic gates (CX, RZ, X), which are fully supported by the quantum node. QTask 1 comprises 100 circuit layers and will execute 4000 shots, while the metrics for QTask 2 are 50 layers and 1000 shots.

Step 1. Initialize the iQuantum core simulation entity by using iquantum.init() method.

Step 2. Create a list of QNode instances. Users can model manually similar to Reference 14, or automatically by using the predefined QNode and adding to qNodeList as follows (Code 1):

```
QNode qNode = IBMQNode.createNode(id,"ibm_oslo",new QTaskSchedulerSpaceShared());
qNodeList = new ArrayList<QNode>();
qNodeList.add(qNodeOslo);
```

Code 1: Sample code for importing ibmq_oslo QNode

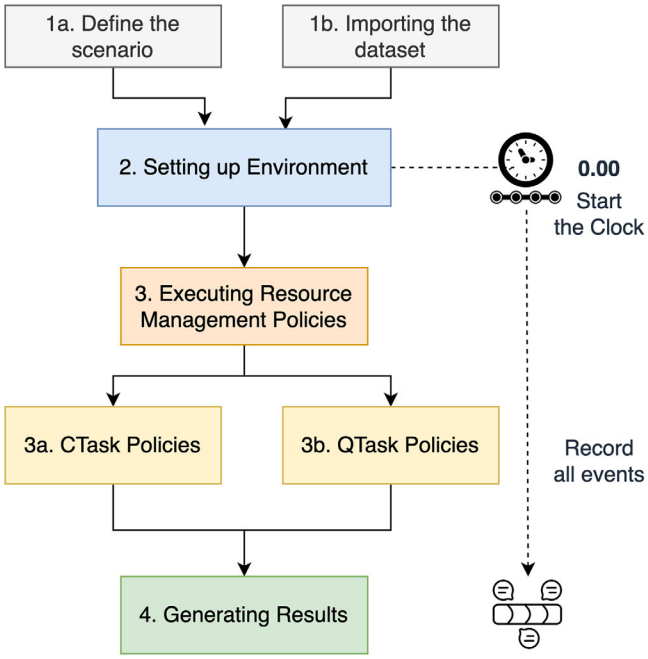

FIGURE 7 Overview of the simulation workflow in iQuantum.

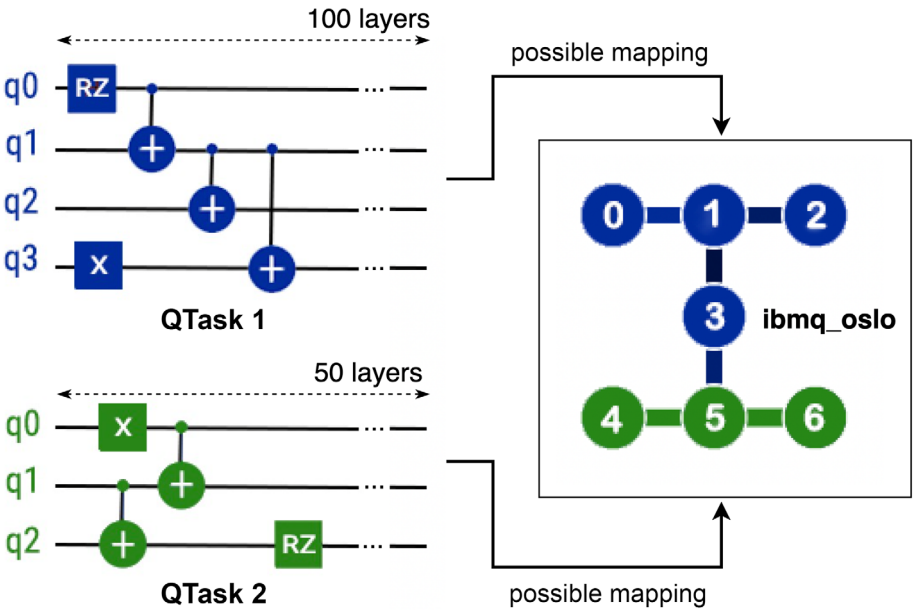

FIGURE 8 Example of two QTasks (left) and possible qubit mapping into ibmq_oslo node (right).

Step 3: Create a QCloudDatacenter. qNodeList and other information, such as cost of execution in the quantum datacenter are modeled in a QDatacenterCharacteristics object (Code 2).

```
QDatacenterCharacteristics characteristics = new QDatacenterCharacteristics(qNodeList, timeZone,
 costPerSec);
QCloudDatacenter qDatacenter = new QCloudDatacenter("QCDatacenter", characteristics);
```

Code 2: Sample code for modeling a QDatacenter

Step 4: Create a CloudGateway and a linked QCloudBroker object as follows (Code 3):

```
QCloudBroker qBroker = createQBroker();
        CloudGateway cloudGateway = new CloudGateway("CloudGateway_0", qBroker);
```

Code 3: Sample code for modeling a broker and a gateway

Step 5: Create a list of quantum tasks (QTask) manually or import the QTask dataset automatically from the CSV file. As this example only consists of two QTasks, they can be modeled manually as follows (Code 4):

```
List<int[]> q1Edges = new ArrayList<>();
q1Edges.add(new int[]{0, 1});
q1Edges.add(new int[]{1, 2});
... [truncated]
QubitTopology q2Tpl = new QubitTopology(3, q2Edges);
ArrayList<String> qg = new ArrayList<>(Arrays.asList("CX", "RZ", "X"));
QTask qtask1 = new QTask(0, 5, 100, 4000, qg, q1Tpl);
QTask qtask2 = new QTask(1, 3, 50, 1000, qg, q2Tpl);
```

Code 4: Sample code for modeling two QTasks

Step 6: Design and implement the resource management policies. For demonstration, we implemented a simple QTaskSchedulerSpaceShare policy by extending the QTaskScheduler class. We estimate the approximate completion time (t^q) of a quantum task inside a quantum node by using the following equation:

$$t^q = \frac{\gamma^d}{q^s} \times \gamma^s \tag{8}$$

where γ^d is the number of circuit layers in the quantum task, q^s is the CLOPS of the quantum node, and γ^s is the number of shots that quantum task need to be executed. It is important to note that other factors, such as transmission time and classical runtime, may be considered to estimate the total completion time. More details about quantum task scheduling are discussed in Section 5.1.

Step 5: Submit all quantum tasks to CloudGateway and start the simulation. Once all the simulation tasks are complete, stop the simulation and print out the final outcome (Code 5).

```
cloudGateway.submitQTasks(qTaskList);
iQuantum.startSimulation();
iquantum.stopSimulation();
List<QTask> rs = qBroker.getQTaskReceivedList();
QTaskExporter.printQTaskList(rs);
```

Code 5: Sample code for starting the simulation and print out the result

The simulator shows all events (with timestamps) happening during the simulation period (Code 6).

```
0.0: CloudGateway : Dispatching 0 CTasks and 1 QTasks from Cloud Gateway to Brokers for processing
0.0: QCBroker: Cloud Resource List received with 1 resource(s)
0.01: QCBroker: Started scheduling all QTasks to QDatacenter
0.01: QBroker: Checking if QNode #0 has enough qubits/gates to execute QTask0
....
153.86: QBroker: QTask 0 result received
173.09: QBroker: QTask 1 result received
173.09: QBroker: All QTasks executed. Finishing...
173.09: Simulation: No more future events
```

Code 6: Sample events in the simulation

The simulation results show that two quantum tasks are submitted to the QNode at timestamp $t = 0.01$ s (minimum interval between two different events). The execution times of QTask 1 and QTask 2 in the QNode are 153.85 and 19.23 s, respectively. According to the Space-shared scheduling policy, QTask 2 can only be executed after QTask 1 finishes its execution. The total execution time of all quantum tasks is 173.08 s. As noted above, these results are the same as when we manually calculated using Equation (8). More large-scale simulation scenarios and evaluation of iQuantum are discussed in the following section.

7 | VERIFICATION AND PERFORMANCE EVALUATION

In this section, we conduct a comprehensive verification and evaluation of iQuantum in different scenarios by using a well-known benchmarking quantum workload dataset to verify the accuracy of the conceptual model's implementation and its performance with the proposed use cases discussed in Section 5. The primary findings of the empirical experiments are discussed in this section.

7.1 | iQuantum simulation verification

Simulation is a cost-effective and less complex alternative to empirical experimentation for understanding real-world systems.[58] However, due to the high complexity of real-world systems, simulators often employ assumptions and abstractions, which can introduce certain inaccuracies while simplifying the model. Therefore, a key aspect of simulation studies is ensuring that these models maintain acceptable accuracy levels, considering their assumptions and abstractions. This involves two critical processes: validation, which assesses if the conceptual model is a true representation of the real world, and verification, which ensures the correct implementation of the model.[27]

For the system model validation, the conceptual model and metrics in quantum are devised based on well-known studies on quantum benchmarking[32,51,59,60] without further modification. Besides, the validation and verification of classical components had been studied in CloudSim.[23] Thus, the quantum system model validation and classical system models are out of the scope of this study. The rest of this section demonstrates a verification to ensure the correctness

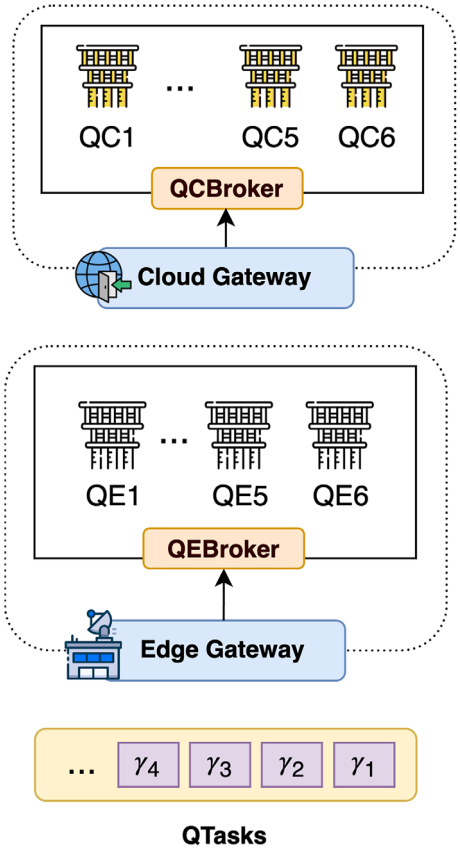

FIGURE 9 Overview of the infrastructure considered in iQuantum's verification and performance evaluation.

of iQuantum implementation for quantum-based features before conducting a large-scale evaluation, following the verification approach of other modeling and simulation studies, such as References 27,58.

7.1.1 | Scenario description

We use the quantum cloud-edge infrastructure as depicted in Figure 9 to comprehend the proposed use cases along with the correctness verification of iQuantum's implementation. Detailed attributes of heterogeneous QNodes, which are adopted from IBM Quantum,[13] are depicted in Table 2. The edge layer consists of a cluster of five quantum nodes, modeled according to information of different 27-qubit IBM Quantum Systems, including *ibm_hanoi, ibm_auckland, ibm_cairo, ibmq_mumbai*, and *ibmq_kolkata*. The cloud layer has a datacenter of 6 QNodes, following the 127-qubit topology of the *ibm_washington* system with different metrics for QV and CLOPS to demonstrate the heterogeneity of the quantum cloud environment.

To facilitate the verification, we use four QTasks that represent different quantum applications extracted from the MQT Bench dataset,[34] in which the circuit attributes are shown in Table 3. All QTasks are initially submitted to the edge gateway for processing. Following the proposed system model and use cases (Section 5), QTask γ_2 (55 qubits) is expected to be offloaded to the quantum cloud layer while the remaining QTasks can be executed at the quantum edge layers. The offloading time between two layers is set to 0.01 s. We also assume that γ_3 and γ_4 are to be scheduled to the same QNode (QE1) to verify the correctness of the proposed space-share task allocation within a QNode of iQuantum.

7.1.2 | Verification discussion

The dynamics of all events that occurred in the simulation of our verification are illustrated in Figure 10 and Table 4, following the similar verification approach of the EdgeSimPy simulator study.[27] At time step $T_1 = 0.01$ s, all QTasks are

TABLE 2 QNodes characteristics for the hybrid quantum environments.

Layer	ID	Qubit model	Qubits	QV	CLOPS
Edge	QE1	ibm_hanoi	27	64	2300
	QE2	ibm_auckland	27	64	2400
	QE3	ibm_cairo	27	32	2400
	QE4	ibmq_mumbai	27	128	1800
	QE5	ibmq_kolkata	27	128	2000
Cloud	QC1-3	ibm_washington	127	64	904
	QC4-6	ibm_washington	127	128	850

TABLE 3 Attributes of QTasks in the iQuantum verification derived from the MQT bench dataset.[34]

QTask	Circuit depth	Initial qubits	Mapped qubits	Application name
γ_1	326	7	27	pricingput
γ_2	74	55	127	graphstate
γ_3	339	8	27	portfoliovqe
γ_4	576	14	27	groundstate

scheduled at the quantum edge layer for execution. However, as all QNodes at the edge layer can only process QTasks up to 27 qubits, QTask γ_2 (55 qubits) needs to be offloaded to the cloud with more powerful QNodes for processing. Besides, γ_1 and γ_3 start the processing at QE5 and QE1, respectively, while γ_4 need to wait at QE1 to be executed after γ_3 completion (following the Space-shared scheduling). At time step $T_2 = 0.02$ s, γ_2 arrived at the cloud layer and is scheduled to QNode QC1 for execution for 8.72 s, then finished at time step $T_3 = 8.74$ s. After the completion of γ_3 for 14.74 s, γ_4 starts its execution at QNode QE1 at time step $T_4 = 14.75$ s. The remaining QTasks, γ_1, and γ_4, continue their execution and finish at time step $T_5 = 16.32$ s and $T_6 = 39.8$ s, respectively. It is obvious that all discrete events that occurred during the verification are correct and meet the initial expectation with the given input.

7.2 | Performance evaluation

To validate the effectiveness of iQuantum with the proposed use cases (discussed in Section 5), we further evaluate its performance and correctness of the implementation in different large-scale scenarios.

7.2.1 | Environment setup

For the quantum cloud-edge system, we set up a similar infrastructure as the previous verification setting (see Figure 9 and Table 2) to further investigate the quantum cloud-edge architecture with large-scale scenarios and use cases. For modeling large-scale quantum task workloads, we used MQT Bench[34] and extracted features of quantum circuits for 28 different quantum algorithms into four sets as shown in Table 5. The original quantum circuits are mapped to IBM Quantum systems (27 qubits and 127 qubits). To simplify the circuit layer extraction, we assume that the number of circuit layers is equivalent to the circuit depth of each QTask. We employed the Lottery-based[61] Backend Selection algorithm for the quantum node selection and the Space-shared policy for QNode scheduling. In the backend selection policy, we use quantum volume (QV) and CLOPS with the same weight (0.5) to determine the number of tickets for each QNode (i.e., a QNode has a higher QV and CLOPS will have a higher chance to be selected). All workload data of each set are sent to the edge gateway for the orchestration. The experiments are conducted on a Ubuntu 22.04 virtual machine hosted by Melbourne Research Cloud with an 8-vCore CPU and 32 GB of RAM. Each evaluation is repeated 1000 times, with the results discussed in the following part.

FIGURE 10 Dynamics of each time step involved in the verification process of iQuantum. The cloud layer in time steps T_4, T_5, and T_6 are truncated for simplicity as there are no new events after T_3.

7.2.2 | Discussion

Figure 11 illustrates the peak memory (RAM) usage and average simulation time (wall-lock time) of all scenarios on different workload datasets measured by using time command in Ubuntu. As iQuantum is an event-based simulation toolkit, its resource usage is relatively low without simulating the quantum operation. For Sets 1 and 2 cases, all tasks are executed at the quantum edge layer as all quantum edge nodes have sufficient qubits for the execution. Each simulation only takes around 0.5 to 1 s and requires from 117.8 MB (Set1) to 241.6 MB (Set2) of RAM for processing. As all tasks in Set 3 and the majority of tasks in Set 4 (5569 tasks) require 127 qubits, they are offloaded from the edge layer to the cloud layer for execution during the simulation. The qubit mapping for 127-qubit tasks requires more memory, peaks at 910.95 MB of RAM, and takes about 8 s to process 7000 QTasks in the hybrid scenario (Set 4). Nevertheless, the average simulation time and resource consumption of iQuantum are lightweight to model and evaluate different resource management policies for quantum computing environments.

TABLE 4 Events and QNodes state during iQuantum's verification simulation.

Time step	QNode ID	QTask status		
		Waiting	Executing	Done
T_1	QE1	γ_4	γ_3	—
	QE5	—	γ_1	—
	QC1	—	—	—
T_2	QE1	γ_4	γ_3	—
	QE5	—	γ_1	—
	QC1	—	γ_2	—
T_3	QE1	γ_4	γ_3	—
	QE5	—	γ_1	—
	QC1	—	—	γ_2
T_4	QE1	—	γ_4	γ_3
	QE5	—	γ_1	—
	QC1	—	—	γ_2
T_5	QE1	—	γ_4	γ_3
	QE5	—	—	γ_1
	QC1	—	—	γ_2
T_6	QE1	—	—	γ_3, γ_4
	QE5	—	—	γ_1
	QC1	—	—	γ_2

TABLE 5 Quantum task workload features, extracted from MQT bench dataset[34] (Qiskit SDK).

Set	QTask count	Circuit depth	Initial qubits	Mapped qubits	Algorithms types
1	300	13–7682	10–27	27	25
2	1000	13–58,861	10–27	27	21
3	3500	10–87,833	7–127	127	28
4	7000	10–161,588	7–127	27–127	28

Figure 12 illustrates the heatmap distribution of all QTask execution on different QNodes. As we employed the lottery-based backend selection policy, which prioritizes the selection of QNode with a higher quantum volume and CLOPS, this heatmap reinforces the correctness of our implementation. In order to use quantum resources more effectively, it is important to develop an advanced resource management strategy. Users can use iQuantum to design and evaluate more advanced backend selection and task scheduling to optimize the resource management strategy.

Additionally, to draw insight into the simulation results to highlight the necessity of the iQuantum toolkit, we illustrate the average results of all QTasks completion time in Figure 13. QTasks completion time indicates the time elapsed for all QNodes to finish executing all QTasks, where different QNodes can execute different tasks at the same time. We also measure the sum of the total execution time from all quantum nodes in each scenario. As the space-shared scheduling policy limits a QNode can execute only one task at a time, and each QTask in our test has a large number of circuit layers, the total completion time, as well as the sum of QPU times, are quite considerable, especially for completing Sets 3 and 4. It requires around 1 h of execution in practical environments for completing all incoming tasks where the actual QPU times are 4.2 h (Set 3) up to 7 h (Set 4). If we consider the cost model of quantum computing providers, such as IBM Quantum, which charges each second of quantum execution at $1.6, the total cost for each set in the actual environment is enormous. Therefore, a simulated environment for the design and evaluation of quantum resource management policies is crucial. Since implementing quantum computing in practical settings can be costly, iQuantum offers a simplified

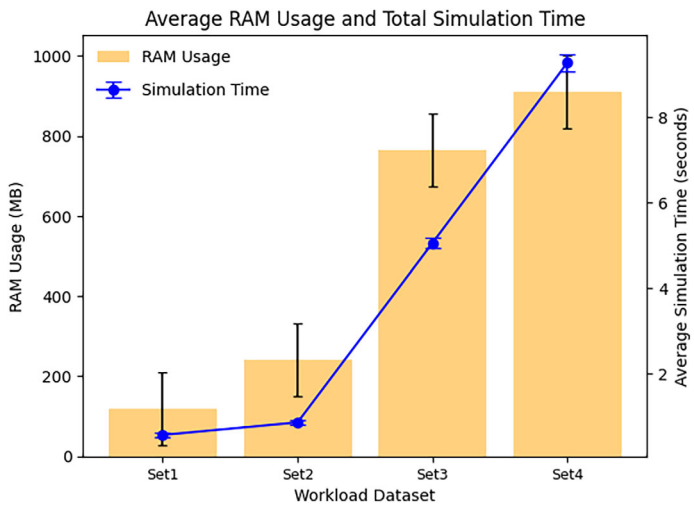

FIGURE 11 Average RAM usage (bar chart) and total simulation time (line chart) of iQuantum on four workload datasets (varying from 300 to 7000 QTasks) over 1000 iterations.

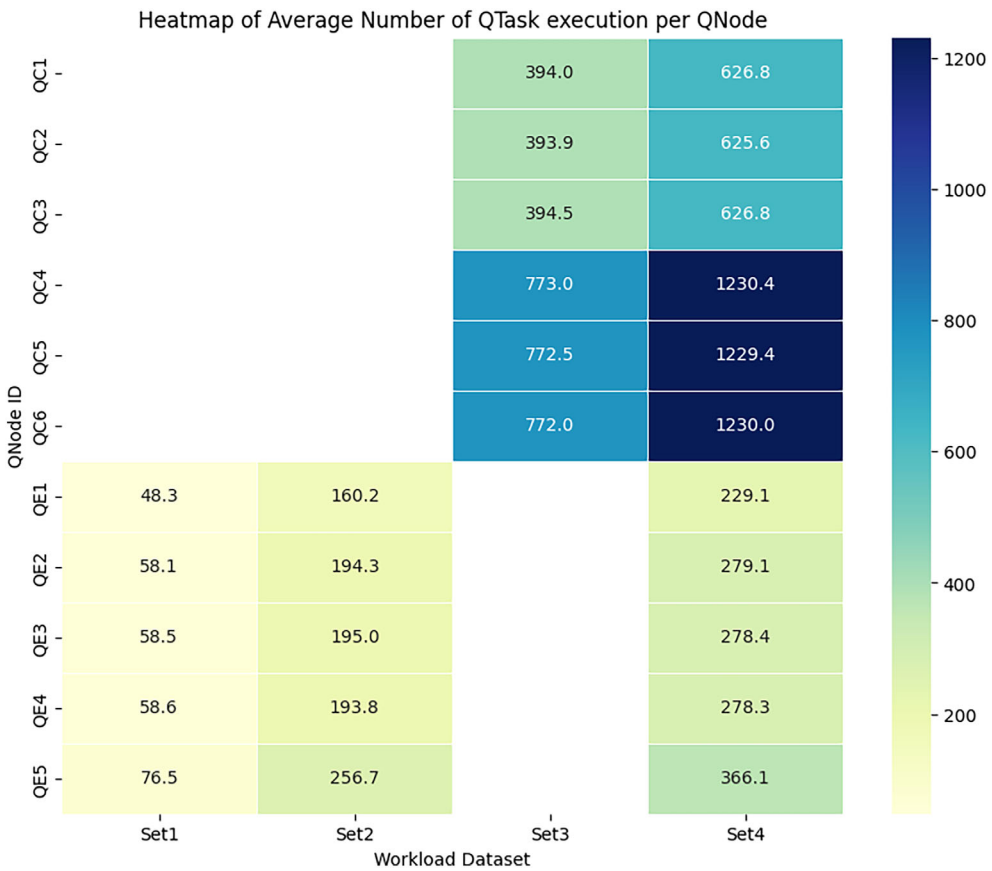

FIGURE 12 Distribution of the average number of QTasks execution per QNode in all scenarios. QC, quantum nodes at the cloud layer; QE, quantum nodes at the edge layer.

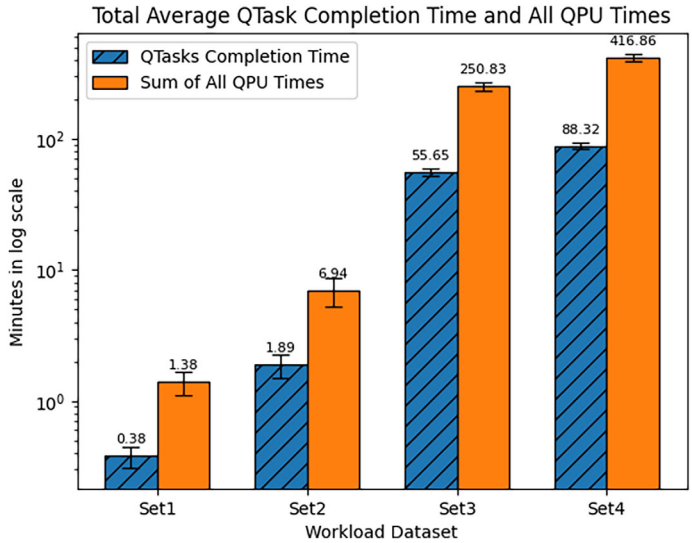

FIGURE 13 Total QTasks completion time (wall-lock time) and cumulative QPU times of all QNodes in minutes required for each workload set's completion. These figures are average values of 1000 iterations.

toolkit for modeling, designing, and assessing resource management policies without incurring massive waiting time or expenses.

8 | LESSON LEARNED AND DISCUSSION

Through the development and empirical evaluation of different use cases of iQuantum, we identify several insights that can be useful for research in quantum resource management and developing modeling and simulation toolkits for quantum computing environments. Performing experiments in practical environments for quantum computing is difficult and time-consuming, mainly due to the limited and costly quantum resources. The use of modeling tools like iQuantum is critical to accelerate research in system design and resource management. Furthermore, iQuantum can be used for educational purposes to help practitioners better understand the quantum system operation and performance.

In order to keep up with the latest advancements in quantum hardware and software, the modeling and simulation toolkit must be easily expandable and support new metrics. Quantum computing is constantly evolving, with new standards and metrics emerging during its development. Recently, metrics like quantum volume and CLOPS have gained popularity, and more may be added in the future. The simulator should be able to support modeling new metrics to reflect the comprehensive environment for testing more advanced resource management strategies. iQuantum allows users to customize and add more features as needed to fulfill their requirements. Users can advance resource management to adapt to current NISQ devices with more comprehensive aspects, such as error rates and connectivity of qubits in QNode. A quantum node in iQuantum can be modeled with detailed information on error rates in each qubit and their connections, which can serve these studies. Besides, different features of quantum circuits can be extracted from QASM files or using other quantum SDKs such as Qiskit[46] and Cirq[47] to the customized format in iQuantum.

Currently, the vision of quantum computing at the edge layer[15,16] is just a theoretical concept, but it may become a reality in the near future as quantum devices become more popular. Nevertheless, our paper also illustrated a simple hybrid model of quantum cloud-edge computing, where resources for quantum computing at the edge are more limited compared to cloud-based quantum computers. Future research should consider various aspects of this hybrid model, such as user mobility, service migration, and network communication. Additionally, we suggest expanding iQuantum to include further aspects and features, such as energy consumption management, network communication, and parallel processing of quantum tasks in multi-QPU quantum computers. It is worth noting that quantum nodes do not currently use virtualization or containerization techniques, unlike classical computing. Instead, they directly execute quantum tasks with the support of classical drivers for circuit compilation and transpilation. As a result, there is no equivalent

conceptual model for virtual machines (VMs) or containers in the quantum computing field at present. Besides, quantum datacenters can contain nodes that are either homogeneous or heterogeneous, depending on their physical properties and underlying technology. While quantum cloud providers offer access to their quantum simulators, these resources are only useful for testing during the NISQ era and not for long-term production phases. Consequently, we do not take these simulators into account for modeling purposes.

It is also important to mention that iQuantum also supports the model of different error rates of quantum nodes, which helps to understand quantum errors in a simulated environment and design more complex resource management strategies. However, the scope of our study does not include a comprehensive benchmarking analysis of the impact on different error rates. Users can consider these error rates and holistic quality metrics of a quantum system, such as quantum volume,[51,60] when designing task scheduling or resource management policies. Quantum error modeling and analysis, which delves into the specific impacts of varying error rates on quantum computation, represents a significant and complex undertaking that is actively developed in the domain of quantum hardware benchmarking,[59,62] and quantum error correction.[63]

In the current landscape of quantum computing, the development of large-scale quantum systems, exemplified by endeavors such as the IBM System Two with more than 1000 qubit quantum chips,[64] represents a significant step forward. Future work with iQuantum will focus on extending its capabilities to model and simulate such large-scale quantum systems. This undertaking will involve enhancing the framework's scalability and computational efficiency to represent and manage the complexities inherent in systems of this magnitude.

It would be beneficial to explore an improved method for measuring the number of circuit layers in quantum tasks. In our evaluation, we assumed that the circuit depth extracted from the original QASM files represented the circuit layers. However, it is important to note that circuit depth is only an approximate metric, and we recommend using a more precise method to extract the number of circuit layers in quantum tasks to adapt to the measurement of the CLOPS metric for quantum nodes. In the study on CLOPS metric benchmarking, a circuit layer was defined as one layer of permutation among qubits and one layer of pair-wise random SU(4) 2-qubit unitary gates. However, this circuit was designed specifically for CLOPS benchmarking, and a technique for estimating the number of circuit layers in a general circuit is still necessary. It is important to note that our work primarily pertains to the features of the circuit dataset used for the simulation, and the circuit layer measurement technique does not impact the core simulation logic of iQuantum. We emphasize the need for a standardized and large-scale quantum workload dataset to investigate further the development of advanced quantum resource management policies in the future. For example, the depth-1 circuit per second[32] metric, which is linearly scaling to CLOPS, can be considered in the future.

9 | CONCLUSIONS AND FUTURE WORK

Our paper introduces iQuantum, a lightweight and versatile toolkit for modeling and simulation of hybrid quantum computing environments. This toolkit focuses on creating and evaluating quantum resource management policies within the cloud-edge continuum. We have extensively developed iQuantum from an initial case proposal to a holistic toolkit that is flexible and adaptable for various use cases in quantum systems research. In addition to the iQuantum toolkit, we also propose a comprehensive system model for quantum computing environments, which serves as a baseline model for quantum entities in the hybrid cloud-edge paradigm. We demonstrate various use cases for iQuantum, including facilitating the design and evaluation of different scenarios in resource orchestration problems such as task scheduling, backend selection, and hybrid task orchestration. Moreover, we also validate and evaluate the performance to highlight the effectiveness of iQuantum using reliable datasets from MQT Bench[34] and IBM Quantum. The targeted users of iQuantum will be students, educators, researchers, and practitioners who want to comprehensively evaluate resourced management algorithms in simulated environments. Such proven algorithms can then be deployed in real quantum computing environments.

Our future work in iQuantum involves ensuring that we can adapt to new advances and standards in quantum hardware and software design. We aim to support the model of large-scale quantum chips with more than 1000 qubits and extend our modeling to analyze the impact of various quantum error rates. We are also considering extending iQuantum's capabilities to model the characteristics of various quantum hardware and quantum networks. We will also consider the modeling of forthcoming techniques, such as circuit-cutting, to allow the allocation of tasks to multiple quantum resources in the future.

AUTHOR CONTRIBUTIONS

Hoa T. Nguyen: Conceptualization, Methodology, Software, Validation, Investigation, Data Curation, Writing – Original Draft, Review & Editing, Visualization; **Muhammad Usman**: Supervision, Validation, Writing – Review & Editing; **Rajkumar Buyya**: Conceptualization, Methodology, Supervision, Validation, Writing – Review & Editing.

ACKNOWLEDGMENTS

This work is the substantial extended work of our 10-page conference paper,[14] which was presented at the 2023 IEEE International Conference on Quantum Software (QSW 2023), held in Chicago, USA. Hoa Nguyen acknowledges the support from the Science and Technology Scholarship Program for Overseas Study for Master's and Doctoral Degrees, Vingroup, Vietnam. Open access publishing facilitated by The University of Melbourne, as part of the Wiley - The University of Melbourne agreement via the Council of Australian University Librarians.

FUNDING INFORMATION

No funding was received for this manuscript

CONFLICT OF INTEREST STATEMENT

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are openly available in CLOUDS Lab's Github at https://github.com/Cloudslab/iQuantum. The iQuantum toolkit with the source code and examples of all proposed use cases can be accessed on our website (clouds.cis.unimelb.edu.au/iquantum) and Github (github.com/Cloudslab/iQuantum) as an open-source tool under the GPL-3.0 license.

ORCID

Hoa T. Nguyen https://orcid.org/0000-0001-6904-6312
Muhammad Usman https://orcid.org/0000-0003-3476-2348
Rajkumar Buyya https://orcid.org/0000-0001-9754-6496

REFERENCES

1. Zinner M, Dahlhausen F, Boehme P, Ehlers J, Bieske L, Fehring L. Quantum computing's potential for drug discovery: early stage industry dynamics. *Drug Discov Today*. 2021;26(7):1680-1688.
2. Griffin P, Sampat R. Quantum computing for supply chain finance. Proceedings of the 2021 IEEE International Conference on Services Computing (SCC), pages 456-459, Chicago, IL, USA. 2021.
3. Moll N, Barkoutsos P, Bishop LS, et al. Quantum optimization using variational algorithms on near-term quantum devices. *Quantum Sci Technol*. 2018;3(3):030503.
4. Debenedictis EP. A future with quantum machine learning. *Computer*. 2018;51(2):68-71.
5. West MT, Erfani SM, Leckie C, Sevior M, Hollenberg LCL, Usman M. Benchmarking adversarially robust quantum machine learning at scale. *Phys Rev Res*. 2023;5(2):023186.
6. West MT, Tsang S-L, Low JS, et al. Towards quantum enhanced adversarial robustness in machine learning. *Nat Mach Intell*. 2023;5(6):581-589.
7. Kaiiali M, Sezer S, Khalid A. Cloud computing in the quantum era. Paper presented at: 2019 IEEE Conference on Communications and Network Security (CNS), volume 2019, pages 1-4. IEEE. 2019.
8. Leymann F, Barzen J, Falkenthal M, Vietz D, Weder B, Wild K. Quantum in the cloud: application potentials and research opportunities. Proceedings of the 10th International Conference on Cloud Computing and Services Science, pages 9-24. SCITEPRESS-Science and Technology Publications. 2020.
9. Rahaman M, Islam M. A review on Progress and problems of quantum computing as a service (QCaaS) in the perspective of cloud computing. *Global J Comput Sci Technol: B Cloud Distrib*. 2015;15(4):23-26.
10. Gill SS, Kumar A, Singh H, et al. Quantum computing: a taxonomy, systematic review and future directions. *Softw Pract Exper*. 2022;52(1):66-114.
11. Microsoft. Azure Quantum. 2023.
12. Gonzalez C. Cloud based QC with Amazon Braket. *Digitale Welt*. 2021;5(2):14-17.
13. IBM. IBM Quantum Computing Services. 2023.

14. Nguyen HT, Usman M, Buyya R. iQuantum: a case for Modeling and simulation of quantum computing environments. Proceedings of the 2023 IEEE International Conference on Quantum Software (QSW), pages 21-30, Chicago, IL, USA, IEEE. 2023.
15. Ma L, Ding L. Hybrid quantum edge computing network. *Quantum Communications and Quantum Imaging XX*. SPIE; 2022:29.
16. Furutanpey A, Vienna T, Barzen J. Architectural vision for quantum computing in the edge-cloud continuum. Proceedings of the 2nd IEEE International Conference on Quantum Software (QSW), pages 88-103, Chicago, IL, USA, IEEE.
17. Wang H, Liu T, Kim B, et al. Architectural design alternatives based on cloud/edge/fog computing for connected vehicles. *IEEE Commun Surv Tutor*. 2020;22(4):2349-2377.
18. Shi W, Cao J, Zhang Q, Li Y, Xu L. Edge computing: vision and challenges. *IEEE Internet Things J*. 2016;3(5):637-646.
19. Qiu X, Zhang W, Chen W, Zheng Z. Distributed and collective deep reinforcement learning for computation offloading: a practical perspective. *IEEE Trans Parallel Distrib Syst*. 2021;32(5):1085-1101.
20. Preskill J. Quantum computing in the NISQ era and beyond. *Quantum*. 2018;2:79.
21. Ravi GS, Smith KN, Gokhale P, Chong FT. Quantum computing in the cloud: analyzing job and machine characteristics. Proceedings of the 2021 IEEE International Symposium on Workload Characterization (IISWC), pages 39-50, Storrs, CT, USA. 2021.
22. Nguyen HT, Usman M, Buyya R. QFaaS: a serverless function-as-a-service framework for quantum computing. *Future Gener Comput Syst*. 2024;154:281-300.
23. Calheiros RN, Ranjan R, Beloglazov A, De Rose CAF, Buyya R. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw Pract Exp*. 2011;41(1):23-50.
24. Sonmez C, Ozgovde A, Ersoy C. EdgeCloudSim: an environment for performance evaluation of edge computing systems: EdgeCloudSim: an environment for performance evaluation of edge computing systems. *Trans Emerg Telecommun Technol*. 2018;29(11):e3493.
25. Qayyum T, Malik AW, Khan Khattak MA, Khalid O, Khan SU. FogNetSim++: a toolkit for modeling and simulation of distributed fog environment. *IEEE Access*. 2018;6:63570-63583.
26. Mahmud R, Pallewatta S, Goudarzi M, Buyya R. iFogSim2: an extended iFogSim simulator for mobility, clustering, and microservice management in edge and fog computing environments. *J Syst Softw*. 2022;190(111351):1-17.
27. Souza PS, Ferreto T, Calheiros RN. EdgeSimPy: python-based modeling and simulation of edge computing resource management policies. *Future Gener Comput Syst*. 2023;148:446-459.
28. Altman E, Brown KR, Carleo G, et al. Quantum simulators: architectures and opportunities. *PRX Quantum*. 2021;2(1):017003.
29. Diadamo S, Notzel J, Zanger B, Bese MM. QuNetSim: a software framework for quantum networks. *IEEE Trans Quantum Eng*. 2021;2:1-12.
30. Coopmans T, Knegjens R, Dahlberg A, et al. NetSquid, a NETwork simulator for QUantum information using discrete events. *Commun Phys*. 2021;4(1):164.
31. Wehner S, Elkouss D, Hanson R. Quantum internet: a vision for the road ahead. *Science*. 2018;362(6412):p.eaam9288.
32. Wack A, Paik H, Javadi-Abhari A, et al. Quality, Speed, and Scale: three key attributes to measure the performance of near-term quantum computers. 2021 arXiv:2110.14108.
33. Dagkakis G, Heavey C. A review of open source discrete event simulation software for operations research. *J Simul*. 2016;10(3):193-206.
34. Quetschlich N, Burgholzer L, Wille R. MQT bench: benchmarking software and design automation tools for quantum computing. *Quantum*. 2023;7:1062.
35. Gubbi J, Buyya R, Marusic S, Palaniswami M. Internet of things (IoT): a vision, architectural elements, and future directions. *Future Gener Comput Syst*. 2013;29(7):1645-1660.
36. Puliafito C, Mingozzi E, Longo F, Puliafito A, Rana O. Fog computing for the internet of things: a survey. *ACM Trans Internet Technol*. 2019;19(2):1-41.
37. Laroui M, Nour B, Moungla H, Cherif MA, Afifi H, Guizani M. Edge and fog computing for iot: a survey on current research activities and future directions. *Comput Commun*. 2021;180:210-231.
38. Gupta H, Vahid Dastjerdi A, Ghosh SK, Buyya R. iFogSim: a toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. *Softw Pract Exp*. 2017;47(9):1275-1296.
39. Huang J, Kong L, Chen G, Cheng L, Wu K, Liu X. B-iot: Blockchain driven internet of things with credit-based consensus mechanism. Paper presented at: 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS). 2019 1348-1357.
40. Sun L, Jiang X, Ren H, Guo Y. Edge-cloud computing and artificial intelligence in internet of medical things: architecture, technology and application. *IEEE Access*. 2020;8:101079-101092.
41. Qu G, Cui N, Wu H, Li R, Ding Y. Chainfl: a simulation platform for joint federated learning and blockchain in edge/cloud computing environments. *IEEE Trans Industr Inform*. 2022;18(5):3572-3581.
42. Malik AW, Qayyum T, Rahman AU, Khan MA, Khalid O, Khan SU. Xfogsim: a distributed fog resource management framework for sustainable iot services. *IEEE Trans Sustain Comput*. 2021;6(4):691-702.
43. Jones T, Brown A, Bush I, Benjamin SC. QuEST and high performance simulation of quantum computers. *Sci Rep*. 2019;9(1):10736.
44. Bian H, Huang J, Tang J, Dong R, Wu L, Wang X. PAS: a new powerful and simple quantum computing simulator. *Softw Pract Exp*. 2023;53(1):142-159.
45. Brennan J, O'Riordan L, Hanley K, et al. QXTools: a Julia framework for distributed quantum circuit simulation. *J Open Source Softw*. 2022;7(70):3711.
46. Aleksandrowicz G, Alexander T, Barkoutsos P, et al. Qiskit: An Open-Source Framework for Quantum Computing. 2019.
47. Cirq Developers. Cirq Framework. 2023.
48. Dahlberg A, Wehner S. SimulaQron-a simulator for developing quantum internet software. *Quantum Sci Technol*. 2019;4(1):15.
49. Steiger DS, Häner T, Troyer M. ProjectQ: an open source software framework for quantum computing. *Quantum*. 2018;2:49.

50. Johansson J, Nation P, Nori F. QuTiP: an open-source python framework for the dynamics of open quantum systems. *Comput Phys Commun*. 2012;183(8):1760-1772.
51. Cross AW, Bishop LS, Sheldon S, Nation PD, Gambetta JM. Validating quantum computers using randomized model circuits. *Phys Rev A*. 2019;100(3):032328.
52. Riley GF, Henderson TR. The ns-3 network simulator. In: Wehrle K, Güneş M, Gross J, eds. *Modeling and Tools for Network Simulation*. Springer; 2010:15-34.
53. Varga A. OMNeT++. In: Wehrle K, Güneş M, Gross J, eds. *Modeling and Tools for Network Simulation*. Springer; 2010:35-59.
54. Tang W, Tomesh T, Suchara M, Larson J, Martonosi M. CutQC: using small quantum computers for large quantum circuit evaluations. Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, pages 473-486, Virtual USA. 2021.
55. Aravanis C, Korpas G, Marecek J. Transpiling quantum circuits using the pentagon equation. 2022 arXiv:2209.14356 [quant-ph].
56. Serrano MA, Cruz-Lemus JA, Pérez-Castillo R, Piattini M. Quantum software components and platforms: overview and quality assessment. *ACM Comput Surv*. 2022;55(8):1-31.
57. Weder B, Barzen J, Leymann F, Vietz D. Quantum software development lifecycle. In: Serrano MA, Pérez-Castillo R, Piattini M, eds. *Quantum Software Engineering*. Springer International Publishing; 2022:61-83.
58. Xiang X, Kennedy R, Madey G, Cabaniss S. Verification and validation of agent-based scientific simulation models. Paper presented at: Agent-Directed Simulation Conference, volume 47, page 55. The Society for Modeling and Simulation International San Diego, CA, USA. 2005.
59. Mills D, Sivarajah S, Scholten TL, Duncan R. Application-motivated, holistic benchmarking of a full quantum computing stack. *Quantum*. 2021;5:415.
60. Pelofske E, Bärtschi A, Eidenbenz S. Quantum volume in practice: what users can expect from NISQ devices. *IEEE Trans Quantum Eng*. 2022;3:1-19.
61. Waldspurger CA, Weihl WE. Lottery scheduling: flexible proportional-share resource management. Proceedings of the 1st USENIX Conference on Operating Systems Design and Implementation, pages 1-es.
62. Resch S, Karpuzcu UR. Benchmarking quantum computers and the impact of quantum noise. *ACM Comput Surv*. 2022;54(7):1-35.
63. Kim Y, Wood CJ, Yoder TJ, et al. Scalable error mitigation for noisy quantum circuits produces competitive expectation values. *Nat Phys*. 2023;19:752-759.
64. Castelvecchi D. Ibm releases first-ever 1000-qubit quantum chip. *Nature*. 2023;624(7991):238.

SUPPORTING INFORMATION

Additional supporting information can be found online in the Supporting Information section at the end of this article.

How to cite this article: Nguyen HT, Usman M, Buyya R. iQuantum: A toolkit for modeling and simulation of quantum computing environments. *Softw: Pract Exper*. 2024;1-31. doi: 10.1002/spe.3331