Serverless and Reinforcement Learning-based Resource Management for Quantum Computing

Thanh Hoa Nguyen

Submitted in total fulfilment of the requirements of the degree of

Doctor of Philosophy

School of Computing and Information Systems
THE UNIVERSITY OF MELBOURNE, AUSTRALIA

September 2025

ORCID: 0000-0001-6904-6312

Serverless and Reinforcement Learning-based Resource Management for Quantum Computing

Thanh Hoa Nguyen
Principal Supervisor: Prof. Rajkumar Buyya
Co-Supervisor: A/Prof. Muhammad Usman

Abstract

Quantum computing promises to address computationally intractable problems beyond the capabilities of classical computers in a variety of industrial sectors, such as drug discovery, finance, machine learning, and cybersecurity. Cloud-based quantum computing has emerged as a solution to the challenges of operating physical quantum systems. This paradigm democratizes access to remote quantum computers through cloud platforms, enabling the deployment of quantum applications. However, current quantum computing environments are characterized by heterogeneous quantum backends with varying capabilities, noise levels, and availability patterns. The limitations of Noisy Intermediate-Scale Quantum (NISQ) devices create distinct scheduling and orchestration complexities that are compounded by the hybrid nature of quantum-classical workflows, where quantum tasks must be seamlessly integrated with classical processing steps. Additionally, current quantum computing resources are limited in capacity and costly, requiring efficient resource utilization. These challenges are further exacerbated by the dynamic nature of quantum computing environments and the need to balance execution fidelity with time constraints in practical quantum applications. Hence, quantum cloud providers require adaptive quantum application deployment and sophisticated resource management techniques to harness the full potential of quantum computing tailored for application specific scenarios while accommodating the inherent limitations of current quantum hardware and software.

This thesis focuses on adaptive resource management solutions for quantum cloud computing environments by developing serverless architectures for seamless quantum application deployment and reinforcement learning-based techniques for efficient task orchestration with time-aware and fidelity-aware optimization. The research addresses key challenges, including quantum hardware heterogeneity, hybrid quantum-classical

workflow integration, dynamic backend selection, quantum task scheduling complexities, and the inherently noisy and time-sensitive nature of NISQ-era quantum systems. The thesis advances the state-of-the-art by making the following key contributions:

1. A comprehensive systematic mapping study and taxonomy of quantum cloud computing from different aspects, including service models, platforms, applications, resource management approaches, security, and privacy.

2. A holistic serverless quantum computing framework that enables seamless integration of quantum computation within classical cloud environments through Quantum Function-as-a-Service (QFaaS) architecture with adaptive backend selection, and cold start mitigation strategies.

3. A comprehensive modeling and discrete-event simulation framework for quantum computing environments that facilitates systematic evaluation of quantum resource management algorithms, incorporating realistic quantum system models and multi-use case support.

4. A novel deep reinforcement learning-based approach for time-aware quantum task placement using the Deep Q-Network technique to adapt to dynamic quantum cloud environments and optimize task completion efficiency.

5. A fidelity-aware quantum task orchestration framework using deep reinforcement learning that effectively balances execution fidelity and time constraints in NISQ-era quantum systems through noise-aware performance modeling and Proximal Policy Optimization approaches.

6. A detailed study outlining challenges and research directions for quantum cloud resource management, establishing foundational approaches for future research to advance quantum computing paradigms.

Declaration

This is to certify that

1. the thesis comprises only my original work towards the PhD,
2. due acknowledgement has been made in the text to all other material used,
3. the thesis is less than 100,000 words in length, exclusive of tables, maps, bibliographies and appendices.

Thanh Hoa Nguyen, September 2025

Preface

Main Contributions

This thesis research has been carried out in Quantum Cloud Computing and Distributed Systems (qCLOUDS) Laboratory, School of Computing and Information Systems, The University of Melbourne. The main contributions of the thesis are discussed in Chapters 2-7 and are based on the following publications:

- **Hoa T. Nguyen**, Prabhakar Krishnan, Dilip Krishnaswamy, Muhammad Usman, and Rajkumar Buyya, "Quantum Cloud Computing: A Review, Open Problems, and Future Directions", *submitted to ACM Computing Surveys (CSUR), April 2024* (Second Revision, available on arXiv:2404.11420).

- **Hoa T. Nguyen**, Muhammad Usman, and Rajkumar Buyya, "QFaaS: A Serverless Function-as-a-Service Framework for Quantum Computing", *Future Generation Computer Systems (FGCS)*, Volume 154, Pages: 281-300, ISSN: 0167-739X, Elsevier Press, Amsterdam, The Netherlands, May 2024.

- **Hoa T. Nguyen**, Muhammad Usman, and Rajkumar Buyya, "iQuantum: A toolkit for modeling and simulation of quantum computing environments", *Software: Practice and Experience (SPE)*, Volume 54, Issue 6, Pages: 1141-1171, ISSN: 0038-0644, Wiley Press, New York, USA, June 2024.

- **Hoa T. Nguyen**, Muhammad Usman, and Rajkumar Buyya, "DRLQ: A Deep Reinforcement Learning-based Task Placement for Quantum Cloud Computing", *Proceedings of the 17th IEEE International Conference on Cloud Computing (CLOUD 2024)*, Pages: 475-481, Shenzhen, China, July 7-13, 2024.

- **Hoa T. Nguyen**, Muhammad Usman, and Rajkumar Buyya, "QFOR: A Fidelity-aware Orchestrator for Quantum Computing Environments using Deep Reinforcement Learning", *submitted to the ACM Transactions on Quantum Computing (TQC), August 2025* (Under Review, available on arXiv:2508.04974).

Supplementary Contributions

During the Ph.D. candidature, I have also contributed to the following supplemental work (this thesis does not claim those as its main contributions):

- **Hoa T. Nguyen**, Muhammad Usman, and Rajkumar Buyya, "iQuantum: A Case for Modeling and Simulation of Quantum Computing Environments", *Proceedings of the 2nd IEEE International Conference on Quantum Software (QSW 2023)*, Chicago, USA, July 2-8, 2023.

- **Hoa T. Nguyen**, Muhammad Usman, and Rajkumar Buyya, "QSimPy: A Learning-centric Simulation Framework for Quantum Cloud Resource Management", *Quantum Computing*, Pages: 165-183, R. Buyya and S. Gill (eds), ISBN: 978-0-443-29096-1, Morgan Kaufmann Press, Cambridge, MA 02139, USA, July 2025.

- **Hoa T. Nguyen**, Bui Binh An Pham, Muhammad Usman, Rajkumar Buyya, "Quantum Serverless Paradigm and Application Development using the QFaaS Framework", *Quantum Computing*, Pages: 139-164, R. Buyya and S. Gill (eds), ISBN: 978-0-443-29096-1, Morgan Kaufmann Press, Cambridge, MA 02139, USA, July 2025.

Acknowledgements

Completing this PhD has been the most challenging and transformative journey of my life, and it would not have been possible without the extraordinary people who supported, guided, and inspired me along the way over the past four years.

First, I am profoundly grateful to my principal supervisor, *Prof Rajkumar Buyya*. From the very first day, you placed trust in me, even when the path ahead was uncertain. Thank you for encouraging me to leap into quantum computing, a field I never imagined I would enter four years ago, and for guiding me with wisdom, patience, and unwavering belief in my potential. Your technical guidance, thoughtful feedback on each of my research projects, and willingness to dive into the details with me made all the difference. I am also especially thankful to my co-supervisor, *A/Prof Muhammad Usman*, whose expertise in quantum computing has been invaluable. The supervision and mentorship from both of you have shaped not only my research but also my way of scientific thinking, my resilience, and my vision for the future of quantum research. Besides, I sincerely thank international collaborators, especially *Prabhakar Krishnan and Dilip Krishnaswamy*, for their contribution to our effort on taxonomy and identification of new research directions in quantum cloud computing.

To my colleagues and friends in the *Quantum Cloud Computing and Distributed Systems (qCLOUDS) Laboratory*, you have been invaluable companions, sharing in both the successes and the difficulties of this journey. Our meetings, lunches, dinners, video filming, bubble tea and coffee breaks, late-night working together, and shared laughter made even the hardest deadlines feel lighter. I am especially grateful to *Dr Mohammad Goudarzi* as my mentor during the very first days at the lab. A special thank you to my labmates who have worked alongside and shared this journey with me, especially

Duneesha Fernando, Ming Chen, Qifan Deng, Siddharth Agarwal, Tharindu Bandara, Zhiyu Wang, Yulun Huang, TianYu Qi, Murtaza Rangwala, Prabhjot Singh, Haoyu Bai, and all other members. Thank you very much for your generosity with time and encouragement.

To my friends and my communities in Melbourne, Vietnam, and beyond, you have been my anchor. Whether through a quick message to check in, a weekend meal, retreat to recharge, or a shared adventure that reminded me there is life outside of research, you kept me grounded and reminded me that I was never alone. I extend my appreciation to all *Vingroup Scholars* in Melbourne, all members of *IO Scholar* and *QuantumVIO* teams, and everyone who has followed and supported my journey. I am also grateful to my colleagues, mentors, mentees, and students at the *University of Melbourne, University of Information Technology - VNU-HCM, QWorld, QOSF,* and the *Qiskit Global Summer Schools* over the years. I also gratefully acknowledge the support from the *Vingroup Scholarship Program for Overseas Study for Doctoral Degrees* and *the ARC Discovery Project*, which made this research journey possible.

Above all, I owe my deepest thanks to my family, especially my mother, father, and my grandmother. To my mother, thank you so much for your steadfast support and for standing by me through every decision, even across distances. To my father and grandmother, although you are no longer here to witness this milestone, your spirit and guidance continue to shape my life. To my brother and sister, thank you for caring for our mother throughout my journey, and I hope you continue to pursue your own paths with determination and excellence. Finally, to my best friend, *Trong Nhan Ly*, thank you for being a constant presence over the past 15 years, for pushing me to start this PhD journey, for believing in me, and for supporting me through every challenge and distance. Your support has meant more than words can ever capture.

This thesis is not just a record of my research, it is a testament to the collective support of everyone mentioned here, and many more who may not be named but have left an indelible mark on my life. *To all of you, from the bottom of my heart, thank you.*

Thanh Hoa Nguyen
September 2025, Melbourne, Australia

To my mentors, family and loved ones, whose unwavering support and encouragement made this journey possible

Contents

List of Figures

List of Tables

List of Acronyms

API Application Programming Interface

CI/CD Continuous Integration / Continuous Deployment

CLOPS Circuit Layer Operations Per Second

CNode Classical Computation Node

CPU Central Processing Unit

CTask Classical Task

DQN Deep Q-Network

DRL Deep Reinforcement Learning

FaaS Function-as-a-Service

NISQ Noisy Intermediate-Scale Quantum

PPO Proximal Policy Optimization

QASM Quantum Assembly Language

QCaaS Quantum Computing-as-a-Service

QML Quantum Machine Learning

QNode Quantum Computation Node

QPU Quantum Processing Unit

QTask Quantum Task

QV Quantum Volume

SDK Software Development Kit

Chapter 1
Introduction

Quantum computing is anticipated to be a transformative computational paradigm that leverages quantum mechanical phenomena, superposition, entanglement, and interference, to solve problems that are computationally intractable for classical computation systems. Foundational algorithms by Shor [2] for integer factorization and Grover [3] for unstructured database search demonstrate theoretical quantum advantage while near-term quantum algorithms, such as the Quantum Approximate Optimization Algorithm (QAOA) [4], Variational Quantum Eigensolver (VQE) [5], and quantum machine learning approaches [6, 7], extend this potential to practical applications across diverse industry domains including finances, pharmaceutical, cybersecurity, and optimization [8]. The global quantum computing market demonstrates a substantial growth trajectory, projected to expand from $1.6 billion in 2025 to $7.3 billion by 2030, representing a compound annual growth rate of 34.6% [9]. The quantum market potential has also catalyzed unprecedented governmental investment, whereas China leads with $15.3 billion in announced funding, followed by Japan at $9.2 billion and the United States at $6.0 billion, according to the McKinsey report as of June 2025 [10]

Quantum computing applications span multiple critical sectors where classical computational approaches face fundamental limitations (as shown in Figure 1.1). In optimization, variational quantum algorithms demonstrate the potential for complex combinatorial problems in logistics, portfolio optimization, and supply chain management [11, 12]. Molecular simulation capabilities will enable unprecedented precision in pharmaceutical research and materials science, accelerating drug discovery and novel material design through quantum-enhanced modeling of electronic structures and chemical interactions [13]. In cybersecurity, quantum algorithms simultaneously threaten existing

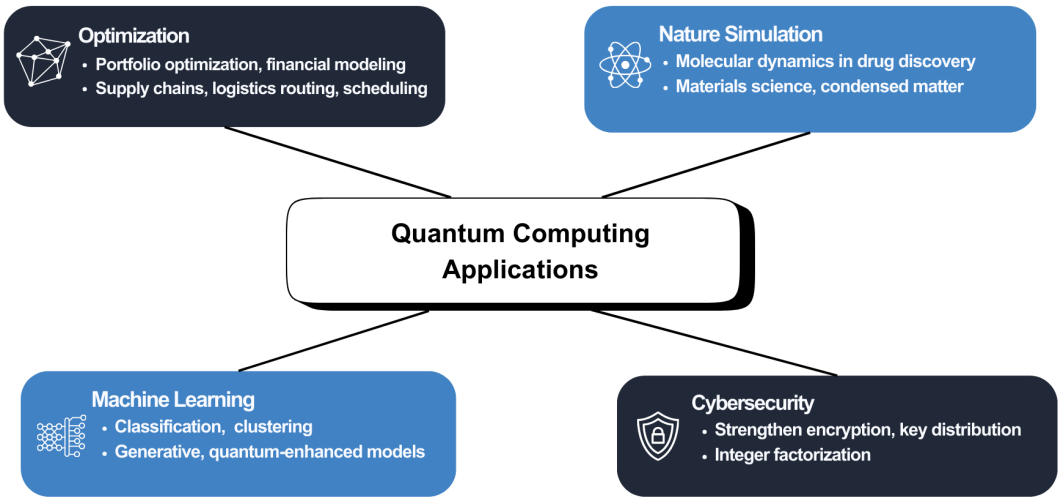


Figure 1.1: Examples of Quantum Computing Applications

public-key cryptographic systems while enabling provably secure quantum key distribution protocols [14]. Furthermore, quantum machine learning algorithms could offer substantial speedups for specific pattern recognition and data analysis tasks, particularly in high-dimensional feature spaces [6]. These and a number of other applications of quantum computing have sparked significant research interest in developing new algorithms and software solutions.

However, the current development of quantum computing applications faces many infrastructure and accessibility challenges that distinguish it from classical computing. Current quantum systems operate within the constraints of the Noisy Intermediate-Scale Quantum (NISQ) era [15], characterized by limited qubit counts, high error rates, and short coherence times. Hardware heterogeneity across competing quantum technologies, such as superconducting, trapped ions, and photonic systems, creates additional complexity for practical deployment. These diverse approaches create fragmented ecosystems with different quantum software stacks and distinct operational requirements. Additionally, cost and accessibility barriers have historically limited quantum computing to the general public.

The emergence of cloud-based quantum computing paradigms has begun to address these fundamental accessibility challenges, offering more democratized computational resources through service-based models. This paradigmatic shift toward distributed quantum computing environments has enabled broader participation in quantum application development while introducing novel challenges in resource management, orchestration, and optimization that form the core focus of cloud-based quantum computing research, including this thesis.

1.1 Background and Motivations

1.1.1 Cloud-based Quantum Computing Environments

The Quantum-as-a-Service (QaaS) model has transformed quantum computing accessibility by eliminating the substantial investment and specialized infrastructure requirements traditionally associated with operating quantum systems. Major cloud providers have established comprehensive quantum cloud platforms that provide remote access to diverse quantum processors through programmatic interfaces. IBM Quantum [16] pioneered public quantum cloud access in 2016, offering researchers and developers access to superconducting quantum processors. Microsoft Azure Quantum [17] and Amazon Braket [18] also provide heterogeneous platforms supporting multiple quantum hardware technologies such as IonQ trapped-ion systems, Honeywell neutral atom processors, and Rigetti superconducting circuits, enabling cross-platform quantum algorithm development and execution across different quantum hardware backends.

The hybrid quantum-classical computing paradigm has emerged as the predominant architectural model for practical quantum applications [19–21]. This paradigm is envisioned to span a continuum of computational layers, extending from cloud and edge computing environments to Internet of Things (IoT) devices, as illustrated in Figure 1.2. Such an architectural model enables seamless orchestration between quantum processors for specialized computational tasks and classical systems for preprocessing, postprocessing, and overall application coordination. This layered integration enhances flexibility, scalability, and the practical deployment of quantum applications across het-

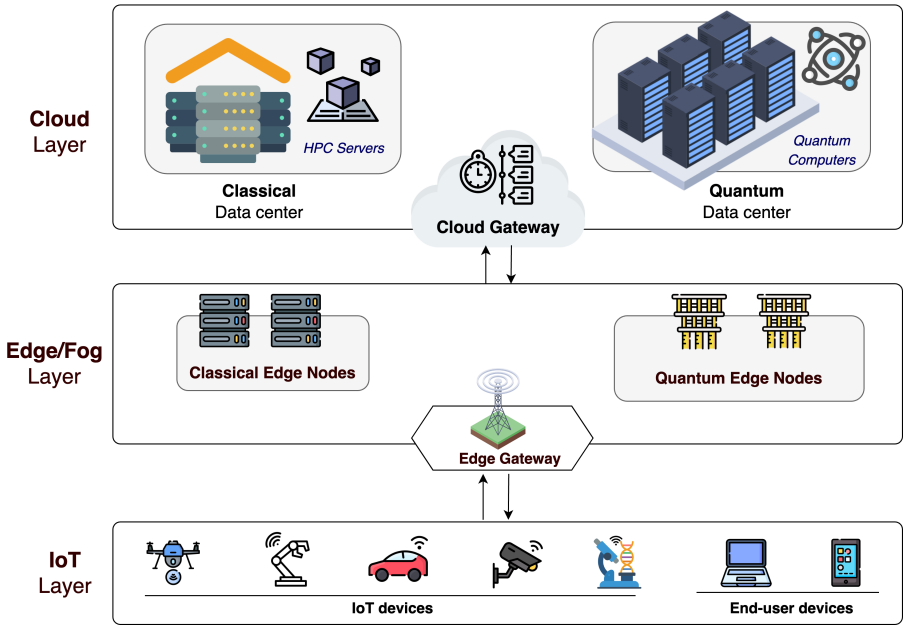


Figure 1.2: High-level overview of hybrid quantum-classical computing environments

erogeneous environments. Several hybrid algorithms, which combine both quantum and classical components, leverage classical optimization techniques to iteratively refine quantum circuit parameters, exemplified by variational quantum algorithms such as VQE [5] and QAOA [4]. Contemporary quantum cloud platforms support these hybrid processes through sophisticated workflow orchestration systems that manage data flow, resource allocation, and execution coordination across heterogeneous quantum and classical infrastructure components. Recent developments in quantum cloud computing have extended beyond basic circuit execution to encompass comprehensive application development environments. For example, serverless quantum computing enables function-as-a-service models that abstract quantum resource management complexity from application developers (see Chapter 3). Many cloud-based quantum software development frameworks and middleware [22–25] have also been proposed to facilitate the hybrid quantum-classical software deployment. Furthermore, distributed quantum computing techniques such as quantum circuit knitting [26] have been proposed to enable the execution of quantum computing applications across different quantum computing resources.

1.1.2 Challenges in resource management for Quantum Computing

Cloud-based quantum computing environments exhibit distinct resource management characteristics that fundamentally differentiate them from traditional cloud computing models. The unique properties of quantum systems introduce multifaceted challenges that require sophisticated approaches to system integration, resource orchestration, and performance optimization across the quantum computing stack. There are several key challenges, which are discussed below.

- *System Integration and Deployment Complexity*: The integration of quantum computing into practical workflows necessitates seamless orchestration between classical preprocessing, quantum execution, and classical post-processing components [20, 25]. Classical-quantum integration challenges arise from the need to coordinate heterogeneous computational resources with vastly different execution models, data formats, and timing constraints. Legacy system compatibility presents additional complexity as quantum applications must integrate with existing enterprise IT infrastructure, data pipelines, and security frameworks that were designed for classical computing paradigms [27]. The adaptation of DevOps practices for quantum computing requires new approaches to continuous integration and deployment that accommodate quantum algorithm development, circuit compilation, and hardware-specific optimization [28].

- *Quantum hardware and software heterogeneity*: Quantum resource management faces unprecedented challenges due to hardware heterogeneity manifested through incompatible instruction sets, varying gate fidelities, different qubit connectivity patterns, and distinct calibration requirements across quantum platforms [29]. Resource fragmentation occurs through platform-specific software stacks such as Qiskit, Cirq, and PennyLane, each utilizing different quantum intermediate representations that hinder cross-platform portability [30]. NISQ limitations significantly impact resource utilization through constrained quantum volume, gate error accumulation, and decoherence effects that limit algorithm complexity and execution time [15]. Dynamic resource characteristics introduce additional scheduling complexity as gate fidelities vary temporally, recalibration schedules affect availability,

and hardware downtimes require adaptive resource allocation decisions [25, 31].

- *Environment Dynamics and Orchestration Priorities*: Time-aware orchestration becomes critical due to quantum coherence constraints that necessitate real-time scheduling and sophisticated queue management for time-sensitive quantum operations. Fidelity-aware placement requires intelligent circuit-hardware matching based on gate error rates, connectivity graphs, and error correlation patterns to optimize computation quality. Multi-objective optimization presents complex challenges in balancing execution time, cost, and fidelity across heterogeneous quantum computing environments. Dynamic adaptation capabilities are essential for real-time adjustment to hardware performance fluctuations, queue dynamics, and failure recovery scenarios that characterize operational quantum cloud environments.

- *Resource Management Modeling, Simulation, and Evaluation Challenges*: Scalability limitations arise from classical simulation bottlenecks that restrict quantum system modeling beyond small scales, necessitating alternative evaluation methodologies. Realistic noise modeling requires an accurate representation of hardware-specific error models and temporal variations of the quantum computing environments that significantly impact quantum application performance. Benchmarking complexities emerge from the need for standardized metrics enabling quantum cloud service evaluation and performance comparison across diverse platforms and hardware technologies [32]. Resource estimation challenges involve predicting quantum resource requirements, including gate counts, circuit depth, and transpilation of quantum circuits before execution.

In this thesis, we address the critical challenges of deployment and execution of quantum applications in a resource-constrained, heterogeneous cloud-based quantum computing environment by developing software system frameworks and techniques for optimizing the overall resource management and task placement. Here, we systematically study the literature of cloud-based quantum computing, recent advances, open problems, and future directions. Then, we develop a serverless framework for quantum application deployment in practical hybrid quantum-classical environments. Furthermore, we propose the modeling and simulation approaches for quantum resource man-

agement evaluation. Afterward, we propose different quantum task placement techniques using deep reinforcement learning to optimize the time and fidelity of the quantum task execution. These collective studies contributed to the fundamentals of software systems and resource management research in quantum computing environments.

1.2 Research Questions and Objectives

The fundamental challenges in cloud-based quantum computing resource management, as outlined in the previous section, necessitate adaptive and efficient approaches that address the unique constraints and dynamics inherent in quantum systems. The objective of this thesis is to improve the overall performance of quantum applications and resource management in cloud-based environments, focusing in optimizing time and fidelity of the quantum execution by utilizing the state-of-the-art serverless models and deep reinforcement learning techniques. To achieve these objectives, this thesis investigates four interconnected research questions (RQ) that span the quantum application workflow, from application deployment to performance optimization, each addressing critical gaps in cloud-based quantum computing paradigms:

- *RQ1. How can the serverless computing model facilitate quantum software development and resource orchestration in cloud-based quantum computing environments?*

  This research question addresses the critical challenge of quantum software complexity and infrastructure management barriers that impede widespread quantum application development. Traditional quantum computing deployments require extensive configurations and complex infrastructure management, creating significant barriers for quantum software developers [28]. The serverless computing paradigm, which has demonstrated success in classical cloud environments by abstracting infrastructure complexity and enabling scalable, event-driven execution models, presents a promising approach for quantum application development [33]. This research question investigates whether the serverless model can effectively abstract quantum resource management complexity while maintaining the performance and flexibility required for practical quantum applications. Key chal-

lenges include adapting classical serverless principles to quantum-specific constraints such as circuit compilation requirements, application deployment, software and hardware heterogeneity across different quantum computation backends and cloud providers. The investigation encompasses the development of quantum function-as-a-service models that enable seamless integration of quantum computation within hybrid classical-quantum workflows, automated quantum backend selection mechanisms, and DevOps-driven deployment strategies for quantum applications. The research addresses practical considerations, including cold start mitigation for quantum functions and multi-cloud orchestration capabilities that can mitigate vendor lock-in issues. Furthermore, it explores how serverless architectures can facilitate the integration of quantum computing with existing software systems, and facilitate the quantum computing adoption without extensive infrastructure redesign.

- *RQ2. How can we design efficient and extensible modeling and simulation frameworks to evaluate resource management strategies in quantum computing environments?*

  The second research question addresses the critical shortage of comprehensive modeling and simulation tools specifically designed for quantum cloud computing environments. While classical cloud computing has benefited from mature simulation frameworks such as CloudSim [34], iFogSim [35], and EdgeSimPy [36], quantum computing lacks equivalent tools that capture the unique characteristics of quantum resources and hybrid quantum-classical workflows. This limitation significantly hinders research and development in quantum resource management that needs to be conducted at a large scale, as practical experimentation with physical quantum hardware is prohibitively expensive and limited in availability.

  This research investigates the design principles and implementation strategies for quantum computing environment simulators that support comprehensive resource management evaluation. The investigation encompasses system modeling approaches that accurately represent quantum processing units, quantum tasks, and their interactions within the cloud-edge continuum. Key challenges include developing lightweight discrete-event simulation architectures that can scale to large

quantum cloud deployments while maintaining accuracy in quantum-specific metrics such as quantum volume [29], Circuit Layer Operations Per Second (CLOPS) [32], and fidelity degradation patterns. The proposed framework design emphasizes extensibility to accommodate emerging quantum technologies and metrics, ensuring relevance as quantum hardware continues to evolve. Practical considerations include dataset integration capabilities for real quantum circuit benchmarks, support for hybrid quantum-classical task orchestration scenarios, and compatibility with machine learning-based resource management algorithms.

- *RQ3. How can deep reinforcement learning be used to develop time-aware resource management techniques for cloud-based quantum computing environments?*

  The third research question addresses the inadequacy of traditional heuristic approaches for quantum task scheduling in dynamic, heterogeneous quantum cloud environments. Classical resource management techniques, including greedy algorithms, round-robin scheduling, and static priority-based approaches, fail to adapt effectively to the temporal variations in quantum device performance, queue dynamics, and task arrival patterns that characterize operational quantum cloud environments [31]. The stochastic nature of quantum computations and the criticality of execution timing due to decoherence constraints necessitate adaptive task placement strategies that can learn from environmental dynamics. This research investigates the application of deep reinforcement learning techniques to quantum task placement and scheduling problems, specifically focusing on temporal optimization objectives. The investigation encompasses the formulation of quantum task placement as Markov Decision Processes that capture the sequential nature of scheduling decisions under uncertainty.

  Key challenges include designing state representations that effectively encode quantum task characteristics, device capabilities, and temporal queue states, while maintaining computational tractability for real-time decision making. The research explores reward function design strategies that balance optimization objectives, including task completion time and scheduling fairness across heterogeneous quantum devices. Time-awareness is incorporated through explicit modeling of queue

waiting times, device availability schedules, and deadline constraints that reflect the temporal sensitivity of quantum applications. The investigation addresses the scalability challenges of applying deep reinforcement learning to quantum cloud environments with varying numbers of quantum processors and diverse task workloads. Practical considerations include training efficiency optimization to enable deployment in operational quantum cloud systems, exploration strategies that balance performance with learning efficiency in expensive quantum environments. The research also investigates the integration of domain knowledge about quantum computing constraints into the learning process to accelerate convergence and improve scheduling decisions

- *RQ4. Can deep reinforcement learning effectively balance execution fidelity and time in quantum task orchestration that adapts to the constraints of current NISQ systems?*

  The fourth research question addresses the fundamental trade-off between computational fidelity and execution time that defines practical quantum computing in the current NISQ era [15]. Unlike classical computing, where performance is primarily constrained by computational resources and time, quantum computing introduces fidelity as a critical quality metric that directly impacts algorithmic success probability. Poor fidelity can render quantum computations meaningless regardless of execution speed, while overly conservative fidelity optimization can result in long execution times. This research investigates whether deep reinforcement learning can effectively navigate the complex fidelity-time optimization landscape in heterogeneous quantum cloud environments. The investigation encompasses the development of comprehensive performance models that integrate device calibration data, circuit transpilation effects, and noise-aware fidelity estimation to support adaptive orchestration decisions. Key challenges include designing multi-objective optimization frameworks which the main focus on high-fidelity computation priorities.

  The research explores the modeling of quantum task execution fidelity through realistic noise characterization based on actual quantum device calibration data, including gate error rates, coherence times, and connectivity constraints. This ap-

proach enables accurate prediction of execution quality across different device-circuit combinations, supporting informed scheduling decisions that optimize for overall performance rather than individual metrics. The investigation addresses the dynamic nature of quantum device characteristics, including temporal variations in error rates and calibration cycles that affect optimal scheduling decisions. Practical considerations include configurable optimization objectives that enable adaptation to different application requirements, noise-aware transpilation effects that capture the impact of circuit mapping on execution fidelity, and real-time adaptation to device performance. The research also investigates the implications of fidelity-time trade-offs, which is essential for efficient resource management in cloud-based quantum computing environments.

1.3 Thesis Contributions

This thesis makes the following contributions to address the research problems mentioned above:

1. Proposes a comprehensive review and systematic mapping study of quantum cloud computing paradigms, identifies research gaps in quantum resource management, open problems, and future directions (addresses the research foundation).

    - Comprehensive literature analysis of quantum cloud computing state-of-the-art, covering cloud computing models, platforms, applications, resource management, and security aspects.
    - Systematic taxonomy of quantum cloud computing research domains through structured mapping and classification of primary studies.
    - Future research directions establishment for quantum cloud computing advancement, including technical challenges and open problems.

2. Develops a holistic serverless quantum computing framework that enables seamless integration of quantum computation within established classical systems while avoiding vendor lock-in problems *(addresses RQ1)*.

- A Quantum Function-as-a-Service (QFaaS) architecture with six extensible system layers supporting multiple quantum SDKs (Qiskit, Cirq, Q#, and Braket) and cloud providers (IBM Quantum and Amazon Braket).
- Adaptive priority-based backend selection policy that automatically determines the most appropriate quantum computation system for executing quantum functions based on task requirements and system capabilities.
- Cold start mitigation strategy through transpilation caching and containerized deployment to reduce quantum function invocation latency.
- DevOps integration for quantum software development, including Continuous Integration / Continuous Deployment (CI/CD) pipelines, and container-based orchestration.
- Hybrid quantum-classical workflow orchestration enabling seamless integration of quantum functions within existing classical application workflows.
- Industry adoption with practical deployment demonstrated through the proposed quantum serverless platform implementation based on QFaaS framework architecture [37].

3. Design and develop a comprehensive modeling and discrete-event simulation framework for quantum computing environments that enables systematic evaluation of resource management strategies *(addresses RQ2)*.

   - Comprehensive system model for quantum computing environments using key metrics of available quantum computers and quantum task execution, serving as a theoretical foundation for quantum resource management.
   - Discrete-event simulation architecture based on CloudSim [34] supporting quantum cloud-edge continuum modeling.
   - Multi-use case support for quantum resource management problems, including task scheduling, backend selection, hybrid task orchestration, and task offloading between edge and cloud layers.
   - Real dataset integration capabilities accommodating IBM Quantum calibration data and MQT Bench quantum circuit datasets for realistic performance

evaluation.

- Open-source contributions of the proposed work, including iQuantum and QSimPy (the Python extended version of iQuantum), providing toolkits for the research community advancement in quantum cloud computing resource management.

4. Proposes a novel deep reinforcement learning-based approach for time-aware quantum task placement that adapts to dynamic quantum cloud environments and optimizes task completion efficiency *(addresses RQ3)*.

    - Deep Q-Network (DQN) framework enhanced with Rainbow DQN, combining advantages of Double DQN, Prioritized Replay, Multi-step Learning, Distributional RL, and Noisy Nets for robust quantum task placement.
    - Markov Decision Process formulation for quantum task scheduling with comprehensive state representation including quantum node features and task characteristics.
    - Adaptive reward function design balancing task completion time minimization and rescheduling attempt reduction through penalty factors and inverse completion time rewards.
    - Significant time-aware performance improvements and scalable training methodology using the Ray RLlib framework with hyperparameter optimization and evaluation on realistic quantum workloads from MQTBench datasets.

5. Develops a fidelity-aware quantum task orchestration framework using deep reinforcement learning that effectively balances execution fidelity and time constraints in NISQ-era quantum systems *(addresses RQ4)*.

    - Noise-aware performance modeling integrating IBM quantum processor calibration data with circuit transpilation effects for accurate fidelity and execution time estimation.
    - Proximal Policy Optimization (PPO) approach for learning adaptive orchestration policies that balance quantum execution fidelity and time across heterogeneous quantum cloud environments.

- Configurable fidelity-aware optimization framework enabling adaptation to different operational priorities through adjustable time penalty weights for fidelity-time trade-off balancing.
- Comprehensive fidelity performance metrics incorporating base execution fidelity, relative performance scoring, and circuit complexity factors for holistic quantum task quality assessment.
- Thorough performance evaluation using IBM Quantum calibration data and MQT Bench quantum circuit datasets across different evaluation scenarios and workload configurations.
- Substantial fidelity improvements in relative fidelity performance compared to conventional scheduling baselines while maintaining comparable quantum execution times.

1.4 Thesis Organization

The structure of this thesis is shown in Figure 1.3. The remaining chapters of this thesis are organized as follows:

- Chapter 2 presents a comprehensive review and systematic taxonomy mapping of quantum cloud computing paradigms, covering state-of-the-art developments, research challenges, and future directions. This chapter provides the foundational background for quantum cloud computing research and identifies critical gaps in resource management that motivate the subsequent technical contributions. This chapter is derived from:

  Hoa T. Nguyen, Prabhakar Krishnan, Dilip Krishnaswamy, Muhammad Usman, and Rajkumar Buyya, "Quantum Cloud Computing: A Review, Open Problems, and Future Directions", *submitted to ACM Computing Surveys (CSUR), April 2024* (Revision).

- Chapter 3 presents QFaaS, a holistic serverless function-as-a-service framework for quantum computing that enables seamless integration of quantum computation

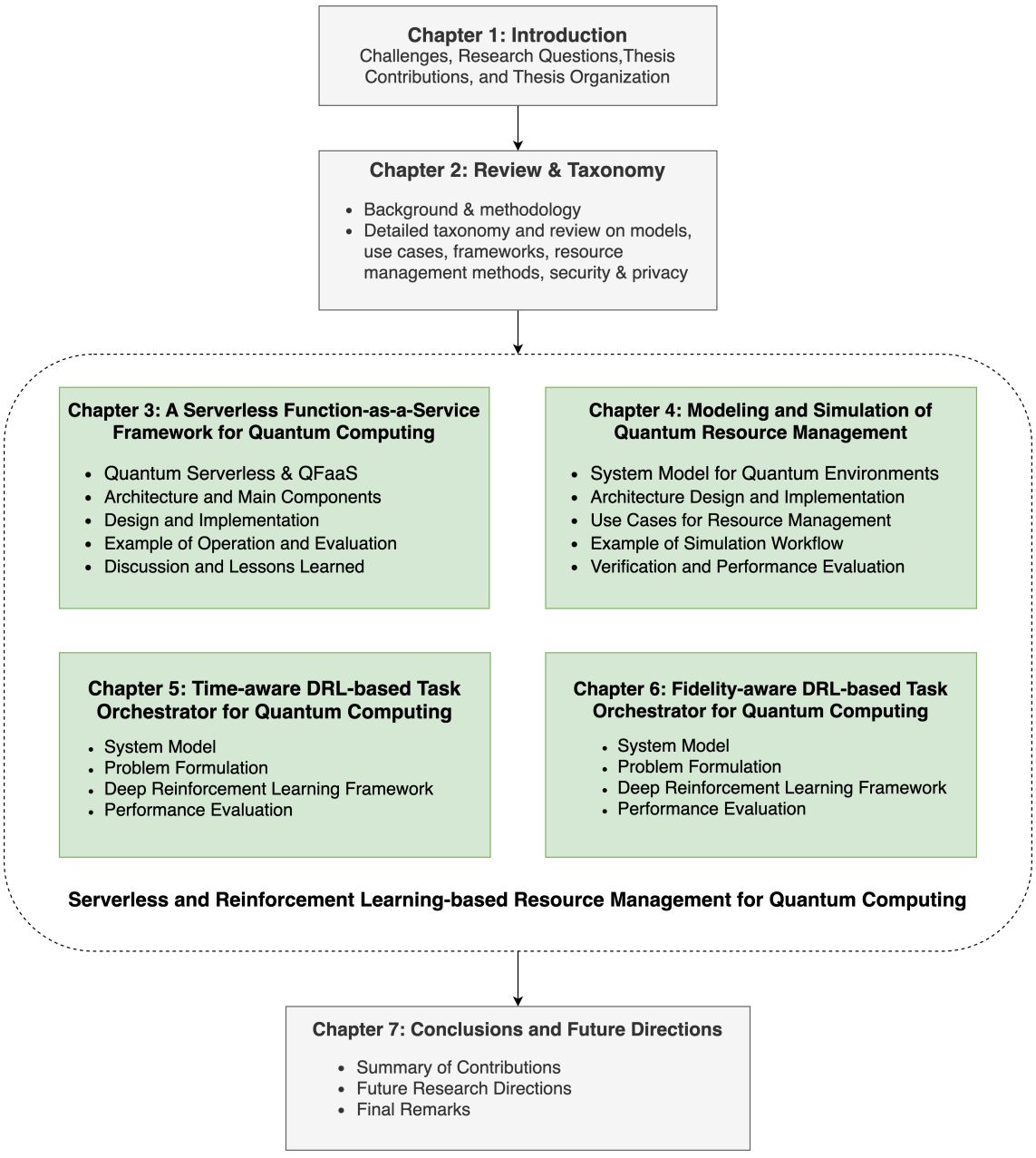


Figure 1.3: Overview of the thesis structure

within established classical systems while avoiding vendor lock-in problems. The chapter introduces quantum functions, adaptive backend selection policies, and DevOps integration for quantum software development. This chapter is derived from:

Hoa T. Nguyen, Muhammad Usman, and Rajkumar Buyya, "QFaaS: A Serverless Function-as-a-Service Framework for Quantum Computing", *Future Generation Computer Systems (FGCS)*, Volume 154, Pages: 281-300, ISSN: 0167-739X, Elsevier Press, Amsterdam, The Netherlands, May 2024

- Chapter 4 presents iQuantum, a comprehensive modeling and simulation framework for quantum computing environments that enables systematic evaluation of resource management strategies without requiring expensive access to physical quantum hardware. The chapter details the discrete-event simulation architecture, system models for quantum entities, and validation using realistic quantum datasets. This chapter is derived from:

  Hoa T. Nguyen, Muhammad Usman, and Rajkumar Buyya, "iQuantum: A toolkit for modeling and simulation of quantum computing environments", *Software: Practice and Experience (SPE)*, Volume 54, Issue 6, Pages: 1141-1171, ISSN: 0038-0644, Wiley Press, New York, USA, June 2024.

- Chapter 5 presents DRLQ, a novel deep reinforcement learning-based framework and technique for time-aware quantum task placement in cloud computing environments. The chapter formulates quantum task scheduling as a Markov Decision Process and employs enhanced Deep Q-Networks with Rainbow DQN to optimize task completion efficiency and minimize rescheduling attempts. This chapter is derived from:

  Hoa T. Nguyen, Muhammad Usman, and Rajkumar Buyya, "DRLQ: A Deep Reinforcement Learning-based Task Placement for Quantum Cloud Computing", Proceedings of the 17th IEEE International Conference on Cloud Computing (CLOUD 2024), Shenzhen, China, July 7-13, 2024

- Chapter 6 presents QFOR, a fidelity-aware quantum task orchestration framework

using deep reinforcement learning that effectively balances execution fidelity and time constraints in NISQ-era quantum systems. The chapter introduces noise-aware performance modeling, configurable fidelity-aware optimization, and Proximal Policy Optimization for adaptive quantum resource allocation. This chapter is derived from:

Hoa T. Nguyen, Muhammad Usman, and Rajkumar Buyya, "QFOR: A Fidelity-aware Orchestrator for Quantum Computing Environments using Deep Reinforcement Learning", *submitted to the ACM Transactions on Quantum Computing (TQC), August 2025* (Under Review)

- Chapter 7 presents the conclusions of this thesis, summarizes the key contributions, discusses the implications of the research findings, and outlines future research directions for advancing cloud-based quantum computing.

Chapter 2

A Review and Taxonomy on Cloud-based Quantum Computing

Cloud-based Quantum Computing, or Quantum Cloud Computing, is an emerging paradigm of computing that empowers quantum applications and their deployment on quantum computing resources without the need for a specialized environment to host and operate physical quantum computers. This chapter provides a systematic review and taxonomy of recent advances in quantum cloud computing. It discusses the state-of-the-art quantum cloud advances, including the various cloud-based models, platforms, and recently developed technologies and software use cases. Furthermore, it discusses different aspects of the quantum cloud, including quantum serverless, hybrid quantum-classical computing, resource management, quantum cloud security, and privacy problems. Our findings offer valuable insights and practical lessons learned from the literature for researchers and practitioners to better understand the current landscape of quantum cloud computing and set the foundations for the remaining chapters of this thesis.

2.1 Introduction

Quantum computing is anticipated to revolutionize many scientific fields by promising to solve intractable computational problems beyond the capabilities of the state-of-the-art classical computers. Although quantum computers are still in the early stages of development, they have already been used to address critical problems, such as simulating the behaviour of molecules [38, 39] and discovering new drugs [40–42]. However,

This chapter is derived from:

- **Hoa T. Nguyen**, Prabhakar Krishnan, Dilip Krishnaswamy, Muhammad Usman, and Rajkumar Buyya, "Quantum Cloud Computing: A Review, Open Problems, and Future Directions", *submitted to ACM Computing Surveys (CSUR), April 2024* (Second Revision).

acquiring a dedicated physical quantum computer and hosting it is challenging as they require an extremely special environment to host and operate, such as extremely cold temperatures and complicated controlling [43]. Due to the complexity of operating a physical quantum computer, hosting them within cloud computing environments is the most common, as they provide easy access to today's quantum computation resources. Thus, quantum machines are placed in a remote data center with particular environmental conditions. These devices can interact with end-users via the vendor cloud platform, which allows for the execution of quantum circuits and the retrieval of the computation result. Engineers and researchers, therefore, have more convenient ways to access and utilize quantum devices remotely from their local computers. This model shortens the pathway to achieving quantum advantages in the current era of noisy intermediate-scale quantum (NISQ) hardware. NISQ devices, characterized by their tens to hundreds of qubit scale, operate with a significant level of noise and limited qubit coherence times, which presents unique challenges for quantum computation [15]. Despite these limitations, NISQ technology offers a promising platform for exploring quantum algorithms and applications prior to the advent of fault-tolerant quantum computing.

The emerging field of quantum cloud computing (QCC) combines the principles of quantum computing with cloud infrastructure, enabling remote access to quantum computers. This integration promises to significantly lower the barrier to utilizing quantum computing resources, making it feasible for researchers and developers to explore quantum algorithms without the need for possession of quantum hardware, facilitating a wide range of cloud-based applications, such as quantum random number generation [44–46] and quantum machine learning [47–50]. As interest in QCC grows, the field has expanded with research focused on novel cloud-based quantum computation models, hybrid architectures, frameworks and platforms, resource management policies, and enhanced privacy and security mechanisms. Given these rapid advancements, a comprehensive review of the field is essential to provide clarity and structure, especially for practitioners and researchers seeking to understand the current landscape and identify new opportunities in quantum cloud computing. Despite the growing body of QCC literature, existing review studies [51–53], however, are often superficial, lacking detailed analysis of recent technological advancements and emerging service models in quantum

cloud computing (discussed in Section 2.2). The interdisciplinary nature of QCC demands a survey that can systematically map the state-of-the-art, covering foundational concepts, service models, practical implementations, resource management, and security to fully capture its scope and potential. This gap highlights the critical need for a systematic and thorough review to encapsulate the latest developments, open problems, and future directions in QCC. To address these gaps, our work systematically examines QCC through the perspective of corresponding concepts and problems in classical cloud computing, using a rigorous systematic mapping method to provide a valuable resource for researchers in both quantum computing and cloud computing domains.

As one of the first literature mapping studies on quantum cloud computing, the major contributions and novelty of our review are:

- We conduct a rigorous review of the quantum cloud computing literature, mapping recent studies and emerging models such as hybrid quantum-classical computing and quantum serverless architectures. Additionally, we provide an overview of leading quantum cloud providers and their contributions to advancements in the field.

- We provide a comprehensive overview of the state-of-the-art in quantum cloud computing, including technologies, frameworks, platforms, and applications that have been developed for quantum cloud computing. Our survey presents various applications of quantum cloud computing in different fields, such as machine learning and cryptography.

- We review recent advances in quantum cloud computing in emerging aspects, including service models, resource management and orchestration, and enhanced security mechanisms specific to quantum cloud environments.

- We identify key challenges and open problems that need to be addressed based on the systematic mapping study and propose future research directions to realise the full potential of quantum cloud computing *(discussed in Chapter 7)*.

The rest of this chapter is organized as follows: Section 2.2 discusses related work to our work. Section 2.3 covers the foundational concepts of quantum computing and

cloud computing to accommodate the diversity of readers' backgrounds. Section 2.4 presents the research methodology and review protocol in detail. In Section 2.5, we comprehensively provide the review results of recent advances in quantum cloud computing. This section is divided into several sub-sections, focusing on different aspects of quantum cloud computing based on the literature classification step in our review protocol. In Section 2.6, we discuss the limitations and threats to validity of our review. Finally, we conclude and summarize the chapter in Section 2.7.

2.2 Related Work

Several recent review articles have addressed various aspects of quantum computing, but few have systematically focused on quantum cloud computing (QCC). To comprehensively assess their rigour, we classified their review methodologies and critically analysed each study using the Database of Abstracts of Reviews of Effects (DARE) criteria [54] as suggested by well-known review guidelines by Kitchenham et al. [55, 56]. These criteria include (**Q1**) inclusion/exclusion criteria reported, (**Q2**) topic coverage and comprehensiveness of the literature review, (**Q3**) systematic quality assessment of included studies, and (**Q4**) adequacy in describing basic data/studies. In Q2 criteria, we evaluated each paper based on topic coverage of key QCC aspects, including fundamental concepts, models, use cases, frameworks and platforms, resource management, security and privacy, open problems and future directions, as summarised in Table 2.1.

The existing literature on quantum cloud computing remains limited in scope, often lacking a comprehensive analysis of recent advancements in various aspects of the field. Several early surveys, such as those by Kaiiali et al. [51], Leymann et al. [53], and Soeparno et al. [52], adopt narrative approaches without formal protocols, resulting in fragmented coverage across fundamental concepts, frameworks, or emerging QCC models like serverless quantum computing. Kaiiali et al. [51] primarily focused on the security implications and high-level impacts of quantum computing on cloud computing but lacked a detailed analysis of the latest advancements, such as computation models and resource management in quantum clouds. Leymann et al. [53] reviewed some application potentials and opportunities of quantum computing on the cloud, focusing

Table 2.1: Summary of related review articles on quantum cloud computing

Refs	Year	Mtd	Q1	Q2							Q3	Q4
				C	M	UC	FP	RM	SP	FD		
Kaiiali et al. [51]	2019	NA	✗	PD		PD					✗	PD
Leymann et al. [53]	2020	NA	✗	PD	PD	PD	PD				✗	PD
Soeparno et al. [52]	2020	NA	✗	PD		PD					✗	✗
Gill et al. [57]	2021	SR	✗	PD		PD				PD	✗	✓
Saki et al. [43]	2021	NA	✗	PD					PD		✗	PD
Serrano et al. [58]	2022	NA	✗			PD	PD				✓	✓
Khan et al. [59]	2023	SR	✓			PD	PD				✓	✓
Yang et al. [60]	2023	NA	✗			PD				PD	✓	✓
Brotherton et al. [61]	2023	NA	✗	PD					PD	PD	✗	PD
Golec et al. [62]	2024	NA	✗	PD		PD			PD	PD	✗	PD
Ours	**2024**	**SR**	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Notes. Mtd=Method; NA=Narrative Analysis; SR=Systematic Review; C=Concepts; M=Models; UC=Use cases; FP=Frameworks and Platforms; RM=Resource Management; SP=Security and Privacy; FD=Open Problems and Future Directions; PD=Partially Described.

on software tools and platforms for collaborating to develop quantum applications, but similarly fell short in addressing the complexities of quantum cloud frameworks and resource management. Soeparno et al. [52] provided insights based on practical experiences with IBM Quantum services, offering a general introduction to the available cloud platforms. While informative, this article remains surface-level and lacks a structured discussion on recent advancements or open challenges in quantum cloud computing.

Other broader surveys, such as those of Gill et al. [57] and Khan et al. [59], offer broader taxonomies and overviews of quantum computing technologies, briefly acknowledging cloud-based quantum services but without analysis into specific aspects of quantum cloud computing. These reviews do not discuss quantum-specific resource management or security frameworks and key components for deploying quantum services in cloud environments. Yang et al. [60] and Caleffi et al. [63] contributed by addressing quantum communication and networking, which are essential for distributed quantum systems. However, their focus diverges from core aspects of quantum cloud computing, such as platform-specific challenges and serverless quantum computing models, which are uniquely covered in our survey. Similarly, Brotherton and Gupta [61] provide a broad survey that focuses exclusively on the security aspects of quantum computing in the cloud, without covering other aspects of quantum cloud computing environments. In a recent review, Golec et al. [62] provided a high-level overview of

the quantum cloud computing concepts, identifying general trends, basic applications, and potential challenges. While acknowledging the paradigms, this review does not provide systematic thematic mapping on critical aspects of QCC such as models, frameworks and resource management.

These limited scopes in existing studies leave significant gaps in understanding the fundamentals, technical advancements, and open challenges in quantum cloud computing, which our study seeks to address comprehensively. Our study distinguishes itself by providing a structured, in-depth review of various facets of quantum cloud computing that remain underexplored in existing literature. Following the rigorous review protocol by Kitchenham et al. [55], we ensure a transparent and systematic review process, with clear inclusion/exclusion criteria, exhaustive literature search, quality assessment, and detailed descriptions of each primary study. We provide full-spectrum coverage of key areas QCC topics, including computation models, use cases, frameworks and platforms, privacy and security, and critically analyse open problems and future directions. This methodological rigour enables us to synthesise valuable insights and uncover underexplored research challenges that remain unaddressed in prior literature. Broader aspects, such as quantum communication channels, network infrastructure, architectural layers (e.g., hypervisor, virtual machines), quantum hardware and quantum solutions for classical cloud, are beyond the scope of this study and are covered in dedicated works on quantum networking and distributed quantum computing, such as [60, 63, 64]. By concentrating on these targeted areas, our survey provides a foundation resource for researchers and practitioners in this emerging domain to further explore and develop systematic reviews on specific aspects of quantum cloud computing.

2.3 A Synopsis of Quantum Computing and Cloud Computing

This section provides a brief overview of quantum computing and cloud computing to take into account the diverse backgrounds of readers.

2.3.1 Quantum Computing

Quantum computing is an emerging paradigm that utilizes quantum-mechanical phenomena, such as superposition and entanglement, to perform operations on data. It has the potential to solve certain problems such as optimization [4, 65, 66], financial modelling [11, 67], molecule simulation [38, 39], and machine learning [47–50, 68] much faster than classical computers. The subsequent discussion focuses on the cornerstone of quantum computing: the quantum bit, computation models, and quantum algorithms.

Quantum bits (Qubits)

A quantum bit (or qubit) is the basic unit of quantum information. Different from classical bits, which can exist in one of two states (0 or 1), qubits are characterized by their ability to exist in a superposition state, i.e., they can be in a combination of both states 0 and 1 simultaneously. This property enables the representation and processing of a richer set of information possibilities compared to classical bits. Another key characteristic of qubits is entanglement, a quantum phenomenon where the state of one qubit becomes inextricably linked to the state of another, regardless of the distance between them. This linkage means the properties of entangled qubits cannot be described independently of each other [69]. These distinctive features of qubits could enable quantum computers to execute certain computations more efficiently than classical computers, potentially solving problems considered intractable for classical computation.

Quantum computation models

Two distinct quantum computing models are being concurrently developed: gate-based computations and quantum annealing-based computations. Gate-based models, also known as circuit-based models, use a set of quantum gates to perform operations on qubits. These gates are used to manipulate the state of qubits and perform operations on them. Table 2.2 briefly summarizes typical quantum gates' characteristics. Gate-based models are based on building a quantum circuit to realize a sequence of gates that perform a specific computation. The action of quantum gates can be represented

Table 2.2: A summary of common quantum gates and their mathematical representation

Quantum gates		Description	Mathematical Representation
Single-qubit	Pauli-X	A bit flip gate, which acts as a rotation by π around the X-axis of the Bloch sphere	$\sigma_X = X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \|0\rangle\langle 1\| + \|1\rangle\langle 0\|$
	Pauli-Y	A bit and phase flip gate, which acts as a rotation by π around the Y-axis of the Bloch sphere	$\sigma_Y = Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} = -i\|0\rangle\langle 1\| + i\|1\rangle\langle 0\| = i.\sigma_X.\sigma_Z$
	Pauli-Z	A phase flip gate, which acts as a rotation by π around the Z-axis of the Bloch sphere	$\sigma_Z = Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \|0\rangle\langle 0\| - \|1\rangle\langle 1\|$
	Hadamard (H)	A gate to create a superposition of $\|0\rangle$ and $\|1\rangle$	$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \frac{1}{\sqrt{2}}(\|0\rangle\langle 0\| + \|0\rangle\langle 1\| + \|1\rangle\langle 0\| - \|1\rangle\langle 1\|)$
	Phase (P_ϕ)	A parametrized gate performs a rotation of ϕ around the Z-axis direction	$P(\phi) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix}$ where ϕ is a real number
	Identity (I)	A gate that have no effect on the qubit state	$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
	S (or $\sqrt{Z}$)	A $P(\phi)$ gate with $\phi = \pi/2$, apply a quarter-turn around the Bloch sphere	$S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{2}} \end{bmatrix}$
	T (or $\sqrt[4]{Z}$)	A $P(\phi)$ gate with $\phi = \pi/4$	$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{bmatrix}$
Multi-qubit	CNOT (or CX)	A 2-qubit gate that flips the target qubit when the control qubit is in the state $\|1\rangle$	$CX = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \|00\rangle\langle 00\| + \|01\rangle\langle 01\| + \|10\rangle\langle 11\| + \|11\rangle\langle 10\|$
	SWAP	A 2-qubit gate that swap (exchange) the state of these two qubits	$SWAP = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
	Toffoli (or CCX)	A 3-qubit gate that flips the third qubit if the first two qubits are both in state $\|1\rangle$	$CCXq_0, q_1, q_2 = I \otimes I \otimes \|0\rangle\langle 0\| + CX \otimes \|1\rangle\langle 1\|$

as a unitary transform on qubits. If a sequence of gates G_1, G_2, $\ldots G_m$ are applied to a set of input qubit state vector Q, then the resultant state is given by the corresponding cascaded sequence of unitary transforms applied to the input vector Q given by the matrix product $G_m \ldots G_2 G_1 Q$. This approach allows for flexibility and can be used to support a wide range of quantum algorithms, such as search and factorization problems.

Quantum annealing-based computations support an alternate paradigm of computation, and they are typically utilized to solve optimization problems. In this approach, an energy function (also known as the Hamiltonian of the system) is used to represent an optimization problem under consideration. For example, a QUBO (Quadratic Unconstrained Binary Optimization) [65] formulation involves self-coupling weights w_{ii} for qubit states x_i and cross-coupling weight terms w_{ij} for pairwise products of qubit state terms $x_i x_j$ to result in an energy function of the form $H = \sum_i w_{ii} x_i + \sum_{ij} w_{ij} x_i x_j$. Many optimization problems can be reduced to such a form for the energy function that is desired to be minimized. There are physical self-coupling and cross-coupling connections in an annealing-based quantum computer. In some simple cases, the optimization problem can be directly mapped to the physical networked connections in the quantum computer. In other cases where this is not feasible (for example, where the desired degree of cross-coupled connectivity is higher than the physically available connectivity), then the problem can be transformed (for example, by splitting a node with high vertex connectivity into multiple nodes) such that the problem can be reduced to a form that can be mapped to the physically available connectivity. The system is initialized to a random or known initial state. Subsequently, the quantum system is allowed to evolve until it settles down or anneals to a low-energy state as it attempts to minimize the energy function. The resultant low-energy state provides the solution to the problem. Due to the likelihood of the system getting trapped in a local minimum, multiple runs of the algorithm are attempted starting with different initial states, and the lowest energy solution across different runs can be utilized as the solution to the problem.

Quantum Algorithms

Quantum computation leverages unique quantum-mechanical properties, such as superposition, entanglement, and interference, to process information encoded in quantum bits (qubits). Unlike classical bits, which are binary, qubits can exist in the superposition of multiple states, enabling quantum algorithms to perform certain computations more efficiently than their classical counterparts. One notable example is Shor's algorithm [2], which demonstrates a significant speedup in factorizing large integers compared to classical approaches. This algorithm exploits quantum superposition and entanglement to find the prime factors of an integer exponentially faster than the best-known classical algorithms. The implications of Shor's algorithm are profound, particularly in cryptography, as many current encryption methods, such as RSA, rely on the high computational complexity of factorizing large numbers. The advent of quantum computing thus poses a potential threat to these encryption systems. Another prevalent quantum algorithm is Grover's algorithm [3], designed for searching unstructured databases. Grover's algorithm achieves a quadratic speedup over classical search algorithms. In a database of N unstructured items, Grover's algorithm can find the desired item in roughly $\sqrt{N}$ steps, compared to the N steps required classically. This speedup, while less dramatic than that of Shor's algorithm, is still significant and has implications for a wide range of search-related problems.

In recent years, advancements in quantum algorithms have expanded beyond these foundational concepts, embracing more complex and application-specific algorithms. Among these, the Variational Quantum Eigensolver (VQE) and Quantum Approximate Optimization Algorithm (QAOA) have gained prominence. The Variational Quantum Eigensolver (VQE) [5] is a hybrid quantum-classical algorithm primarily used in quantum chemistry for finding the ground state energy of molecules. VQE operates by preparing a quantum state on a quantum computer and measuring its energy, while a classical computer variably adjusts the quantum state to minimize its energy. This approach leverages the quantum system for what it does best – representing complex quantum states – and uses classical optimization techniques to navigate the solution space efficiently. The Quantum Approximate Optimization Algorithm (QAOA) [4] is designed for solving combinatorial optimization problems, like the Max-Cut problem.

QAOA works by encoding the problem into a Hamiltonian and then applying a series of quantum gates that approximate the problem's solution. Its performance improves with the number of quantum layers used, offering a promising approach for achieving quantum advantage in optimization tasks. Quantum Machine Learning (QML) [68, 70] represents another emerging field. QML algorithms leverage quantum computing for machine learning tasks, potentially offering speedups in data processing and pattern recognition. Quantum machine learning is particularly compelling because it merges two cutting-edge fields, quantum computing and artificial intelligence, potentially leading to breakthroughs in both domains.

While these recent quantum algorithms show great promise, they are still in the developmental stage and face implementation challenges. They often require a significant number of qubits and sophisticated error correction methods to be practically viable. Unlike classical computing, where errors are managed through straightforward bit-flip corrections, quantum computing's fault tolerance is more complex, addressing both bit-flip and phase-flip errors without direct measurement to avoid collapsing the quantum state [71]. These unique challenges in fault tolerance highlight the need for advanced quantum error correction codes to achieve reliable, large-scale quantum computation. However, as quantum technology continues to advance, these algorithms are likely to play a pivotal role in realizing the practical applications of quantum computing.

2.3.2 Cloud Computing

Cloud computing has revolutionized how software and IT infrastructure capabilities are made available as subscription-oriented computing utilities on a pay-as-you-go basis to consumers over the Internet. This paradigm has brought numerous benefits to society, enabling enterprises to expand globally faster and supporting scientific research advancement [72]. In this section, we briefly discuss their computing models along with the recent emergence of the serverless computing model.

Cloud computing models

There are several service models and deployment models in cloud computing:

1. **Service models**: Cloud computing can be delivered through three main service models, including Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [73]. IaaS provides users access to physical infrastructures like servers, networking, and data storage. IaaS enables users to outsource the infrastructure maintenance and management to a cloud provider and flexibly adjust the resources as needed. PaaS abstracts away the infrastructure and allows developers to think about creating applications on top of a platform that provides different services, such as authentication, database storage, load-balancing, auto-scaling, etc., to provide an environment that enables ease of development. SaaS enables applications to be hosted on a cloud to directly provide services to a user, with the platform and infrastructure aspects abstracted away from the user of the service.

2. **Deployment models**: Cloud computing can be deployed in different manners, including public clouds, private clouds, and hybrid clouds. For the public cloud, services are provided over the public Internet and are available to anyone willing to pay for them. This model offers scalability, flexibility, and efficiency. In terms of private cloud, infrastructure is dedicated to a single organization, offering more control and security. It can be hosted internally or externally. Hybrid clouds are combinations of public and private clouds, allowing data and applications to be shared between them. This model provides greater flexibility and optimization of existing infrastructure.

From Serverful to Serverless computing model

The transition from traditional (or serverful) models to serverless models marks a significant shift in cloud computing. The serverful computing model requires users to manage servers and computation resources, demanding setup, maintenance, and scaling, which offers control but adds complexity [74]. Serverless computing, however, abstracts server management, focusing developers on code rather than infrastructure. It dynamically allocates resources, optimizing costs by charging only for actual usage, and automatically scales to demand [75]. This shift simplifies operations, reduces costs, and allows

for more efficient resource use, emphasizing development efficiency over infrastructure management [76]. There are two common serverless paradigms:

1. **Function-as-a-Service (FaaS)**: FaaS allows users to execute code in response to specific events or triggers, typically through an application programming interface (API). The cloud provider automatically allocates and sets up the underlying infrastructure required to run the code [77]. Examples of FaaS-based commercial platforms include Amazon Web Services (AWS) Lambda, Google Cloud Functions, and Microsoft Azure Functions. Besides, there are numerous open-source FaaS frameworks, such as OpenFaaS, OpenWhisk, and Knative.

2. **Backend as a Service (BaaS)** provides a set of pre-built backend services, such as database management, user authentication, and push notifications, which developers can use to build and run their applications. Examples of popular BaaS platforms include Firebase, AWS Mobile Hub, AWS Amplify, and Microsoft Azure Mobile Apps.

Quantum and Post-Quantum Cryptography-based Techniques for Cloud Computing

Alongside the development of quantum cloud computing, significant research has focused on quantum-based and post-quantum cryptography (PQC) techniques to enhance security and efficiency within classical cloud computing environments. Quantum-based techniques leverage quantum algorithms to improve computational tasks for cloud computing, such as resource management optimization [78] and cloud workload prediction [79]. Additionally, PQC aims to secure cloud computing against potential quantum attacks by developing cryptographic algorithms resistant to quantum decryption, ensuring future-proof security for sensitive data [80]. PQC also intersects with QCC by securing classical components of cloud infrastructure against quantum-enabled attacks. Although these approaches are essential for advancing classical cloud security and efficiency, they fall outside the primary scope of this review study and can be found with further details in relevant studies, such as [81–83].

2.4 Review Methodology

This section describes the rigorous multi-step review process applied in our study, following well-established guidelines for systematic mapping studies (SM) [55, 84, 85], and other systematic mapping studies in quantum computing domain [86, 87]. The overall review protocol is outlined in Figure 2.1.

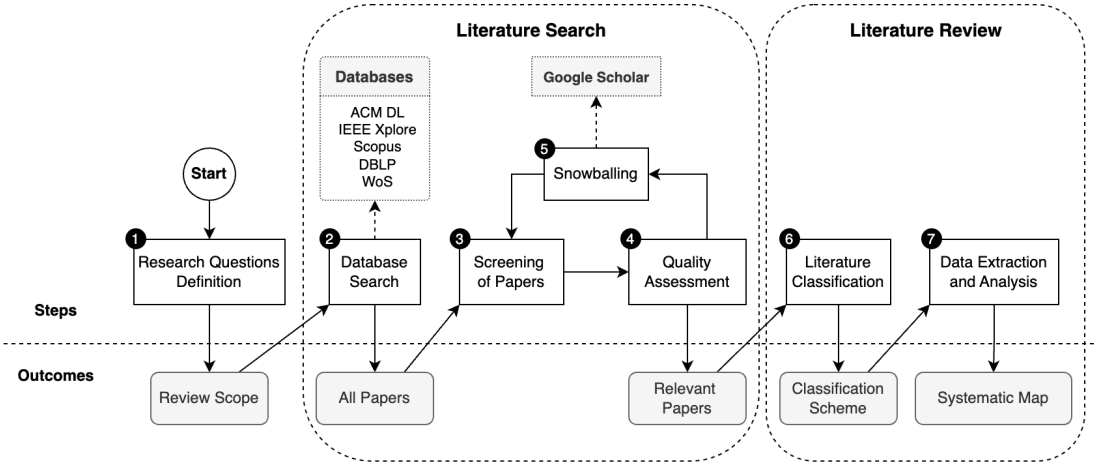


Figure 2.1: Overview of our multi-step review protocol

2.4.1 Research Questions and Scope of the Review

The main objective of our systematic mapping study is to analyze the current state of quantum cloud computing research and development comprehensively, covering both established computing paradigms and emerging trends. Our goal is to classify key research topics within the field and identify open challenges and future directions that are relevant to quantum cloud computing as opposed to the broader area of quantum computing. To achieve this, we carefully designed our research questions (RQs) to focus explicitly on quantum cloud computing based on an extensive search and review of primary studies specifically within this domain:

1. **Research Question 1 (RQ1):** *What is the state-of-the-art paradigm of quantum cloud computing?*

2. **Research Question 2 (RQ2)**: *What research topics are currently being explored in quantum cloud computing?*
   To ensure thoroughness, we divided this research question into two sub-questions:

   (a) RQ2.1 *Which key aspects of quantum cloud computing are being studied?*

   (b) RQ2.2 *What are advances in each identified aspect of quantum cloud computing?*

3. **Research Question 3 (RQ3)**: *What open problems and future research directions in quantum cloud computing?*
   The response to this question is discussed in Chapter 7.

These research questions establish a structured and comprehensive scope for this review, focusing on quantum cloud computing. Our study focuses explicitly on essential quantum cloud computing aspects, including cloud computing models, use cases, frameworks, platforms, resource management, security and privacy to provide a comprehensive mapping, addressing both the current state and the potential future directions. Areas such as quantum networking, quantum hardware, quantum-inspired algorithms, and post-quantum cryptography are explicitly excluded, as they belong to distinct research domains. While related, their core challenges, such as entanglement distribution in networking or device-level design in hardware, lie outside the cloud abstraction layer. Further review and analysis of those areas can be found in dedicated surveys, such as [60, 83, 88]. This exclusion ensures a focused and methodologically coherent review of the quantum cloud computing domain.

2.4.2 Review Protocol

Based on the defined research questions, we conducted an intensive sequential searching and selection process comprising four main steps labelled from (2) to (5) as shown in Figure 2.1. This structured approach is intended to minimize selection bias and ensure comprehensive coverage of the relevant literature.

Database Search

In Step 2, we performed a formal search across five major electronic databases: ACM Digital Library[1], IEEE Xplore[2], DBLP[3], Scopus[4], and Web of Science[5], following Kitchenham et al. [55] and other relevant quantum review studies [86, 87]. We excluded ArXiv to limit the inclusion of non-peer-reviewed studies, hence mitigating the introduction of selection bias in unpublished works. The following search string was used to ensure specificity to our topic: `"quantum" AND "cloud" AND "computing"`. The search string was applied uniformly across all databases, and search results were refined based on the inclusion/exclusion criteria specified below.

Inclusion and Exclusion Criteria

To establish a rigorous selection process for relevant primary studies, we defined the following inclusion (✓) and exclusion (✗) criteria:

- ✓ Peer-reviewed publications that directly cover quantum cloud computing.
- ✓ Publications written in English have been published and made available in the databases between January 2017 and July 2024, covering the period after IBM's introduction of one of the first public cloud-accessible quantum systems in 2017 [86, 89]. Therefore, most relevant research starting in 2017 will primarily use these available cloud-based quantum resources.
- ✓ Technical reports and publicly available information from quantum cloud vendors' websites on the current development of quantum cloud computing services.
- ✓ Publications accessible through open access or our institutional access.
- ✗ Publications that focus on unrelated topics, including the application of quantum computing for classical cloud computing, quantum-inspired algorithms, and post-quantum cryptography techniques.

[1] https://dl.acm.org/
[2] https://ieeexplore.ieee.org/Xplore/home.jsp
[3] https://dblp.org/
[4] https://www.scopus.com/
[5] https://www.webofscience.com

✗ Publications that focus on other related aspects that overlap significantly with previously reviewed systematic literature on other quantum computing topics, such as quantum communications, quantum networks, quantum hardware, and quantum error mitigation techniques.

✗ Publications that were only available in the form of abstracts or presentation slides.

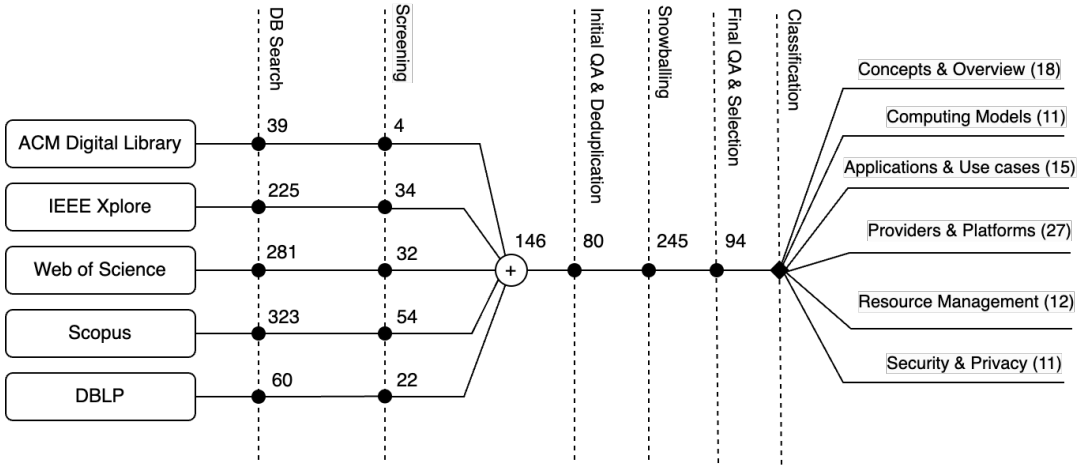


Figure 2.2: Results of our multistep review protocol for search, selection, and classification of primary studies

Screening, Quality Assessment, and Snowballing

In Step 3, we screened the titles and abstracts of all retrieved studies, removing those that did not meet the inclusion criteria and discarding duplicates. This initial screening helped refine our dataset, ensuring that only studies relevant to the RQs were retained. Step 4 involved an in-depth quality assessment (QA) of the remaining studies based on our inclusion criteria. To ensure methodological rigour, we prioritized peer-reviewed publications offering significant insights into quantum cloud computing. In cases where multiple papers reported the same research, we retained only the most recent and comprehensive work. In Step 5, we employed Google Scholar[6] for backward and forward snowballing [85] on key studies to identify additional relevant literature that may have

[6]https://scholar.google.com

been missed during the database search, similar to other systematic mapping studies [87]. This choice was motivated by Google Scholar's extensive and inclusive indexing of scholarly literature to maximise the likelihood of identifying all relevant studies, including potentially impactful works that may not be indexed in conventional databases. This step was repeated iteratively with steps 3 and 4 until no new relevant articles were found, ensuring comprehensive coverage and minimising the risk of missing relevant literature.

Literature Classification

In this stage, we categorized the selected studies into a structured classification scheme based on their primary focus areas within quantum cloud computing. This classification was designed to align with our research questions and identify patterns within the field. The final categories were iteratively developed and refined based on initial findings to ensure that they reflected both the research questions and the current landscape of quantum cloud computing. As shown in Figure 2.2, the main categories of the literature include *Concepts and Overview, Computation Models, Applications and Use Cases, Providers and Platforms, Resource Management, Security and Privacy.* Each primary study was assigned to one or more of these categories based on its primary focus, enabling a systematic analysis in subsequent stages. Additional relevant studies were also included as supporting references to enhance the contextual foundation of the review.

Data Extraction and Analysis

In the final step, we analysed, extracted and synthesized data systematically from the classified studies to create a comprehensive systematic map and discussion in Section 2.5. This extraction included key details on study objectives, methodologies, findings, and identified research gaps.

2.5 Recent Advances in Quantum Cloud Computing

This section provides an overview of quantum cloud computing (QCC), addressing **RQ1** by examining the state-of-the-art paradigms, foundational principles and concepts of QCC in Section 2.5.1. Then, the rest of this section addressed **RQ2** by reviewing current research topics that are currently being explored in QCC and their recent advances.

2.5.1 Overview of Quantum Cloud Computing

Quantum Cloud Computing Paradigm

Quantum cloud computing represents an emerging computational paradigm that integrates the principles of quantum computing with cloud computing, enabling quantum computations to be executed on cloud-based quantum platforms [51, 53]. This model allows for quantum computational resources to be either publicly accessible via the Internet (public quantum cloud) or privately within a specific organization through a secured network (private quantum cloud), thereby democratizing access to quantum computing capabilities [90]. QCC allows users to perform quantum computing tasks without investing in their own quantum hardware. As depicted in Figure 2.3, users interact with these quantum computational resources through a cloud interface from classical computers, utilizing APIs to access software as a service. This paradigm facilitates the decomposition of large-scale quantum applications into microservices [91] and quantum functions (see Chapter 3), which can then be efficiently deployed and managed on a cloud-based platform. The cloud platform comprises essential components for orchestrating the quantum runtime and execution environment, resource allocation, storage, and networking. Ultimately, the quantum workload is processed in quantum computers and managed at remote data centers. A key innovation in QCC is the potential integration with the Quantum Internet [92, 93] and Quantum Key Distribution techniques [94]. This paradigm promises to revolutionize data communication by utilizing quantum principles for network communications, potentially eliminating the need for classical intermediaries in quantum data exchange. However, the realization of a mature Quantum Internet remains a future goal, with current challenges including the

development of robust quantum communication protocols and the integration of quantum and classical systems [92, 93]. Presently, the interconnection between cloud-based quantum computers and users still predominantly relies on classical Internet and computing technologies [95], leading to the emergence of hybrid quantum cloud computing models [90]. These hybrid models combine quantum computational power with classical networking and data processing, offering a pragmatic step towards fully realizing quantum cloud computing's potential.

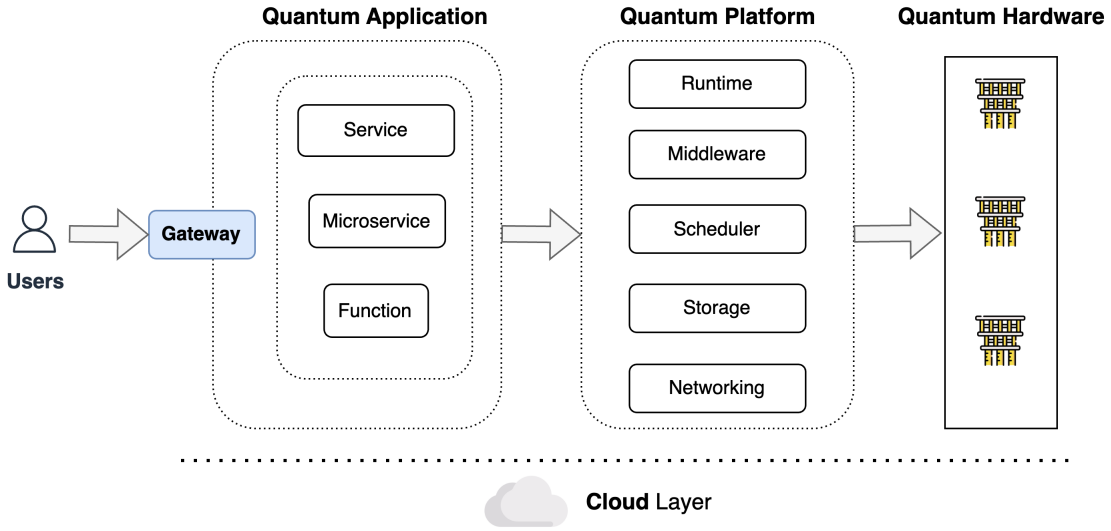


Figure 2.3: A high-level view of quantum cloud computing

Quantum cloud computing is still an emerging field; however, it is witnessing increasing interest from both industry and academic entities focused on developing and deploying these services. Singh et al. [96] highlighted the idea of quantum-cloud integration and implied this combination would be the potential approach of future quantum computing. One of the first trials to bring quantum resources to the cloud was conducted by the Center for Quantum Photonics (CQP) at the University of Bristol [97] in early 2016. They introduced a two-qubit optical-based quantum computer, accessible through the Internet for testing purposes, marking the initiation of quantum resources into the cloud domain. Subsequent to this, industry giants such as IBM [89], Amazon Web Services (AWS) [98], and Microsoft Azure [17] began to provide quantum computing services to the public as part of their cloud services offerings. These initiatives

underscored the viability and growing accessibility of quantum computing resources via cloud platforms. Karalekas et al. [19] detailed the architecture of a typical quantum cloud computing platform, specifying four essential components: 1) an apparatus for accommodating physical qubits, 2) a control system for manipulating the apparatus, 3) an executor for orchestrating the control system, and 4) a compiler for compiling quantum circuit for the executor. Faro et al. [20] introduced a concept for a hybrid quantum cloud computing architecture that incorporates middleware designed to facilitate the integration and management of quantum and classical computations. Furthermore, several works have been proposed to enhance the error robustness of cloud-based quantum computer interfaces. For example, Carvalho et al. [99] achieved substantial error reduction in quantum logic operations using optimized pulses through a cloud quantum computing interface, marking a pivotal advancement in enhancing the reliability and performance of quantum computations. These developments collectively represent the initial steps towards establishing a standardized approach for cloud-based quantum infrastructure across academia and industry, aiming to realize the full potential of quantum cloud computing. Due to the noisy intermediate-scale nature of available quantum machines [15, 100, 101], high-performance quantum simulators also need to be provided through the cloud to create the experiment environment for quantum-related research. Most quantum cloud providers offer access to large-scale quantum simulators for prototyping quantum applications. For example, IBM Quantum [89] offers access to various simulators, including the statevector simulator (32 qubits), QASM simulator (32 qubits), and the Clifford simulator, which supports up to 5,000 qubits for the Clifford circuit. Similarly, Amazon Braket and Azure Quantum provide access to numerous quantum simulators, such as state vector simulators, density matrix simulators, and tensor network simulators [7]. Additionally, Intel also offers a cloud-ready quantum simulator [102] with high-performance capabilities for simulating up to 42-qubit quantum circuits, supporting numerical studies, noise/error modeling, and parallel quantum device emulation.

[7] https://docs.aws.amazon.com/braket/latest/developerguide/braket-devices.html

Cost of Quantum Cloud Computing

Given the significant costs associated with cooling and manufacturing quantum computers [103, 104], their usage costs can typically be higher than utilizing resources on a classical computer. However, quantum computers can execute certain computationally hard tasks significantly faster than classical computers. It is advantageous to execute such tasks on a quantum computer in such cases where the overall cost to execute the task is significantly lower than on a classical computer. Besides the execution cost, one must consider latency constraints to access a quantum computer. Preparing a task for submission to a quantum computer can incur additional latency as well. A quantum computer's resources are typically shared across different users, so additional considerations, such as a job queuing latency or task compilation latency, would also need to be considered. In general, one has to carefully consider different constraints, such as the overall cost for execution and the overall latency to execute a task on a quantum computer, before choosing a quantum computer over a classical computer to perform a given task.

2.5.2 Quantum Cloud Computing Models

Table 2.3 maps studies that focus on various service and computation models within the context of quantum cloud computing in the selected literature, including Quantum Computing-as-a-Service (QCaaS), Quantum Serverless and Quantum Function-as-a-Service, and Hybrid Quantum-Classical Computing.

Table 2.3: Mapping of Studies Considering Quantum Cloud Computing Models

Covered aspects of QCC Models	**Studies**
Quantum Computing-as-a-Service (QCaaS)	[105–107]
Quantum Serverless and Quantum Function-as-a-Service	[91, 108, 109]
Hybrid Quantum-Classical Computing	[20, 28, 110–113]

Quantum Computing as a Service (QCaaS)

Quantum Computing as a Service (QCaaS), also referred to as Quantum as a Service (QaaS), represents a general quantum cloud service model within the evolving quantum cloud computing landscape. This model provides users with remote access to quantum computing resources, facilitating the exploration and application of quantum computing without the need to own and maintain quantum hardware. Stefano et al. [105] proposed the concept of Quantum-Algorithms-as-a-Service (QAaaS) for hybrid quantum-classical applications. QAaaS abstracts the quantum computing elements from the software development process, enabling developers to focus more on application logic rather than the intricacies of quantum computation. This abstraction is crucial for integrating quantum computing into broader IT infrastructures and for making quantum resources more accessible to a diverse range of users, including those with limited expertise in quantum mechanics. Obst et al. [106] highlight the challenges practitioners face when navigating these quantum-specific cloud service offerings, which vary significantly in capabilities and requirements. As such, QCaaS is pivotal in democratizing access to quantum computing resources on the cloud and accelerating the development of quantum applications across various industries [107].

Quantum Serverless

Serverless Quantum Computing is the emerging adoption of a serverless computing model for empowering quantum computing, making it more usable and reliable by abstracting the application deployment and infrastructure setup from users. Following this paradigm, users only need to focus on developing quantum applications using cloud-based services. The serverless applications are highly scalable and effectively utilize the resources, therefore, optimizing the total cost with the pay-per-use model. The serverless quantum computing model is well-suited to the hybrid quantum-classical application deployment. In 2021, IBM proposed the proof of concept for Quantum Serverless architecture, which allows the incorporation of quantum and classical tasks in a single application. They have claimed that direction is the future of quantum programming and have planned to introduce Quantum Serverless with intelligent orchestration, cir-

cuit knitting toolbox, and circuit libraries by 2023 [114].

Cloud-native technologies such as containerization, Application Programming Interface (API) gateway, continuous integration, and continuous delivery can be adopted to develop a quantum serverless platform. Garcia-Alonso et al. [108] proposed a proof of concept for Quantum API Gateway (QAPI), which adapts API gateway for exposing quantum services. They also demonstrated that a quantum service could not be deployed permanently inside a quantum computer like its classical counterpart. Instead, the quantum circuit needs to be compiled and sent to an appropriate quantum device for execution at the run time. They also proposed an execution time forecasting model to recommend the best quantum computer for each task and evaluated their proof-of-concept on Amazon Braket service. Dreher et al. [109] proposed a simple container-based prototype to encapsulate Qiskit codes into a Docker container on the local computer and interact with the cloud-based IBM Quantum service. In [91], the authors demonstrated their trials on deploying hybrid quantum microservices of the Travelling Salesman Problem (TSP) to reveal the limitations of quantum service engineering. They particularly deployed adiabatic and gate-based quantum implementations of TSP on the Amazon Braket platform using IonQ, Rigetti, and D-Wave hardware. The evaluation of this research showed many challenges of quantum service engineering with current NISQ devices, including the limited number of qubits, error rates, response time, and service costs. Focusing on the serverless integration aspect, we also proposed the concept of Quantum Function-as-a-Service (QFaaS) in Chapter 3, which is one of the first full-stack serverless function-as-a-service frameworks that support multiple quantum SDKs and quantum computing services. The evaluation of QFaaS with practical use cases on multiple quantum software kits and cloud vendors demonstrates the versatility and potential of bringing serverless techniques for accelerating quantum software engineering.

Hybrid Quantum-Classical Computing

The current development of almost all quantum cloud computing platforms still depends on classical architecture, generally referred to as the hybrid quantum-classical

cloud (HQCC) [19]. The sample workflow in an HQCC system is illustrated in Figure 2.3. In this architecture, users submit their quantum circuits through web API to the classical cloud services (such as IBM Cloud or Amazon Web Services). These circuits are queued before being forwarded to a quantum processor for execution. Finally, the results will be returned to the classical cloud and forwarded to the original user. All popular quantum cloud providers, such as IBM Quantum, Amazon Braket, and Azure Quantum, are employing this hybrid architecture for their quantum computing service. HQCC utilizes classical and quantum computing resources in a cloud environment to solve specific problems more efficiently by combining the strengths of both types of computing. The aim is to improve the system's performance by employing the unique capabilities of quantum computing, like quantum parallelism and entanglement, along with the power and flexibility of classical computing. Kumara et al. [110] propose Quantum Service-Oriented Computing (QSOC), which uses a model-driven approach to integrate quantum and classical components. Moguel et al. [28] emphasize the technical challenges of hybrid classical-quantum integration, using a case study on Amazon Braket to illustrate limitations in developing quality quantum services, including issues of platform independence and scalability in service-oriented quantum systems. Several studies, such as Britt et al. [111] and Maring et al. [113] also explore integration pathways for quantum processing units (QPUs) within high-performance computing (HPC) systems, identifying integration approaches based on infrastructure constraints and highlighting the importance of quantum interconnects in enhancing QPU performance within hybrid architectures. Faro et al. [20] proposed the middleware architecture for orchestrating hybrid quantum-classical computing systems. This heterogeneous architecture highlights the benefits of combining the advantages of both quantum and classical computing to optimize cloud-based resource utilization. Pfandzelter et al. [112] also proposed a Kernel-as-a-Service (KaaS) programming model to support heterogeneous workflows involving classical and quantum computation tasks. This prototype shows promising development of HQCC architecture design in the future.

2.5.3 Quantum Cloud Applications and Use Cases

Quantum cloud computing is pivotal in the development and deployment of quantum software. As quantum cloud computing is currently the only way to access quantum computers outside developers, almost all empirical studies have been conducted using quantum cloud resources. Table 2.4 provides the mapping of key studies to demonstrate practical use cases of quantum software in the cloud environments.

Table 2.4: Mapping of studies on a subset of key use cases and applications of quantum cloud computing

Category	**Studies**
Quantum Random Number Generation (QRNG)	[44–46]
Quantum Machine Learning (QML)	[47–50, 115]
Chemical Simulation and Modeling	[116–119]
Cybersecurity and Blockchain Applications	[120–123]
Optimization Problems	[124]

Quantum Random Number Generation (QRNG), an important application, has seen diverse implementations. For example, Huang et al. [44] enhanced cloud cybersecurity by integrating four types of quantum random number generators on Alibaba Cloud servers, combining their outputs for robust random number generation in high-security applications like Alipay. Li et al. [45] developed a quantum random number generator (QRNG) on IBM's cloud-based quantum computers, addressing the challenge of noise-induced randomness errors. Inspired by source-independent QRNG in optics, their method estimates errors in superposition state preparation, ensuring randomness even with readout errors, and optimizes parameters for increased random bit generation rate. Similarly, Kumar et al. [46] demonstrated a Quantum True Random Number Generator (QTRNG) on IBM's cloud-based quantum computing platform, showcasing a practical application of cloud quantum computing in cryptographic operations.

Quantum Machine Learning (QML) has emerged as another impactful application of quantum cloud computing, particularly in areas requiring vast data processing and complex model training. In this context, Gomez et al. [47] developed an Automated QML (AutoQML) framework in a classical-quantum hybrid cloud architecture, enabling parallelized hyperparameter exploration and model training. They demonstrated train-

ing a quantum Generative Adversarial neural Network (qGAN) for generating energy prices, showcasing the potential of QML in the energy economics sector. Gong et al. Similarly, [48] proposed a quantum k-means algorithm that employs quantum homomorphic encryption for security, demonstrating its effectiveness in reducing client-side computational burden and protecting data privacy in the cloud. Fadli et al. [49] presented the Quantum Integrated Cloud Architecture (QICA), aimed at enhancing quantum computing for aerospace applications, especially in satellite networks. They explore the computational benefits of a Hybrid Quantum-Classical machine learning architecture using IBM Quantum and Qiskit Machine Learning, demonstrating QICA's potential in advancing aerospace technology. Yarter et al. [115] explore the use of quantum neural networks (QNNs) within a quantum cloud paradigm for real-time audio classification, highlighting the feasibility and challenges of integrating quantum computing simulations with edge devices for processing live audio data. Additionally, Hibat-Allah et al. [50] leverage quantum cloud computing to compare the efficacy of classical and quantum generative models, particularly Quantum Circuit Born Machines (QCBMs). These applications highlight cloud-based QML deployment potential in handling large datasets more efficiently than traditional cloud-based machine learning by leveraging quantum parallelism and entanglement to speed up model training and data processing.

One of the most promising applications of quantum computing lies in simulating molecules and chemical reactions, tasks that are computationally intensive for classical systems. With quantum computational devices accessible through cloud platforms, researchers in computational chemistry can now perform simulation experiments more efficiently [118]. Notable milestones include Google's simulation of the hydrogen molecule's ground state using three qubits [116] and IBM Quantum's simulations of hydrogen, lithium hydride, and beryllium hydride (BeH2) with six qubits [117]. These advancements mark significant progress for new possibilities in drug discovery and materials science. Further illustrating this progress, Kim et al. [119] demonstrate the practical utility of quantum computing through experiments on a cloud-based superconducting quantum processor, showcasing its application in simulating quantum many-body systems beyond classical computational limits, thereby demonstrating quantum cloud com-

puting's potential in fields that require high-fidelity quantum simulations.

Quantum cloud computing also plays a crucial role in cybersecurity applications, optimization, and other domains. Fang et al. [120] developed cloud-assisted quantum protocols for enhanced security in applications like Anonymous Voting and Multiparty Private Set Intersection, leveraging quantum cryptography. Gong et al. [121] introduced a novel Quantum Homomorphic Encryption Ciphertext Retrieval (QHECR) scheme based on Grover's algorithm, addressing the challenge of efficiently retrieving homomorphically encrypted data in quantum cloud environments. Azzaoui et al. [122] proposed a Quantum Cloud-as-a-service for Smart Healthcare, combining Quantum Terminal Machines (QTM) and Blockchain for enhanced security and feasibility. Their architecture offers a scalable and secure solution for complex healthcare computations, highlighting the practicality and robust security of Q-OTP encryption. Similarly, Barletta et al. [123] propose a Quantum as a Service (QaaS) architecture that leverages quantum computing approach for enhancing security in smart cities, using quantum classifiers on the D-Wave Leap and IBM Quantum Cloud to detect threats in real-time. These advancements indicate that quantum cloud computing can offer more robust security than classical cloud computing in scenarios requiring high data integrity and confidentiality. In the domain of optimization and complex problem-solving, quantum cloud computing has demonstrated its utility in areas that surpass classical computing's capabilities. For instance, Zhang et al. [124] propose a quantum solution to the Exactly-1 3-SAT problem, leveraging the IBM Quantum platform and Grover's algorithm to experimentally verify the feasibility of solving NP-complete problems through quantum cloud resources.

In addition to these applications, quantum cloud computing is increasingly being explored for broader commercial sectors, such as logistics, financial risk analysis, and satellite communication, as outlined by Bova et al. [27] and Hassija et al. [125]. These studies suggest that quantum cloud services may soon provide capabilities that classical cloud infrastructure cannot efficiently handle, especially in fields requiring substantial computational power and high-speed data processing. This growing list of use cases demonstrates quantum cloud computing's unique potential to surpass traditional cloud computing, positioning it as a transformative tool for computationally demanding applications.

2.5.4 Quantum Cloud Providers and Platforms

Cloud-enabled quantum hardware providers

Most quantum hardware vendors offer cloud-based access to their computation resources. We summarize the latest developments of popular cloud-enabled quantum hardware vendors in Table 2.5. Several vendors, such as IBM Quantum [89], Rigetti [19], and IonQ [126], offer cloud-based computing services to their quantum computers, while other providers like Amazon Braket [98] and Azure Quantum [17] mainly collaborate with third-party hardware vendors to offer quantum computing services. Various technologies such as superconducting, trapped ions, neutral atoms, and quantum annealing have been employed in building quantum chips (or quantum processing units - QPUs). Each technique leverages distinct physical systems to implement qubits and quantum operations [103].

The superconducting technique leverages the properties of superconducting circuits to create qubits. This approach uses Josephson junctions to exploit quantum mechanical phenomena at macroscopic scales, enabling the manipulation of quantum states. Superconducting qubits are known for their relatively more straightforward integration into electronic systems and potential for scalability [138]. This approach can be considered the most popular technique for developing quantum computers by many leading companies such as IBM Quantum [89], Rigetti [19], Oxford Quantum Circuits (OCQ) [131], and Google [130]. In 2019, Google claimed quantum supremacy with their Sycamore 54-qubit superconducting processor, demonstrating it could perform a specific task in 200 seconds that would take the best classical supercomputer approximately 10,000 years, marking a significant breakthrough in quantum computing [139]. However, access to Google's quantum processor has remained limited to the general public compared to other hardware vendors. IBM Quantum offers a range of quantum processors available on their cloud-based platform, from 5 qubits to 433 qubits. In 2023, they released the Condor processor with 1,121 superconducting qubits and a high-performance 133-qubit Heron processor, marking a significant milestone in quantum hardware development [114]. IBM also demonstrated the utility in quantum computing with their 127-qubit superconducting processor, showcasing accurate expectation values measurement beyond

Table 2.5: A summary of quantum hardware vendors with cloud-based access.

Hardware vendor	**QPU technology**	**Max qubit count**	**Qubit topology**		**Supported SDKs**	**Cloud access**	
			PC	FC		Vendor's Cloud	External Cloud
Alpine Quantum Technologies [127]	Trapped Ion	20		✓	Qiskit, Pennylane	✓	-
Atom Computing [128]	Neutral Atom	1180	✓		-	-	-
D-Wave [129]	Quantum Annealing	5000*	✓		Ocean	✓	✓
Google [130]	Super-conducting	54	✓		Cirq	✓	-
IBM Quantum [89]	Super-conducting	1121	✓		Qiskit	✓	✓
IonQ [126]	Trapped Ion	36		✓	Qiskit, Cirq, Pennylane, tket, Q#, *etc*	✓	✓
Oxford Quantum Circuits (OCQ) [131]	Super-conducting	32	✓		tket	✓	✓
Pasqal [132]	Neutral Atom	100	✓		Pulser	✓	✓
Quantum Computing Inc (QCI) [133]	Photonics (Entropy)	949**		✓	Qatalyst	✓	-
Quantinuum [134]	Trapped Ion	56		✓	tket, lambeq	-	✓
QuEra [135]	Neutral Atom	256	✓		Braket	-	✓
Quantum Inspire (QuTech) [136]	Super-conducting, Solid-state spins	5	✓		cQASM, QI	✓	✓
Rigetti [19]	Super-conducting	84	✓		Quil	✓	✓
Xanadu [137]	Photonics	24	✓		Pennylane, Strawberry Fields, *etc*	✓	-

Notes. * Annealing qubit; **: Number of variables (qudit); ✓: Yes, -: Not available, PC: Partially connected, FC: Fully Connected (Data compiled from publicly available vendor sources as of July 2024).

classical computation capabilities, highlighting advancements in coherence, calibration, and noise management as critical enablers for pre-fault-tolerant quantum computing applications [119]. Other companies such as Rigetti [19], Oxford Quantum Circuits (OCQ) [131], and Quantum Inspire (QuTech) [136] also employed superconducting techniques for developing their cloud-based quantum hardware. QuTech also demonstrates their second hardware technique to develop a 2-qubit quantum processor based on single electron spin qubits in silicon [140, 141].

Another popular quantum hardware technique is trapped ion, which leverages ions (charged atoms) trapped in electromagnetic fields to function as qubits. Lasers are used to perform qubit initialization, manipulation, and measurement. This method is known for its long coherence times and high-fidelity operations. IonQ [126], Alpine Quantum Technologies (AQT) [127], and Honeywell Quantum (now part of Quantinuum) [134] are prominent vendors in this area, with all fully-connected qubit devices developed, demonstrating advanced quantum computing systems based on trapped ions. Besides, neutral atom technology, which uses lasers to trap and cool neutral atoms (atoms with no net electric charge) in a 2D or 3D array, also caught attention. This technology promises scalability and high qubit numbers, as Atom Computing [128] claimed that they successfully developed a 1,180-qubit quantum computer in 2023 and are preparing to make their system available for cloud-based access. Pasqal [132] and QuEra [135] are other notable examples of companies exploring quantum computing with neutral atoms to develop their cloud-based hardware. Furthermore, photonics is also a promising technique for developing quantum chips [142]. Quantum Computing Inc (QCI) introduced their Dirac-3 quantum system as a new approach in quantum computing through Entropy Quantum Computing (EQC) [133] to support higher-order interactions among qudits and leveraging entropy and noise, enabling operation at room temperature without cryogenic environment, and promising enhanced computing speed and capacity. Utilizing photonics, QCI's system advances beyond traditional two-level qubits by implementing multi-level qudits [143], with every single photon enabling up to 10,000 levels through various degrees of freedom, like polarization and orbital angular momentum [144]. This capability allows for the management of up to 949 photons/qudits, where each photon's degrees of freedom are harnessed to represent multiple levels, ef-

fectively turning each variable into a qudit with its level signifying the variable's value [133]. Research into EQC is underway, with the potential to significantly enhance the energy efficiency of quantum cloud computing by minimizing the cooling requirements of quantum computational systems. Another company, Xanadu, [137] has developed its nanophotonic quantum chip that can operate at room temperature, offering advantages in stability and scalability [145].

While superconducting, trapped ion, photonics, and neutral atom techniques are employed to develop gate-based quantum computers, the quantum annealing approach is designed to solve optimization problems by naturally finding a quantum system's ground state. It uses a quantum mechanical process to minimize energy states and find solutions to optimization problems. D-Wave Systems [129] is the most well-known company specializing in quantum annealing technology, offering quantum cloud computing services for specific optimization tasks. However, they recently added the gate-based quantum model to their roadmap towards developing devices for universal quantum computation.

The selection of quantum hardware technology plays a pivotal role in determining the efficiency and type of computations that can be executed, directly affecting quantum computing systems' scalability, coherence time, and error rates. As this field progresses, the refinement and interaction of these technologies will be fundamental in developing practical and broadly accessible quantum cloud computing services. Furthermore, the cloud-based provision of quantum computational resources simplifies the integration with high-performance computing (HPC) resources, promising enhanced computational capabilities and broader application potential.

Quantum cloud computing frameworks and platforms

Apart from cloud services offered directly from the hardware vendors, as discussed in the previous section, we summarize other cloud services and frameworks for quantum computing in Table 2.6.

Most cloud-based quantum computing platforms provide commercial quantum computing as a service. Major well-known cloud platforms such as Amazon Web Services

Table 2.6: Representative commercial services and studies on frameworks/platforms for quantum cloud computing

Platforms/ Framework	Type	Quantum Backends	Supported SDKs	Serverless Model	Quantum Computing Models		
					Simulation	Annealing	Gate-based
1Qloud [146]	CQS	1Qbit	1Qbit		✓		
Amazon Braket [98]	CQS	Rigetti, OCQ, IonQ, QuEra	Braket, Qiskit, Pennylane	✓	✓		✓
Azure Quantum [17]	CQS	Quantinuum, Rigetti, IonQ, Pasqal	Q#, Qiskit, Cirq,		✓		✓
Google Cloud [130]	CQS	Google, IonQ	Cirq		✓		✓
IBM Cloud [147]	CQS	IBM Quantum	Qiskit	✓	✓		✓
PlanQK [148]	CQS	IBM Quantum, Amazon Braket, Azure Quantum	Qiskit, Pennylane	✓	✓	✓	✓
QEMIST [149]	CQS	1Qbit	OpenQEMIST		✓		
QFaaS (Chapter 3)	OSF	IBM Quantum, Strangeworks	Qiskit, Cirq, Q#	✓	✓		✓
QuantumPath [22]	CQS	IBM Quantum, Amazon Braket, D-Wave, QuTech	Qiskit, Ocean, Braket, Q#		✓	✓	✓
Strangeworks [150]	CQS	IBM Quantum, Amazon Braket, Azure Quantum, Hitachi, Toshiba,...	Qiskit, Braket, Rigetti	✓	✓	✓	✓

Notes. CQS: Commercial Quantum Service, OSF: Open-source Framework, QSim: Quantum simulator, ✓: Yes

(AWS), Microsoft Azure, Google Cloud, and IBM Cloud have started offering quantum computing services in their cloud ecosystem, opening opportunities for integrating quantum computation tasks with classical ones. IBM offers access to its self-developed quantum computation resources via IBM Cloud [147] and IBM Quantum platform [89], while AWS and Azure provide cloud services and platforms to access other quantum hardware vendors, such as Rigetti, IonQ, and Quantinuum, OCQ, Pasqal, and QuEra. These quantum cloud platforms also offer a range of Software Development Kit (SDK) for designing, developing, and evaluating quantum applications. For example, IBM Quantum provides users with multiple tools, such as IBM Quantum Composer and IBM Quantum Lab, for designing, visualizing, executing, and analyzing quantum applications. Amazon Braket, Azure Quantum, and Google Cloud also offer their quantum SDKs, naming Braket, Microsoft Quantum SDKs (along with Q# programming language), and Cirq, respectively. Strangeworks collaborates with almost all major quantum computing hardware and software vendors to create a comprehensive cloud platform that offers users access to nearly all available quantum hardware and other software platforms.

Besides, several cloud-based quantum software-oriented frameworks have been proposed. Hevia et al. [22] proposed the QuantumPath platform to support the development of quantum applications. This framework integrates multiple visual editors to aid the design of quantum circuits and supports different SDKs, with the tendency to become an agnostic software framework for quantum computing. Our proposed QFaaS framework (see Chapter 3) is another open-source quantum software framework that supports developing serverless quantum function-as-a-service applications. QFaaS incorporates multiple quantum SDKs and programming languages and can be extended to work with different quantum cloud providers, such as IBM Quantum and Amazon Braket. Furthermore, PlanQK [148] is a comprehensive quantum platform and ecosystem to support the development of quantum workflow applications with a vision towards establishing a quantum application marketplace. Besides, several orchestration tools, such as Orquestra [151] and QuantMe [152], are proposed for integrating classical components and workflow with quantum algorithms. Xin et al. [153] also introduced their quantum cloud computing service, named NMRCloudQ, in which quantum hard-

ware is developed based on a nuclear magnetic resonance (NMR) spectrometer. Apart from offering quantum computing services, most quantum cloud platforms and frameworks provide simulation environments for prototyping and developing quantum applications. Several cloud platforms, such as 1QCloud [146] and QEMIST [149], are dedicated to providing quantum simulation services for specific problems such as optimization and quantum chemistry.

2.5.5 Quantum Cloud Resource Management

Similar to traditional cloud computing, quantum resources must be efficiently managed and allocated as quantum cloud computing continuously advances. However, quantum cloud resource management is facing more complicated challenges. The first challenges arise due to the heterogeneity of quantum resources in terms of qubit numbers, qubit connectivity, error rates, and quantum processor speed. Indeed, each quantum hardware technology has different advantages and limitations in its performance and scalability. For example, trapped ion quantum devices can achieve higher quantum volume (qubit quality) but face difficulty achieving high processing speed (CLOPS) [32, 154]. In contrast, the opposite can be seen in spin-based quantum devices. Second, quantum resources are almost fixed, i.e., they cannot be divided and scaled flexibly in the same ways as classical ones. We can create multiple virtual machines or containers inside a single classical host machine or a cluster of machines as isolated environments for executing independent tasks concurrently. However, no corresponding techniques for quantum resources have been proposed yet. Besides, each quantum task has different requirements, which are unknown or unpredictable, to allocate an appropriate quantum device for execution. The uncertainty of quantum task requirements and the fixed resource amount of quantum computers accelerate the complexity of quantum resource management problems and result in underestimated or overestimated resource allocation. While this is an emerging topic in quantum cloud computing, several studies have begun investigating various aspects of quantum cloud resource management problems, as shown in Table 2.7.

Table 2.7: Mapping studies in resource management for quantum cloud computing

Category	Approach	Studies
Task Scheduling/Placement	Statistical Analysis	[31, 155]
	Rule-based Policy	[156]
	Round Robin Algorithm	[157, 158]
	Greedy Algorithm	[159]
	Reinforcement Learning	[160]
Task Orchestration	Parallel Task Allocation	[161–164]
	Load Balancing	[165]
	Middleware	[20]

Quantum Task Scheduling/Placement

Quantum task scheduling, or placement, involves efficiently assigning quantum tasks to available quantum processors based on factors like resource availability, execution time, and hardware compatibility to maximize computational efficiency and minimize resource contention in the quantum cloud. When designing a comprehensive resource allocation for the quantum cloud computing paradigm, multiple factors must be considered. In practice, IBM Quantum uses fair-share scheduling algorithm[8] to ensure fairness in their Open plan quantum cloud service, which is freely public for the research community. When a new quantum job arrives, it is placed in the waiting queue from all users, and its order to be executed will be dynamically determined using the fair-share algorithm. Ravi et al. [155], analyzed multiple quantum jobs and resource utilization characteristics using IBM Quantum Cloud services for two years. They evaluated the significance of the execution times, waiting times, and compilation times of quantum circuits, fidelity, error rates, and the utilization of quantum machines. This study gave insights into numerous factors affecting quantum task execution on quantum cloud devices. In the following study [31], they proposed a prediction model to predict the fidelity of quantum computers and queueing time for each device based on historical data of IBM Quantum computers. To schedule incoming jobs to an appropriate quantum computer, this work compiles and transpiles the corresponding quantum circuit for each quantum computer, extracts the key features of the circuit, and maps it with the characteristics of the machine. Then, using correlation coefficient techniques, the best-

[8]https://quantum-computing.ibm.com/lab/docs/iql/manage/systems/queue/

suited quantum computer will be selected for job execution. Although the proposed method, which uses statistical analysis, is straightforward, this work can be considered the first work on resource management for quantum cloud computing. Salm et al. [156] presented the NISQ Analyser as a novel tool designed to optimize the quantum backend selection process by evaluating the specific requirements of quantum tasks. It assesses essential parameters, including qubit requirements, circuit depth, and gate complexity, to ensure the most efficient alignment of quantum algorithms with available quantum computing resources, thereby enhancing the practicality of quantum computing applications. In addition to proposing the quantum serverless framework, we also introduced a policy-based quantum backend selection that prioritizes fidelity and execution speed based on properties of quantum tasks and available quantum computation backends (see Chapter 3). Focusing on the quantum network resource aspects, Cicconetti et al. [157] presented another resource allocation technique for distributed quantum computing. They leveraged the Weighted Round Robin (WRR) algorithm to design the network resource allocation technique for quantum applications. The key idea is to pre-calculate the weight of all traffic flows in each quantum application and use the round-robin strategy to assign network resources. They also proposed a quantum network provisioning simulator for the evaluation and showed the trade-offs between fairness and time complexity of the network resource allocation algorithm. Similarly, Zhang et al. [158] proposed a new task scheduling scheme based on a round-robin algorithm for cloud-based quantum computing platforms, focusing on user and task classification. This approach aims to reduce waiting times for high-priority users, thereby enhancing their experience and addressing the challenge of limited access to high-quality quantum computing resources. Furthermore, Cicconetti et al. [159] also proposed a Greedy-based algorithm to achieve fair resource allocation among quantum computation nodes, ensuring service differentiation across a networked quantum infrastructure. Recent studies have applied reinforcement learning techniques to optimize task scheduling efficiency. We also proposed the DRLQ framework (see Chapter 5) and QFOR framework (see Chapter 6), which leverages deep reinforcement learning algorithms, including Deep Q Network (DQN) and Proximal Policy Optimization (PPO) for dynamic task placement. These method enhances adaptability to changing resource conditions, reducing task comple-

tion time and rescheduling attempts while maximizing the quantum execution fidelity. Similarly, Li and Zhao [160] utilized reinforcement learning with a Graph Convolutional Network (GCN) model to improve resource allocation through device allocation and circuit deployment, achieving a significant reduction in quantum task execution time.

Quantum Task Orchestration

Quantum task orchestration is a critical aspect of quantum resource management, focusing on the efficient orchestration of quantum (and classical) tasks across multiple computation resources in cloud environments. Due to the error-prone nature of NISQ devices, only quantum circuits with a limited number of qubits can be precisely executed. It leads to the common problem of under-utilizing quantum resources as only more qubits are wasted, and only one circuit can be executed each time. Therefore, parallel processing and multi-programming techniques are essential for maximizing the resource utilization for quantum cloud computing [161]. Das et al. [162] presented their studies on quantum multi-programming and proposed several solutions for enabling the multi-programmed NISQ devices. Specifically, they proposed three methods, including 1) a qubit partitioning method to ensure fairness in qubit allocation, 2) a Delayed Instruction Scheduling policy to reduce the multi-program interference, and 3) an Adaptive Multi-Programming to allow flexible switching between single- and multi-programming. Ohkura et al. [163] proposed *palloq*, a parallel allocation protocol for quantum circuits, which accelerates the performance of quantum multi-programming in NISQ devices. They also considered error detection using randomized benchmarking methods. Nguyen et al. [164] developed a full-stack software framework to enable parallel quantum computing for hybrid quantum workloads. This framework supports three modes to distribute quantum tasks, including Message Passing Interface (MPI) protocol and local and cloud-based quantum accelerators, which are built based on nitrogen-vacancy (NV) centres in the diamond. Alvarado-Valiente et al. [165] propose a quantum orchestrator designed to streamline the execution of quantum circuits across multiple service providers, facilitating load balancing and simplifying resource access for developers working with cloud-based platforms, including Amazon Braket and IBM Quan-

tum. Faro et al. [20] propose a heterogeneous computation architecture that integrates quantum and classical computing, demonstrating how a middleware solution can facilitate seamless task orchestration and improve computational efficiency across quantum and classical resources.

Simulation Toolkits for Quantum Cloud Resource Management Problems

In the evolving landscape of quantum cloud computing, where accessibility to physical quantum resources is limited, simulation toolkits like CloudSim [34] in the classical domain are pivotal for designing and evaluating quantum cloud resource management algorithms. We also proposed iQuantum (see Chapter 4), a toolkit for modelling and simulating resource scheduling algorithms on the cloud. This toolkit is particularly beneficial for simulating quantum computing scenarios that integrate cloud-based quantum resources, focusing on job scheduling and hybrid quantum-classical task orchestration. This simulator highlights the importance of standardization and the creation of such simulators and toolkits for cloud-based quantum computing, as well as the extension to edge computing environments.

2.5.6 Quantum Cloud Security and Privacy

As quantum cloud computing is in its infancy, a limited number of works in the literature focus on its security and privacy. On the contrary, many studies pay extensive attention to quantum security [43, 61, 166–169], post-quantum cryptography [170, 171], and quantum-safe techniques [172].

Security and data privacy threats of quantum cloud computing

In quantum cloud computing, the integration of quantum services via cloud platforms can introduce several security and privacy challenges. The accessibility of cloud-based quantum computing services potentially enables adversaries to exploit these resources for unauthorized access to sensitive data without needing proprietary quantum hardware. Initial attack vectors can target the exploitation of quantum resources to compro-

mise non-quantum-safe infrastructures or the theft of credentials securing cloud-based quantum services, enabling service manipulation or compromise [173]. The necessity for secure data transfer channels in remote quantum computing emphasizes the importance of network authenticity, especially in cloud environments where physical network characteristics are abstracted [174]. Malicious third-party cloud computers and the vulnerabilities they introduce, such as hacking, data leaks, and insecure APIs, are well-known threats in the classical domain [175]. As the quantum computing ecosystem expands in scope and practicality, an increase in providers, including potentially unreliable third-party vendors, offering quantum computing as a service is anticipated. This situation presents multiple security challenges. Some emerging providers might attract customers with the promise of cheaper access to quantum computing resources and reduced waiting times. However, if the security measures of these services are not rigorously assessed, they could introduce significant risks. This scenario is similar to the risk of employing untrusted compilers for constructing quantum circuits, which could potentially expose or compromise proprietary information [173, 176]. Additionally, quantum-specific challenges such as crosstalk noise present significant security threats, as adversaries can exploit it to interfere with and manipulate quantum computations on cloud-based platforms [177]. Furthermore, the shared nature of cloud quantum computing resources raises concerns over intellectual property (IP) security. Quantum computers are susceptible to attacks like fault injection in multi-tenant computing environments. Compute performance can also be degraded for denial-of-service attacks if third-party calibration services provide inaccurate error rates of qubits or if the qubits are miscalibrated. Access to trustworthy quantum computing providers often involves long wait times and high costs, tempting users to opt for more affordable, readily available alternatives that may compromise security, risking intellectual property theft or tampering with computation results. The rise of efficient yet unreliable compilation services poses a significant threat to the integrity of quantum circuits by potentially introducing malicious code [43, 178]. Baseri et al. [169] also provides a comprehensive analysis of quantum-induced cybersecurity risks for critical infrastructure and cloud-based environments, proposing a security framework that highlights the need for proactive quantum-resilient strategies across cloud service layers. Despite quantum computing's

capacity to address strategically critical problems involving sensitive data, its security and privacy aspects remain underexplored.

Recent advances in securing and privacy-preserving quantum cloud computing

Advancements in techniques for securing and ensuring data privacy for quantum cloud computing draw from quantum-specific and traditional cloud security measures, focusing on robust authentication, verifiability, trustworthiness, and malicious attack detection and mitigation. Table 2.8 maps relevant studies on securing and privacy-preserving quantum cloud computing.

Table 2.8: Mapping studies in securing and privacy-preserving quantum cloud computing.

Problem	Approach	Studies
Enhancing privacy and verifiability	Blind quantum computation	[179, 180]
	Cryptographic verification	[181]
Securing remote quantum execution	Remote state rotation via classical communications	[182]
	Quantum key distribution (QKD)	[183]
Enhancing trust and computation fidelity	Quantum Physically Unclonable Function (QuPUF)	[184]
	Ensemble of Diverse Mappings (EDM)	[185]
Data encryption retrieval	Grover's algorithm-based scheme	[121]
Securing inputs for QML tasks in QCC	Subcircuit encryption	[186]
Malicious error rate changes detection	Test points injection	[187]
Adversarial tampering	Iteration distribution & parameter re-initialization	[178]

Several studies considered privacy and verifiability for quantum computations in cloud computing environments. Li et al. [180] introduced a verifiable quantum cloud computation scheme leveraging blind computation, addressing privacy and verifiability for clients in cloud-based quantum computing with a novel use of cluster states, enhancing the scheme's suitability for cloud architectures and promoting data confidentiality. Distributed approaches for protecting quantum circuits through classical interfaces have also been proposed, specifically for prime factorization, relying on shared entanglement. Huang et al. [179] conducted an experiment demonstrating blind quantum computing for entirely classical clients, showing that clients without quantum devices can perform computations securely on quantum servers using entanglement, including tasks like factorization with built-in verification for server honesty and correctness, marking a crucial step towards secure cloud quantum computing. Utilizing a cryptographic verifi-

cation approach, Chen et al. [181] proposed a novel scheme to ensure the authenticity of quantum computing within cloud services, differentiating between actual quantum processing and classical simulation. Another aspect that has been explored is securing communications for executing tasks on quantum cloud platforms. QEnclave [182] introduces a secure, classical-controlled cloud-based quantum hardware for remote quantum process execution, bringing classical secure enclave principles to quantum computing and ensuring user privacy. In a recent study, Huang and Emeakaroha [183] propose a framework for distributed quantum calculations over a multi-cloud architecture, using the Quantum Key Distribution (QKD) protocol to secure inter-cloud communications and enhance the reliability of data transfers across heterogeneous quantum service providers.

Besides, few studies proposed solutions for enhancing trust and computation fidelity for quantum cloud tasks. Phalak et al. [184] explored security concerns in cloud-based quantum computing, proposing Quantum Physically Unclonable Functions (QuPUF) to ensure trustworthiness and security. Addressing the risk of users being allocated inferior quality quantum hardware by untrustworthy third parties or due to malicious tampering, they developed QuPUF variants based on superposition and decoherence and tested them on IBM Quantum hardware. Their solution demonstrates the feasibility of using QuPUFs to distinguish between quantum hardware, enhancing user trust in cloud quantum computing environments. Tannu et al. [185] proposed the Ensemble of Diverse Mappings (EDM) method to counteract the vulnerability of NISQ computers to correlated errors, improving the reliability of quantum computations. By diversifying qubit allocations across multiple trials, EDM decreases the likelihood of consistent errors, thereby enhancing the accuracy of output inference. Their approach demonstrated a significant improvement in inference quality compared to existing mapping algorithms. Adapting classical security technologies like homomorphic encryption for quantum resilience is crucial, as conventional encryption might falter against quantum capabilities. Considering cloud-based quantum data encryption ciphertext retrieval, Gong et al. [121] present a Grover algorithm-based quantum homomorphic encryption scheme designed for secure ciphertext retrieval in quantum cloud environments, enhancing data privacy through efficient encrypted search without requiring client-server interaction. Wang et

al. [186] introduced PristiQ, a cross-layer framework for data security in Quantum Machine Learning (QML) under Quantum-as-a-Service (QaaS), utilizing a combination of encryption circuits and reinforcement learning to secure data encoding while maintaining model performance.

Furthermore, quantum cloud attack detection and mitigation have been studied recently. Acharya et al. [187] addressed the reliability challenges in NISQ computers by proposing a method to detect malicious changes in error rates that could alter quantum circuit outputs. Their lightweight approach involves inserting test points into circuits to monitor error rates relative to qubit allocations, employing superposition, classical, and un-compute tests for side-channel analysis, and offering a security enhancement for NISQ computing. Upadhyay and Ghosh [178] propose a security framework to counteract adversarial tampering in hybrid quantum-classical computations by distributing computations across trusted and untrusted hardware, along with adaptive iteration distribution and parameter re-initialization to enhance resilience against suboptimal tampered outputs.

2.6 Limitations and Threats to validity

Systematic mapping studies inherently suffer from certain limitations due to the research design [55, 188]. In this section, we discuss possible limitations that may threaten the validity of the results.

Completeness and Selection Bias: A significant challenge in conducting systematic mapping studies is ensuring comprehensive coverage of relevant literature while mitigating selection bias. In this study, we implemented a rigorous search protocol utilizing five major databases, which yielded a substantial corpus of related papers. However, it is essential to acknowledge the potential for unintentional omissions, particularly regarding studies that lie at the intersection of quantum computing and cloud computing. Furthermore, while our search strings were optimized to capture a maximum number of known quantum cloud computing studies, there remains a possibility that some research not explicitly proposing solutions for quantum cloud computing may have been overlooked. This limitation underscores the complexity of defining precise boundaries

for emerging interdisciplinary fields. We have also omitted a search for other underlying aspects of quantum cloud computing, such as quantum networks and communications, quantum hardware, and quantum error mitigations, as these can be considered dedicated research domains and have been studied in other systematic studies.

Generalization: Besides, the threat of redundancy and limited generalizability has also been considered. By conducting the rigorous literature selection and quality assessment in our review protocol (see Section 2.4), we minimise the possibility of duplicated publication in the final selected papers in the review. Additionally, we examined and derived the publication trend to identify the field's key research areas. However, as quantum cloud computing is an emerging area, it is possible to have more subtopics that can be explored in the near future, such as virtualisation and containerisation for quantum computing resources.

Data Extraction and Analysis: To enhance the comprehensiveness of our analysis regarding the practical status of quantum cloud computing, we conducted a thorough examination of the characteristics of available quantum cloud services offered by industry vendors. However, it is important to note that our findings may not fully reflect the most recent developments for future readers due to the rapid advancements in quantum computing, particularly in quantum hardware. This limitation is inherent to the dynamic nature of the field. Furthermore, it is worth acknowledging that some service providers did not explicitly disclose comprehensive information about their offerings. Consequently, there is a possibility that certain aspects of quantum cloud computing may have been inadvertently omitted from our analysis. This potential information gap underscores the challenges of conducting exhaustive reviews in emerging technological domains. Another potential issue is the depth of analysis of each paper. However, as a systematic mapping study, we aim to provide a holistic view of the big picture of quantum cloud computing and make it the cornerstone for further systematic review studies with more depth regarding each aspect of quantum cloud computing.

2.7 Summary

In this chapter, we carried out a comprehensive review of recent advances and open problems in quantum cloud computing. We introduced emerging concepts and models of quantum cloud, such as hybrid classical-quantum cloud, quantum computing as a service, and quantum serverless. Besides, we summarized and mapped studies in each aspect to highlight the importance of combining quantum and cloud computing to accelerate quantum engineering. We discussed the potential applications of quantum clouds in different areas. We explored key research problems in quantum clouds, including resource management, distributed computation, and quantum cloud security. We also highlighted several research challenges to advance the quantum cloud. Although there are still challenges to be addressed, quantum cloud computing has the potential to drive innovation and bring significant benefits to realize the advantage of practical quantum computing. We believe that quantum cloud computing is a promising strategy to revolutionize how quantum computing is used to solve intractable problems for classical computing in the coming years. This calls for access to quantum hardware, robust network connectivity, quantum software tools, security measures, scalability, interoperability, and cost-effectiveness.

Chapter 3

A Serverless Function-as-a-Service Framework for Quantum Computing

*This chapter proposes QFaaS, a holistic **Q**uantum **F**unction-**a**s-**a**-**S**ervice framework, which leverages the advantages of the serverless model, DevOps lifecycle, and the state-of-the-art software techniques to advance practical quantum computing for next-generation application development in the NISQ era. Our framework provides essential elements of a serverless quantum system to streamline service-oriented quantum application development in cloud environments, such as combining hybrid quantum-classical computation, automating the backend selection, cold start mitigation, and adapting DevOps techniques. QFaaS offers a full-stack and unified quantum serverless platform by integrating multiple well-known quantum software development kits (Qiskit, Q#, Cirq, and Braket), quantum simulators, and cloud providers (IBM Quantum and Amazon Braket). In this chapter, we propose the concept of quantum function-as-a-service, system design, operation workflows, implementation of QFaaS, and lessons learned on the benefits and limitations of quantum serverless computing. We also present practical use cases with various quantum applications on today's quantum computers and simulators to demonstrate our framework's capability to facilitate the ongoing quantum software transition.*

3.1 Introduction

Recent breakthroughs in quantum hardware development are creating opportunities for its use in many applications, making it becoming a critical future technology attracting

This chapter is derived from:

- **Hoa T. Nguyen**, Muhammad Usman, and Rajkumar Buyya, "QFaaS: A Serverless Function-as-a-Service Framework for Quantum Computing", *Future Generation Computer Systems (FGCS)*, Volume 154, Pages: 281-300, ISSN: 0167-739X, Elsevier Press, Amsterdam, The Netherlands, May 2024

significant investment at the global level [57]. The rapid advancements in quantum hardware trigger more investments in quantum software engineering and quantum algorithms development to maximize the practical use of quantum computers. Quantum computers have demonstrated their abilities to solve many complex problems which are challenging to tackle with classical supercomputers, such as molecule simulations [117], machine learning [68], cryptography [189], and finances [190]. Some notable algorithms have been proposed in the last few decades, such as Deutsch-Jozsa's [191], Shor's [2], and Grover's [3]. Some of these algorithms have been directly applied to problems of practical relevance, albeit at the proof-of-concept level, due to quantum hardware limitations.

Despite the inevitable prospect of quantum computing for future-generation computation, quantum software engineering is an early-emerging domain with numerous open challenges. First, the development of quantum applications is complicated and time-consuming for software engineers, mainly because of the requirement for prior quantum knowledge. Indeed, quantum programming is underpinned by the principles of quantum mechanics, which are quite different from the traditional models. For example, the basic difference between quantum and classical computing comes from their fundamental unit: a classical bit has one state, either 0 or 1, whereas a quantum bit (or qubit) could also be placed in a *superposition* state, i.e., a combination state of 0 and 1 [69]. A software engineer must overcome the hurdle of learning quantum mechanics to develop quantum applications.

Second, quantum computing services are still heavily relying on classical servers for circuit compilation due to the lack of quantum data storage methods in a quantum computer. In classical computing, the compiled binaries of applications can be installed or deployed on persistent storage mediums, allowing them to remain available for re-execution without the need to recompile the code for each use. By contrast, quantum computing services currently do not possess an analogous capability for persistent storage of quantum programs. Consequently, when quantum computation is invoked, it necessitates a classical driver to compile a quantum circuit tailored to the specific quantum processor [108]. This circuit is then loaded into the quantum processing unit (QPU) for execution, with the outcome derived from one or more iterations (shots). Addition-

ally, classical computing resources can be requisitioned for post-processing and storing the results of the quantum execution.

Besides, the existence of many quantum software development kits (SDKs) tightly coupled with specific vendor platforms presents challenges such as data lock-in and limitations in migrating software deployments across various platforms. Each SDK and programming language has distinct requirements for environment configuration, syntax, and interaction methods with their respective quantum simulators and computers. Additionally, there is no well-known standard or lifecycle in quantum software engineering similar to practices like Agile and DevOps in the traditional realm [192]. Although some efforts have been made to deal with this issue, such as preliminary approximations based on Model-Driven Engineering (MDE) [193], a comprehensive solution to establish a unified quantum software platform capable of seamlessly working with multiple quantum SDKs and providers remains necessary.

Presently, quantum computing resources remain constrained within the noisy intermediate-scale quantum (NISQ) era [15], characterized by limitations in both the quantity and quality of available qubits. Also, access to quantum computing services is exclusively facilitated through cloud-based platforms, which often incur substantial costs. Indeed, the most widely adopted way to access today's quantum computers is through a cloud service from external vendors, such as IBM Quantum [16], Amazon Braket [98], and Azure Quantum [17]. To ensure a mutually advantageous relationship between quantum cloud providers and clients, it is crucial to establish a win-win paradigm that maximizes the benefits of quantum computing while optimizing both budgetary and quantum resource considerations. In this context, the current pay-per-use pricing model offered by cloud vendors must be complemented by an appropriate computing model that effectively balances the advantages for both parties involved.

Furthermore, the ongoing evolution of new computational paradigms raises a strategic decision in the transition towards incorporating quantum computing: determining the extent of integration within existing classical systems. It is currently impractical to envision an entire replacement of quantum systems for all computational tasks due to efficiency considerations. For example, basic arithmetic operations, such as summing two integers, are far more efficiently executed on classical computers. Thus, the fore-

seeable future points to a hybrid approach, where quantum computing is employed for specific, complex problems that are beyond the capabilities of classical computers, while routine tasks remain with classical solutions. This integration strategy is particularly applicable during the NISQ era, where leveraging both technologies' strengths is essential for optimizing performance [59].

3.1.1 Our Contributions

To address the research challenges highlighted above, we propose **QFaaS** - a novel and versatile ***Q**uantum **F**unction-**a**s-**a**-**S**ervice* framework without the vendor lock-in problem. The major contributions and novelty of our proposed work are:

- QFaaS framework represents a holistic serverless framework for quantum computing, enabling the seamless integration of quantum computation within established classical systems. We tackle the challenges associated with platform and data lock-in in serverless quantum computing by incorporating multiple quantum SDKs, namely Qiskit, Cirq, Q#, and Braket, to perform the hybrid computation on classical computers, quantum simulators, and quantum computers provided by multiple cloud vendors, including IBM Quantum and Amazon Braket.

- We introduce the concept of quantum functions, quantum function-as-a-service, sample workflows, which involve both classical and quantum computation, and discuss their benefits for quantum software development. Our framework provides a comprehensive reference for quantum software engineers and industries to design and develop their service-oriented quantum platforms.

- We propose the practical quantum serverless system architecture with six extendable system layers, a core API set, and a quantum function programming library. In addition to the main framework architecture, we propose an adaptive quantum backend selection policy that determines the most appropriate quantum computation system for executing the quantum function. Besides, we present a caching-based policy to mitigate the cold start problem, which helps to reduce the latency of quantum function invocation. We utilize the state-of-the-art software and sys-

tem techniques for quantum software development, such as containerization and DevOps lifecycle. By leveraging Kubernetes as the underlying orchestration, our framework is portable and scalable for further advancement.

- We empirically validate the proposed framework through the practical implementation of all its components, following an open-source-oriented approach. Additionally, we showcase two sample operational workflows within the system, catering to both quantum software engineers and end-users. These workflows demonstrate how our framework can effectively support the development and utilization of hybrid quantum-classical applications.
- We conduct thorough experiments using various quantum algorithms on different quantum simulators and quantum computers to evaluate the performance of our framework. Through this evaluation, we provide practical insights into the current state of NISQ devices and discuss the limitations and lessons learned from the serverless quantum computing model.

The rest of this chapter is organized as follows: Section 3.2 introduces the current state-of-the-art in quantum software engineering, the quantum computing as a service (QCaaS) model, and serverless computing. Section 3.3 proposes the concept of quantum functions and quantum function-as-a-service. Then, Section 3.4 presents the details of the QFaaS framework, including system architecture and main components. Section 3.5 describes the design and implementation of the QFaaS framework. Then, Section 3.6 demonstrates the operation and validation of QFaaS with practical use cases. We discuss the benefits of using QFaaS for quantum service-oriented application development and lessons learned on the limitations of the quantum serverless approach in Section 3.7. Section 3.8 discusses the related work and compares our framework's advantages with existing work. Finally, we conclude the chapter in Section 3.9.

3.2 Background

This section outlines the state-of-the-art development of quantum software engineering, quantum computing service model, and serverless quantum computing. A brief intro-

duction to quantum computing and gate-based quantum model for broad readers can be found in other well-known books such as [69], [194]. It is important to note that our focus in this work is on gate-based quantum computing SDKs and platforms due to their broad applicability and active development by many well-known quantum cloud providers, such as IBM Quantum [16] and Amazon Braket [98].

3.2.1 Quantum SDKs and programming languages

Some popular quantum software development kits (SDKs) and programming languages that originated from well-known companies are:

- **Qiskit** [195] (by IBM) is one of the most popular Python-based open-source SDKs for developing gate-based quantum programs. It offers a wide range of additional libraries and support tools, particularly tailored to the IBM Quantum platform [16].

- **Cirq** [196] (by Google) is a prevalent open-source SDK for quantum programming. This SDK supports writing, manipulating, and optimizing quantum gate-based circuits. Cirq programs can run on built-in simulators and Google's quantum processors.

- **Q#** [197] (by Microsoft) is a new programming language from Microsoft for developing and executing quantum algorithms. It comes along with Microsoft's Quantum Development Kit, which includes a set of toolkits and libraries for quantum software development.

- **Braket** [98] (by Amazon) is an emerging Python-based SDK to interact with Amazon Braket service [98]. This SDK provides multiple ways to prototype and develop hybrid quantum applications, then run them on simulators or quantum computers.

Besides, there are numerous quantum languages and SDKs proposed by research groups over the world, such as Forest and pyQuil by Rigetti [198], Strawberry Fields [199] and PennyLane [200] by Xanadu, Quingo [201], QIRO [202], and qcor [203].

3.2.2 Current state of quantum computing: The NISQ era

John Preskill proposed the *"Noisy Intermediate-Scale Quantum (NISQ)"* term in 2018 [15] to describe the current state of quantum computers. This term indicates two characteristics of today's quantum devices, including *"noisy,"* i.e., unstable and error-prone quantum state due to the affection of various environmental actions, and *"intermediate scale,"* i.e., the quantum volume is at the intermediate level, with about a few tens of qubits [192]. Due to the NISQ nature, the typical pattern for developing today's quantum programs combines quantum and classical parts [192]. In this hybrid model, the classical components are mainly used to pre-process and post-process the data. In contrast, the remaining part is sent to quantum computers for computation. The quantum execution parts are repeated many times and measure the average values to mitigate the error caused by the noisy quantum environment. An example of the hybrid quantum-classical model is Shor's algorithm [2] to find prime factors of integer numbers. In this algorithm, we execute the period-finding part, leveraging the Quantum Fourier Transform on quantum computers and then performing the classical post-process to measure the prime factors based on the outcome of the quantum part.

3.2.3 Quantum Computing as a Service (QCaaS)

Today's quantum computers are made available to the industry and research community as a cloud service by a quantum cloud provider [108]. This scheme is well known as Quantum Computing as a Service (QCaaS or QaaS) [107], which corresponds with well-known paradigms in cloud computing such as Platform as a Service (PaaS) or Infrastructure as a Service (IaaS). In terms of QCaaS, software engineers can develop quantum programs and send them to quantum cloud providers to execute those programs on appropriate hardware. After finishing the computation, the users only need to pay for the actual execution time of the quantum program (pay-per-use model). In this way, QCaaS is an efficient way that optimizes the user's budget for using quantum services and the provider's resources. Many popular cloud providers nowadays offer quantum computing services using their quantum hardware, such as IBM Quantum [16], which is publicly accessible for everyone in their early phase. Besides, other quantum comput-

ing services (such as Amazon Braket [98], and Azure Quantum [17]) collaborate with other hardware companies such as D-Wave, Rigetti, and IonQ to provide commercial services. However, this paradigm still faces many challenges before solving real-world applications due to the limitations of today's NISQ computers [15]. These devices have a small number of qubits that are error-prone and limited in capabilities. Therefore, improving the quality and quantity of qubits for quantum computers will accelerate the QCaaS model and quantum software development.

3.2.4 Serverless Computing and Function as a Service (FaaS)

In the classical computing domain, serverless is the state-of-the-art computing model, which can be considered as a second phase for cloud computing [204]. This computing model fits with modern software architecture, especially the microservice applications, where the overall application is decomposed into multiple small and independent modules [205]. The serverless computing concept generally incorporates both Function-as-a-Service (FaaS) and Backend-as-a-Service (BaaS) models. FaaS refers to the stateless ephemeral function model where a function is a small and single-purpose artifact with few lines of programming code. BaaS is a concept to describe serverless-based file storage, database, streaming, and authentication services.

As FaaS is a subset of the serverless model, its main objective is to provide a concrete and straightforward way to implement software compared with traditional monolith architecture. FaaS allows the software engineer to focus only on coding rather than environmental setup and infrastructure deployment. A function can be triggered by a database, object storage, or deployed as a REST API and accessed via an HTTP connection. Functions also need to be scalable, i.e., automatically scaling in when idle and scaling out when the request demand increases. In this way, a FaaS platform can be an efficient way to optimize the resources for providers and reduce costs for customers. There are numerous open-source FaaS platforms in the cloud-native landscape, such as OpenFaaS, OpenWhisk, Kubeless, Knative, and many commercial platforms such as AWS Lambda, Azure Functions, Google Cloud Functions [206].

3.3 Quantum Serverless and Quantum Function-as-a-Service

3.3.1 Serverless Quantum Computing

A serverless quantum computing model is a viable solution for effectively utilizing contemporary quantum computers. Each quantum device, characterized by inherently limited resources, is made accessible globally via quantum cloud services. Indeed, by decomposing a monolith application into multiple single-purpose functions, we can distribute them to various backend devices. Furthermore, we can implement a hybrid quantum-classical model by combining quantum functions and classical functions in a unified application. This approach can leverage the power of existing quantum computers to facilitate new promising techniques, such as hybrid quantum-classical machine learning [207].

The adaptation of the serverless model to quantum computing must account for key differences in deployment and execution when compared to traditional computing services. In classical computing, a service can be deployed once to a server, whether physical or virtual, and then it can be repeatedly invoked by end-users. This permanent deployment is not yet feasible with current quantum computing technology., i.e., a quantum program cannot be deployed persistently in a specific quantum computer [108]. Instead, an appropriate quantum circuit needs to be built every time we execute a specific task. Then, that circuit will be transpiled to corresponding quantum system-level languages (such as QASM [208]) before being sent to a quantum cloud service for execution. Therefore, an adaptable serverless model for executing quantum tasks is needed to address this challenge. By leveraging the ideas of the serverless model and combining quantum and classical parts in a single service, we can adapt to the current nature of quantum cloud services, accelerate the software development process, and optimize quantum resource consumption. This kind of computing model could be a potential approach to enable software engineers to realize the advantages of quantum computing and explore more complicated quantum computation in the future.

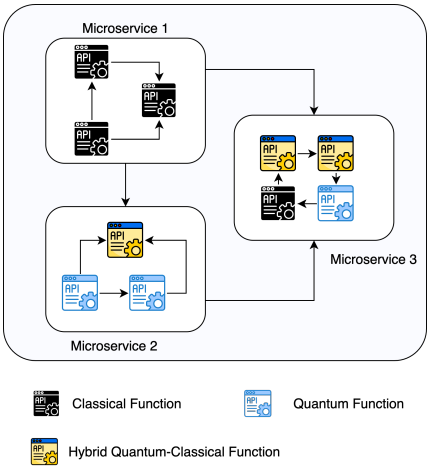


Figure 3.1: Service-Oriented Quantum-Classical Application Model

3.3.2 Quantum Function-as-a-Service (QFaaS)

In serverless computing, the Function-as-a-Service (FaaS) indicates a service-oriented cloud computing model in that the software engineer only needs to focus on coding without worrying about server configuration. A function is made by small pieces of programming code and is deployed as a service, which can be triggered and executed on demand [209].

By bringing the FaaS model to quantum computing, we propose the concepts of Quantum FaaS. A *quantum function* can be considered an ephemeral, event-triggered, and single-purpose quantum program with few pieces of quantum code. Due to the limitation of the NISQ devices and the difference in the software deployment model, we need to leverage the classical resources and techniques for developing and executing the quantum function. Specifically, a software engineer can still focus solely on coding quantum functions with high-level quantum SDKs (such as Qiskit, Cirq, Q#, or Braket) without needing to care about the quantum programming environmental setup or server deployment. The function code is automatically deployed in a containerized environment and is published as a service with an API endpoint for invoking. Whenever that service is triggered, the programming logic defined in the function will be executed, where both classical and quantum computation is involved. The classical parts include the pre-processing of input data, quantum circuit generating, and post-processing, where

the quantum part indicates the circuit execution on quantum backends.

By adopting the FaaS approach and classical resources for creating a quantum function, we can seamlessly integrate multiple quantum functions together or with classical ones to construct a service or microservice in a large application (see Figure 3.1). The serverless and service-oriented application model is a potential approach to bring the advantages of quantum computation to solve intractable problems of classical computing without replacing the whole system. For example, we can replace a classical random number generation function in a finance microservices application with the quantum counterpart to yield truly random numbers [45], which we cannot do in classical programming. However, the application of serverless computing models to quantum applications needs to be carefully considered and adjusted, especially in the current NISQ era, in which quantum hardware limitations can hinder quantum execution. Ultimately, the potential benefits and challenges of the emerging quantum serverless computing model motivate us to explore and empirically evaluate in this study.

3.4 QFaaS Architecture and Main Components

3.4.1 Software Requirements and Design Principles

The design of the QFaaS framework is guided by several key requirements and design principles, which contribute to its benefits for streamlining service-oriented quantum application development and help software engineers to plan, develop, and improve their quantum software applications:

- **Serverless**: Quantum software engineers only need to focus on developing and improving their functions, while the framework automatically carries out the rest of other procedures, including the environmental setup, function deployment, selecting the appropriate quantum computation system for the function execution (backend selection), managing the function operation and scaling.

- **Service-Oriented**: Each quantum function can be deployed as a service, which can be accessed through the cloud-based API gateway in multiple methods. This ap-

proach simplifies further expansion and maintenance, drawing inspiration from the microservices architecture and the "everything-as-a-service" (XaaS) paradigm. Consequently, new functionalities can be easily integrated into the existing application without disrupting other services.

- **Flexibility**: Users can choose their preferred quantum programming languages, libraries, and cloud providers to avoid potential vendor lock-in situations. The framework needs to support the current NISQ computers and quantum simulators. Its architecture provides the flexibility to implement possible extensions to support other quantum technologies as they emerge in the future.

- **Seamlessness**: The framework needs to support continuous integration and continuous deployment, which are two of the essential characteristics of DevOps to continuously deliver value to end-users. Utilization of this model boosts application development and becomes more reliable when compared with the traditional paradigm [210].

- **Reliability**: The framework implementation should use the state-of-the-art software technologies to ensure high availability, security, fault tolerance, and trustworthiness of the overall system. The execution results are stored in the database for comparison purposes and to optimize the execution parameters of the quantum function.

- **Service Scalability**: As one of the critical characteristics of the serverless model, the quantum service is scalable and adapts to the actual user requests. However, the scalability of current quantum devices is limited by the number of qubits, and NISQ devices cannot execute practical-scale networking and computational instructions beyond small instances of a few specific problems [211]. Therefore, it is important to note that within the context of our framework, this requirement is manifested in the size scalability (i.e., the ability for a system to effectively expand its size as more resources or users are added) [212] of the classical resources, wherein the quantum function is deployed. We primarily focus on horizontally scaling in/out function deployment by adjusting the replication of function instances, in response to user requests. This approach provides a practical and

adaptable solution to handle varying workloads depending on the number of concurrent function invocations.

- **Transparency**: The operation workflow of the framework needs to be transparent to both the software engineers and end-users. The information provided by the framework is sufficient for troubleshooting, logging, and monitoring purposes, and can be used for further investigations if needed.

3.4.2 Main Components

The architecture design of QFaaS comprises six extendable components: the QFaaS APIs and API Gateway, the Application Deployment Layer, the Classical Cloud Layer, the Quantum Cloud Layer, the Monitoring Layer, and the User Interface. Figure 3.2 illustrates the overall design, including the architecture and principal components of our framework.

QFaaS APIs and API Gateway

We design two types of APIs that expose to the authenticated user through a secure HTTPS connection:

- **Service APIs** are the set of APIs corresponding to the deployed functions. Each function running on the classical cloud layer has a unique API endpoint accessible to an authorized end-user. These APIs can be integrated seamlessly into existing software workflow.

- **Core APIs** set is one of the most essential components in the QFaaS framework. It comprises a set of secure REST APIs, which provide principal operations and interactions among all components of the whole system. These APIs facilitate function development, invocation, job monitoring, and interaction with the external quantum providers and backend management. Core APIs also facilitate the main functionalities of the QFaaS UI. We explain the detailed design and implementation of the Core APIs in Section 3.5.1.

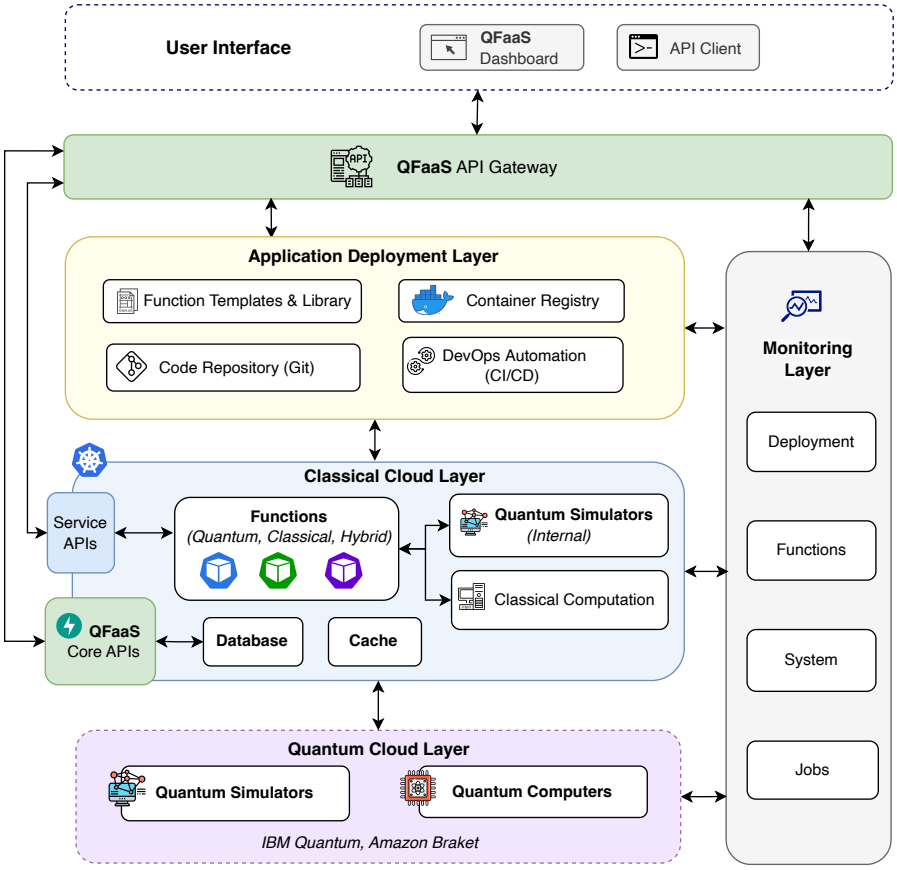


Figure 3.2: Overview of QFaaS Architecture Design and Main Components

The *API gateway* serves as a centralized entrance where users can interact with other components. This API gateway routes users' requests to suitable components for processing and delivers the result back to the users with a common data format after completing the execution.

Application Deployment Layer

This layer serves as a bridge between quantum software engineers and the cloud layers to deploy and expose each function as a service with an API endpoint. It takes the principal responsibility for code version control, containerizing, and deploying functions by incorporating four key components:

- **Code Repository** is a Git-based platform to manage function codes with version

control, which is essential in software development for collaboration, issue tracking, and further workflow automation integration.

- **Function Templates and Library** provides container-based quantum software environment configuration, support library, and common function templates for well-known quantum SDKs and languages, including Qiskit, Cirq, Q#, and Braket. We developed *QFaaS Library*[1], a Python-based programming library that supports essential interactions to the Core APIs and provides common pre-built data pre-processing, and output post-processing for the function development.

- **DevOps Automation** employs CI/CD integration following the DevOps manner to automate the function of deploying and updating, ensuring the continuous delivery of reliable quantum functions.

- **Container Registry** stores immutable container images of functions and environmental setup for function deploying, migrating and scaling.

Classical Cloud Layer

This layer is a cluster of cloud-based classical computers (physical servers or virtual machines), where the quantum functions are deployed and triggered. All the classical computation tasks are executed here, including backend selection, data pre-processing, and post-processing.

We employed Kubernetes to orchestrate all the *pods* (the container-based unit of Kubernetes) for the deployed function across all cluster nodes. Each function will be run on a pod and can be scaled up horizontally by replicating the original pod to serve multiple incoming requests simultaneously. Following the proposed architecture, we can use all built-in quantum simulators of employed SDKs directly inside a pod at the Kubernetes cluster. We call this kind of simulator the *internal quantum simulator*, while the term *external quantum simulator* denotes simulators offered by quantum cloud providers.

We also deployed a NoSQL database on this layer to permanently store the processed job result data and information of users, functions, and backends. In a production de-

[1]https://pypi.org/project/qfaas/

ployment, it is recommended that the database be placed externally to ensure the high availability for permanent data storage. Besides, the cached data for quantum circuits can be stored at this layer.

Quantum Cloud Layer

This layer is an external part, indicating the quantum cloud providers, such as IBM Quantum and Amazon Braket, where the quantum job can be executed in a physical quantum backend. Quantum providers can provide either quantum simulators or actual quantum computers through their cloud services, which can be accessed from the Classical Cloud layer. We develop the corresponding APIs in *QFaaS Library* and *Core APIs* to interact with each external quantum cloud platform. The result processing data from all cloud providers is standardized in a common JSON format, ensuring data consistency and preventing data lock-in issues within the serverless-based platform.

Monitoring Layer

This layer incorporates monitoring techniques to check the status of other QFaaS components, including quantum backends, quantum providers, function execution (job), and function deployment. As the classical cloud layer employs Kubernetes as the container orchestration, we can also seamlessly integrate additional open-source monitoring techniques, such as Prometheus[2], Grafana[3], and Lens[4] for observing other system aspects of the classical cloud layer such as resource consumption, networks, and system logs.

User Interface

We created a user-friendly web application (QFaaS Dashboard) using React[5] to interact with the QFaaS system (examples can be found in Section 3.6). This user interface visualizes the essential functionalities of QFaaS, such as function developing, deploying,

[2]https://prometheus.io/
[3]https://grafana.com/
[4]https://k8slens.dev/
[5]https://reactjs.org/

monitoring, and invoking; job results, quantum providers, and backends connection.

3.5 Design and Implementation

This section provides the technical design, procedures, and implementation of the QFaaS Core APIs and quantum function development, development, invocation, and backend selection in the QFaaS framework.

3.5.1 QFaaS Core APIs

QFaaS Core APIs take responsibility for primary functionalities in the QFaaS framework. We have developed this API set using Python 3.10 with FastAPI[6], a high-performance Python-based framework supporting the Asynchronous Server Gateway Interface (ASGI) for concurrent execution. We used the MongoDB database to store the persistent data in JSON format. Figure 3.3 depicts the overall class diagram, with attributes and methods of each object in QFaaS Core APIs.

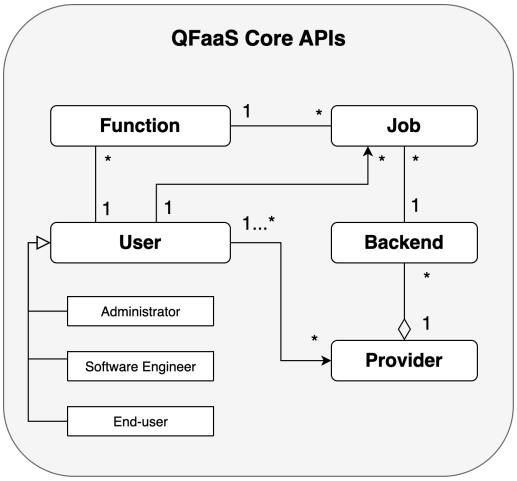


Figure 3.3: Class Diagram of QFaaS Core APIs

- **User:** This class defines user attributes and methods to facilitate access control and role management features. We categorized three different users: *administrator,*

[6]https://fastapi.tiangolo.com/

software engineer, and *end-user* with different privileges in the system. Administrators control all components; software engineers can develop and deploy functions, while end-users can only use their appropriate functions. Each active user is assigned a unique token (using OAuth2 Bearer[7]), which is used for authentication, authorization, and dependency check for each interaction with the core components of the QFaaS. This implementation enhances security for the whole framework and provides a multiple-user environment for taking advantage of the framework.

- **Function:** This class defines each function's properties and supported methods. Each function belongs to a software engineer (author) and its access can be granted to a specific end user. The *CRUD*, *invoke()* and *scale()* methods of this object interacts directly with other architectural components such as Code Repository, Container Registry and the Classical Cloud layer to handle the function deployment, management, and invocation.

- **Job:** A job in QFaaS is a computation task submitted to a quantum backend for execution. All properties and methods of a job are defined in the Job class. Each Job has a unique *Job ID* assigned by QFaaS and can be associated with a *providerJobID* given by an external provider. The function invocation initializes the job object. After finishing the execution, job results can be post-processed and stored in the database for further retrieval.

- **Provider:** The provider class handles a user's authorization to external quantum providers, including IBM Quantum and Amazon Braket. The design of this class ensures that each user has the specific privilege to access their quantum providers only.

- **Backend:** A backend is a quantum computation node, such as a quantum simulator, or a quantum computer, which takes responsibility for the quantum execution. The Backend class defines the attributes and methods to interact with the backend provided by the classical cloud layer or external quantum cloud layer. We also im-

[7]https://datatracker.ietf.org/doc/html/rfc6750

plement the *Backend Selection* policy in this class for automating selecting the most appropriate quantum backend for each quantum task execution (Section 3.5.3).

3.5.2 QFaaS Quantum Function Structure

This section describes the structure of a quantum function in the QFaaS framework for the development process. Our framework provides a set of pre-configured function templates, each encapsulated in a Docker image with the necessary quantum software development kit (SDK) environment, to streamline the development of quantum functions. Each function has a single working directory, including main components following the common pattern of the serverless platform (such as Lambda [213]). Function handler code includes classical parts (using Python) and quantum parts. When end-users invoke the function, QFaaS executes the function handler and starts the computation as defined. Handler for Qiskit, Cirq, and Braket function can be defined at `handler.py` file while Q# function requires an additional Q# code at `handler.qs` file and then to import it to the main `handler.py` file. The sample structure for the function handler with classical pre-processing and post-processing is described in Code 3.1. In the function handler code, we import `qfaas` library[8] and all additional libraries (including compiled binary for Q# function). Then, we define the function handling procedure (as shown in Code 3.1) as follows:

```
import qfaas, [additional_libraries]

def handle(event, context):
    # 1. Pre-processing (optional)
    data = pre_process(event.data)
    # 2. Generate Quantum Circuit
    qc = generate_circuit(data.input)
    # 3. Verify/select quantum backend
    backend = Backend(data, qc)
    # 4. Submit job for execution
    job = backend.submit_job(qc)
    # 5. Post-process (optional)
    result = post_process(job.result)
```

[8]https://pypi.org/project/qfaas/

```
14    return result
```

Code 3.1: Sample structure of a hybrid quantum-classical function

1. **Input Pre-processing**: Users can optionally define the pre-processing for input data handling in `event` object. The `event` is JSON-based data that contains user input raw data, followed by the QFaaS JSON format, while `context` provides HTTP methods (such as GET/POST), HTTP headers, and other properties, which are optional for the request. This pre-processing is executed on classical computers and the processed data is used for the circuit generation process.

2. **Quantum Circuit Generating and Compilation**: The engineer can define quantum circuit code that takes `input` parameters as input data. Based on the given parameters, an appropriate quantum circuit is built. As aforementioned in section 3.3.2, each quantum function is a single-purpose and event-triggered quantum program to solve a specific problem that produces the output based on the user's input. To ensure the versatility of the quantum function, the quantum function code needs to be designed to generate the quantum circuit dynamically based on the user's input (so we called *variable circuits* for short). Otherwise, if the quantum circuit is fixed regardless of different input data, the same circuit will be generated every time the quantum function is invoked. The characteristic of the quantum circuit is used for the backend selection in the next stage. After that, the initial circuit may need to be transpiled to be compatible with the supported gates and qubit topology of the selected backend. The conversion is also required in the case of the quantum cross-platform execution model.

3. **Quantum Backend Selection**: QFaaS provides a built-in quantum backend verification and selection in `Backend` class (detailed implementation in Section 3.5.3) to ensure the most appropriate backend is selected for the execution. It is worth noting that our proposed backend selection strategy is a best-effort approach as some quantum jobs take longer to execute than others, and no quantum provider discloses any information about the pending jobs' characteristics at the moment. However, our framework also allows users to define their customized backend

selection strategy, which paves the way for more advanced techniques to be designed when more information is available in the future.

4. **Quantum Job Submission and Execution**: We also provide the `submit_job` method in the `Backend` class to perform the job submission to the selected quantum backend in the previous step. This method involves the quantum circuit transpilation to ensure the submitted circuit is compatible with all supported gates of the quantum backend. The outcome of this method is a `job` with a unique ID for result retrieval and further inspection. As today's quantum computers are NISQ devices [15], each quantum execution should be run many times (shots) to mitigate quantum errors. Besides, due to the limited number of available quantum computers, a quantum task (job) needs to be queued at the cloud provider (from seconds to hours) before execution. After the quantum computation is finished, the raw result is retrieved from the provider to the function handler for the post-processing step.

5. **Output Post-Processing**: Users can define the customized post-processing method for further analyzing the raw result from the quantum computer before generating the final result for end-users. We provided several sample post-processing methods based on result counts, which will be discussed in section 3.6.

After finishing all the processing, the function handler returns the result to end-users, including the HTTP Status Code and response data in common JSON format, regardless of the difference of targeted backends. The job result is also kept in the MongoDB database for further retrieval.

3.5.3 Quantum Backend Selection

Quantum Backend Selection is an essential procedure in the function invocation process to determine which quantum backend is suitable for the quantum circuit execution. In the initial version of QFaaS, we have incorporated a scoring-based policy, as outlined in Algorithm 1. This policy empowers users to prioritize their selection of the quantum backend based on factors such as result precision or the speed of function execution.

Algorithm 1: QFaaS Quantum Backend Selection

Input : qc: quantum circuit, sdk: quantum SDK,
a: autoselect option, t: preferred backend type,
bName: backend name (for manual selection),
γ: list of all parameter weights, u: current user
Output: be: quantum backend instance

procedure ***BackendSelection:***
  be $\leftarrow$ null
  q $\leftarrow$ qc.getNumQubit()
  if t *is internal* **then**
    be $\leftarrow$ `getInternalSimulator` (sdk, q, u)
  else
    if a *is True* **then**
      # Auto Backend Selection
      Pre-select backends and get all transpiled qc
      $\mathcal{B} \leftarrow$ `getBackendList` (u)
      $\mathcal{B} \leftarrow$ `preSelect` ($\mathcal{B}$, q, sdk, t)
      $\mathcal{T} \leftarrow$ `getTranspile` (qc, $\mathcal{B}$)
      Scoring all backend in $\mathcal{B}$
      for $b \in \mathcal{B}$ **do**
        $\tau \leftarrow$ get transpiled circuit for b ($\tau \in \mathcal{T}$)
        Normalize depth, QV, workload & CLOPS
        v $\leftarrow$ `norm` (getQV(b), $\mathcal{B}$)
        d $\leftarrow$ `norm` (getDepth(τ), $\mathcal{T}$)
        c $\leftarrow$ `norm` (getCLOPS(b), $\mathcal{B}$)
        w $\leftarrow$ `norm` (getWorkload(b), $\mathcal{B}$)
        Compute precision, speed & overall score
        p $\leftarrow v\gamma_v + d\gamma_d$
        s $\leftarrow w\gamma_w + c\gamma_c$
        $\epsilon \leftarrow p\gamma_p + s\gamma_s$
        Update all backend scores
        $\mathcal{B} \leftarrow$ `updateScore` (b, ϵ)
      end for
      be $\leftarrow$ `getMaxScore` ($\mathcal{B}$)
    else
      # Manual Backend Selection
      be $\leftarrow$ `verifyBackend` (bName, u, qc)
    end if
  end if

In our backend selection logic, users can specify their preferred backend type as `internal` to use the internal quantum simulator for testing and prototyping purposes. They can also manually select a specific quantum backend when invoking a function. Otherwise, the QFaaS framework will process the automatic backend selection strategy as follows:

1. **Backend Pre-selection and Circuit Transpilation:**
   QFaaS filters a list of backends based on key requirements to execute the quantum task. These requirements include (1) scale of the quantum backend (i.e., the number of qubits must be sufficient), (2) availability (i.e., the quantum backend must be operational), and (3) and compatibility (i.e., the quantum backend must support the quantum SDKs used by quantum circuit). After pre-selecting the list of appropriate quantum backends ($\mathcal{B}$), the circuit will be transpiled to adapt to all quantum backends in $\mathcal{B}$, and the characteristics of each corresponding transpiled circuit can be used for determining the most suitable quantum backend.

2. **Backend Scoring based on execution priority**
   Considering the current state of NISQ devices and the quantum computing service model of quantum providers, we determine two priorities to select a quantum backend when invoking a function, i.e., precision and speed.
   *(a) The **precision** of the execution results* depends on the quality of qubits in a quantum system, which can be determined by the quantum volume (QV, denoted by v). Quantum volume [29] is a holistic metric that indicates how well a quantum circuit can be executed in a quantum system, measured by the largest random square circuit that can be successfully run. We also consider the depth (denoted by d) of the circuit as a shallower circuit has a higher chance of being executed faithfully inside a quantum system [32, 214]. In other words, a finer quantum backend for generating better result precision has a higher quantum volume and requires a quantum circuit to be transpiled with a smaller circuit depth. For equally weighting two metrics with different scales in our backend scoring algorithm, we normalize v and d of the i-th backend in $\mathcal{B}$ to be between 0 and 1 (using min-max normalization method) by the following formulas:

$\bar{v}_i = \frac{v_i - \min(v)}{\max(v) - \min(v)}$ and $\bar{d}_i = \frac{\max(d) - d_i}{\max(d) - \min(d)}$
where $\bar{v}_i$ and $\bar{d}_i$ are normalized values of quantum volume and transpiled circuit depth.

The precision score (p_i) of the i-th quantum backend can be calculated by $p_i = \bar{v}_i\gamma_v + \bar{d}_i\gamma_d$ where $\gamma_v + \gamma_d = 1$ and γ_v, γ_d are the weight of quantum volume and circuit depth, respectively.

*(b) The **speed** of execution* relies on how fast a quantum system can execute quantum circuits, which can be measured by Circuit Layer Operations Per Second [32] (CLOPS, denoted by c). We also consider the current workload (i.e., the total number of pending jobs to be executed, denoted by w) of a quantum backend to determine the speed score, as the waiting time can be typically shorter. Similar to the calculation of precision score, we normalize c and w of the i-th backend in $\mathcal{B}$ to be between 0 and 1 by the following formulas:
$\bar{c}_i = \frac{c_i - \min(c)}{\max(c) - \min(c)}$ and $\bar{w}_i = \frac{\max(w) - w_i}{\max(w) - \min(w)}$
where $\bar{c}_i$ and $\bar{w}_i$ are normalized values of CLOPS and current workload of the quantum backend.

The speed score (s) of an i-th quantum backend can be determined by $s_i = \bar{c}_i\gamma_c + \bar{w}_i\gamma_w$, where $\gamma_c + \gamma_w = 1$; γ_c and γ_w are the weight of CLOPS and workload of the quantum backend, respectively.

After scoring the precision (p_i) and speed (s_i), we calculate the overall score (ϵ_i) of the i-th backend by using the following formula: $\epsilon_i = p_i\gamma_p + s_i\gamma_s$ where $\gamma_p + \gamma_s = 1$; γ_p and γ_s are the weight of precision and speed priority, respectively. This approach can dynamically select the most suitable quantum backend based on user priority, the characteristics of current function invocation, and the status of all available quantum backends. It is important to note that although all weight parameters can be adjusted, users can only need to adjust two primary weights γ_p and γ_s based on their priority on either the precision of the execution result or how fast the quantum circuit will be executed. For example, if a user prioritizes the precision of the result rather than the speed, they can set γ_p close to 1, in which the quantum can be queued in a specific backend (which can be executed with the highest accuracy) even if the availability of the least

busy or highest quantum backend is not the best one. Otherwise, the default value of all weight parameters can be set to 0.5 to maintain the balanced contribution of all factors to the backend selection decision. To validate the operation of this policy, we provide examples of backend selection for three consecutive quantum function invocations in Section 3.6.3.

It is important to note that we have already utilized the most accessible circuit information and key benchmarking metrics of quantum computers (qubit number, quantum volume, CLOPS) [32] for implementing the backend selection strategy. Due to the limitations to accessing the public information of pending jobs at the other cloud provider (e.g., Amazon Braket through Strangeworks), this backend selection procedure is only supported for IBM Quantum providers. Supporting the backend selection for Amazon Braket and other quantum cloud vendors and considering other aspects, such as costs, is our future plan.

3.5.4 Quantum Function Cold Start Mitigation

Cold start is a typical problem of serverless computing [206]. It is referred to when the framework needs to initialize a new container instance and prepare the function execution when handling a new invocation [33, 215]. This latency is even more significant in the context of quantum functions, as each function typically requires a specific environment setup and circuit compilation optimization to adapt to the targeted quantum backend.

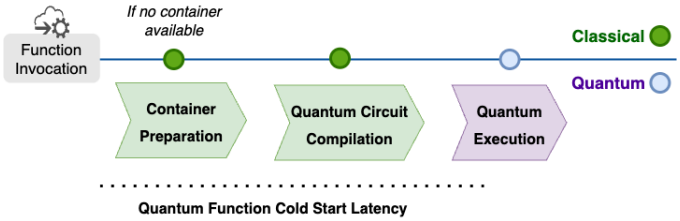


Figure 3.4: Quantum function cold start latency process during the invocation

The quantum function cold start latency can be illustrated in Figure 3.4. If there is no container of a specific function is available at the time of its invocation, a new in-

stance needs to be initialized, which can cause a notable delay. After setting up the environment for the quantum SDK, a corresponding quantum circuit needs to be generated and compiled. As most available quantum computers do not have fully connected qubit topology and do not support all quantum gates, quantum circuit transpilation is typically required to tailor it to specific qubit topology and native gate set of targeted quantum backend. This process can also result in substantial delays in the quantum circuit compilation phase before its execution. Thus, mitigating cold start latency in a quantum serverless platform, particularly in the NISQ era, is unavoidably essential.

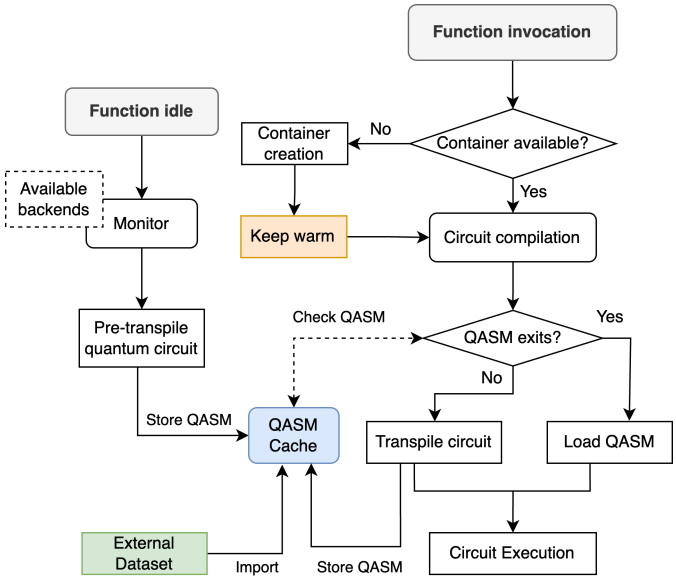


Figure 3.5: QFaaS Transpilation Caching approach for mitigating the cold start latency of container preparation and quantum circuit compilation

Focusing on the nature of current quantum execution, we design an agile and practical strategy to mitigate the cold start issue and enhance quantum function execution as a complement to the QFaaS framework (see Figure 3.5). To reduce the container preparation latency, we keep the function's container "warm" (i.e., keep at least one instance up and running) after its creation to avoid the long container initialization latency. To mitigate the quantum circuit compilation latency, we utilize the cache-based approach, which is a popular strategy in classical serverless computing [215]. To enhance the reusability and flexibility of cached data, we use Quantum Assembly Language (QASM) to store a copy of the pre-transpiled quantum circuits. Open QASM is a lightweight as-

sembly language to represent universal quantum circuits [208], which can be imported to or generated from different quantum SDKs (e.g., Qiskit). During the circuit compilation, QFaaS will check the availability of the corresponding transpiled circuit and load it into the current function instance for execution. Otherwise, the quantum circuit has to be transpiled, but a copy of the transpiled circuit will be stored for further reuse. Besides, during the idle time period of the function, we can proactively pre-transpile quantum circuits of that function to optimize with the qubit topology and native gate set of available quantum backends. Another source to enrich the transpiled QASM files is to integrate the external quantum circuit dataset, such as MQTBench [1], which comprises over 70,000 quantum circuits in QASM format of different applications. As Open QASM is a lightweight and platform-agnostic quantum assembly language [208], this caching approach also brings its potential to future expansion of QFaaS to support other quantum computing systems. Also, the circuit loading time from a pre-transpiled QASM file is by far faster than transpile the circuit; this approach can significantly reduce the overhead for the function repeated execution, hence reducing the cold start latency for the quantum function execution. The empirical result to validate this strategy can be found in Section 3.6.4.

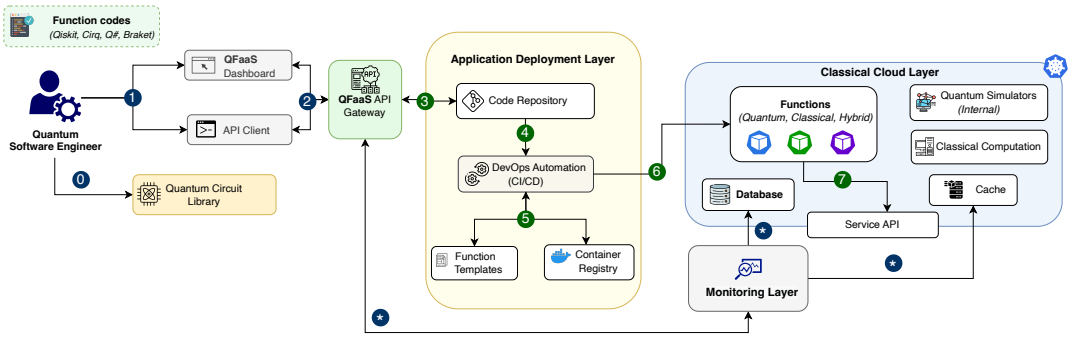
3.5.5 QFaaS Sample Operation Workflows

This section provides two sample operation workflows, including developing/deploying and invoking quantum functions using the QFaaS framework.

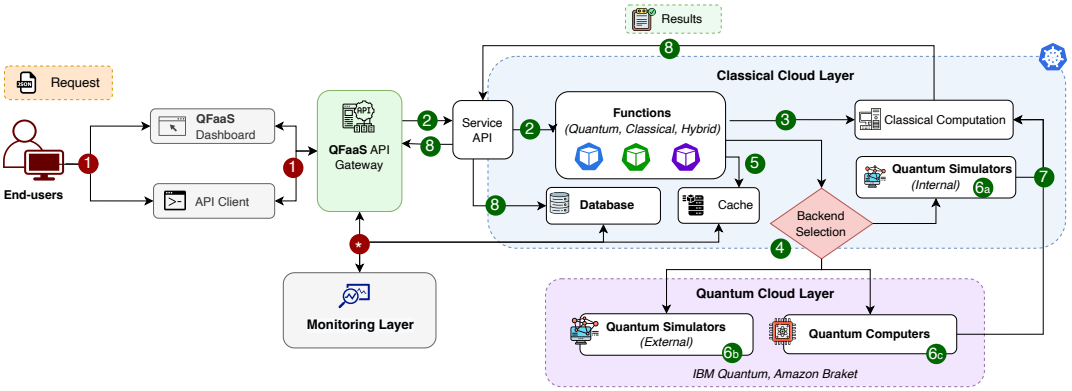
Developing and deploying quantum functions

QFaaS simplifies the function development process for quantum software engineers. They can utilize the following workflows to create new functions, update existing functions, and troubleshoot issues during the development process. Figure 3.6a depicts the function development process, which consists of seven key steps as follows:

1. Create a new function by using the QFaaS UI. The engineers specify which quantum SDK will be used (Qiskit, Cirq, Q#, or Braket), include the required library, and write their quantum function code.



(a) Function development and deployment for Quantum Software Engineers.



(b) Function invocation for End-users.

Figure 3.6: Overview of two main operation workflows in QFaaS: (a) Function development and deployment, (b) Function invocation.

2. Push function codes to the Application Deployment layer through the QFaaS API Gateway.

   After these steps, QFaaS automatically takes responsibility for the rest of the deployment procedure by performing the following steps:

3. The API gateway forwards the function code and pushes it to the Code Repository.

4. After the function code is pushed, it triggers the Automation components to start the continuous deployment.

5. Pull the function template and combine it with function code to build up and containerize it into a Docker image. Then, those images will be pushed to a Container Registry to be stored for further utilization (such as migrating or scaling in a function).

6. Deploy the function as a container-based service into the Kubernetes cluster at the classical cloud layer.

7. Expose the service API URL endpoint corresponding to the deployed function. After this stage, the function serves as a service and is ready for invoking from end-users.

During the development stage of the quantum function, engineers can test the deployed services with different backends several times and analyze the results, which are stored in the database for comparison and further improve function configurations (e.g., shots) to achieve optimal results. QFaaS also monitors the operation of quantum functions and performs the cold start mitigation strategy as described in Section 3.5.4

Invoking quantum functions

The users can invoke the deployed function through the QFaaS API gateway. Figure 3.6b demonstrates the overall workflow for the function invocation, including seven steps as follows:

1. *Sending request*: In the requested data, the user can clarify their preferred backend or let the framework automatically select the suitable backend, the result retrieval method, and the number of shots they want to repeat the quantum task.

   After receiving the user's requested data, QFaaS automatically accomplishes the rest of the process.

2. *Routing the requests*: The API Gateway routes user requests to appropriate available functions. In the event that a function is not yet initiated or undergoing scaling in to zero, QFaaS takes charge of initializing and activating the function to handle the incoming user request. This scenario is anointed as a *cold start* in serverless jargon.

3. *Input Pre-Processing and Quantum Circuit Compilation*: The user's input data undergoes pre-processing at the classical computation node. Then, a corresponding quantum circuit is generated based on the provided input.

4. *Backend Selection*: An appropriate backend is selected based on user requests and availability at the quantum provider, using our decision policy (Algorithm 1).

5. *Executing the quantum job*: The quantum circuit is transpiled and dispatched to the chosen backend, which can be an internal quantum simulator (6a), an external quantum simulator (6b), or a quantum computer (6c). Once the backend completes the execution, the outcome is transmitted back to the function handler for post-processing on classical resources in the same invocation. If users want to check the response later or when the waiting time at the provider is longer than a predefined timeout (1 minute by default), the function will send back the QFaaS Job ID and backend information after successfully submitting the quantum circuit to the selected backend. This delayed scenario is expected to happen often when submitting a job to an external quantum cloud provider during peak hours due to the current shared nature of available quantum resources.

6. *Output Post-Processing*: The outcome from quantum backends could be analyzed and post-processed before being sent back to end-users and stored in the database.

7. *Returning the results*: Following the previous step, the final result is returned to end-users through the API Gateway, following the same approach as when they initially submitted the request. End-users can obtain the final result data and information regarding the quantum backend device utilized during the quantum execution.

3.6 QFaaS Example of Operation and Evaluation

This section provides explanatory examples of the operation and performance evaluation of QFaaS in function deployment, resource consumption, and scalability. Besides, we validate the proposed quantum backend selection and cold start mitigation strategy with various quantum algorithms. We also use QFaaS to evaluate the performance of popular simulators and computers to give practical insights into the limitations and challenges of quantum software engineering in the NISQ era (see Appendix B).

3.6.1 Environment Setup

We deployed the core components of QFaaS on a set of four virtual machines (VMs) offered by the Melbourne Research Cloud[9]. We set up the Kubernetes cluster using microk8s[10] with *containerd* as the underlying containerization technology on a three-VM cloud cluster (one master node with 4 vCPU, 16GB RAM, and two worker nodes with 8vCPU, 32GB RAM each). The function deployment component is built on top of OpenFaaS [216]. The QFaaS Code Repository and Automation components are deployed on the last VM (4 vCPU, 16GB RAM) with Gitlab as the underlying Git-based platform. For the quantum computation, we have tested the Qiskit functions with the built-in QASM simulator on the classical computers at the classical cloud layer and quantum backends provided by IBM Quantum [16]. For Q# and Cirq functions, we used their built-in quantum simulators and executed them on the classical cloud layer. For Braket functions, we used their local simulator and external backends at Amazon Braket through the support of Strangeworks Backstage program [150]. The circuits and transplied QASM files for other quantum algorithms in the backend selection and cold start mitigation validation are adopted from the MQT Bench dataset [1].

3.6.2 Example of Operation and Performance Evaluation

To demonstrate the practical operation of QFaaS, we utilize an explanatory quantum circuit to generate truly random numbers, utilizing the superposition characteristic of qubits. It is evident that random numbers play an essential role in cryptography, finances, and many other fundamental scientific fields [217]. By leveraging quantum principles, Quantum Random Number Generation (QRNG) is a reliable way to provide true randomness, which cannot be achieved by classical computers [44]

We deployed the QRNG circuits in four popular quantum SDKs (Qiskit, Cirq, Q#, and Braket). The main idea of this circuit is to leverage the Hadamard gate to create the superposition state of each qubit and then measure to get a random value (0 or 1) with the same possibility (50%). To validate the hybrid quantum-classical integration feature,

[9]https://cloud.unimelb.edu.au/
[10]https://microk8s.io/

we implemented sample post-processing by analyzing all possible outcomes when the function is executed multiple times (shots) and returning the most frequent result to the user.

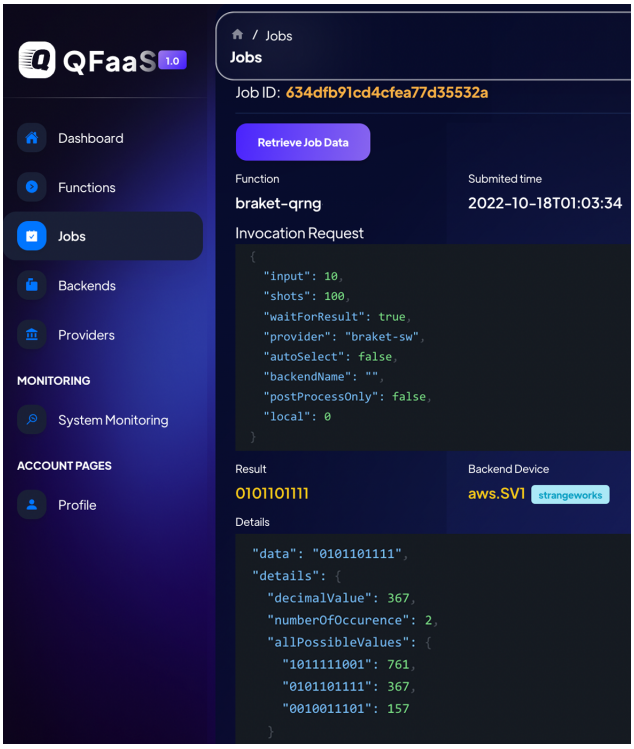


Figure 3.7: Sample QRNG Function Invocation Result on QFaaS Dashboard interface using Amazon Braket backend (aws.SV1)

Table 3.1: QFaaS functions deploying and re-deploying (updating) time

Function	**Function Image Size** *(MB)*	1st Building time *(second)*	**1st Deploying time** *(second)*	Re-building time *(second)*	**Re-Deploying time** *(second)*
Qiskit QRNG	755.85	180.62	262	8.35	31
Cirq QRNG	**561.19**	**105.57**	**168**	8.53	**30**
Braket QRNG	1010.07	354.04	496	6.36	**30**
Q# QRNG	2200	267.75	466	**5.77**	34

The request for invoking the QRNG function using all supported SDKs and languages follows the QFaaS format. Upon completion of the processing, the sample response, as illustrated in Figure 3.7, indicates that a 10-qubit random number, specifically 367 (0101101111 in binary), has been generated. This particular random number occurs most frequently, appearing twice, during the execution of the function using the Amazon Braket Simulator (aws.SV1).

Table 3.2: QFaaS function resource consumption during the idle time and busy time with 1 and 10 concurrent users

Function	Idle time		1 user		10 concurrent users	
	CPU (vCore)	**RAM** (MB)	**CPU** (vCore)	**RAM** (MB)	**CPU** (vCore)	**RAM** (MB)
Qiskit QRNG	**0.001**	**87.9**	0.052	**87.93**	0.082	**88.04**
Cirq QRNG	**0.001**	124.56	**0.024**	128.605	**0.053**	129.07
Braket QRNG	**0.001**	96.063	0.042	99.586	0.078	99.96
Q# QRNG	0.037	688.77	0.056	735.16	-	-

Function Deployment Evaluation

In this evaluation, we measured the image size, average image building time, and the total deploying time in the first and later update at the Application Deployment layer (using Gitlab) to provide insight into QFaaS system performance. Table 3.1 records the detailed result of this evaluation.

As all function's essential components are compressed in the container images, its size is varied from 561 MB (Cirq) up to 2.2 GB (QSharp). The total deploying time includes the function image-building time and image-deploying time, which is below ten minutes for creating the function deployments the first time. Deploying the Cirq QRNG took the shortest time (below 3 minutes) whereas Braket and Q# functions required 7-8 minutes to complete. However, when we update the function handler code, the re-deploying times are significantly faster (around 30 seconds) for all functions. It is mainly because a container image comprises multiple layers and the updated image can inherit multiple layers from the previous images. We also utilized caching technique to optimize the continuous integration and deployment process. These function deploying and updating times are reasonable in practice as the software engineer can focus on coding and offload the configuration and deployment to the Application Deployment layer. The corresponding service endpoint of the quantum function is then ready for invoking after several minutes in the first deployment and below a minute for the following update.

Function Resource Consumption

We monitored the resource consumption, including the CPU and memory (RAM) used by the pod associated with the deployed function in QFaaS. We measured the aver-

age value of maximum CPU and RAM usage, provided by Kubernetes Dashboard and K8sLens 6[11] monitoring tool. In this evaluation, we kept a single pod for each function and considered 3 scenarios: the idle time (i.e., keep the pod running without any invocation), 1 user, and 10 concurrent users.

As the results are shown in Table 3.2, the required resource to keep Qiskit, Cirq, and Braket functions running are kept low, which are 0.001 CPU vCore and roughly 100 MB of memory. In contrast, the Q# function requires more additional resources to keep its pod running, even during idle time. Then, we used JMeter 5[12] to continually generate 1000 requests to obtain 10-qubit random numbers (using the internal quantum simulators) with 100 shots for each invocation. The maximum CPU and RAM used are slightly increased in cases of Qiskit, Cirq, and Braket while the corresponding increment is higher with the Q# function. We also note that the figures for the Q# function in the last columns are disregarded as its pod frequently crashed when we constantly send requests from 10 concurrent connections. Therefore, we suggest using Qiskit, Cirq, or Braket for prototyping a quantum function to achieve better performance and maintain proper resource consumption.

Table 3.3: Backend Selection for Job 1 (Deutsch-Jozsa's algorithm - using 5 qubits. ibmq_kolkata backend is chosen with the highest $\epsilon = 0.61$.

Backend	**Qubits**	**QV (v)**	$\bar{v}$	**QC depth (d)**	$\bar{d}$	**CLOPS (c)**	$\bar{c}$	**Workload (w)**	$\bar{w}$	**p**	**s**	ϵ
ibm_washington	127	64	0.33	20	0	850	0	2	1	0.17	0.5	0.34
ibmq_kolkata	27	128	1	19	0.17	2000	0.56	40	0.68	0.59	0.62	**0.61**
ibm_hanoi	27	64	0.33	17	0.5	2300	0.71	88	0.27	0.42	0.49	0.46
ibmq_guadalupe	16	32	0	14	1	2400	0.76	103	0.14	0.5	0.45	0.48
ibm_perth	7	32	0	15	0.83	2900	1	120	0	0.42	0.5	0.46

Table 3.4: Backend Selection for Job 2 (GHZ State - using 10 qubits). ibmq_hanoi backend is chosen with the highest $\epsilon = 0.8$.

Backend	**Qubits**	**QV (v)**	$\bar{v}$	**QC depth (d)**	$\bar{d}$	**CLOPS (c)**	$\bar{c}$	**Workload (w)**	$\bar{w}$	**p**	**s**	ϵ
ibm_washington	127	64	0.33	13	1	850	0	5	1	0.67	0.5	0.59
ibmq_kolkata	27	128	1	13	1	2000	0.74	212	0	1	0.37	0.69
ibm_hanoi	27	64	0.33	13	1	2300	0.94	22	0.92	0.67	0.93	**0.8**
ibmq_guadalupe	16	32	0	13	1	2400	1	103	0.53	0.5	0.77	0.64
ibm_perth	7	-	-	-	-	-	-	-	-	-	-	-

[11]https://k8slens.dev/
[12]https://jmeter.apache.org/

Table 3.5: Backend Selection for Job 3 (Shor's algorithm to factorize number 9 - using 18 qubits). ibmq_kolkata backend is chosen with the highest $\epsilon = 0.7$.

Backend	Qubits	QV (v)	$\bar{v}$	QC depth (d)	$\bar{d}$	CLOPS (c)	$\bar{c}$	Workload (w)	$\bar{w}$	p	s	ϵ
ibm_washington	127	64	0.33	38381	0	850	0	4	1	0.17	0.5	0.34
ibmq_kolkata	27	128	1	37776	1	2000	0.79	130	0	1	0.4	**0.7**
ibm_hanoi	27	64	0.33	38190	0.32	2300	1	57	0.58	0.33	0.79	0.56
ibmq_guadalupe	16	-	-	-	-	-	-	-	-	-	-	-
ibm_perth	7	-	-	-	-	-	-	-	-	-	-	-

Function Scalability Evaluation

As the underlying orchestration technique is based on Kubernetes, we can enable the auto-scaling feature to scale out the function deployment horizontally (i.e., increase the number of function replications), dealing with the scenario when the request workload grows significantly from multiple concurrent connections. To evaluate the effectiveness of different scalability levels, we perform a set of evaluations on the 10-qubit Qiskit QRNG function. In this evaluation, we increase N - the number of concurrent users from 8 to 64, using JMeter 5. In each case, we consider a set of three different scenarios: *non-scale* (1 pod/function), scale out to N/2 pods, and scale out to N pods (we note that the number of pods is fixed for evaluation purposes only). For example, suppose there are 64 users (N) invoking the function simultaneously; we will conduct three test cases: 1 pod, 32 pods (N/2), and 64 pods (N), and record the average response time and the standard deviation.

Figure 3.8 demonstrates the result of our benchmarking. Overall, it is clear that if the function is non-scalable, the average response times for high-demand scenarios significantly increase. The previous section shows that the average response time for the 10-qubit Qiskit QRNG function is 81 ms. This figure jumps dramatically, up to 1703 ms, if 64 users use the function simultaneously. However, thanks to the containerization approach in our framework, we can quickly scale out deployment in seconds to ensure the response time is maintained. We can see that the average response time fluctuates between 87 to 148 ms if we scale out to N pods or from 102 to 180 ms when the number of pods is N/2.

It is important to notice that by scaling a quantum function, our approach is to replicate its classical instance deployment (i.e., Kubernetes pod), which is comparable to any existing serverless system. Its main objective is to handle concurrent requests from mul-

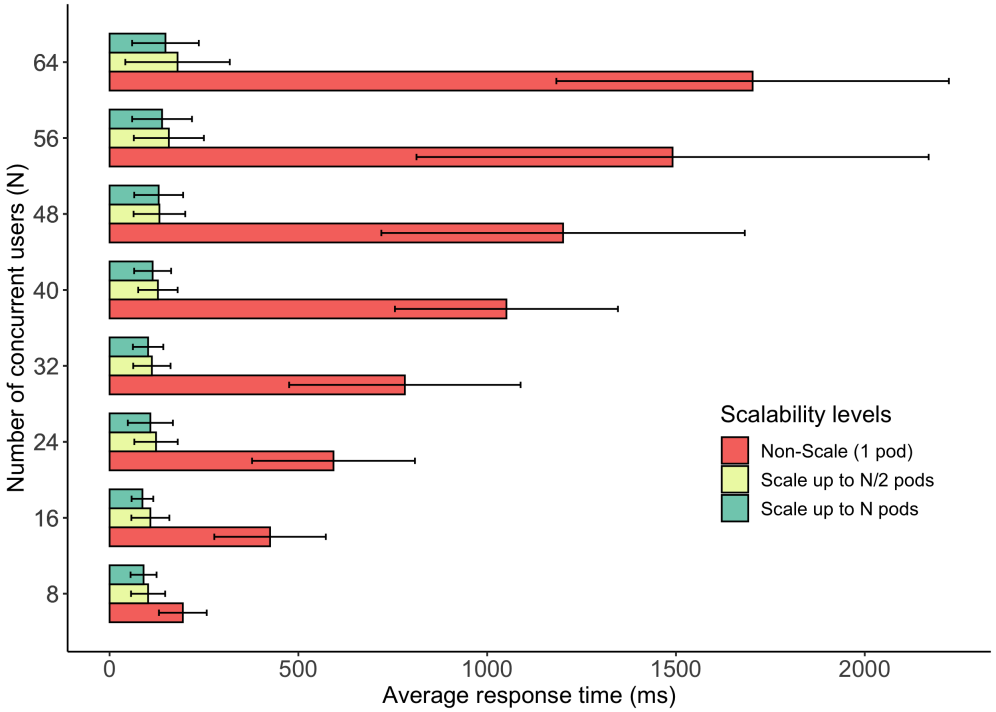


Figure 3.8: Scalability evaluation on 10-qubit Qiskit QRNG function.

tiple connections efficiently by reducing total response time. As shown in Figure 3.8, this scaling approach shows significant performance improvement in the case of using *internal quantum simulators* (for function testing or prototyping purposes) as the corresponding quantum simulator is incorporated into each function instance.

This scaling approach has no impact on the general performance in a special situation when all concurrent users manually select the same quantum backend for execution, as all quantum circuits will be forwarded to the same backend. However, this limitation can be addressed by designing an automatic backend selection algorithm based on the availability of quantum resources at the provider as our proposed algorithm (see Section 3.5.3). This way, each time the new instance of the same function is triggered, it will execute the backend selection individually to determine the best-suited quantum backend for execution without overflowing the same quantum backends. Considering the limitations of NISQ devices and available information about quantum jobs provided by the quantum cloud vendor, our scaling and backend selection strategy can still offer a best-effort and viable approach to facilitate the service-oriented quantum application

requirements. We have planned to advance these techniques in future releases of QFaaS when more information about waiting jobs is publicly accessible from the provider.

3.6.3 QFaaS Backend Selection Validation

To illustrate and validate the operation of the proposed Backend Selection policy, we present a case involving three invocations of different quantum functions (Deutsch-Jozsa's algorithm [191] using 5 qubits, the GHZ state [218] using 10 qubits, and Shor's algorithm[2] using 18 qubits) that need to be executed on the most suitable quantum backend. We assign equal weight to all factors, with $\gamma = 0.5$, thereby ensuring a balanced contribution of all aspects (including quantum volume, circuit depth, CLOPS, and workload) to the final backend decision. It is important to note that users can adjust these weight parameters according to their preferences, prioritizing either precision or execution speed. We assume that five quantum backends are available for the backend selection, each supporting a different number of qubits ranging from 7 to 127. Quantum volume and CLOPS are fixed values associated with each quantum backend, sourced from IBM Quantum [16], while the quantum circuit depth and workload are variable. Therefore, the quantum backend can be dynamically selected based on the current quantum circuits to be executed and the status of all available quantum backends (once per function invocation). Table 3.3 illustrates the backend selection process for the first quantum job. While the *ibm_washington* backend has the largest number of qubits and is less busy, its quantum volume and CLOPS are significantly lower compared to other 27-qubit quantum computers. Considering all the factors, the *ibm_kolkata* backend is chosen as it has a higher chance of achieving both higher precision and faster execution. For the second quantum job (see Table 3.4), only four backends with more than 10 qubits are considered. Since all the transpiled quantum circuits have the same depth, the depth scores are normalized to 1. The *ibm_hanoi* backend is eventually selected as it has the highest score of 0.8. Lastly, the third quantum job involves Shor's algorithm to factorize 9, which requires 18 qubits and a significant number of layers in the transpiled circuit (Table 3.5). Only three backends meet the key requirement for the number of qubits, and among them, the *ibmq_kolkata* backend with the highest score is chosen.

It is also essential to highlight that the backend selection policy presented here is an approximation approach, as certain metrics, such as the workload of a quantum backend, can be tricky. For example, a backend with a smaller number of pending jobs does not necessarily guarantee faster execution, as those pending jobs could be large tasks requiring a longer processing time. However, users have the flexibility to disregard or assign a lower priority (close to 0) to this metric when determining the speed score of a backend. In our future plans for advancing the QFaaS framework, we aim to incorporate more advanced techniques, such as machine learning, which automatically adjust these metrics to enhance the backend selection process. This will further refine the effectiveness of the policy, offering improved decision-making capabilities within the QFaaS framework.

3.6.4 QFaaS Cold Start Mitigation Evaluation

We evaluate the cold start mitigation strategy with Grover's algorithm to diverse the use cases of QFaaS. Grover's algorithm [3] is a quantum search algorithm that provides a quadratic speedup over classical counterparts. Its complexity grows with the number of qubits, which in turn affects the depth of the quantum circuit and the qubit connectivity.

We deploy different variations of Grover's algorithm from 2 qubits to 7 qubits, using the quantum circuits and transpiled QASM files of the MQT Bench dataset [1] (no ancilla qubit version, transpilation optimization level 3 to IBM Quantum 27-qubit quantum computers). We measure the average start-up latency from the function invocation until the function is ready for the quantum execution (i.e., finish the circuit compilation and transpilation) with the results as shown in Figure 3.9.

It is obvious that our transpilation caching approach (warm start with caching) significantly reduces the latency compared to both the cold start and the warm start without caching scenarios. For example, in the Grover-7 function (7 qubits), the cold start latency is around 15s, the warm start without transpilation caching is around 8s, while the QFaaS transpilation caching approach brings it down to under 0.628s, demonstrating a substantial improvement in execution preparation time. This pattern of reduction holds consistently across all evaluated Grover-n functions, with the caching approach offering a significant decrease in function preparation latency. This evaluation suggests

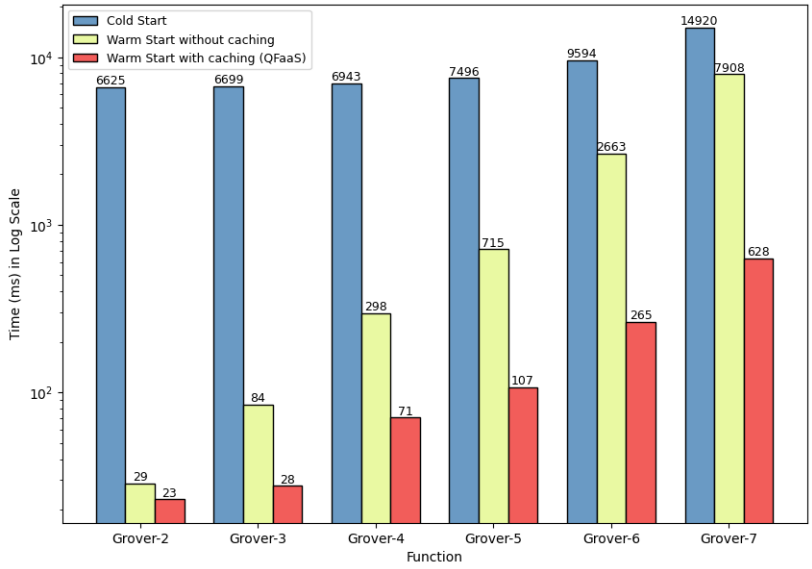


Figure 3.9: Quantum function cold start latency mitigation evaluation with Grover-n function, where n is the number of qubits. Grover's algorithm circuits and transpiled QASM files adopted from the MQT Bench dataset [1], transpilation optimization level 3 to IBM Quantum 27-qubit devices

that the QFaaS's transpilation caching strategy can effectively handle the classical cold start problem for quantum function execution, thereby enhancing the system's responsiveness and performance for end-users.

3.6.5 Industry Use Cases of QFaaS

It is imperative for industries to start investing in quantum computing services to stay competitive in the cloud-based computing market. QFaaS can serve as a proof-of-concept case study and reference system for further practical development. For example, the company Citynow Asia in Japan [37] has already started using QFaaS to develop their quantum serverless platform (QuaO). We expect to see growing interest in quantum serverless computing and the adoption of QFaaS in the future.

3.7 Discussion and Lessons Learned

This section presents key lessons learned regarding both the opportunities and limitations of the QFaaS framework, providing a roadmap for future advancements in QFaaS and the broader quantum serverless domain. Throughout the empirical development and rigorous validation of our QFaaS framework, we have gained pivotal insights into the prospects of the serverless approach to quantum computing, including:

- ***L1:*** *Serverless computing holds great potential to accelerate quantum software development.*

  From the software developers' perspective, serverless approaches can help to obviate the necessity for quantum software environment setup, service deployment, and quantum infrastructure configuration, enabling them to concentrate on application development and experimentation without worrying about the underlying system. Quantum functions can be developed and deployed in a manner analogous to classical functions, ensuring seamless integration into existing application workflows. For quantum cloud providers, serverless models can provide an efficient, cost-effective means for allocating resources, optimizing utilization, and reducing idle time. By effectively implementing serverless models and offering a competitive pay-per-use cost model, providers can enhance user engagement and encourage long-term commitment to their services.

- ***L2:*** *The state-of-the-art software techniques and workflows can be effectively leveraged to expedite the quantum serverless paradigm.*

  Our work not only theorizes but also empirically demonstrates the adaptation of the DevOps methodology and techniques within the QFaaS framework, such as containerization, continuous integration, and continuous deployment. The development process of a quantum function can further be split into multiple stages. Quantum simulators can be used in the initial prototyping and testing phase, while quantum computers can be used later on during the production stage.

- ***L3:*** *The hybrid architecture of QFaaS represents one of the adaptive and practical approaches to facilitate quantum serverless systems, reflecting the current reliance on quan-*

tum execution on classical runtimes.

This architecture highlights the crucial cooperation between classical and quantum computation resources. In this setup, the classical counterpart acts as the runtime server, responsible for storing and compiling the quantum code prior to its execution on the designated quantum computer. The abstraction for decomposing a complex application into multiple smaller quantum functions adapts to the NISQ devices. The development and deployment of quantum functions are simplified and streamlined by the employment of DevOps techniques. The quantum function performance can also be optimized based on the needs of users thanks to the lightweight yet adaptive backend selection strategy.

Despite the promising potential of the quantum serverless approach, the current state of quantum hardware and software presents several significant challenges. These serve as crucial lessons learned for further exploration in our work:

- ***L4:*** *Current NISQ hardware is expensive, unreliable, and constrained, making it significantly challenging to incorporate the quantum serverless paradigm into production environments.*

  The intrinsic noise in these devices poses substantial challenges to the reliability and accuracy of quantum operations, where techniques such as quantum error correction and mitigation can be employed. Furthermore, the execution of tasks on real quantum devices is often associated with long queuing times and inconsistent execution durations, making them less suitable for time-sensitive and real-time applications. Throughout our empirical study with QFaaS, even the execution of Shor's algorithms requires minutes on current platforms, which is unanticipated for a common serverless execution. However, the rapid advances in quantum hardware in recent years hold promise for addressing this problem in the near future.

- ***L5:*** *Quantum resources cannot be scaled in the same manner as their classical counterparts.*

  Contrary to the classical domain, where serverless offers clear scalability and resource management benefits, the quantum realm introduces unique challenges.

The scaling constraint of quantum hardware underscores the need for unique approaches to resource management in quantum serverless architectures. Our research recognizes that the advantages seen in classical serverless computing may not translate directly to quantum computing without significant adaptations and optimizations, specifically concerning the scalability of quantum resources. Techniques such as virtualization and containerization for quantum resources are potential directions to improve the utilization and scalability of quantum computing resources in the future.

- ***L6:*** *The quantum serverless model is still in its early stages, and there are numerous open problems that require extensive research and attention.*
  The application of serverless computing for quantum applications is still an emerging area where more knowledge is needed to understand the full extent of its suitability and benefits. A serverless approach may introduce challenges for non-trivial quantum applications that require iterative adjustment and optimization, such as quantum machine learning. Besides, the nature of current NISQ devices can also limit the potential of a quantum serverless approach. Despite this, the serverless approach has gained prominence due to its ability to reduce costs, improve scalability, and eliminate the need for hardware-side management, which can be particularly beneficial for future quantum software development. Indeed, quantum vendors, such as IBM Quantum, have placed quantum serverless as their key priority in the development roadmap [114]. There are numerous open problems that require further extensive research and attention to exploit the potential of a quantum serverless approach. Key areas that necessitate additional exploration include circuit cutting, optimizing the orchestration of hybrid quantum-classical tasks, circuit transpilation caching, reducing quantum cold start, and developing adaptive backend selection mechanisms. Resolving these challenges will play a critical role in facilitating the widespread adoption of the quantum serverless paradigm in the near future.

3.8 Related Work

This section discusses the related work in the context of frameworks for developing service-oriented quantum software. To the best of our knowledge, QFaaS can be considered one of the pioneers in serverless-based function-as-a-service frameworks for quantum computing. Table 3.6 summarizes the difference between QFaaS and related work in the context of their various capabilities.

It is important to note that all related frameworks [22, 219], empirical studies [108, 221] and minimum viable product (MVP) [222] designs did not provide an open-source or detailed description for reproducible purposes. Besides, two other SDKs [201, 220] are related but their classification is not a quantum serverless framework. Therefore, we can only use the feature information reported in these studies to compare with our framework capabilities for reference purposes but cannot fully validate the features of related work in practice.

Existing quantum software platforms lack various features to provide a universal environment for developing service-oriented quantum applications. Hevia et al. proposed QuantumPath (QPath) [22], which is a software development platform aiming to support multiple quantum SDKs for gate-based and annealing quantum applications, and provide multiple design tools for creating a quantum algorithm. However, QPath does not support a serverless computing model and scalability features for further expansion, and it is still in the preliminary phase without providing a performance evaluation to validate the proposed design. Fu et al. [201] proposed the overall framework for developing heterogeneous quantum-classical applications, adapting with NISQ devices. Instead of working with popular quantum languages and SDKs, they also proposed a new programming language for quantum computing, called Quingo. Although this is an exciting direction for further quantum framework development, it can face many challenges when developing a new programming language compared to improving well-known languages. For example, expanding the support for large developer communities, security testing for potential vulnerabilities, and covering all aspects of quantum and classical computation. In another way, Claudino et al. [219] proposed the XACC framework, which extended the C++ programming language to support quan-

Table 3.6: A summary of related work and their comparison of system and software engineering aspects for quantum computing

Criteria		**QPath** [22]	**Quingo** [201]	**XACC** [219]	**QAPI** [219]	**t|ket** [220]	**algo2qpu** [221]	**SCQ** [222]	**QFaaS** *(Proposed)*
Type (Main category)		F	SDK	F	ES	SDK	ES	ES	F
Systems	1. Serverless without Vendor lock-in	×	×	×	×	×	×	×	✓
	2. Modular/Microservices Components	×	×	×	×	✓	×	✓-	✓
	3. Support multiple Quantum Clouds	✓	×	×	×	✓	×	×	✓
	4. Support multiple Quantum Simulators	✓	×	×	×	✓	×	×	✓
	5. Containerization/Kubernetes Integration	×	×	×	×	×	×	✓-	✓
	6. Distributed and Scalable System	×	×	×	×	×	×	×	✓
	7. Evaluation on NISQ devices	×	×	✓	✓	✓	✓	×	✓
	8. Permanent Data Storage	✓	×	×	×	×	×	×	✓
	9. Security & User Authentication	×	×	×	×	×	×	×	✓
	10. Monitoring Integration	×	×	×	×	×	×	×	✓
Software	11. Hybrid Quantum-Classical Integration	×	✓	✓	×	×	✓	✓-	✓
	12. Support multiple Quantum SDKs	✓	×	×	×	✓	×	×	✓
	13. Built-in Software Library/Templates	✓	✓	✓	×	✓	×	×	✓
	14. Quantum Software Workflows	×	✓	✓	×	×	✓	×	✓
	15. Quantum Backend Selection	×	×	×	×	×	×	×	✓
	16. REST API support with API Gateway	×	×	×	✓	×	×	×	✓
	17. Full-stack Software Framework	✓	×	×	×	✓	×	×	✓
	18. DevOps (CI/CD) Integration	×	×	×	×	×	×	×	✓
	19. Practical Use Cases	✓	✓	✓	×	✓	✓	×	✓
	20. Open-Source Software	×	✓	×	×	✓	×	×	✓

Notes. F: Framework, SDK: Software Development Kit, ES: Empirical Study, ✓: Yes, ×: No, -: Not validated

tum chemistry simulations. However, this framework focused solely on designing multiple quantum algorithms for chemical problems and simulating quantum computation on GPU backends. Cambridge Quantum Computing proposed $t|ket\rangle$ [220], which is an open-source language-agnostic quantum compiler for NISQ devices. This framework focuses on circuit optimizations, transformation, and qubit mapping features and does not consider the service-oriented quantum application approach. In light of the Quantum Computing as a Service approach, Garcia-Alonso et al. [108] proposed the proof-of-concept about Quantum API Gateway with two simple API endpoint validations (execution and feedback) using Python and Flask platform on the Amazon Braket platform. This work also presented an execution time forecasting model and quantum computer recommendation. Still, no information about what kind of quantum SDKs, quantum problems, or datasets are used for the forecasting is provided. There is also some Proof-of-Concept (PoC) designs for cloud-based quantum software development that have been proposed in recent years. For instance, Sim et al. [221] proposed *algo2qpu*, a hardware and software agnostic framework that supports designing and testing hybrid quantum-classical algorithms on the Rigetti cloud-based quantum computer. Using their proposed framework, they implemented two applications in quantum chemistry and machine learning. However, similar to the work mentioned above, *algo2qpu* also did not apply the serverless quantum computing model but considered the standalone quantum application model instead. Another minimum viable product (MVP) design proposed by Grossi et. al [222] suggested using IBM services such as IBM Cloud Functions and IBM Containers to integrate serverless features for Qiskit programs. However, this work only considered single-SDK and single-vendor environments with no implementation or validation provided. This work did not consider the software workflow or the backend selection strategy and depended on a specific quantum SDK and platform technology. This approach can lead to a data lock-in problem, which is one of the most serious challenges of serverless computing.

As shown in Table 3.6, we use ten criteria (1-10) to evaluate in terms of system architecture and computing model. The major contribution of QFaaS is one of the first frameworks adopting the serverless function-as-a-service model for quantum computing, which avoids vendor lock-in problems of serverless computing by supporting mul-

tiple quantum SDKs/programming languages, quantum cloud providers, and simulators (1). The idea of serverless integration is also proposed in [222] using Qiskit SDK and IBM techniques. Still, no implementation and evaluation are provided to validate the design, and it can also lead to the vendor lock-in problem as that design relies on single vendor techniques. The core elements of the QFaaS system are developed on top of open-source cloud-native technology, such as Docker container, Kubernetes, OpenFaaS, and Gitlab, with flexibility for further expansion and integration (2). This system design leverages the state-of-the-art techniques in classical cloud computing to support the quantum computing as a service (QCaaS) model, which adapts to the limitations of the NISQ computers. We validated the system design and used QFaaS to evaluate the performance of multiple quantum simulators and computers to provide insight into the current state of the NISQ era (3-7). The database integration with a common data scheme for all different SDKs in QFaaS allows users to perform further analyses and avoids data lock-in issues (8). Furthermore, we also incorporate other essential system components, such as monitoring and security for QFaaS, to demonstrate the completed quantum software stack in practice (9-10).

Regarding software engineering aspects to facilitate the Quantum FaaS model, we consider the ten remaining features (11-20). Our framework simplifies the hybrid quantum-classical integration model by providing function templates and a supported software library (11-12). It allows users to write both classical and quantum code in a single file. Besides, we also propose an adaptable quantum software life cycle with seven stages of a quantum function and demonstrate its implementation in practice. This is the first quantum function life cycle, as other proposals in the literature focused on the general, standalone quantum software (14). We design a backend selection strategy and directly apply it to QFaaS to automatically select the most suitable quantum backends for executing the quantum computation parts (15). As several works in the literature only proposed the proofs-of-concept (PoC) [108, 221, 222], we do not only propose the PoC but also fully implement and validate the proposed PoC with practical use cases (19) of well-known quantum circuits. Apart from adopting open-source cloud-native techniques, we also develop full-stack software components for quantum functions. QFaaS backend included a complete OpenAPI set with a centralized API gateway (16),

a Python-based supported library, and multiple Docker-based function templates (13), where its front end is a user-friendly web application. We demonstrate the first integration of continuous integration and continuous deployment (CI/CD) of DevOps into the quantum software workflow (18). Finally, as we incorporated several latest open-source techniques, our framework is also designed to be a part of the quantum open-source software ecosystem to contribute to our effort in the early development of serverless quantum computing (20).

In summary, as no existing quantum function-as-a-service framework in the literature consider the multi-SDK, multi-cloud environment, QFaaS is one of the first practical platforms that seriously investigated and prepared the initial steps toward a universal serverless quantum computing architecture. Our major innovation also includes bringing state-of-the-art system design and software engineering techniques in classical computing to support service-oriented quantum software development and mitigate the challenges in the current NISQ era.

3.9 Summary

In this chapter, we proposed QFaaS - a holistic serverless framework for developing quantum function as a service, enabling software engineers to leverage their knowledge and experience to adapt to quantum counterparts in the *Noisy Intermediate-Scale Quantum* era quickly. Our framework brings state-of-the-art methods such as containerization, DevOps, and the serverless model to reduce the burden of quantum software development and pave the way toward combining hybrid quantum and classical components. QFaaS provides essential features with multiple quantum software environments, leveraging the well-known quantum SDKs and languages to develop quantum functions running on multiple quantum simulators or cloud-based quantum computers. The current implementation of QFaaS demonstrates the possibility and advantages of our framework in developing quantum function as a service without vendor lock-in issues, and can be seamlessly integrated into the current software workflow. Throughout our empirical implementation and evaluation, we also highlight the lessons learned and limitations of the quantum serverless approach that need to be further investigated.

Software Availability:

The QFaaS Framework with the source code of all components and sample functions code can be accessed from our iQuantum Initiative website:
http://clouds.cis.unimelb.edu.au/iquantum.

Chapter 4

Modeling and Simulation of Quantum Computing Environments

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, in this chapter, we propose 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.

This chapter is derived from:

- **Hoa T. Nguyen**, Muhammad Usman, and Rajkumar Buyya, "iQuantum: A toolkit for modeling and simulation of quantum computing environments", *Software: Practice and Experience (SPE)*, Volume 54, Issue 6, Pages: 1141-1171, ISSN: 0038-0644, Wiley Press, New York, USA, June 2024.

4.1 Introduction

Quantum computing holds enormous promise for solving computationally intractable problems, revolutionizing various domains such as drug discovery [41], finance [190], optimization [223], and machine learning [68, 70, 224]. The emergence of the cloud-based quantum computing [51, 53] and quantum computing-as-a-service (QCaaS) [225] models has enabled access to quantum computation resources without a massive up-front investment in quantum hardware, leading to tremendous progress in quantum software and algorithm fronts [57]. Major cloud providers, such as Microsoft Azure [17], AWS [98], and IBM [16], now offer cloud-based access to their quantum computing services. Moreover, quantum computation resources are predicted to be extended to the edge network [226] when quantum hardware becomes popular in the future, envisioning the emergence of the hybrid paradigm of quantum cloud-edge continuum [227], whose main components are illustrated in Figure 4.1. 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 [228]. 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 [229, 230]. 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 [15].

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 [155]. However, there are several challenges to designing and evaluating system and resource orchestration policies in practical environments, as previously discussed in Chapter 3. First, access

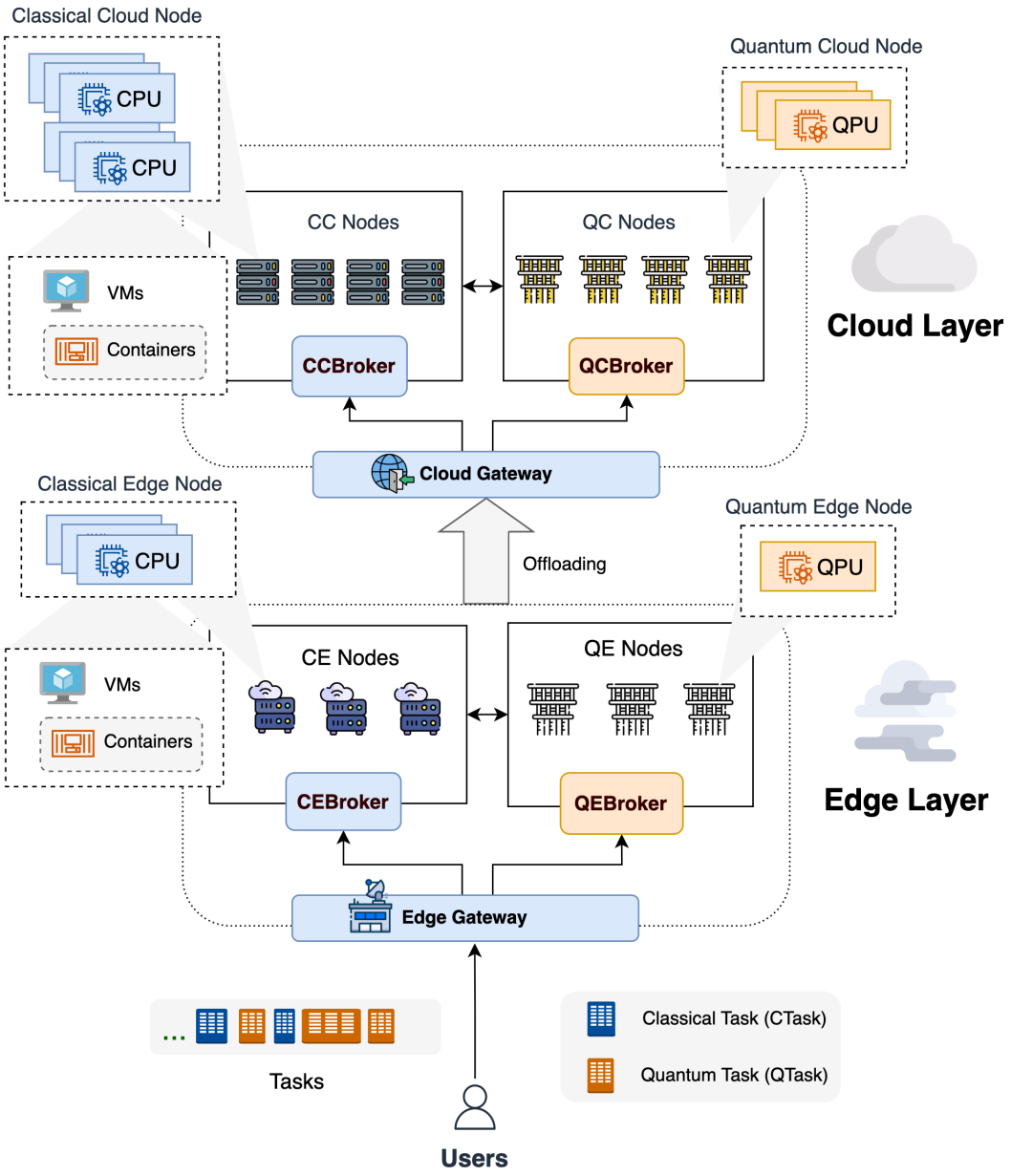


Figure 4.1: Overview of Hybrid Quantum Computing Paradigm envisions the seamless integration of quantum and classical computation resources across different cloud and emerging edge layers, with edge resources being geographically closer to end-users (data sources), albeit with computational limitations compared to their cloud counterparts.

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 [155]. 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 [15], 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 [34] 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 [231], FogNetSim [232], iFogSim [35], EdgeSimPy [36]. 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 [233]. Besides, several quantum simulators focus on quantum communications, which support modeling quantum network

protocols [234, 235]. While quantum networks represent a promising direction for the future of quantum computing [93], 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 [236]. We leverage the core engine and classical components of the latest version of CloudSim [34] 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.

In this chapter, 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 envi-

ronments in the cloud-edge continuum. The major contributions of our extended study are as follows:

- 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.

- 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.

- We validate and evaluate iQuantum in different scenarios using trustworthy datasets, including IBM Quantum [16] calibration data for quantum systems and the MQT Bench dataset [1] 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.

- 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 chapter is structured as follows: Section 4.2 reviews related work and identifies gaps in the literature. Section 4.3 outlines the system model for the quantum computing environment. Section 4.4 proposes the architecture design and imple-

mentation of iQuantum. Section 4.5 presents different models for various use cases of iQuantum in resource management and orchestration problems. Section 4.7 shows an explanatory example, sample workflow operation of iQuantum, and performance evaluations from various scenarios and workload datasets. Section 4.8 discusses the lessons learned from developing iQuantum. Finally, Section 4.9 concludes the chapter.

4.2 Related Work

Table 4.1 summarizes the overall comparison of iQuantum and other related work in terms of modeling and simulation toolkit as well as quantum computing simulation.

In the classical computing domain, modeling and simulation tools such as CloudSim [34] 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) [237], paradigms such as edge and fog computing are gaining prominence [238, 239]. 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.* [35] proposed iFogSim2, which extends the first version of iFogSim [240] to support mobility, clustering, and microservices management policies in fog and edge computing. Similarly, Souza *et al.* [36] 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) [241] and medical applications [242] 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 [243] and xFogSim [244], 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.

Table 4.1: A feature comparison overview of related works with our iQuantum toolkit

Toolkits	Focus	Environment Modeling			Policies Modeling	Datasets Support	Event Simulation	Integration Possibility	Language
		Quantum	*Cloud*	*Edge*					
QuNetSim [234]	Network	E	×	×	Network Protocols	-	-	✓	Python
NetSquid [235]	Network	E	×	×	Network Protocols	-	✓	✓	Python
QuEST [245]	Circuit Operation	E	×	×	-	-	-	✓	C++
PAS [246]	Circuit Operation	E	×	×	-	-	-	-	-
QXTools [247]	Circuit Operation	E	×	×	-	✓	-	-	Julia
iQuantum *(Our toolkit)*	System & Workload	S	✓	✓	Resources Management	✓	✓	✓	Java

✓: Supported; ×: Unsupported; -: N/A; E: Emulation ; S: Simulation
(*Classical cloud features derived from CloudSim)

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 [34].

Industry-standard quantum simulators, such as IBM's quantum simulator integrated with Qiskit [195] and Google's quantum simulator employed with Cirq [196], 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 [34], iFogSim [35], and EdgeSimPy [36]. 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 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 [1]) 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 4.7).

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.* [234] 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 [248], ProjectQ [249], and QuTiP [250]), its main objective is to develop quantum network protocol simulation rather than distributed quantum system modeling. Similarly, NetSquid [235] 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 [245] to support simulating the behavior of quantum systems with high performance. Similarly, Bian *et al.* [246] proposed PAS as a lightweight

quantum simulator, and the authors argued that PAS outperforms QuEST in terms of quantum operation simulation. QXTools [247] 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 [34], 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 (see Chapter 3) 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 [195], 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[1], 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.

4.3 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 4.2 illustrates these entities and main attributes, which are described in detail below.

4.3.1 QPUs and Quantum Computation Nodes

Quantum Processing Units (QPUs)

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 $\mathcal{Q}$ can be defined as follows:

$$q_i = \{q^w, q^v, q^s, q^g, q^t, q^e\} \tag{4.1}$$

where:

- 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.
- 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.* [29] proposed the measurement of QV as follows:

$$q^v = 2^{\min(d,m)} \tag{4.2}$$

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

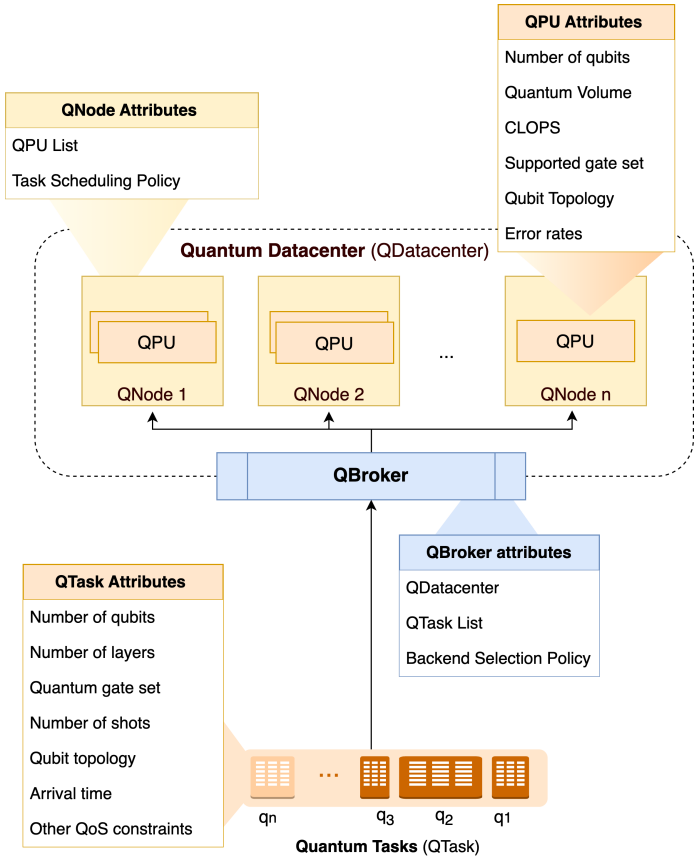


Figure 4.2: Overview of the system model and key attributes of entities in quantum computing environments

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.

- 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

$$q^s = \frac{M \times K \times S \times D}{time\_taken} \tag{4.3}$$

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.

- 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.

- 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.

- 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.

Quantum Nodes (QNodes)

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

$$\mathcal{Q} = [\{q_i | 1 \leq i \leq n\}, \pi_s] \tag{4.4}$$

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[2] 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 4.5.1.

4.3.2 Quantum Datacenters and Brokers

Quantum Datacenters (QDatacenters)

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:

$$\mathcal{D}^{\mathcal{Q}} = [\{\mathcal{Q}_i | 1 \leq i \leq |\mathcal{Q}|\}, \xi, \chi] \tag{4.5}$$

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

[2]IBM Quantum System Two

at $1.6 USD per second of quantum runtime[3], whereas Amazon Braket [98] 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 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.

Quantum Brokers (QBrokers)

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:

$$\mathcal{B}^{\mathcal{Q}} = \{\mathcal{D}^{\mathcal{Q}}, \Gamma, \pi_b\} \tag{4.6}$$

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.

4.3.3 Quantum Tasks (QTasks)

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. 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

[3]https://www.ibm.com/quantum/access-plans

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:

$$\gamma_i = \{\gamma^a, \gamma^g, \gamma^w, \gamma^d, \gamma^s, \gamma^e, \gamma^t\} \tag{4.7}$$

where:

- γ^a is the arrival time of the quantum task.
- 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.
- γ^w is the quantum circuit width, which is measured by the number of qubits that need to be used.
- γ^d is the number of circuit layers, which can be assumed to be directly related to the depth of the circuit.
- γ^s is the number of shots (*i.e.*, execution repetition) that circuit need to be executed.
- 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 [1] and mapped them to corresponding QTasks to represent the workload needed to be processed within the quantum computing environment (see Section 4.7).

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 4.5.

4.4 iQuantum Architecture Design and Implementation

4.4.1 Architecture and main components

We developed iQuantum using the *discrete-event* simulation approach, leveraging the core components of the CloudSim toolkit [34]. The initial design of iQuantum aimed to demonstrate the possibility of modeling quantum computing environments. 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 4.3. We have significantly enhanced the original design through extensive refactoring and added new features to enhance the simulation process for both quantum and hybrid quantum-classical environments.

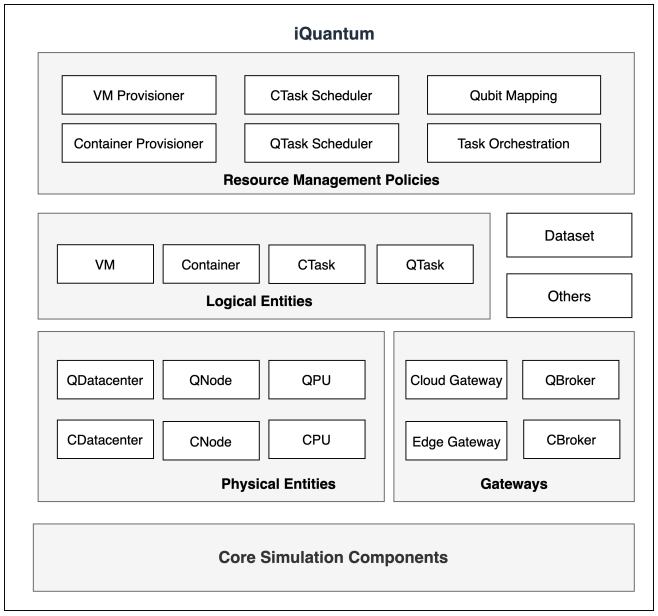


Figure 4.3: Architecture Design of iQuantum incorporates five main layers, dataset support, and other auxiliary components.

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.

Core Simulation Layer

The Core Simulation layer in iQuantum is built on the discrete event management framework in CloudSim [34], 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 toolkits, including iFogSim2 [35], ns-3 [251], and OMNET++ [252]. 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.

Physical Layer

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

Processing Units in iQuantum comprise both Central Processing Unit (CPU) and Quantum Processing Unit (QPU). 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) [34]. 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 4.2). 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.

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 Task (CTask). 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.

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.

Resource Management Layer

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

- *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.
- *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.
- *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.

4.4.2 Implementation of quantum components

The overview of the main class diagram for the quantum entities in iQuantum is shown in Figure 4.4. 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.

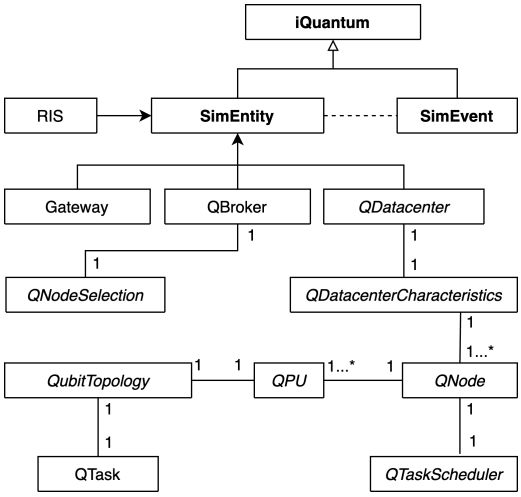


Figure 4.4: Overview of class diagram for quantum components

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 mul-

tiple 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` is defined in the `QDatacenterCharacteristics` class.

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 4.5.1.

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.

4.4.3 Quantum Nodes and Workload Datasets

Initially, simulation scenarios in iQuantum can be defined manually, similar to CloudSim [34]. 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.

- **Quantum nodes**: iQuantum supports the use of a calibration datasheet format, adopted from IBM Quantum [16], to model all attributes of a QNode, including detailed qubit topologies and error rates.
- **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 [1].

4.5 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 chapter.

4.5.1 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 2. 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:

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

Algorithm 2: 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 ;

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 [253], 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 [254] 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.

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.

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 [34]. Examples of comparable strategies for quantum tasks are illustrated in Figure 4.5. 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.

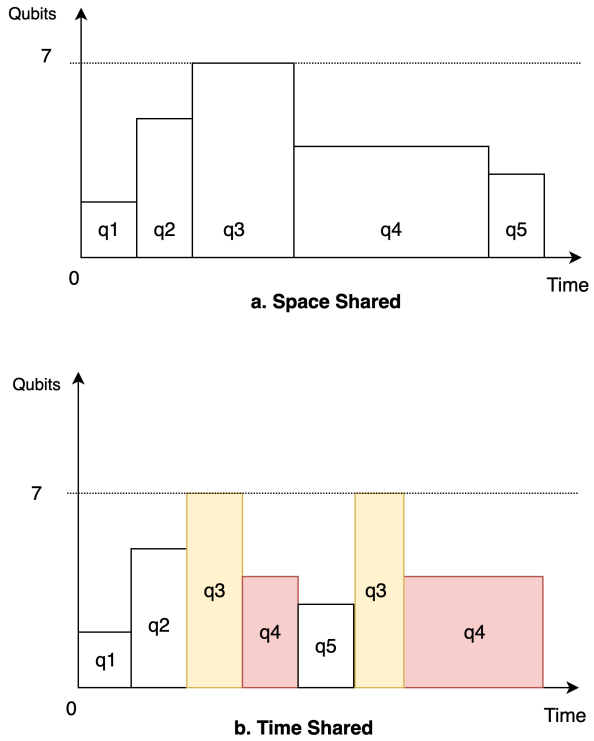


Figure 4.5: Example of two allocation strategies for five different quantum tasks (sequenced as q1 to q5) within a 7-qubit quantum node. a) The Space Shared policy dedicates separate quantum resources to each task. b) Time Shared policy allocates tasks in even time slots using a round-robin approach, resulting in pausing and resuming of tasks q3 and q4, rendering it impractical for quantum computing.

- 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.

- 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 [15] as it can be affected by noise and complicated qubit reset control.

4.5.2 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 [58]. This approach utilizes the advantages of classical computing for tasks such as data preprocessing and post-processing while assigning more computationally demanding tasks to quantum resources [255].

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 3. 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 de-

Algorithm 3: Hybrid Task Orchestration Logic

Input : All incoming tasks: $\Lambda = [\lambda_1, \lambda_2, ..., \lambda_n]$
QNodes: $\mathcal{Q} = [q_1^c, q_2^c, ..., q_m^c, q_1^e, q_2^e, ..., q_k^e]$
CNodes: $\mathcal{C} = [c_1^c, c_2^c, ..., c_p^c, c_1^e, c_2^e, ..., c_l^e]$
Backend Selection Policies: $\pi_{\mathcal{Q}}, \pi_{\mathcal{C}}$
All VM/Container Provision Policies

Output: List of placements: $\Sigma = [\sigma_1, \sigma_2, ..., \sigma_n]$

Initialize simulation clock $t \leftarrow 0$;
$\mathcal{C}_V^e, \mathcal{C}_V^c \leftarrow$ Apply VM/Container Provision Policies;
Submit all tasks to Edge gateway;
$\Sigma \leftarrow []$;
for $i \leftarrow 1$ *to* n **do**
    Update t & all processing time;
    $\lambda_i \leftarrow \Lambda[i]$;
    $b_T \leftarrow$ null;
    if $\lambda_i.type = QTask$ **then**
        Send λ_i to QEBroker;
        Apply QTask scheduling policy *(Algo. 1)*
        $b_T \leftarrow \pi_{\mathcal{Q}}(\lambda_i, \mathcal{Q}^e)$;
    else
        Send λ_i to CEBroker;
        Apply CTask scheduling policy;
        $b_T \leftarrow \pi_{\mathcal{C}}(\lambda_i, \mathcal{C}_V^e)$;
    end if
    $\sigma_i \leftarrow \{\lambda_i, b_T\}$;
    if $\sigma_i \leftarrow$ ***false*** **then**
        Offload λ_i to Cloud Gateway;
        $\sigma_i \leftarrow$ Repeat 8-18 on Cloud resources;
    end if
    Append σ_i to Σ;
end for
return Σ;
Distribute tasks according to Σ ;
Perform local scheduling process at nodes ;

scribed in Algorithm 2 (Section 4.5.1). For classical tasks, more discussion about task scheduling and resource management policies can be found in [34].

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

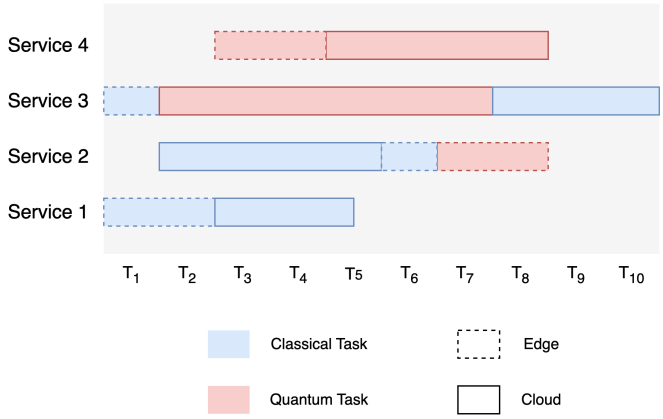


Figure 4.6: An example of hybrid task orchestration in Cloud-Edge continuum. Four services incorporate different quantum and classical tasks, varying in execution time and resource requirements. Each task can be executed on edge computing resources or offloaded to the cloud if edge resources are insufficient for execution.

An example of hybrid task orchestration in the cloud-edge continuum is illustrated in Figure 4.6. 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. Service 1 and Service 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. Service 2 and Service 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 used with CloudSim/iFogSim, one can model and simulate a hybrid quantum-classical computing environment and evaluate new hybrid task orchestration algorithms.

4.6 Example of Simulation Workflow

Figure 4.7 depicts an overview of the simulation workflow in iQuantum, which consists of four main stages.

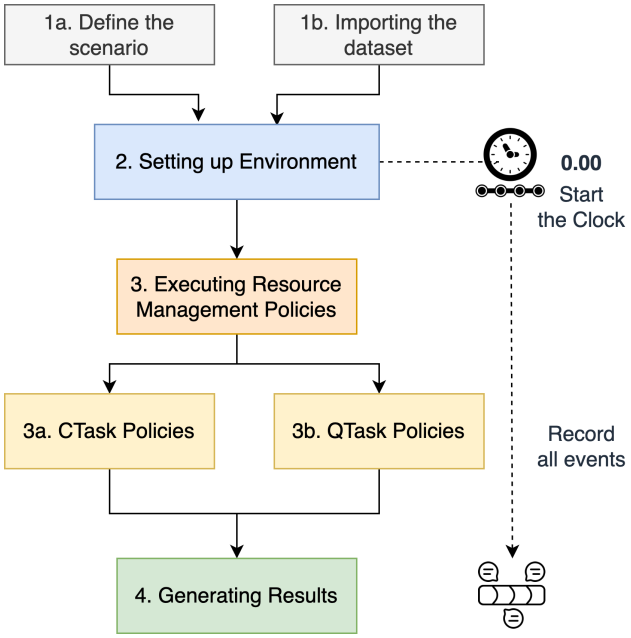


Figure 4.7: Overview of the Simulation Workflow in iQuantum

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 generate the outcomes for the user 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 4.8. 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 4,000 shots, while the metrics for QTask 2 are 50 layers and 1,000 shots.

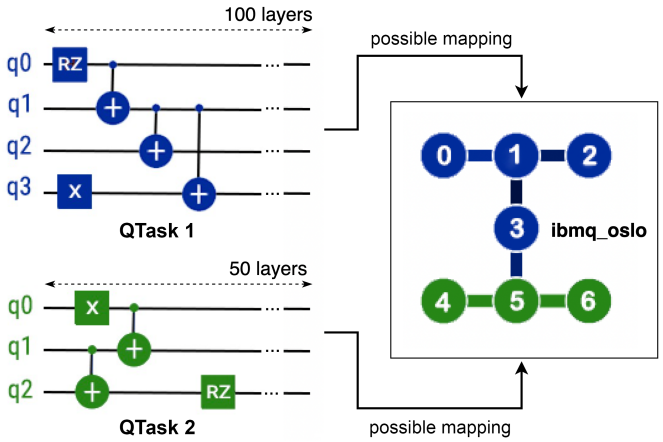


Figure 4.8: Example of two QTasks (left) and possible qubit mapping into ibmq_oslo node (right)

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 or automatically by using the predefined QNode and add to `qNodeList` as follows (Code 4.1):

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

Code 4.1: Sample code for importing ibmq_oslo QNode

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

object.

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

Code 4.2: Sample code for modeling a QDatacenter

Step 4: Create a `CloudGateway` and a linked `QCloudBroker` object as follows:

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

Code 4.3: 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:

```
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 4.4: 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:

$$t^q = \frac{\gamma^d}{q^s} \times \gamma^s \tag{4.8}$$

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 the quantum task needs 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 4.5.1.

Step 7: 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.

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

Code 4.5: Sample code for starting the simulation and print out the result

The simulator shows all events happening during the simulation period.

```
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 4.6: Sample events in the simulation

The simulation results show that two quantum tasks are submitted to the QNode at timestamp t = 0.01s (minimum interval between 2 different events). The execution times of QTask 1 and QTask 2 in the `QNode` are 153.85s and 19.23s, 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 173.08s. As noted above, these results are the same as when we manually calculated using Equation 4.8.

More large-scale simulation scenarios and evaluation of iQuantum are discussed in the following section.

4.7 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 4.5. The primary findings of the empirical experiments are discussed in this section.

4.7.1 iQuantum Simulation Verification

Simulation is a cost-effective and less complex alternative to empirical experimentation for understanding real-world systems. [256]. 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 [36].

For the system model validation, the conceptual model and metrics in quantum are devised based on well-known studies on quantum benchmarking [29, 32, 257, 258] without further modification. Besides, the validation and verification of classical components had been studied in CloudSim [34]. 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 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 [36, 256].

Scenario description

We use the quantum cloud-edge infrastructure as depicted in Figure 4.9 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 [16], are depicted in Table 4.2. The edge layer consists of a cluster of 5 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.

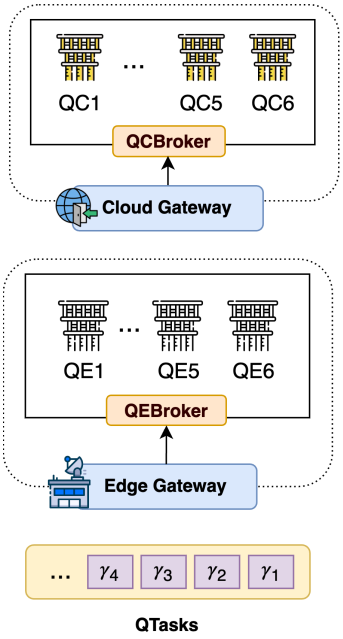


Figure 4.9: Overview of the infrastructure considered in iQuantum's verification and performance evaluation

To facilitate the verification, we use 4 QTasks that represent different quantum applications extracted from the MQT Bench dataset [1], in which the circuit attributes are shown in Table 4.3. All QTasks are initially submitted to the edge gateway for processing. Following the proposed system model and use cases (Section 4.5), 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 0.01s. 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.

Table 4.2: 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

Verification discussion

The dynamics of all events that occurred in the simulation of our verification are illustrated in Figure 4.10 and Table 4.4, following the similar verification approach of the EdgeSimPy simulator study [36]. At time step T_1 = 0.01s, all QTasks are 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 needs to wait at QE1 to be executed after γ_3 completion (following the Space-shared scheduling). At time step T_2 = 0.02s, γ_2 arrived at

Table 4.3: Attributes of QTasks in the iQuantum verification derived from the MQT Bench dataset

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

Table 4.4: 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

the cloud layer and is scheduled to QNode QC1 for execution for 8.72s, then finished at time step T_3 = 8.74s. After the completion of γ_3 for 14.74s, γ_4 starts its execution at QNode QE1 at time step T_4 = 14.75s. The remaining QTasks, γ_1, and γ_4, continue their execution and finish at time step T_5 = 16.32s and T_6 = 39.8s, respectively. It is obvious that all discrete events that occurred during the verification are correct and meet the initial expectation with the given input.

4.7.2 Performance Evaluation

To validate the effectiveness of iQuantum with the proposed use cases (discussed in Section 4.5), we further evaluate its performance and correctness of the implementation in different large-scale scenarios.

Figure 4.10: 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

Environment Setup

For the quantum cloud-edge system, we set up a similar infrastructure as the previous verification setting (see Figure 4.9 and Table 4.2) 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 [1] and extracted features of quantum circuits for 28 different quantum algorithms into four sets as shown in Table 4.5. 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 [259] 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 1,000 times, with the results discussed in the following part.

Table 4.5: Quantum Task Workload features, extracted from MQT Bench Dataset

Set	QTask count	Circuit Depth	Initial qubits	Mapped qubits	Algorithms types
1	300	13 - 7682	10-27	27	25
2	1000	13 - 58861	10-27	27	21
3	3500	10 - 87833	7-127	127	28
4	7000	10 - 161588	7-127	27-127	28

Discussion

Figure 4.11 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 Set 1 and Set 2 cases, all

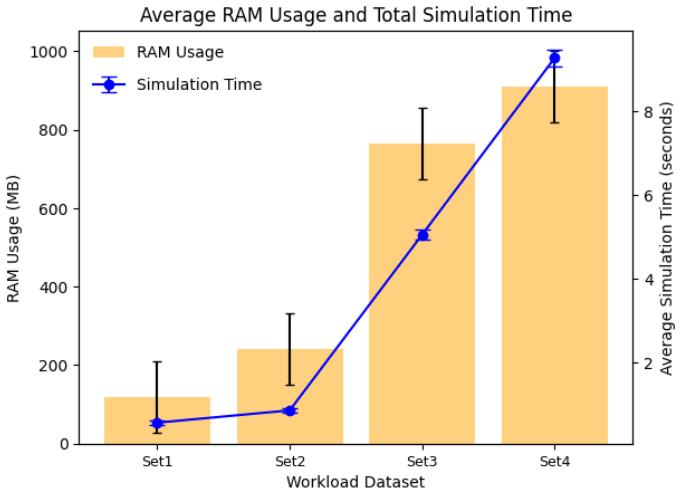


Figure 4.11: Average RAM Usage (bar chart) and Total Simulation Time (line chart) of iQuantum on four workload datasets (varying from 300 to 7,000 QTasks) over 1000 iterations.

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 second 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 (5,569 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 seconds to process 7,000 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.

Figure 4.12 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

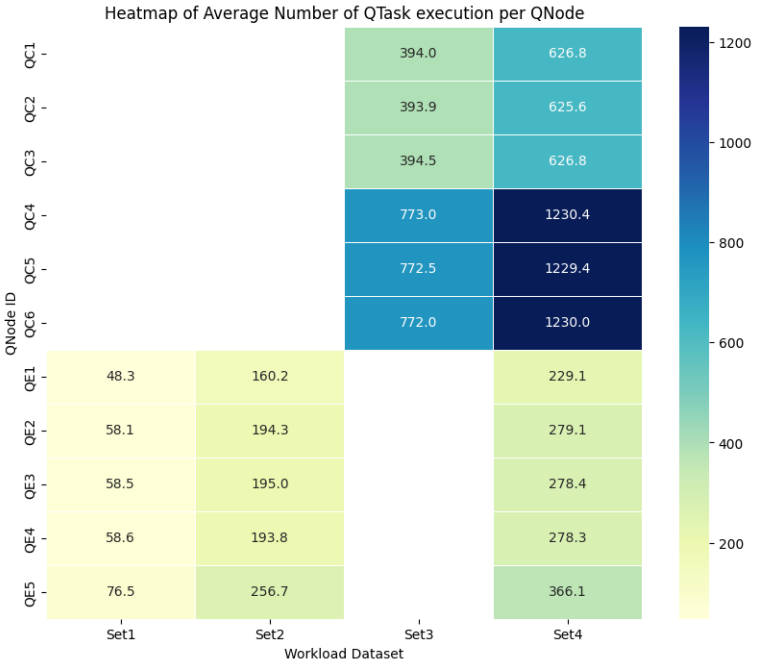


Figure 4.12: Distribution of the average number of QTasks execution per QNode in all scenarios. (QE: quantum nodes at the edge layer, QC: quantum nodes at the cloud layer)

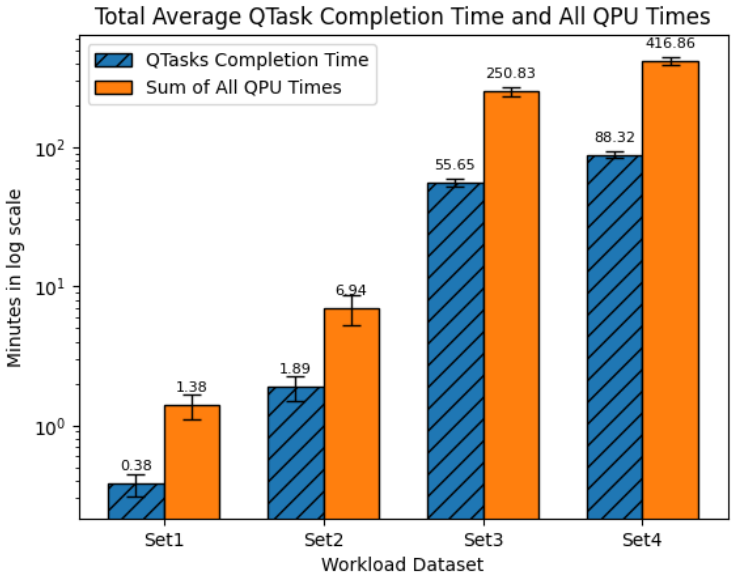


Figure 4.13: Total QTasks completion time (wall-lock time) and cumulative QPU times of all QNodes in minutes required for each workload set's completion. These figures are average values of 1000 iterations.

Figure 4.13. 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 Set 3 and Set 4. It requires around 1 hour of execution in practical environments for completing all incoming tasks where the actual QPU times are 4.2 hours (Set 3) up to 7 hours (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 toolkit for modeling, designing, and assessing resource management policies without incurring massive waiting time or expenses.

4.8 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 [195] and Cirq [196] to the customized format in iQuantum.

Currently, the vision of quantum computing at the edge layer [226, 227] is just a theoretical concept, but it may become a reality in the near future as quantum devices become more popular. Nevertheless, our work 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 simu-

lated 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 [29, 258], 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 [101, 257], and quantum error correction [260].

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 1,000 qubit quantum chips [261], 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.

4.9 Summary

In this chapter, we introduce 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 [1] 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.

Software availability

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.

Chapter 5

Time-aware DRL-based Task Orchestrator for Quantum Computing

This chapter proposes DRLQ, a novel Deep Reinforcement Learning (DRL) based technique for task placement in quantum cloud computing environments, addressing the optimization of task completion time and quantum task scheduling efficiency. It leverages the Deep Q Network (DQN) architecture, enhanced with the Rainbow DQN approach, to create a dynamic task placement strategy. This approach is one of the first in the field of quantum cloud resource management, enabling adaptive learning and decision-making for quantum cloud environments and effectively optimizing task placement based on changing conditions and resource availability. We conduct extensive experiments using the QSimPy simulation toolkit to evaluate the performance of our method, demonstrating substantial improvements in task execution efficiency and a reduction in the need to reschedule quantum tasks. Our results show that utilizing the DRLQ approach for task placement can significantly reduce total quantum task completion time by 37.81% to 72.93% and prevent task rescheduling attempts compared to other heuristic approaches.

5.1 Introduction

Quantum computing is at the forefront of technological innovation, with the potential to drive advances in fields such as cryptography [262], finance [12], machine learning [68], and complex chemical simulation [117]. It has the capability to solve numerous problems that are currently intractable by classical computers. Besides, the emergence of

This chapter is derived from:

- **Hoa T. Nguyen**, Muhammad Usman, and Rajkumar Buyya, "DRLQ: A Deep Reinforcement Learning-based Task Placement for Quantum Cloud Computing", Proceedings of the 17th IEEE International Conference on Cloud Computing (CLOUD 2024), Shenzhen, China, July 7-13, 2024

quantum cloud computing (as discussed in Chapter 2) represents a major advancement in providing access to the computational capabilities of quantum computing. This integration enables users worldwide to execute quantum algorithms on remotely accessible quantum computers, thereby overcoming the significant barriers of cost and physical access associated with quantum hardware [51]. The quantum cloud paradigm has not only extended the accessibility of quantum computing but has also presented new challenges and opportunities in optimizing the utilization of these cloud-based quantum computation resources.

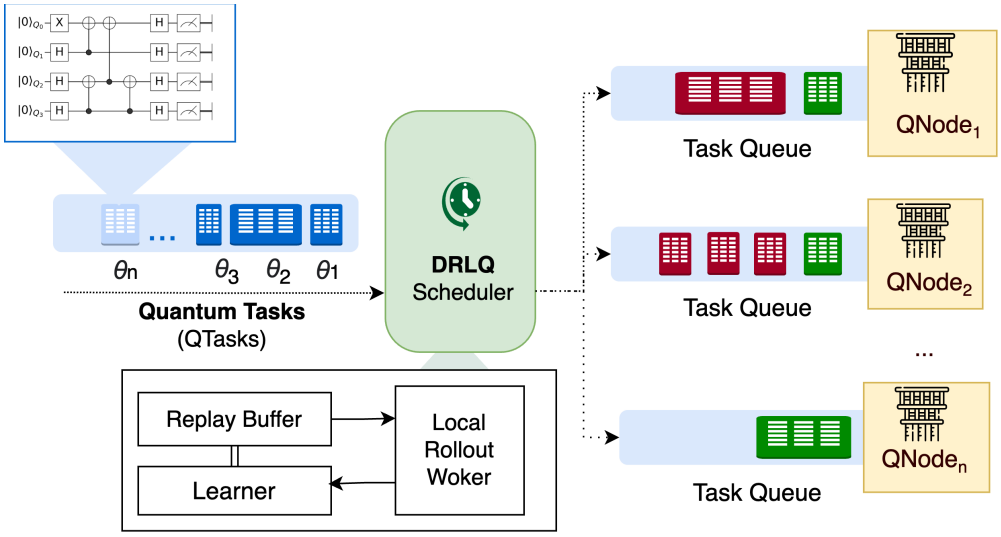


Figure 5.1: Overview of the system model for the task placement problem in quantum cloud environments

Despite its significant potential, the efficient utilization of quantum cloud computing resources faces substantial challenges, particularly in the context of quantum cloud resource orchestration. Specifically, quantum task placement, i.e., selecting an appropriate quantum backend or physical hardware and associated parameters for executing quantum tasks, is crucial for the performance and reliability of quantum computations. However, the current landscape of quantum task placement is denoted by dependence on heuristic approaches or manually crafted policies [31]. Although practical in certain contexts, these approaches do not take full advantage of the dynamic capabilities of quantum cloud computing environments. They lack the flexibility and adaptability required to optimize performance in the face of ongoing advancements in quantum

hardware and the increasing complexity of practical quantum applications. [263].

This scenario underscores the critical need for novel, adaptive techniques in resource management capable of harnessing the potential of quantum cloud computing. Integrating DRL into the quantum task placement process presents a promising approach to address these challenges. By incorporating the principles of DRL, which is well-suited for navigating complex and dynamic environments such as cloud-edge in the classical domain, with several successful examples such as [264–266], a DRL-based task placement strategy can potentially enhance quantum cloud resource management. As far as we know, this study is the first attempt to apply a deep reinforcement learning-based technique designed for the task placement problem in quantum cloud computing environments. Our approach aims to navigate the complexities of quantum systems' dynamics, balancing performance metrics and adaptively learning the optimal task placement policy through continuous interactions with the quantum computing environment. Our proposed methodology leveraging Deep Q Networks (DQN) architecture and empowered by Rainbow DQN approach [267], which combines advantages of different DQN algorithms, including Double DQN, Prioritized Replay, Multi-step learning, Distributional RL, and Noisy Nets, seeks to optimize task placement in quantum cloud computing environments.

The major contributions and novelty of our work are:

- We propose one of the first applications of DRL techniques to address the task placement problem in quantum cloud computing, leveraging the enhanced combining improvement of the Rainbow DQN technique [267] for robust and adaptive decision-making.

- Through extensive experimentation, we have shown that our DRLQ approaches can significantly reduce the total completion time by 37.81% to 72.93% and minimize the need for task rescheduling compared to other popular heuristic-based approaches.

- Our findings emphasize a possible method for tackling the problem of quantum task placement in order to optimize resource management in quantum cloud computing environments. This provides a starting point for additional thorough re-

search that considers more quantum-specific properties, such as execution accuracy, quantum circuit transpilation, and quantum error rates.

The rest of this chapter is organized as follows: Section 5.2 reviews related work, identifying gaps our research addresses. Section 5.3.1 describes the system model and problem formulation for task placement in quantum cloud computing, then explains our DRL model, focusing on the DQN architecture and Rainbow approach. Section 5.4 evaluates our method's performance through simulations and discusses the implications of our findings. Section 5.5 concludes the chapter by summarizing our contributions along with future work.

5.2 Related Work

The literature on task placement in quantum cloud computing is nascent, with only few works beginning to address the unique challenges posed by this emerging paradigm. This section briefly reviews the existing studies, highlighting the pioneering efforts in task placement within quantum cloud computing and the potential of DRL to innovate in this area. We summarise several related works in the quantum cloud domain and representative works in the classical cloud-edge domain in Table 5.1.

Table 5.1: Representative works related to our study

References	**Resource Management Problem**	**Environment**	**Approach**
[268]	Task Placement	Classical Cloud	DRL
[266]	Task Placement	Classical Edge	DRL
[31]	Task Placement	Quantum Cloud	Heuristics
[269]	Qubit allocation	Quantum Cloud	Heuristics
[157]	Resource Allocation	Quantum Network	Heuristics
Our work	Task Placement	Quantum Cloud	DRL

The traditional task placement strategies used in classical cloud computing cannot be directly applied to the quantum context due to differences in quantum computational tasks and resource characteristics. For example, the fundamental difference between the characteristics of a quantum task and a classical task is their information unit, i.e., quantum bits and classical bits [53]. Besides, current quantum computation processors can

be characterized by different benchmarking metrics, such as circuit layer operations per seconds (CLOPS) and quantum volume (QV) [32]. Meanwhile, applying deep reinforcement learning (DRL) in optimizing such tasks represents a potential approach in similar resource management problems in the classical domain [266, 268]. It offers promising yet unexplored solutions for dynamic and efficient resource management in quantum cloud environments.

A few studies have focused on heuristic approaches to designing quantum cloud resource management policies. Ravi et al. [155] analyzed quantum job characteristics of IBM Quantum cloud systems and then proposed an adaptive job scheduling approach based on a basic statistical analysis of historical data in their subsequent work [31]. Ngoenriang et al. [270] proposed a two-stage stochastic programming technique to allocate resources for distributed quantum computing. The technique aims to minimize the deployment cost and maximize quantum resource utilization while accounting for uncertainties like quantum task demands and computation power. Kaewpuang et al. [269] proposed a new method for allocating qubits in a quantum cloud that considers the uncertainties of quantum circuit requirements and expected waiting time. The method consists of two stages: reservation and on-demand. In the reservation stage, historical data is used to determine the allocation of resources, while in the on-demand stage, actual requirements are considered. Cicconetti et al. [157] proposed a resource allocation technique for distributed quantum computing, focusing on quantum network aspects. They used the Weighted Round Robin algorithm to assign network resources based on pre-calculated traffic flow weights. They developed a network provisioning simulator for evaluation, showing trade-offs between fairness and time complexity.

However, these existing works do not consider the characteristics of heterogeneous quantum computing systems and circuit-based metrics of quantum tasks when designing the scheduling algorithm. Furthermore, none of the existing works leverage machine learning-based approaches for quantum task placement problems in cloud-based environments. Our work fills this important gap and contributes to the foundational knowledge and advancement of task placement strategies in the quantum cloud computing domain.

5.3 System Model and Problem Formulation

5.3.1 System Model

Figure 5.1 represents an overview of our system model in quantum cloud computing environments, which is derived from our studies on the QSimPy toolkit[1] for quantum cloud resource management. Since quantum applications cannot be permanently installed in a quantum computer, classical cloud resources are required to host these applications. Additionally, these applications can be made available as a service (as our proposed QFaaS framework in Chapter 3). Users send task requests from their local devices to the classical cloud layer, where the quantum application is deployed. A corresponding quantum task (QTask), which comprises single or multiple quantum circuits, will be created for each incoming request. The broker (or scheduler) then makes a placement decision for each QTask based on its requirement and the current state of available quantum cloud computation resources.

Quantum Nodes: The set of available quantum computation nodes (QNodes) at a quantum data center is defined as $\mathcal{Q} = q_1, q_2, ..., q_m$, where $m = |\mathcal{Q}|$ indicates the number of available QNodes. We assume that each QNode has a single quantum processing unit (QPU), reflecting the current state of available quantum computers. Each QNode has different properties, such as qubit number (q^w), quantum volume (QV) (q^v) [29], circuit layer operation per second (CLOPS) (q^s) [32], supported gates (q^g), and qubit topology (q^t).

Quantum Tasks: We consider each quantum task (QTask) to comprise a gate-based quantum circuit. Thus, a set of incoming quantum tasks can be defined as $\Theta = \{\theta_1, \theta_2, ..., \theta_n\}$, where $n = |\Theta|$ is the number of QTasks. Each QTask θ_i has various properties, such as qubit number (θ^w), circuit depth (θ^d), used quantum gates (θ^g), number of shots (θ^s), qubit topology (θ^t), and arrival time (θ^a). The circuit depth (θ^d) of a quantum circuit is defined as depth-1 circuit layer, where the depth of the following components can be considered as 1: a) a single-qubit gate from native gate set, b) a measurement, c) a reset, d) a 2-qubit gate from native gate set [32].

[1] https://github.com/Cloudslab/qsimpy

5.3.2 Problem Formulation

The placement configuration of QTask $\theta_i \in \Theta$ can be define as $\xi_i = \{\theta_i, q_j\}$ where $q_j \in \mathcal{Q}$ and $1 \leq j \leq |\mathcal{Q}|$ is the selected quantum node index. If the placement fails due to a violation of resource constraints, such as placing a task to a QNode that does not have enough qubits, the broker needs to do the replacement (or rescheduling) to find another suitable QNode for the task execution.

Completion Time Model: The total completion time (or the makespan) represents the total waiting time for each request from the submission to the completion, including executing time and queueing time as follows:

$$t_{\theta_i} = t_{\theta_i}^{wait} + t_{\theta_i}^{exec} \tag{5.1}$$

where $t_{\theta_i}^{wait}$ is the queuing time (from the arrival time till the execution start time) and $t_{\theta_i}^{exec}$ is the quantum execution time. The execution time of a quantum task depends on the corresponding computer's required quantum circuit layer and QPU speed (CLOPS). Based on IBM Quantum study [32], we use depth-1 circuit layer operation per second (D1CPS) instead of CLOPS for q^s, which is used to measure the number of depth-1 circuit layers (or circuit depth) of a quantum circuit that can be executed per second. The execution time of a QTask θ_i in QNode q_j can be estimated as follows:

$$t_{\theta_i}^{exec} = \frac{\theta_i^d \times \theta_i^s}{q_j^s} \tag{5.2}$$

where θ_i^d is the circuit depth (or number of depth-1 circuit layers), θ_i^s is the number of shots to be executed, and q_j^s is D1CPS of the QNode.

Problem Statement: Given a data center with heterogeneous quantum computation nodes (QNodes) and a continuous incoming workload (QTasks), design the task placement policy to select the most appropriate QNode for each incoming QTask object to minimize the total response time of all QTasks and mitigate the replacement frequency due to violation of the execution constraints. The objective can be defined as:

$$\Omega(\Xi) = \min \sum_{i=1}^{n} t_{\theta_i} \tag{5.3}$$

s.t.

$$C1 : Size(\theta_i) = 1, \forall \theta_i \in \Theta \tag{5.4}$$

$$C2 : t_{\theta_j}^{start} \geq t_{\theta_h}^{start} + t_{\theta_h}, \forall \theta_h, \theta_j \in \Theta, h < j \leq |\Theta| \tag{5.5}$$

$$C3 : \theta_i^w \leq q_j^w, \forall \theta_i \in \Theta, q_j \in \mathcal{Q} \tag{5.6}$$

where $C1$ specifies that each QTask can only be assigned to one QNode at the time for the execution; $C2$ indicates that the QTask θ_j can only be executed in the selected QNode after the completion of its predecessor at the same QNode (QTask θ_h); and $C3$ requires the selected QNode q_j to have enough qubits for the execution of QTask θ_i. If the QNode does not have enough qubits, the placement will fail, and rescheduling will be necessary.

5.3.3 Deep Reinforcement Learning Model

Deep Reinforcement Learning (DRL) employs deep neural networks to tackle decision-making with high-dimensional states, formulated as Markov Decision Processes (MDP) [271]. An MDP is denoted as $(\mathbb{S}, \mathbb{A}, \mathbb{P}, \mathbb{R}, \gamma)$, with $\mathbb{S}$ and $\mathbb{A}$ representing the sets of states and actions, respectively. $\mathbb{P}$ defines the transition probabilities, $\mathbb{R}$ is the reward function, and $\gamma \in [0, 1]$ is the discount factor, indicating the preference for future rewards. During discrete time steps t, a DRL agent observes a state s_t, selects an action a_t from policy $\pi(a_t|s_t)$, and transitions to a new state s_{t+1}, receiving a reward r_t. The agent aims to maximize the expected return $\mathbb{V}^\pi(s_t) = \mathbb{E}_\pi[\sum_t \gamma^t r_t], t \in \mathbb{T}$, the sum of discounted rewards obtained by following policy π from s_t. The policy, often a neural network, is refined through training to optimize performance.

State Space $\mathbb{S}$: A state of the agent's observation from the quantum cloud computing environment, which includes 1) information on all available QNodes and 2) information on current QTasks to be placed (or scheduled).

The feature vector of m quantum nodes in $\mathcal{Q}$, each quantum node has o features, at time step t can be presented as:

$$\mathcal{F}_t^{\mathcal{Q}} = \{f^{q_i^z} | \forall q_i \in \mathcal{Q}, 1 \leq i \leq m, 1 \leq z \leq o\} \tag{5.7}$$

where i is the index of a quantum node, and z is the index of a quantum node's feature.

The feature vector of the current quantum task θ_j in Θ, with p features at time step t can be presented as:

$$\mathcal{F}_t^{\theta_j} = \{f^{\theta_j^k} | \theta_j \in \Theta, 1 \leq k \leq p\} \tag{5.8}$$

where k is the index of the current quantum task's feature.

All features of quantum nodes and quantum tasks are described in Section 5.3.1. Each QTask specification is only sent to the DRLQ agent as part of the state after its arrival. Therefore, the State space of the system can be defined as:

$$\mathbb{S} = \{s_t | s_t = (\mathcal{F}_t^{\mathcal{Q}}, \mathcal{F}_t^{\theta_j}), \forall t \in \mathbb{T}\} \tag{5.9}$$

Action Space $\mathbb{A}$: An action can be defined as the placement of a QTask on an available quantum node. The action a_t at time step t is an placement of QTask θ_j to quantum node q_i can be defined as

$$a_t = \xi_i = \{q_i, \theta_j\}, q_i \in \mathcal{Q}, \theta_j \in \Theta \tag{5.10}$$

Thus, the Action space is equivalent to the set of all available quantum nodes at the data center:

$$\mathbb{A} = \mathcal{Q} \tag{5.11}$$

Reward Function $\mathbb{R}$: The main goal is to minimize the total completion of all incoming tasks. Besides, we also aim to mitigate task replacement attempts (or maximise the success rate of the task placement). To achieve these objectives, we define the reward r_t at time step t as follows:

$$r_t = \begin{cases} \dfrac{1}{t_{\theta_i}} \times (1 - \alpha\kappa) & \text{if done} = 1 \\ \Delta \times (1 + \alpha\kappa) & \text{if done} = 0 \end{cases} \tag{5.12}$$

where t_{θ_i} is the total completion time of QTask θ_i, α is the penalty factor, and κ is the replacement count. If the QTask is successful ($done = 1$), the inverse value of its total completion time is assigned for the reward to encourage the policy to find a better

placement that has a shorter total completion time to get a higher reward r_t. Otherwise, if the task execution fails for any reason, we apply a large negative value Δ for penalty and advise the policy to avoid similar action in the future. Besides, we also consider κ - the number of replacements (or rescheduling attempts) of QTask θ_i and define a penalty factor α when assigning the reward in order to mitigate the replacement. The penalty factor acts as an additional discount factor when a QTask needs more than one placement to be successful and also magnifies the penalty if that QTask fails multiple times. Thus, our reward function can be used to achieve the main objective of minimizing the total completion time and reducing the number of task replacements.

5.3.4 DRLQ Framework

Our DRLQ framework employs an enhanced deep reinforcement learning technique, combining Deep Q-Networks (DQN) and the Rainbow approach [267] to optimize task placement in quantum cloud computing environments.

The Deep Q-Network (DQN) algorithm, introduced by Mnih et al. [271, 272], represents a significant advancement in reinforcement learning, utilizing deep neural networks to approximate the action-value function $Q(s, a; \theta)$. The objective is to minimize the loss function:

$$L(\theta) = \mathbb{E}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta)\right)^2\right] \tag{5.13}$$

where $y_i = r + \gamma \max_{a'} Q(s', a'; \theta^-)$ defines the target for a state-action pair (s, a), with r as the immediate reward, γ as the discount factor, s' as the subsequent state, and θ^- denotes the parameters of a target network that is periodically updated to stabilize the learning process. The overall process of the DRLQ framework can be represented in the Algorithm 4.

First, we initialize the quantum cloud environment following the reinforcement learning setting. Due to the limitation of managing the practical quantum cloud environment setup, we utilize a simulated quantum cloud environment by utilizing the QSimPy simulator and a quantum application dataset, such as MQTBench [1], for generating the synthetic QTask data for the environment. Then, we register the environment with Ray

Algorithm 4: DRLQ Framework for QTask Placement

```
Initialize the quantum cloud environment in QSimPy;
Loading dataset for the QTask generator module;
Register the environment with Ray;
Initialize replay buffer 𝒟 to capacity N and priority replay configuration;
Initialize action-value policy Q with random weights;
for each hyperparameter configuration do
    Set hyperparameters using Ray Tune;
    for episode = 1, M do
        Perform DQN process, defined in [272];
        Adjust rewards and transitions for n-step;
        Add parameter noise to network weights for exploration;
        Update Q using a distributional approach;
    end for
end for
Select the best configuration from Ray Tune results;
```

[273], a comprehensive machine-learning framework for managing the training and tuning. We utilize a replay buffer to enhance learning efficiency by decoupling consecutive training samples, thereby providing a diverse set of experiences for more robust neural network training. The replay buffer configuration and other training hyperparameters need to be defined for the hyperparameter tuning process. We leverage the Rainbow DQN approaches [267] to combine all the advantages of the different DQN approaches, such as Multi-step Learning [274], Distributional RL [275], Prioritized Replay [276], and Noisy Nets [277]. These enhancements of DQN can collectively improve the training efficiency and effectiveness in quantum task placement, offering a potential approach to learning in the complex and dynamic environments of quantum cloud computing. Finally, we determine the best hyperparameter configuration based on the tuning process using Ray Tune [278].

5.4 Performance Evaluation

5.4.1 Environment Setup

To evaluate the performance of our DRLQ technique, we set up a simulated environment that reflects the actual quantum cloud environment due to current limitations on accessing and managing a quantum data center. We use QSimPy, a learning-centric framework derived from the iQuantum toolkit (see Chapter 4), to create the quantum cloud environment. All experiments are conducted on a computation instance with 16 vCPUs and 64GB of RAM at the Melbourne Research Cloud.

We model a quantum data center with 10 heterogeneous quantum nodes, ranging from 16 qubits to 127 qubits, using quantum benchmarking metrics [32] from IBM Quantum and backend instances in Qiskit. The modeled quantum nodes included *ibm_sherbrooke*, *ibm_washington*, *ibm_brisbane*, *ibm_osaka*, *ibm_nazca*, *ibm_kyoto*, *ibm_cusco*, *ibm_kolkata*, *ibm_hanoi*, *ibm_guadalupe*. We used Qiskit [279] for the transpilation of circuits in incoming QTasks to the selected QNode and extracted the circuit metrics after transpilation to mimic the process when a quantum circuit reaches the quantum node for further execution. To simulate stochastic incoming quantum tasks with metrics from actual quantum applications, we selected 12 quantum applications from the MQTBench dataset [1], which contains quantum circuits in QASM files ranging from 2 to 50 qubits each. The selected quantum applications from the MQTBench dataset include:

1. Amplitude Estimation (AE)
2. Deutsch-Jozsa algorithm
3. Greenberger–Horne–Zeilinger (GHZ) state
4. Quantum Fourier Transformation
5. Entangled Quantum Fourier Transformation
6. Quantum Neural Network (QNN)
7. Quantum Phase Estimation (QPE) exact

8. Quantum Phase Estimation (QPE) inexact
9. Random circuits
10. Real Amplitudes ansatz with Random Parameters
11. Efficient SU2 ansatz with Random Parameters
12. Two Local ansatz with random parameters

The number of shots is set to 1024 by default. We randomly select QTasks for each episode in the reinforcement learning process from the synthetic QTask dataset. The QTask arrival times were generated following a Poisson distribution.

After setting up the Gymnasium-based environment for the quantum cloud, we used Ray RLlib [280], an industry-grade reinforcement learning framework, to implement the proposed method of DRLQ and several baseline algorithms for performance comparison. We evaluated the performance of DRLQ against other popular heuristic approaches, including:

- *Greedy*: QTasks are greedily assigned to the QNode with the shortest waiting time, similar to an approach in our QFaaS framework (see Chapter 3). If these tasks fail, they are assigned to the most powerful QNode (i.e., the one with the largest number of qubits).
- *Round Robin*: QTasks are assigned to QNodes in a cyclic order, ensuring a balanced distribution of tasks across all available QNodes.
- *Random*: QTasks are randomly assigned to QNodes.

5.4.2 Evaluation and Discussion

We used a similar evaluation approach following other DRL-based task scheduling works in classical computing, such as [266, 281], to evaluate the performance of our framework using reward values after 100 training iterations, which involves 100,000 time steps. We conducted extensive experiments and used Ray Tune [278] to optimize hyperparameters through the grid search method. The results of the best configuration of DRLQ are shown in Figure 5.2.

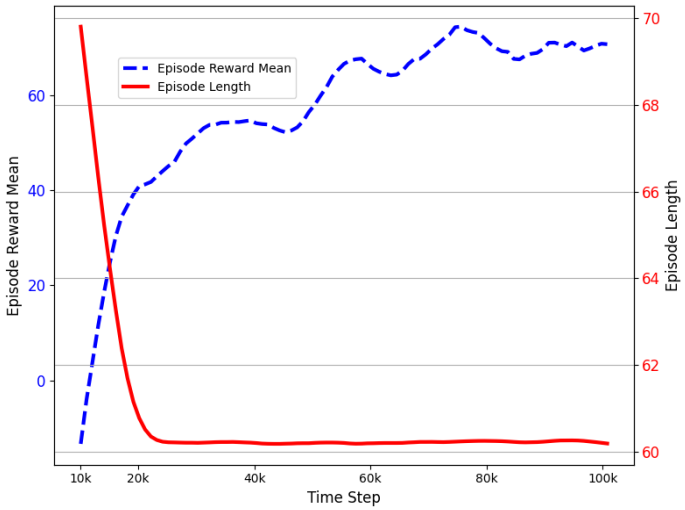
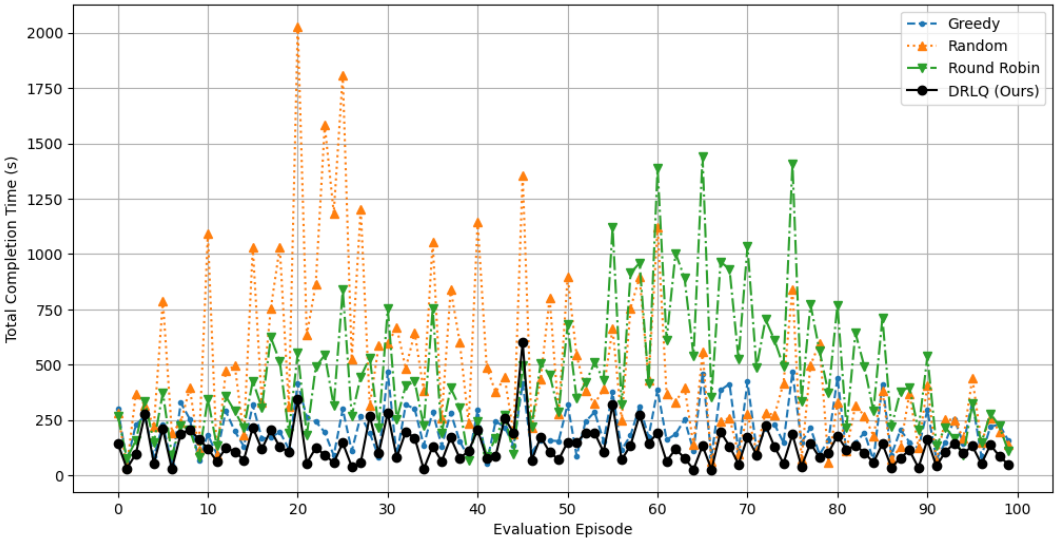


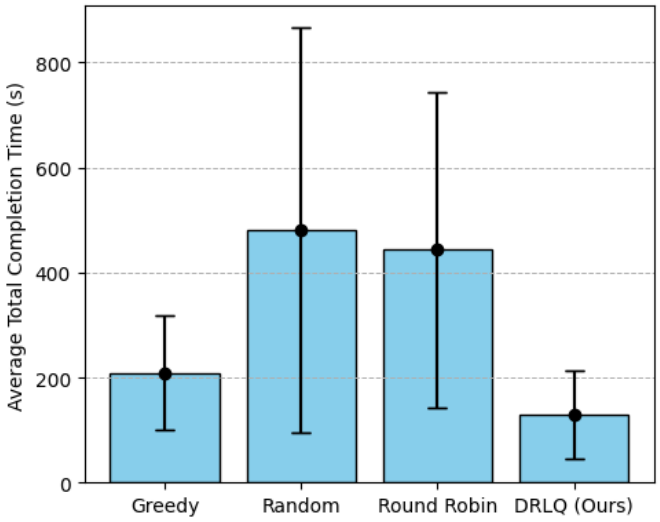
Figure 5.2: Episode reward means and episode lengths during the training of the DRLQ policy over 100,000 time steps, total training time is 8.34 hours. Each episode consists of 60 random QTasks that arrive randomly within a 1-minute time window. The tuned hyperparameters are set as follows: learning rate (lr) = 0.01, number of atoms = 10, train batch size = 180, n_step = 3, v_min = -10, v_max = 10, penalty (Δ) = -10, and penalty factor (α) = 0.1.

Our main objective is to minimize the total completion time and the number of task rescheduling (or replacement) attempts. Figure 5.2 clearly demonstrates the efficient learning process of the DRLQ method. As the reward is inversely proportional to the total completion time, the upward trend in the reward indicates a reduction in total completion time during the training episodes. The reward continuously increases after the first 20 training iterations, reaching convergence after 90 training iterations (each iteration consists of 1,000 time steps). Simultaneously, the episode length significantly reduces and converges around 60, which is the minimum length of an episode, after the first 25 training iterations. We then exported the trained policy after 100 training iterations for evaluation on different QTask workload datasets to compare the effectiveness of DRLQ against other heuristic approaches.

Figure 5.3a compares DRLQ with other baseline techniques for QTask placements, considering the total completion time of all QTasks over 100 episodes, each consisting of 60 random incoming QTasks different from the training set of DRLQ. The average total completion times of all QTasks in each episode after 100 evaluation episodes across



(a) Total completion time of all QTasks in each episode during the evaluation.



(b) Average values of total completion time of all QTasks over 100 evaluation episodes.

Figure 5.3: Total completion times of all QTask over 100 evaluation episodes among DRLQ and other heuristic approaches.

DRLQ and other baselines are shown in Figure 5.3b. Our evaluation results indicate that the DRLQ algorithm significantly improves efficiency by minimizing the total completion time. It achieves a 37.81% reduction in the total completion time compared to the Greedy algorithm, a 72.93% reduction compared to the Random approach, and a 70.71% reduction compared to the Round Robin algorithm over 100 evaluation episodes.

We also evaluate the performance of DRLQ by considering the number of task rescheduling attempts per episode. Figure 5.4 shows the average number of task rescheduling attempts over 100 evaluation episodes for all approaches.

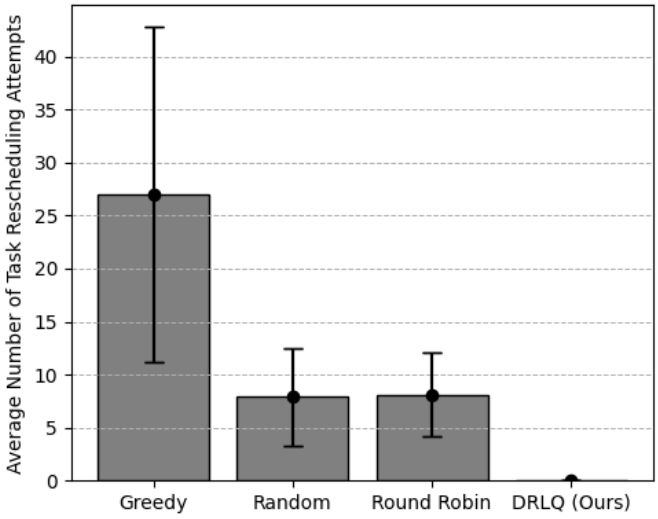


Figure 5.4: Average number of task rescheduling attempts after 100 evaluation episodes of DRLQ and other approaches

The results show that our DRLQ approach significantly outperforms other methods in mitigating task rescheduling attempts. Specifically, DRLQ achieves zero rescheduling attempts, which is a substantial improvement over the Greedy, Random, and Round Robin approaches, with average rescheduling attempts of 26.95, 7.88, and 8.11, respectively. These improvements are especially important in quantum cloud computing environments, where minimizing task completion time is crucial due to the high costs of quantum resources and the inherent variability of quantum computations.

5.5 Summary

In this chapter, we have explored the effectiveness and potential of utilizing a deep reinforcement learning approach in proposing a novel DRLQ framework for task placement in quantum cloud computing environments. Our results showcase the significant improvement of the DRLQ technique in quantum task placement compared to other heuristic approaches. It highlights the potential of using a DRL-based approach for robust quantum cloud resource management. Our work is one of the first studies on the resource management problem of quantum cloud computing, which requires more attention from the research community in the quantum cloud domain.

Based on the foundation and insights gained in this chapter, we further refine our deep reinforcement learning methodology and introduce a more comprehensive approach for estimating quantum execution fidelity and execution time. These enhancements are designed to better reflect the characteristics of quantum computing environments in the NISQ era, and will be discussed in detail in Chapter 6.

Chapter 6

Fidelity-aware DRL-based Task Orchestrator for Quantum Computing

*This chapter proposes QFOR, a **Q**uantum **F**idelity-aware **O**rchestration of tasks across heterogeneous quantum nodes in cloud-based environments using Deep **R**einforcement learning. We model the quantum task orchestration as a Markov Decision Process and employ the Proximal Policy Optimization algorithm to learn adaptive scheduling policies, using IBM quantum processor calibration data for noise-aware performance estimation. Our configurable framework balances overall quantum task execution fidelity and time, enabling adaptation to different operational priorities. Extensive evaluation demonstrates that QFOR is adaptive and achieves significant performance with 29.5-84% improvements in relative fidelity performance over heuristic baselines. Furthermore, it maintains comparable quantum execution times, contributing to cost-efficient use of quantum computation resources.*

6.1 Introduction

The rapid advancement of quantum computing promises to solve computationally intractable problems across critical technology domains, including cryptography [282], drug discovery [41], optimization [211], and machine learning [6]. However, given the significant challenges associated with operating physical quantum hardware, such as stringent environmental requirements and high costs, Quantum Cloud Computing

This chapter is derived from:

- **Hoa T. Nguyen**, Muhammad Usman, and Rajkumar Buyya, "QFOR: A Fidelity-aware Orchestrator for Quantum Computing Environments using Deep Reinforcement Learning", *submitted to the ACM Transactions on Quantum Computing (TQC), August 2025* (Under Review)

(QCC) has emerged as an access paradigm (see Chapter 2). QCC platforms, offered by major providers like IBM Quantum, Amazon Braket, Google Cloud, and Microsoft Azure, democratize access to Quantum Processing Units (QPUs) in a Quantum-as-a-Service (QaaS) model. This allows users to execute quantum applications remotely without the need for on-premise hardware.

Notably, current QPUs are not fully fault-tolerant processors and operate within the constraints of the Noisy Intermediate-Scale Quantum (NISQ) era [15], and executing quantum applications on them is a hybrid process, involving both classical and quantum execution. For instance, Variational Quantum Algorithms (VQAs) necessitate a classical optimisation step for convergence [223]. Even fully quantum algorithms require classical transpilation to be compatible with the qubit topology and native gate set of the targeted QPU. Furthermore, realizing quantum advantage requires seamless integration with classical high-performance computing (HPC) infrastructure to form hybrid quantum-classical systems [283, 284]. These hybrid systems leverage classical HPC resources for pre-processing, optimization, and post-processing while utilizing quantum processing units (QPUs) for quantum-specific computations. Figure 6.1 presents a high-level architecture of a quantum-classical hybrid cloud system, wherein user requests are routed via an API gateway to a corresponding service or application deployment at the middleware layer. Classical tasks for pre-processing and post-processing are queued and dispatched to CPUs and GPUs, while quantum tasks are orchestrated by a quantum task orchestrator component (for example, our proposed QFOR orchestrator in this work), which manages quantum processing units (QPUs) through dedicated queues and APIs.

In these hybrid quantum cloud computing environments, effective quantum task orchestration is crucial for three key reasons. First, quantum computation resources exhibit extreme scarcity as quantum hardware development is still in its early stages. Second, quantum resources incur costs significantly higher than equivalent classical compute time, making efficient utilization economically critical. For example, each minute of quantum execution on IBM Quantum hardware costs $96 USD[1], while each hour of IonQ

[1]https://www.ibm.com/quantum/pricing

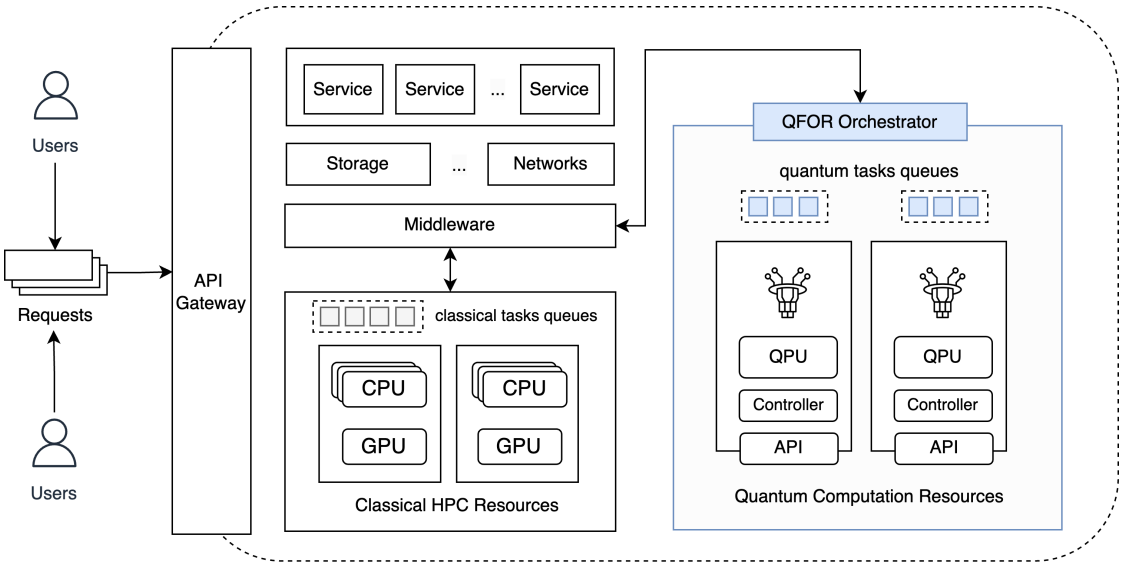


Figure 6.1: High-level view of a Quantum-HPC system in the cloud-based environment. The QFOR Orchestrator (proposed in this work) manages quantum task scheduling and coordination across quantum processing units (QPUs).

device reservation on Amazon Braket costs $7,000 USD[2] (as of June 2025). Third, quantum fidelity degradation due to poor scheduling can invalidate entire hybrid computations, regardless of classical processing quality [25, 285]. Current quantum hardware exhibits significant variability across multiple dimensions that directly impact quantum task execution. Multiple QPU technologies exist, including superconducting qubits, trapped ions, and neutral atoms, each with distinct performance and noise characteristics. Even within the same technology, different QPU models possess varied qubit layouts, native gate sets, and qubit connectivity. Architecturally, devices differ in qubit connectivity constraints how quantum circuits can be efficiently mapped and executed. Additionally, gate durations and error rates vary not only between devices but also over time due to calibration cycles and environmental factors [101, 286]. Device availability is another critical consideration, as queue times can fluctuate based on user demand and system maintenance [31]. These factors collectively introduce substantial heterogeneity and uncertainty into the quantum cloud environment. As a result, effective orchestration must be both fidelity-aware, dynamically select quantum computation nodes that

[2]https://aws.amazon.com/braket/pricing/

optimize for execution reliability while maintaining efficiency with reasonable execution time. This orchestration approach is essential for harnessing the potential of quantum cloud resources and ensuring consistent, high-quality results and cost efficiency for users.

Despite the increasing interest in QCC, existing research in quantum cloud resource management exhibits several critical gaps, particularly concerning heterogeneous quantum computation resources and noise-aware execution fidelity. Traditional resource management approaches fall into two categories, both with critical limitations. Traditional HPC scheduling algorithms excel at managing classical computation resources but can struggle to account for quantum-specific constraints such as fidelity optimization, decoherence, and calibration-dependent performance [284, 287]. Conversely, quantum-specific heuristics [31, 288] are inherently limited in their ability to adapt to the dynamic and uncertain nature of quantum cloud environments. Recent AI-driven approaches, such as deep reinforcement learning (DRL), have shown promise for addressing this problem with several successful cases in the classical cloud-edge [266, 281] and high-performance computing domain [265, 289], as it is well-suited for sequential decision-making in environments with incomplete information and stochastic dynamics. However, DRL approaches for quantum cloud orchestration remain limited in scope. Existing works mainly focused on completion time and device allocation [160] or overall fidelity of the targeted quantum device [290]. To our knowledge, no existing work considers the trade-off between circuit execution fidelity, time, and complexity of quantum tasks using DRL-based approaches with real quantum circuit workload.

Furthermore, as quantum cloud resources are scarce and limited, designing and evaluating resource management in a practical environment is extremely challenging. Therefore, simulation frameworks for modeling and simulating quantum cloud computing environments are essential for this research area. However, existing approaches mainly focus on high-level metrics for the performance estimation of quantum tasks, such as quantum volume [29] and circuit layer operation per second (CLOPS) [32]. Although this approach is promising, it lacks the adaptability to different structures of quantum circuits and their transpilation to different qubit topologies of quantum hardware. Indeed, specific information on gate errors and durations of individual qubits in quantum

hardware can be used to enhance the estimation process. Thus, there is a clear need for resource estimation that systematically explores this fidelity-runtime tradeoff to aid optimal quantum resource orchestration.

To address the above-mentioned challenges, we propose QFOR, a **Q**uantum **F**idelity-aware **O**rchestrator using a Deep **R**einforcement learning-based approach that optimizes the overall performance of task orchestration in quantum cloud environments. QFOR models quantum task placement as a Markov Decision Process and employs Proximal Policy Optimization (PPO) [291] to learn adaptive policies that balance overall execution fidelity and time of quantum tasks.

The major contributions and novelty of our work are:

- We propose a novel deep reinforcement learning-based task orchestration framework for quantum computing in cloud-based environments. Our method considers the critical trade-off between execution fidelity and quantum execution time, with a primary focus on maximizing fidelity-aware overall performance.
- Our approach employs a systematic quantum task execution and fidelity estimators, and mimics the execution of noisy devices based on calibration data and actual quantum circuit properties, enabling more rigorous quantum cloud modeling and simulation of quantum hardware behavior and improved orchestration decisions.
- We provide configurable orchestration objectives that balance execution fidelity and latency, accounting for circuit complexity and task priority. Extensive evaluation demonstrates that QFOR is flexible and adaptive, achieving 29.5–84% improvements in relative fidelity compared to heuristic baselines, while maintaining comparable quantum execution times, thus supporting cost-efficient quantum resource usage.

The rest of the chapter is organized as follows: Section 6.2 reviews existing works related to our study. Section 6.3 describes the system model and formulates the problems of quantum task orchestration in heterogeneous quantum cloud computing environments. We provide details of the methodology and design of the QFOR framework in Section 6.4. Then, Section 6.5 describes the evaluation study of our proposed framework,

followed by and further discussion on the results. Finally, we conclude the chapter with key insights, limitations, and future directions in Section 6.6.

6.2 Related Work

In this section, we review existing literature on quantum resource management and orchestration, categorizing approaches by their methodology and highlighting critical gaps that our work addresses. Table 6.1 provides a comprehensive comparison across key technical dimensions.

Table 6.1: Overall comparison of related works on quantum cloud task orchestration and resource management problem.

Works	**Quantum Tasks**		**Quantum Nodes**		**Method**	**Optimization config**			**Env**
	Real Circuit	*Dataset/Task Generator*	*Noise Aware*	*Qubit*		*Fidelity*	*Time*	*Weighted*	
QFaaS (Chapter 3)	✓	Qiskit	✓	7-127	H	◐	◐	✗	E
Ravi et al. [31]	✓	Qiskit	✓	1-65	H	✓	◐	✗	S
iQuantum (Chapter 4)	✓	MQTBench	✗	27-127	H	✗	✓	✗	S
Qonductor [25]	✓	MQTBench	✓	27	H	✓	◐	✗	E
QuSplit [288]	✓	Qiskit (VQE)	✓	N/A	H	✓	✗	✗	S
DRLQ (Chapter 5)	✓	MQTBench	✗	27-127	DRL	✗	✓	✗	S
Moirai [160]	✓	Qiskit	✗	5-7	DRL	✗	✓	✗	E
Luo et al. [290]	✗	Random data	✓	127	DRL	✓	◐	✗	S
QFOR	✓	**MQTBench**	✓	**27-127**	**DRL**	✓	✓	✓	**E**

Notes. H: Heuristic, DRL: Deep Reinforcement Learning, N/A: Not available, ◐: Partially Addressed/Described, Env: Environments, E: Emulation with real quantum circuit compilation and execution, S: Simulation with circuit feature or synthetic data

Early work in quantum cloud orchestration have primarily relied on heuristic approaches that extend classical scheduling paradigms to quantum environments. Ravi et al. [31] introduced an adaptive job and resource management for quantum clouds, focusing on fidelity optimization through device selection. However, their approach relies on predetermined heuristics and statistical analysis that can struggle to adapt to temporal variations in quantum device performance. QFaaS (proposed in Chapter 3) is one of the first serverless quantum computing framework, establishing a Function-as-a-Service model for quantum task execution. While QFaaS demonstrates practical hybrid quantum-classical integration across multiple cloud providers, it employs primarily heuristic models for resource allocation decisions based on execution priority (speed and accuracy) using Quantum Volume (QV) [29], CLOPS [32], and queue metrics with-

out considering dynamic fidelity-runtime tradeoffs. Pioneering the discrete-event quantum cloud modeling and simulation, iQuantum (see Chapter 4) provides comprehensive toolkits for quantum cloud resource management design and evaluation, but has not fully considered noise-aware modeling. This gap is also one of the key motivations for QFOR to address, providing an emulation approach to mimic the noisy-quantum hardware in practical quantum cloud environments. Qonductor [25] represents one of the most sophisticated heuristic frameworks, offering a hybrid quantum-classical orchestration. Despite introducing important concepts like hybrid resource estimation, Qonductor relies on heuristic scheduling policies and a prediction model based on historical data that can be challenging to adapt to dynamic quantum environments. Focusing only on quantum optimization applications, QuSplit [288] focuses on optimizing fidelity and throughput through job splitting using a genetic algorithm. However, the limitation of heuristic approaches lies in their ability to adapt to the dynamic and stochastic nature of quantum cloud environments. As quantum hardware and the characteristics of quantum tasks evolve, static scheduling policies become harder to adapt to, requiring adaptive learning-based approaches.

Recent research has begun exploring deep reinforcement learning as a solution to quantum orchestration challenges, demonstrating significant improvements over heuristic baselines while revealing important limitations. Our DRLQ framework (see Chapter 5) pioneered the use of deep reinforcement learning [267] for quantum task scheduling, demonstrating significant improvements over heuristic baselines in terms of completion time. Similarly, Moirai [160] employed policy gradient methods within the OpenWhisk framework to schedule quantum circuits on small-scale quantum devices. However, neither DRLQ nor Moirai incorporated comprehensive noise-awareness in the decision-making process. Recently, Luo et al. [290] employed a DRL approach in a simulated environment to maximize targeted device fidelity on different 127-qubit quantum nodes. However, their work relied on CLOPS-based estimation for quantum execution with synthetic random data for the training and evaluation, rather than real quantum circuits, and did not consider the execution time factor in the DRL policy design to fully address the tradeoff between time and fidelity of task execution.

As summarized in Table 6.1, our work addresses several limitations in the exist-

ing approaches. Existing approaches lack comprehensive performance modeling that integrates circuit features, device calibration data, and noise-aware fidelity and execution time estimation for improving scheduling decisions. Besides, current DRL-based methods operate on limited-scale systems or synthetic random data in simulated environments rather than emulating the execution with quantum circuit compilation and execution, which have limitations in practical applicability to quantum cloud environments. Our work provides a comprehensive, deep reinforcement learning-based framework that considers noise-aware performance modeling using device calibration data and configurable optimization with evaluation on realistic quantum circuit workloads from a well-known quantum circuit dataset (MQTBench [1]). Our work aims to contribute valuable approaches and insights for future works in quantum cloud orchestration, enabling learning-driven resource management that adapts to the dynamic, heterogeneous nature of NISQ-era quantum computing environments while optimizing for practical deployment requirements.

6.3 System Model and Problem Formulation

This section presents our system model and problem formulation for quantum cloud orchestration, focusing on fidelity-aware performance metrics.

6.3.1 Quantum Task Model

In quantum cloud environments, a quantum task (QTask) represents the unit of quantum computation requiring adaptive resource orchestration to optimise the performance. A QTask can comprise one or multiple quantum circuits that need to be executed with specific qubit requirements, gate operations, and circuit depth. In the context of this work, we consider each QTask as a single independent quantum circuit. These QTasks can encompass different quantum algorithms, each with distinct characteristics.

Let $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_N\}$ denote a sequence of quantum tasks arriving for execution.

Each task $\tau_i \in \Gamma$ is characterized by the following properties:

$$\tau_i = \left(a_i, q_i, d_i, s_i, g_i^{(1)}, g_i^{(2)}, \mathcal{C}_i\right) \tag{6.1}$$

where:

- $a_i \in \mathbb{R}_{\geq 0}$: arrival time of the task into the system.
- $q_i \in \mathbb{N}$: number of qubits required for execution.
- $d_i \in \mathbb{N}$: circuit depth, i.e., the longest gate dependency path.
- $s_i \in \mathbb{N}$: number of shots (i.e., repetition of the execution)
- $g_i^{(1)} \in \mathbb{N}$: total number of single-qubit gates.
- $g_i^{(2)} \in \mathbb{N}$: total number of two-qubit gates.
- $\mathcal{C}_i$: the quantum circuit representation as a Directed Acyclic Graph (DAG).

Each circuit $\mathcal{C}_i$ is modeled as a DAG $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$, where $\mathcal{V}_i$ is the set of all quantum operations (gates), and $\mathcal{E}_i \subseteq \mathcal{V}_i \times \mathcal{V}_i$ represents directed edges indicating gate dependencies.

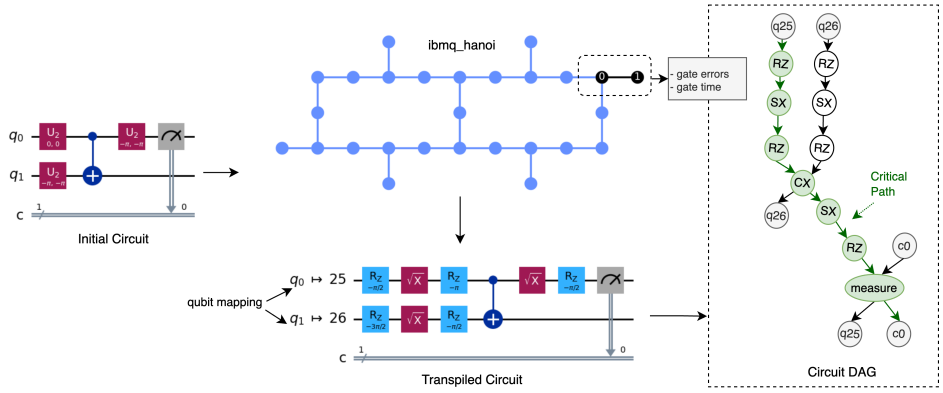


Figure 6.2: Example of an initial quantum circuit and its transpilation, qubit mapping to a 27-qubit QNode (ibmq_hanoi), and DAG of the transpiled circuit with the critical path illustration.

Example of a quantum circuit within a QTask and its transpilation, mapping and DAG representation are illustrated in Figure 6.2. The DAG representation enables crit-

ical path analysis for the estimation of QTask execution time and the execution fidelity based on the calibration data of QNodes. Based on the current state of typical quantum cloud environments as previously discussed in Chapter 2, we assume each task requires exclusive access to a set of qubits on a quantum processor and no preemption or interruption occurs during task execution. The orchestration focuses exclusively on quantum circuit execution, which dominates in terms of resource sensitivity compared to classical resources.

6.3.2 Quantum Computation Resource Model

The quantum cloud infrastructure consists of heterogeneous quantum nodes (QNodes) with one or multiple quantum processing units (QPUs) with distinct performance characteristics that directly impact scheduling decisions. In the context of this work, we consider a single QPU per QNode to reflect the current state of available quantum hardware. Each QNode exhibits unique hardware specifications and properties, which must be taken into account during scheduling to achieve reliable and efficient quantum task execution.

Let the set of quantum resources be denoted by: $\mathcal{N} = \{n_1, n_2, \ldots, n_M\}$, where each node $n_j \in \mathcal{N}$ is defined as:

$$n_j = \left(q_j, \mathcal{G}_j, \mathcal{D}_j, \mathcal{E}_j, \rho_j(t)\right) \tag{6.2}$$

where:

- $q_j = |\mathcal{Q}_j| \in \mathbb{N}$: number of physical qubits and $\mathcal{Q}_j$ is the set of all available qubits at QNode n_j.
- $\mathcal{G}_j = (\mathcal{V}_j, \mathcal{E}_j)$ is the qubit connectivity graph, where V_j is the set of all qubits and E_j is the set of all edges connecting these qubits.
- $\mathcal{D}_j : \mathcal{O}_j \times \mathcal{Q}_j \to \mathbb{R}_{>0}$ maps available gates $\mathcal{O}_j$ and qubits $\mathcal{Q}_j$ to corresponding gate execution durations
- $\mathcal{E}_j : \mathcal{O}_j \times \mathcal{Q}_j \to [0, 1]$ maps all available gates $\mathcal{O}_j$ and qubits $\mathcal{Q}_j$ to corresponding error probabilities.

- $\rho_j(t)$: other dynamic state of the QNode at time t, such as next available time and queuing information.

This abstraction allows us to capture both static and time-dependent dynamics of each quantum node. These features are critical for orchestrator decisions in dynamic workload settings and heterogeneous quantum cloud environments.

6.3.3 Fidelity-aware Orchestration Performance Model

For quantum cloud orchestration in the NISQ era, there are two critical metrics that indicate the performance of the orchestration decision: execution fidelity and time. First, execution fidelity can be considered as one of the most critical factors. Quantum execution exhibits high sensitivity to noise, where small fidelity degradations can render results meaningless regardless of execution speed [32]. Poor device selection as well as poor selection of qubits on the targeted device can reduce algorithmic success probability, necessitating multiple re-executions that far exceed any time savings from faster scheduling. Second, quantum task execution time in the quantum node is also critical, as quantum resources are scarce and extremely expensive. Besides, quantum states decay exponentially with time, making execution time a fundamental physical constraint rather than merely an optimization preference [292]. Longer execution sequences suffer increased error accumulation, creating a direct coupling between execution time and fidelity. Therefore, we define the orchestration performance metric that mainly focuses on the fidelity of the execution, while maintaining the execution time and considering the complexity of the task that needs to be executed, as well as how good it compares to other available decisions.

Fidelity Performance Score

The execution fidelity performance score $\mathcal{F}_{i,j}$ is the key objective in this work and captures the comprehensive quality of executing task τ_i on node n_j through three complementary components:

For task τ_i assigned to node n_j, the base execution fidelity $F_{i,j}$ is approximately esti-

mated as:

$$F_{i,j} = \prod_{g \in \mathcal{C}_i'} \left(1 - \mathcal{E}_j(g, \vec{q}_g)\right) \tag{6.3}$$

where $\mathcal{C}_i'$ represents the transpiled circuit and $\vec{q}_g$ are the assigned physical qubits.

The base fidelity score R_{rf} normalizes this against expected fidelity performance $F_{i,j}^{'}$ based on the average gate errors of all available quantum nodes, preventing hardware-specific biases while rewarding above-expected performance.

$$R_{rf} = \frac{F_{i,j}}{F_{i,j}^{'}} \tag{6.4}$$

In quantum cloud environments, simpler circuits (i.e., those with shallow depth and fewer gates) tend to achieve higher fidelity due to reduced exposure to noise. As a result, an orchestration policy that solely maximizes fidelity may develop a bias toward such circuits, systematically deprioritizing more complex tasks. To address this issue, we introduce a small complexity bonus R_{cb}, which encourages the orchestrator to also consider more complex circuits:

$$R_{cb} = w_d \cdot \frac{d_i}{D_{max}} + w_g \cdot \sum \frac{g_i}{G_{max}} \tag{6.5}$$

where d_i is circuit depth, $\sum g_i$ is total gate count in the circuit of the scheduled QTask, w_d and w_g are adjustable weights, and D_{max}, G_{max} are normalization bounds for the maximum depth and gates of the circuit. This ensures fairness and task diversity within the orchestration policy while remaining computationally efficient and easy to integrate into learning-based decision frameworks. We also introduce a ranking bonus R_{rb} that captures the relative quality of selecting a specific quantum node compared to all available options for a given task to ensure that the orchestration policy not only aims for high absolute fidelity but also makes competitively optimal decisions based on the current system state. It is defined as:

$$R_{rb} = \frac{F_{i,j} - F_{worst}}{F_{best} - F_{worst}} \tag{6.6}$$

where F_{best} and F_{worst} represent the highest and lowest achievable fidelity across all nodes for task τ_i.

The comprehensive execution fidelity performance score (or relative fidelity) combines these components:

$$\mathcal{F}_{i,j} = \alpha_1 \cdot R_{rf} + \alpha_2 \cdot R_{cb} + \alpha_3 \cdot R_{rb} \tag{6.7}$$

where α_1, α_2, and α_3 are configurable weights and were set by default, and $\alpha_1 = 0.8, \alpha_2 = \alpha_3 = 0.1$, emphasizing the key focus of our orchestration on execution fidelity while maintaining circuit complexity and decision ranking awareness. By aggregating these complementary components, $\mathcal{F}_{i,j}$ provides a robust fidelity-based performance measure. It supports fair and adaptive orchestration across diverse workloads and heterogeneous quantum hardware, and serves as a principled reward signal in reinforcement learning-based policy optimization of this work.

Time Penalty Score

The time penalty score $\mathcal{T}_{i,j}$ captures the temporal cost of quantum task execution, encompassing both waiting and actual quantum execution time. For task τ_i assigned to node n_j, the quantum execution time is estimated by analyzing the critical path of the transpiled circuit:

$$T_{i,j}^{exec} = s_i \cdot \sum_{g \in \mathrm{CP}(\mathcal{C}_i')} \mathcal{D}_j(g, \vec{q}_g) \tag{6.8}$$

where $\mathrm{CP}(\mathcal{C}_i')$ represents the critical path (longest execution sequence) of the transpiled circuit, and $\mathcal{D}_j(g, \vec{q}_g)$ is the gate execution duration along the critical path, s_i is the number of shots (execution iterations). An example of the longest path of a quantum circuit is illustrated in Figure 6.2. The total completion time includes queuing delays of QTask until it can be executed at the targeted QNode $\mathcal{T}_{i,j} = T_{i,j}^{wait} + T_{i,j}^{exec}$ and is normalized based on the maximum completion time bound to enable stable policy training.

6.3.4 Problem Formulation

Given the fidelity performance score $\mathcal{F}_{i,j}$ and time penalty score $\mathcal{T}_{i,j}$ defined above, we formulate the quantum cloud orchestration problem as a sequential decision-making

process that maximizes the combined orchestration performance across all quantum tasks.

The quantum cloud orchestration problem can be formally stated as follows: Given a sequence of quantum tasks $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_N\}$ arriving dynamically in a quantum cloud environment with heterogeneous quantum nodes $\mathcal{N} = \{n_1, n_2, \ldots, n_M\}$, find an optimal orchestration policy $\pi : \Gamma \rightarrow \mathcal{N}$ that assigns each task τ_i to an appropriate quantum node $n_j = \pi(\tau_i)$ to maximize the overall orchestration performance. For each assignment decision $\pi(\tau_i)$, the overall orchestration performance is quantified by combining the fidelity performance score and time penalty score, with a negative value of time score indicating the secondary goal to minimize the time penalty:

$$\mathcal{P}_{i,\pi(\tau_i)} = \mathcal{F}_{i,\pi(\tau_i)} - \beta \cdot \mathcal{T}_{i,\pi(\tau_i)} \tag{6.9}$$

where $\mathcal{F}_{i,\pi(\tau_i)}$ represents the relative fidelity performance score, β is the configurable time penalty weight and $\mathcal{T}_{i,\pi(\tau_i)}$ represents the time penalty, ensuring that higher fidelity and lower execution time both contribute positively to the overall performance score. The primary optimization objective is to maximize the cumulative orchestration performance across all quantum tasks can be defined as follows:

$$\max_{\pi} \sum_{i=1}^{N} \mathcal{P}_{i,\pi(\tau_i)} = \max_{\pi} \sum_{i=1}^{N} \left[\mathcal{F}_{i,\pi(\tau_i)} - \beta \cdot \mathcal{T}_{i,\pi(\tau_i)} \right] \tag{6.10}$$

subject to:

$$C1 : Size(\pi(\tau_i)) = 1, \forall \pi(\tau_i) \in \{1, \ldots, M\} \tag{6.11}$$

$$C2 : q_{\tau_i} \leq q_{\pi(\tau_i)}, \forall i \in \{1, \ldots, N\} \tag{6.12}$$

$$C3 : \mathcal{C}_{\tau_i} \hookrightarrow \mathcal{G}_{\pi(\tau_i)}, \forall i \in \{1, \ldots, N\} \tag{6.13}$$

where N is the total number of quantum nodes, M is the total number of tasks that need to be scheduled. $C1$ shows that each QTask will be allocated to exactly one QNode at a time, $C2$ indicates that the number of qubits in the targeted QNode needs to be larger than or equal to the number of qubits required by the allocated QTask, and $C3$ implies that the quantum circuit of the QTask can be mapped to the QNode through

the transpilation process. This optimization problem exhibits several challenges. First, quantum node characteristics ($\mathcal{D}_j, \mathcal{E}_j$) vary with calibration cycles, and queue states $\rho_j(t)$ change with task arrivals and completions, requiring adaptive decision-making capabilities. Second, the discrete assignment decisions combined with non-linear relationships between circuit characteristics and performance metrics create a combinatorial optimization problem. Furthermore, the sequential arrival of quantum tasks with unknown future characteristics necessitates an adaptive optimization without complete future information. The tension between fidelity maximization and time minimization requires sophisticated balancing strategies that adapt to different priorities.

Given these complexities, traditional heuristic approaches cannot effectively navigate the dynamic trade-offs inherent in quantum cloud orchestration. Therefore, we propose a deep reinforcement learning approach that models this problem as a Markov Decision Process, enabling the learning of adaptive orchestration policies that can balance fidelity and time objectives with a set of configurable weights that can be adjusted based on the priority of the orchestration. The sequential nature of the decision-making process, combined with the need for adaptive policy learning, makes reinforcement learning particularly well-suited for this orchestration challenge, which is widely used effectively for resource management in the classical computing environments [266, 268, 281].

6.4 QFOR Framework and Technique

6.4.1 Main components and Design

The QFOR framework implements an adaptive orchestration system that applies deep reinforcement learning to optimize quantum task placement in heterogeneous cloud environments. Figure 6.3 illustrates the framework architecture, consisting of six integrated components that collectively enable adaptive fidelity-aware orchestration decisions.

1. *QTask Generator and Dataset*: The QTask Generator serves as the workload interface, consuming quantum circuit representations from standardized datasets (for example, MQTBench [1]) in OpenQASM format [293]. This component simulates

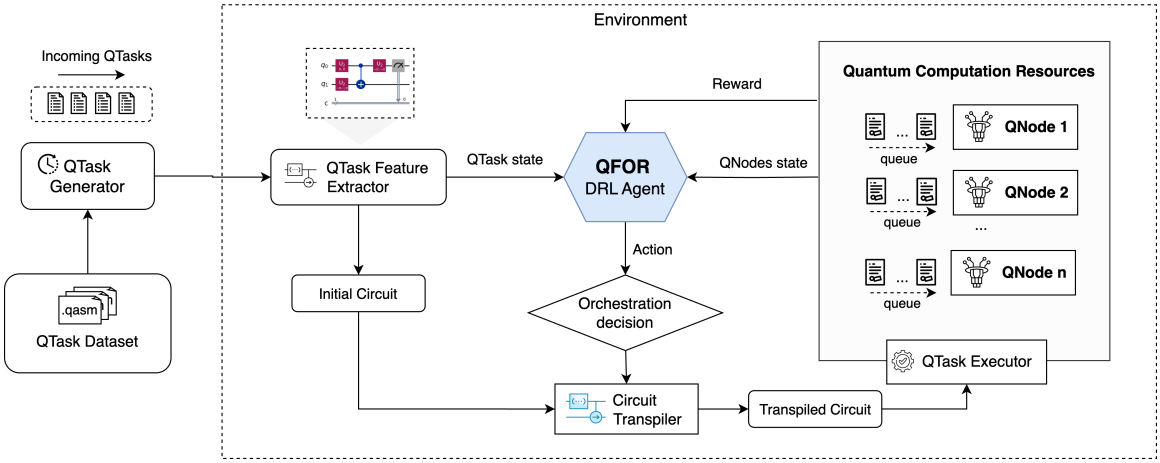


Figure 6.3: Overview of the main components of the QFOR framework

realistic quantum cloud workloads by generating tasks with diverse complexity characteristics—varying qubit requirements, circuit depths, and algorithmic patterns. The generator also supports configurable quantum task arrival patterns to model different cloud workload scenarios.

2. *QTask Feature Extractor:* This component analyzes quantum circuits to extract critical features, including qubit count, circuit depth, gate statistics (single-qubit, two-qubit gate, measurement counts), and circuit structure. These features are normalized and encoded into a concise representation that characterizes the computational requirements and complexity of each QTask, enabling the DRL agent to make informed scheduling decisions based on circuit characteristics.

3. *Deep Reinforcement Learning Agent*: The key component that implements the DRL-based decision-making policy. It observes the current state of quantum tasks and available quantum nodes, processes this information through a DRL agent, and produces orchestration decisions that assign tasks to appropriate quantum nodes. The orchestrator continuously learns from execution outcomes through the defined reward function that optimizes the overall performance of the orchestration decision.

4. *Heterogeneous Quantum Cloud Environment:* The framework extends the capabili-

ties of QSimPy[3] to models of heterogeneous quantum computing resources (QNodes) with varying capabilities, including qubit counts, connectivity graphs, gate durations, and error rates. Each node maintains dynamic state information such as queue length and availability, enabling realistic simulation of quantum cloud environments.

5. *Circuit Transpiler:* This component transforms logical quantum circuits into hardware-specific implementations optimized for the target quantum node's topology and native gate set. The transpilation process is guided by the orchestration decision and affects both execution time and fidelity. We utilized the Qiskit Transpiler[4] with optimisation level 3 as the default transpilation mode.

6. *QTask Executor:* This component mimics the execution of a quantum task on noisy quantum nodes by analysing the transpiled quantum circuit and estimating execution fidelity and time metrics based on the system model defined in Section 6.3 and the calibration snapshot data of IBM Quantum systems (with Qiskit Fake Provider[5]).

The framework provides a learning system loop where the DRL agent observes environment states, selects quantum node assignments, receives performance feedback through the reward obtained, and iteratively improves scheduling policies. The integration of realistic transpilation, noise-aware execution modeling, and configurable performance objectives enables the system to learn nuanced scheduling strategies that adapt to varying operational priorities while maintaining practical applicability to practical quantum cloud systems.

This modular architecture supports extensibility for different quantum hardware backends, alternative circuit datasets, and enhanced performance models while maintaining the core orchestration capability through learning-based algorithms with design principles similar to existing works (see Chapter 4).

[3]https://github.com/Cloudslab/qsimpy
[4]https://quantum.cloud.ibm.com/docs/en/api/qiskit/transpiler
[5]https://quantum.cloud.ibm.com/docs/en/api/qiskit-ibm-runtime/fake-provider

6.4.2 Deep Reinforcement Learning Model

Based on the system model and problem formulation in Section 6.3, we model the quantum task orchestration as a Markov Decision Process (MDP) to enable adaptive policy training through deep reinforcement learning. The MDP is defined by the tuple $(\mathbb{S}, \mathbb{A}, \mathbb{P}, \mathbb{R}, \gamma)$, where $\mathbb{S}$ is the state space, $\mathbb{A}$ the action space, $\mathbb{P}$ the state transition probability, $\mathbb{R}$ the reward function, and $\gamma \in [0, 1]$ the discount factor that balances immediate and future rewards. At each discrete time step t, the agent observes the current state s_t of the environment, selects an action a_t according to a policy $\pi(a_t|s_t)$, and moves to the next state s_{t+1} while receiving a reward r_t. The objective of the agent is to maximize the expected cumulative discounted reward, defined as $\mathbb{V}^{\pi}(s_t) = \mathbb{E}_{\pi}\left[\sum_t \gamma^t r_t\right]$, by learning an optimal policy π. Typically, the policy is parameterized by a neural network and is improved iteratively through training based on observed transitions and rewards.

State Space $\mathbb{S}$: The state s_t at time $t \in \mathbb{T}$ is a concatenation of the feature vector of the current quantum task τ_i and the feature vectors of all available quantum nodes (QNodes) $\mathcal{N}$ in the environment.

$$s_t = \left(\mathbf{f}_t^{\tau_i}, \mathbf{f}_t^{\mathcal{N}}\right) \tag{6.14}$$

where $\mathbf{f}_t^{\tau_i} \in \mathbb{R}^p$ is the p-dimensional feature vector of the current QTask τ_i, and $\mathbf{f}_t^{\mathcal{N}} \in \mathbb{R}^{m \times o}$ is the concatenated feature matrix of all m QNodes, each with o features. All features are normalized to $[0, 1]$ to ensure stable learning dynamics across heterogeneous scales. Thus, the State space can be defined as:

$$\mathbb{S} = \{s_t | s_t = \left(\mathbf{f}_t^{\tau_i}, \mathbf{f}_t^{\mathcal{N}}\right), \forall t \in \mathbb{T}\} \tag{6.15}$$

Action Space $\mathbb{A}$: The action space is discrete and corresponds to the assignment of a suitable QNode for the placement of the current incoming QTask. At each timestep, the agent chooses action $a_t \in \{0, 1, \ldots, m-1\}$, where $m = |\mathcal{N}|$ and $a_t = i$ denotes assigning the task to QNode $n_i \in \mathcal{N}$. Thus,

$$\mathbb{A} = \mathcal{N} \tag{6.16}$$

Reward Function $\mathbb{R}$: The reward function directly implements the orchestration per-

formance model defined in Equation 6.9. For each assignment decision to allocate QTask τ_i to QNode n_j at time t, the reward $r_t \in \mathbb{R}$ combines the fidelity score $\mathcal{F}_{i,j}$ and time penalty $\mathcal{T}_{i,j}$ with configurable weight β:

$$r_t = \begin{cases} \mathcal{F}_{i,j} - \beta \cdot \mathcal{T}_{i,j} & \text{if the execution is successful} \\ P_{fail} & \text{otherwise} \end{cases} \tag{6.17}$$

A failure penalty P_{fail} is applied when task τ_i cannot be executed on the selected node for any reason, providing negative feedback for infeasible assignments to improve the decision of the reinforcement learning policy.

6.4.3 QFOR Policy

The QFOR technique extends the Proximal Policy Optimization algorithm [291] to the quantum cloud orchestration problem, with the overall training workflow shown in Algorithm 5. The algorithm operates on the principle of learning an optimal scheduling policy $\pi_\theta : \mathcal{S} \to \Delta(\mathbb{A})$ that maximizes long-term cumulative reward while maintaining stable learning dynamics through trust region constraints.

Initially, the algorithm initializes three components: (i) a parameterized policy network π_θ with parameters θ that maps states to action probability distributions, (ii) a value function approximator V_ϕ with parameters ϕ that estimates state values for advantage computation, and (iii) an experience buffer $\mathcal{B}$ for storing trajectory data. Besides, the quantum computing environment instantiation involves configuring heterogeneous quantum nodes to mimic the realistic NISQ hardware parameters, including error rates, gate durations, and qubit connectivity constraints.

The outer training loop iterates over K policy improvement cycles, where each iteration corresponds to one complete policy update using collected experience data. This structure follows the standard PPO algorithm [291] and Ray RLlib [280] of alternating between data collection and policy optimization phases. The experience buffer is cleared at the beginning of each iteration to ensure on-policy learning. Parallel rollout collection across W workers enables efficient data gathering and improved sample diversity. Each worker operates independently, reducing correlation between consecutive experiences

Algorithm 5: QFOR Training Workflow with Proximal Policy Optimization

Input: QTask dataset $\mathcal{QD}$, QNodes $\mathcal{N}$, hyperparameters Θ
Output: Trained policy π_θ

Initialize: Policy network π_θ, value network V_ϕ, experience buffer $\mathcal{B}$;
Initialize: Quantum Cloud Environment in QSimPy (QSimPyEnv);
for *iteration* $k = 1, 2, \ldots, K$ **do**
  $\mathcal{B} \leftarrow \emptyset$ // Clear experience buffer
  for *worker* $w = 1, 2, \ldots, W$ **do**
    Reset environment: $(s_0, \tau_0) \sim \text{QSimPyEnv}(\mathcal{QD})$;
    for *step* $t = 0, 1, \ldots, T-1$ **do**
      $s_t \leftarrow f_{\text{state}}(\tau_t, \mathcal{N}, t)$ // Normalized features
      $a_t \sim \pi_\theta(\cdot|s_t)$ // Sample action from policy
      ProcessTask(τ_t, n_{a_t});
      $r_t \leftarrow$ CaculateReward$()$;
      $s_{t+1}, \tau_{t+1} \leftarrow$ NextTask$()$ // Get next QTask
      $\mathcal{B} \leftarrow \mathcal{B} \cup \{(s_t, a_t, r_t, s_{t+1})\}$ // Store exp.
      if *episode terminated or* $t = T-1$ **then**
        break;
      end if
    end for
  end for
  Policy Update:;
  Compute advantages: $\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \ldots$;
  Compute returns: $\hat{R}_t = \hat{A}_t + V_\phi(s_t)$;
  for *epoch* $e = 1, 2, \ldots, E$ **do**
    for *minibatch* $\mathcal{B}_m \subset \mathcal{B}$ **do**
      $r_t(\theta) \leftarrow \dfrac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ // Probability ratio
      $L^{\text{CLIP}}(\theta) \leftarrow \mathbb{E}\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$;
      $L^{\text{VF}}(\phi) \leftarrow \mathbb{E}\left[(V_\phi(s_t) - \hat{R}_t)^2\right]$;
      $L^{\text{ENT}}(\theta) \leftarrow \mathbb{E}\left[H(\pi_\theta(\cdot|s_t))\right]$ // Entropy bonus
      $\theta \leftarrow \theta + \alpha_\pi \nabla_\theta (L^{\text{CLIP}}(\theta) + c_1 L^{\text{ENT}}(\theta))$;
      $\phi \leftarrow \phi - \alpha_v \nabla_\phi L^{\text{VF}}(\phi)$;
    end for
  end for
  $\theta_{\text{old}} \leftarrow \theta$ // Update old policy parameters
end for
return *Trained policy* π_θ

and enhancing the robustness of gradient estimates during policy updates. Environment reset initializes a new episode by sampling the initial QTask τ_0 from the QASM-based [293]circuit dataset $\mathcal{QD}$ and computing the corresponding initial state s_0. The stochastic nature of task arrival ensures diverse training scenarios and prevents overfitting to specific task sequences.

The inner episode loop processes quantum tasks sequentially until episode termination at the final timestep t or when the task limit is reached. Each step corresponds to scheduling one incoming quantum task to an available quantum node. Action sampling follows the current policy distribution, where a_t represents the selected QNode index. The stochastic policy enables exploration while the learned parameters θ bias the selection toward high-reward actions based on accumulated experience. Each QTask execution involves a circuit transpilation process and performance metrics estimation on the selected quantum node n_{a_t} based on the calibration data of the quantum device. Then, the reward function combines multiple performance indicators as defined (see Equation 6.17) in the previous section.

State transition involves advancing to the next quantum task in the episode sequence and updating the environment time. The next state s_{t+1} incorporates updated quantum nodes metrics and the new quantum task's characteristics, maintaining the Markov property essential for policy gradient convergence. Experience tuple storage enables subsequent policy optimization through gradient-based updates. Each tuple (s_t, a_t, r_t, s_{t+1}) provides a complete transition record necessary for advantage estimation and policy gradient computation. Episode termination logic ensures proper boundary handling when task limits are reached or no additional tasks are available. This prevents infinite episodes while maintaining consistent episode lengths for stable learning dynamics.

For the policy optimization phase, our QFOR technique leverages the standard PPO algorithm [291] and adapts to quantum task orchestration. The advantage computation using Generalized Advantage Estimation (GAE) [294]:

$$\hat{A}_t = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}$$

where $\delta_t = R_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$ represents the temporal difference error. GAE bal-

ances bias and variance in advantage estimation through the λ parameter.

During the policy update phase, PPO's clipped surrogate objective function is used to ensure bounded policy updates:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$

where $r_t(\theta) = \dfrac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$. The value function loss $L^{\text{VF}}(\phi) = \mathbb{E}[(V_\phi(s_t) - \hat{R}_t)^2]$ and entropy bonus $L^{\text{ENT}}(\theta) = \mathbb{E}[H(\pi_\theta(\cdot|s_t))]$ provide additional optimization objectives for stable learning and adequate exploration. The clipped objective function ensures monotonic policy improvement with high probability, while the adaptive reward structure maintains the Markov property essential for convergence and provides scale-invariant rewards across diverse circuit complexities, enabling consistent learning signals.

6.5 Performance Evaluation

6.5.1 Environment Setup

We use our QSimPy framework[6] for the simulation of quantum cloud environments. We also extended it further to support the modeling of noisy quantum nodes using device calibration data and mimic the practical execution process of a quantum task, which comprises the circuit transpilation to selected QNodes before execution. This approach allows us to emulate heterogeneous quantum cloud computing environments and quantum task execution more comprehensively compared to other existing work, such as [290] and our iQuantum toolkit (see Chapter 4). For quantum cloud computation resources, we created a cluster of 5 different quantum nodes ranging from 27 to 127 qubits using the calibration data of IBM devices (using Qiskit FakeBackend instances[7]), including `ibm_auckland`, `ibm_hanoi`, `ibm_kolkata`, `ibm_brisbane`, and `ibm_sherbrooke`.

For quantum tasks, we created different training and evaluation datasets with 16 different quantum benchmark algorithms with qubit numbers ranging from 2 to 27 and

[6] https://github.com/Cloudslab/qsimpy
[7] https://quantum.cloud.ibm.com/docs/en/api/qiskit-ibm-runtime/fake-provider

initial circuit depths (before transpilation) of 3–30 layers, derived from the MQT Bench dataset [1]. The QTask arrival times were generated following a Poisson distribution, similar to other works [295, 296] to mimic the task arrival at the cloud data center. We use Gymnasium to wrap the QSimPy-based environment and use Ray RLlib [280] for implementing the reinforcement learning training and using Ray Tune [278] for hyperparameter tuning. All experiments are conducted at the Melbourne Research Cloud computation node with an AMD EPYC 9474F 48-core CPU and 128GB of RAM.

6.5.2 Performance Study

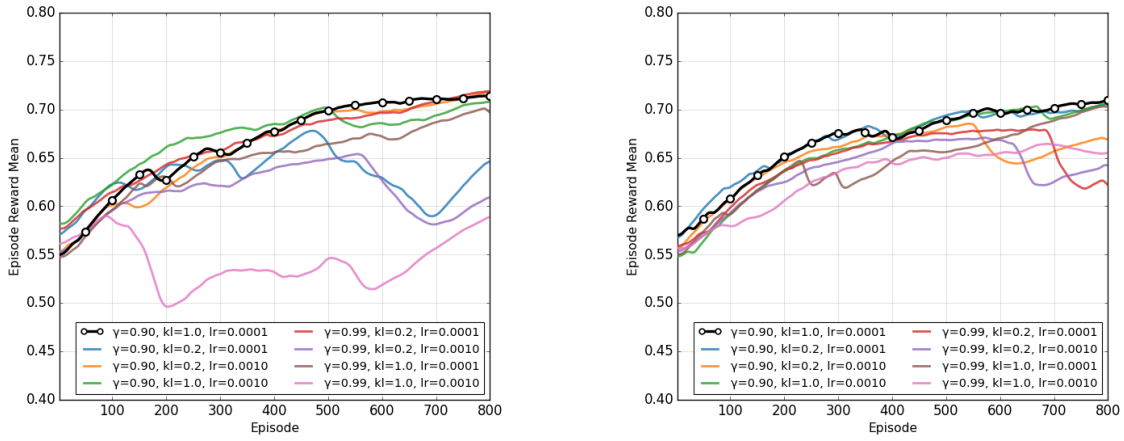
QFOR Policy Training Performance

To thoroughly evaluate the adaptability of QFOR across different operational priorities in quantum cloud environments, with the main priority to optimise the fidelity performance, we trained separate policy instances under different time weights $\beta \in \{0.5, 1.0\}$ and different hyperparameters to explore the orchestration trade-off and determine the balance configuration for optimizing the overall orchestration performance. We employed Ray Tune [278] for systematic hyperparameter optimization across all three β configurations, evaluating key parameter combinations to identify optimal settings. The optimal hyperparameter configuration was selected based on convergence stability and final performance across all training modes, with the tuning result shown in Table 6.2. Other hyperparameters and settings are based on the default configuration of PPO in Ray RLlib [280].

Table 6.2: QFOR Training Hyperparameters

QFOR Parameters	Value
Learning Rate	0.0001
Discount Factor (γ)	0.9
KL Coefficient	1.0
GAE Parameter (λ)	0.95
PPO Clip Parameter	0.3
Entropy Coefficient	0.01
Train Batch Size per Learner	180
Fidelity Reward Weight ($\alpha_1, \alpha_2, \alpha_3$)	0.8, 0.1, 0.1

Figure 6.4 shows training convergence across 800 training episodes. The optimal configuration across all configurations demonstrates superior performance with three key advantages: (1) stable reward convergence to approximately 0.70 by training episode 400-500 across all modes, (2) minimal variance indicating robust optimization dynamics, and (3) consistent performance across different β values, demonstrating hyperparameter robustness. Other configurations exhibited training instability and performance degradation after episode 500-600, particularly in balanced and high-performance modes. The lower discount factor ($\gamma = 0.9$) proves advantageous for quantum orchestration by appropriately balancing immediate scheduling decisions with long-term efficiency in dynamic environments.



(a) $\beta = 0.5$ **(b)** $\beta = 1.0$

Figure 6.4: Training convergence comparison across all configurations. The optimal hyperparameter configuration (black line with markers) achieves more consistent convergence in both time weight β settings.

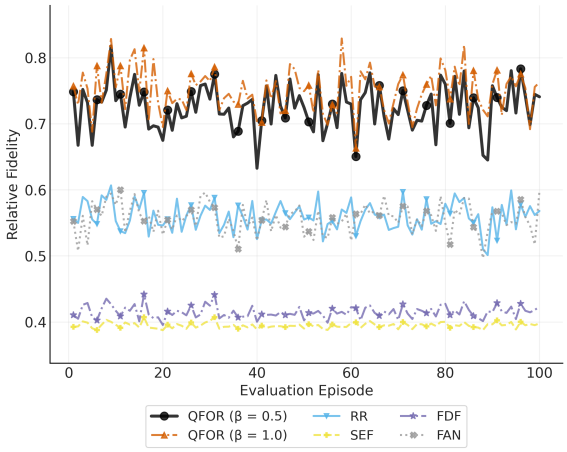
Execution Fidelity Performance Analysis

To evaluate the effectiveness of the QFOR policy, we conducted a comprehensive performance evaluation across 100 evaluation episodes with 6,000 quantum tasks using an independent test dataset distinct from the training data to ensure an unbiased assessment of the learned policies' generalization capabilities and practical applicability. We compared QFOR against four representative baseline policies, similar to the evaluation

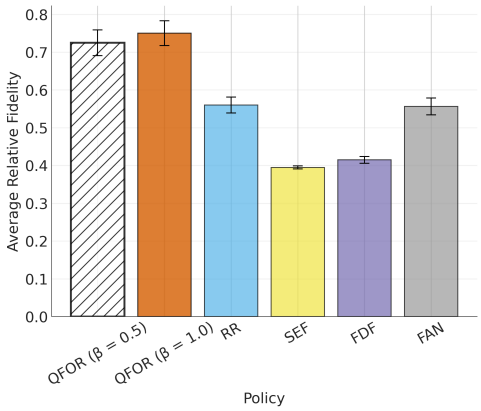
approach of Chapter 5 and other related works [160, 290]:

- *Round Robin (RR)*: Distributes quantum tasks cyclically across nodes, ensuring fair resource allocation while potentially ignoring node-specific characteristics.
- *Smallest Error First (SEF)*: Assigns QTask to QNode with the smallest average error rates of all gate operations.
- *Fastest Duration First (FDF)*: Assigns QTask to QNode with the fastest average gate duration times.
- *First Available Node (FAN)*: Assigns QTask to the first idle quantum node to minimize waiting time.

These baselines represent the spectrum of conventional scheduling strategies commonly employed in distributed computing environments. Figure 6.5 and Table 6.3 present the relative fidelity performance comparison across all policies.



(a) Execution fidelity in all evaluation episodes



(b) Average execution fidelity of all policies

Figure 6.5: Relative execution fidelity performance comparison across all policies over 100 evaluation episodes.

QFOR demonstrates substantial fidelity advantages across all operational configurations. The episode-wise analysis in Figure 6.5a shows QFOR's consistent enhancement throughout the evaluation period. Notably, all three QFOR configurations maintain stable performance in the 0.70-0.75 range, while baselines cluster in the 0.394-0.56 range.

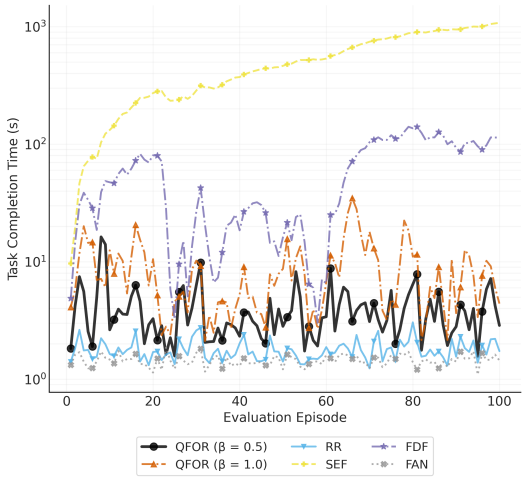
As shown in Figure 6.5b, with error bars denoting standard deviation, QFOR achieves consistently higher relative fidelity performance, with an average fidelity score at 0.725 using $\beta = 0.5$, and 0.75 using $\beta = 1.0$. These results represent significant improvements over the best-performing baseline (RR at 0.56), with enhancement margins of 29.5% and 34%, respectively. In contrast, the worst-performing policy, SEF, which greedily selects the QNode with the smallest average error rate regardless of the scheduled quantum circuit structure, fails to maintain a high fidelity performance score, achieving a mean performance score of 0.395. This highlights that final execution fidelity depends not only on the average error rate of the QNode but also critically on the transpilation process, including the mapping of the QTask to the specific qubit topology of the QNode. These results indicate the key limitations of conventional scheduling approaches in quantum cloud environments and demonstrates robust policy learning that generalizes effectively to unseen quantum tasks and dynamic device conditions. The consistent high performance across different β values demonstrates the ability of QFOR to learn context-aware scheduling policies that adapt optimization priorities while maintaining overall effectiveness.

Table 6.3: Detailed performance comparison of all policies, regarding average relative fidelity score, execution time, and total completion time (± standard deviation) over 100 evaluation episodes

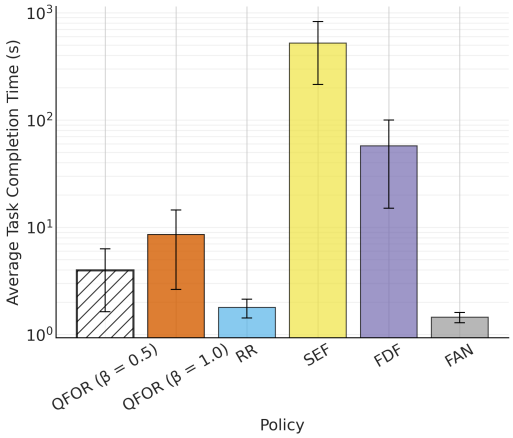
Policy	Average Fidelity Score	Average Execution Time (s)	Average Completion Time (s)
QFOR ($\beta = 0.5$)	0.725 ± 0.034	1.104 ± 0.131	3.967 ± 2.336
QFOR ($\beta = 1.0$)	0.750 ± 0.033	1.412 ± 0.161	8.580 ± 5.946
RR	0.560 ± 0.021	1.523 ± 0.174	1.784 ± 0.357
SEF	0.395 ± 0.004	1.220 ± 0.123	521.675 ± 306.631
FDF	0.415 ± 0.009	1.039 ± 0.114	57.502 ± 42.483
FAN	0.556 ± 0.022	1.444 ± 0.157	1.448 ± 0.159

Execution Fidelity-Time Trade-off Analysis

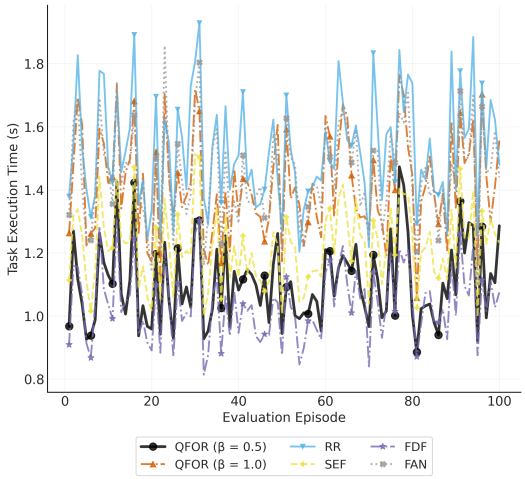
As the main priority of the orchestration is optimizing the fidelity performance, with the secondary consideration being to balance the execution time required, we conducted a comprehensive analysis of total completion time and quantum execution time of all QTasks in the evaluation across all policies to find the optimal time weight to achieve



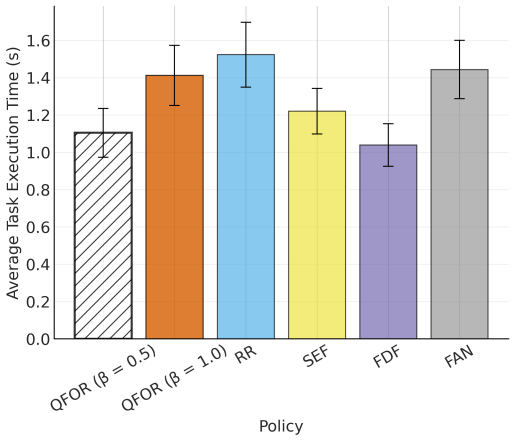
(a) Task completion time across evaluation episodes



(b) Average task completion time comparison



(c) Task execution time across evaluation episodes



(d) Average task execution time comparison

Figure 6.6: Task completion and quantum execution time analysis of all policies over 100 evaluation episodes.

this orchestration goal and explore the trade-off between fidelity and time in the decision. Figure 6.6 and Table 6.3 present a detailed timing analysis across all policies, demonstrating both average performance and episode-wise progression over 100 evaluation episodes.

The results indicate that greedy policies focusing exclusively on gate error rates (SEF) or gate duration time (FDF) lead to substantial waiting times, resulting in significantly longer total completion times compared to other approaches. In contrast, policies such as First Available Node (FAN) and Round Robin (RR) effectively minimize waiting times by distributing tasks more evenly, thereby reducing overall completion time.

As quantum execution time cost is critical and task queueing before execution can be handled by a classical controller, it is more valuable and cost-efficient to optimise the quantum execution time. Figures 6.6c and 6.6d show the actual quantum execution time of all policies. The result shows that the balanced time weight $\beta = 0.5$ achieves a reasonable average execution time, which is slightly higher than the execution time greedy policy (FDF), while maintaining the fidelity score consistently higher than all of the other baselines, up to approximately 84% higher relative fidelity performance score compared to the SEF baseline. This result suggests the optimal and balanced configurations of the time weight that encourage the QFOR policy to achieve high fidelity while maintaining the balance of fidelity-time tradeoff. Furthermore, minimising the execution time also inherently optimises the monetary cost of using quantum resources, which is essential in the current landscape of quantum cloud computing environments.

The results demonstrate that adaptive orchestration fundamentally balances the execution fidelity-time trade-off. While traditional approaches force a binary choice between speed and quality, QFOR policy identifies strategies that balance both objectives. This analysis establishes three key insights for quantum cloud orchestration: (1) Fidelity should be the key consideration in the current NISQ era as error sensitivity makes quality optimization essential as fast but inaccurate operations are ultimately ineffective, (2) Our adaptive, configurable policy enable flexible resource management optimization strategies which can be further extended, and (3) DRL-based is a potential approach that discovers non-obvious scheduling patterns that outperform conventional heuristics.

6.6 Summary

This chapter presents QFOR, a novel fidelity-aware deep reinforcement learning framework for quantum task orchestration in heterogeneous cloud-based environments with NISQ computation resources. We developed a holistic orchestration technique specifically designed for optimizing overall fidelity performance, which is critical for quantum task execution. Our systematic approach integrates comprehensive quantum task execution and fidelity performance estimators based on device calibration data, enabling rigorous emulation of noisy quantum hardware behavior. The configurable orchestration objectives successfully balance execution fidelity and time across different operational priorities, with extensive evaluation demonstrating 29.5-84% fidelity performance score improvements over traditional baseline methods. This work establishes the foundation and facilitates learning-driven quantum resource management research in hybrid quantum-HPC systems, which require additional effort to develop a robust and adaptive framework and techniques to keep up with the advances of quantum hardware development.

Chapter 7
Conclusions and Future Directions

This chapter serves as the conclusion of the thesis, providing a comprehensive summary of the primary works and contributions presented. Additionally, it outlines essential future directions for the continued advancement of cloud-based quantum computing and its efficient resource management.

7.1 Summary of Contributions

The rapid evolution of quantum computing technologies has catalyzed the emergence of quantum cloud computing as a paradigm that democratizes access to quantum computational resources without requiring specialized infrastructure. Today, quantum cloud computing enables researchers, developers, and organizations to leverage quantum capabilities through cloud-based platforms, addressing the prohibitive costs and technical complexities of maintaining dedicated quantum hardware. Major cloud providers such as IBM Quantum, Amazon Braket, and Microsoft Azure Quantum have established quantum cloud services that provide remote access to various quantum processors and simulators. The integration of quantum computing with cloud infrastructure presents unique resource management challenges fundamentally different from classical cloud environments. Unlike traditional cloud computing, quantum cloud environments are characterized by heterogeneous quantum backends with varying capabilities, noise levels, and availability patterns. The inherently fragile nature of quantum states, the limited quantum coherence times, and the current limitations of NISQ-era quantum devices [15] create distinct scheduling and orchestration complexities. These challenges are compounded by the hybrid nature of quantum-classical workflows, where quantum tasks must be seamlessly integrated with classical preprocessing and postprocessing steps. As

a result, achieving efficient resource utilization and optimal task execution in quantum cloud environments requires novel approaches that account for quantum-specific constraints while maintaining compatibility with established cloud computing paradigms.

Chapter 1 briefly introduced the emerging paradigm of quantum cloud computing and its distinctive resource management challenges arising from the unique characteristics of quantum systems. The chapter highlighted the growing importance of quantum cloud platforms in democratizing quantum computing access and presented the critical need for specialized resource management techniques. Furthermore, it outlined the research questions addressing the gaps in quantum cloud resource management and summarized the thesis contributions towards advancing this rapidly evolving field.

Chapter 2 investigated the existing quantum cloud computing landscape through a comprehensive systematic mapping study, encompassing service models, platforms, applications, and resource management approaches. A detailed taxonomy of quantum cloud computing research domains was presented, categorizing studies according to their focus areas, including quantum serverless architectures, hybrid quantum-classical computing, and quantum cloud security. The chapter provided a thorough analysis of recent literature and identified critical research gaps in quantum resource management, highlighting the need for comprehensive frameworks and intelligent orchestration mechanisms.

Chapter 3 developed a holistic serverless quantum computing framework that enables seamless integration of quantum computation within classical cloud environments while avoiding vendor lock-in constraints. The proposed Quantum Function-as-a-Service (QFaaS) architecture provides multi-SDK support and adaptive backend selection policies, complemented by cold start mitigation strategies and DevOps integration capabilities. This contribution establishes the foundation for practical quantum serverless deployments in heterogeneous cloud environments.

Chapter 4 designed and implemented a comprehensive modeling and simulation framework for quantum computing environments that facilitates systematic evaluation of resource management strategies without requiring expensive physical quantum hardware access. The framework incorporates realistic quantum system models, discrete-event simulation architecture, and multi-use case support for various quantum resource

management scenarios. This contribution provides researchers and practitioners with essential tools for developing and validating quantum cloud resource management approaches.

Chapter 5 proposed a novel deep reinforcement learning-based approach for time-aware quantum task placement that adapts to dynamic quantum cloud environments and optimizes task completion efficiency. The developed Deep Q-Network framework, enhanced with Rainbow DQN techniques, addresses the complex decision-making requirements of quantum task scheduling through comprehensive state representation and adaptive reward mechanisms. This contribution demonstrates significant improvements in time-aware performance for quantum workload management.

Chapter 6 proposed a fidelity-aware quantum task orchestration framework using deep reinforcement learning that effectively balances execution fidelity and time constraints in NISQ-era quantum systems. The framework incorporates noise-aware performance modeling and Proximal Policy Optimization approaches to learn adaptive orchestration policies across heterogeneous quantum cloud environments. This contribution addresses the critical challenge of maintaining quantum execution quality while meeting operational time requirements, providing substantial fidelity improvements compared to conventional scheduling approaches.

Together, these research contributions establish a comprehensive foundation for adaptive and intelligent resource management in quantum cloud computing environments, representing significant advances in this rapidly evolving field. These developments provide essential building blocks for realizing the full potential of quantum cloud computing and pave the way for widespread adoption of quantum technologies in practical cloud-based applications.

7.2 Future Research Directions

While our work in this thesis demonstrates potential in quantum cloud resource management, we recognized several limitations that present opportunities for future research. This section outlines critical open problems and potential future directions based on insights from our research, aiming to improve the potential of quantum computing.

7.2.1 Quantum Computing Resource Management

Analogously to its classical counterpart, resource management emerges as a critical issue in the quantum cloud computing landscape, attributed to the diversity of quantum hardware technologies. Due to the heterogeneity of different quantum hardware technologies, multiple aspects of the quantum resource management problem, such as allocation, scheduling, and utilization, must be addressed. It is essential to develop algorithms that can efficiently allocate and schedule resources across varied quantum tasks while optimizing their utilization [31]. Quantum computation resources are inherently expensive, emphasizing the importance of designing efficient and adaptable resource allocation and scheduling algorithms. The resource utilization must also be optimized while ensuring the reliability of quantum devices to produce correct computation results. This challenge parallels those encountered in classical cloud computing, yet it is magnified by the unique complexities of quantum technology. Addressing this challenge is crucial for harnessing the full potential of quantum cloud computing, necessitating innovative approaches to navigate the intricacies of quantum resource management and facilitate the growth and application of quantum cloud computing services. Furthermore, the transition from NISQ to future fault-tolerant quantum hardware is expected to fundamentally reshape quantum resource management models, enabling new abstraction layers, scheduling strategies, and reliability guarantees.

Several directions can be considered to further extend the robustness of our work in Chapter 5 and 6. First, dynamic device calibration data can be considered to mimic continuously updated device characteristics and adapt scheduling policies in real-time. Second, we can extend the evaluation of QFOR on large-scale quantum cloud infrastructures with dozens of quantum devices and circuits that require a larger number of qubits. Additionally, circuit knitting [26] for distributing large quantum tasks across distributed quantum systems can be enhanced by utilizing realistic quantum circuit simulations instead of synthetic data, making it more suitable for practical environments. Furthermore, the centralized policy learning architecture may become a bottleneck in distributed quantum computing environments, motivating future exploration of distributed reinforcement learning architectures [230, 266] where multiple agents coordinate scheduling decisions across geographically distributed quantum resources. These

future directions will be essential for realizing the full potential and optimizing the resource management of quantum cloud computing as the technology matures toward practical quantum advantage applications.

7.2.2 Modeling and Simulation of Quantum Computing Environments

Accurate modeling and simulation remain indispensable for designing and evaluating resource management strategies in quantum computing. Existing simulators such as our iQuantum toolkit (see Chapter 4) have demonstrated the feasibility of modeling hybrid quantum-classical environments, enabling the evaluation of scheduling and orchestration policies without incurring the cost of real hardware access. Future research should extend simulation capabilities toward large-scale, distributed, and networked quantum computing infrastructures [297], incorporating models of quantum communication channels, entanglement distribution, and quantum internet protocols. Such extensions are critical for exploring the scalability of quantum cloud-edge systems [227] and for studying multi-tenant interactions under realistic workload dynamics. Furthermore, integrating communication latency models will provide more robust testbeds for evaluating quantum task orchestration frameworks. In addition, the development of visualization features, akin to tools such as CloudAnalyst [298], would enable intuitive analysis of quantum workloads, network behavior, and scheduling outcomes, thereby enhancing the accessibility and impact of simulation frameworks. Advancing modeling and simulation in these directions will accelerate the prototyping of algorithms and policies for quantum cloud resource management, supporting the transition from NISQ-era devices to upcoming fault-tolerant, large-scale quantum computing environments.

7.2.3 Quantum Serverless Computing

Leading companies in quantum computing, such as IBM, have claimed that quantum serverless is potentially the future of quantum programming [299]. To empower the quantum serverless model in the NISQ era, some open challenges must be addressed.

First, resource orchestration should be optimized to deal with various heterogeneous backends and the high load of requests from multiple users. Resource orchestration

refers to the collaboration of multiple processes of allocating, scheduling, and monitoring the usage of quantum and classical resources. This orchestration problem is essential for all cloud-based systems and is more challenging for serverless quantum computing systems, as both quantum and classical resources are involved. They must be flexibly orchestrated to fit all the task requirements and ensure resource utilization while minimizing the total cost. Second, supplement techniques, such as quantum circuit knitting [26], should be developed to enable large-scale quantum circuit execution on multiple NISQ devices. Circuit knitting can segment a large quantum circuit into smaller, manageable circuits for execution on different devices before integrating the outcomes, which is a vital area of research. Other approaches, including entanglement forging [300] and circuit cutting [253], are under exploration to address these challenges. These techniques are essential in overcoming the limitations of NISQ devices, enabling the execution of complex quantum computations, and advancing the serverless quantum computing model.

Several promising directions can also be explored to further extend the capabilities of our QFaaS framework in Chapter 3. First, QFaaS can be extended to support additional quantum cloud providers, enabling cross-platform execution capabilities. More advanced orchestration techniques can be incorporated to improve the automatic selection of quantum backends for hybrid quantum-classical applications. Additionally, enhancements to the cold start mitigation policy for quantum function execution can be developed to further optimize performance. Furthermore, strengthening the security and scalability features of QFaaS will be essential to support large-scale deployments and accommodate the growing demands of multi-user quantum cloud environments.

7.2.4 Distributed and Parallel Quantum Computation

The integration of multiple quantum computing resources to harness their collective power is essential [301, 302]. However, this direction faces significant challenges that require extensive effort to address. First, the coherence of quantum states needs to be maintained while transferring quantum data across different devices. Today's quantum computers are error-prone [15] due to the high sensitivity of quantum states, which are easily affected by environmental factors, such as electromagnetic noise and tempera-

ture, so maintaining the qubit quality in individual quantum devices is difficult. The challenge lies in ensuring the integrity of qubits across potentially vast distances, demanding advancements in quantum networking and data communication to achieve reliable inter-device connectivity. Besides, the scalability of the system and the requirement of designing effective distributed quantum algorithms are challenging for large distributed quantum systems. This requires innovative approaches to algorithm design that can navigate the intricacies of distributed quantum computations, optimizing performance across a heterogeneous network of quantum processors [104]. In addition, parallel quantum processing of multiple quantum tasks simultaneously on a single quantum computer is still an open problem without an efficient solution devised. Current quantum computing paradigms struggle to efficiently parallelize tasks due to the quantum decoherence and error rates associated with simultaneously manipulating multiple qubits. Developing methodologies for parallel quantum processing that can mitigate interference and maximize the utilization of quantum resources remains a critical area of research, promising to enhance the computational throughput of quantum systems significantly. Addressing these challenges is pivotal for realizing the full potential of distributed and parallel quantum computing [111].

7.2.5 Quantum Cloud Applications

The advent of quantum cloud computing has poised software engineering at the brink of significant evolution, offering access to quantum computational resources analogous to classical infrastructure. However, the emerging field of quantum software development confronts several limitations, particularly when leveraging cloud-based quantum resources. A primary problem is the lack of a standardized quantum programming model. Currently, each quantum cloud provider operates on distinct software platforms, development toolkits, and standards [58]. This absence of uniformity complicates the development of cross-platform quantum applications, impeding their ability to operate seamlessly across diverse cloud environments. Additionally, the deployment paradigm for quantum applications diverges significantly from that of traditional software. Unlike classical applications, which can reside on servers for on-demand invocation, quantum

applications require recompilation and transmission to quantum computers for each execution. This discrepancy underscores the need for innovative approaches to quantum software lifecycle management, emphasizing the critical role of standardization and the development of deployment methodologies tailored to the quantum computing domain.

Furthermore, quantum databases have recently been explored as a potential application within quantum cloud computing, offering a new approach for data storage and retrieval that leverages quantum principles for enhanced search and optimization capabilities [303]. Implementing practical quantum databases on cloud platforms presents unique challenges, including the high resource demands of quantum memory and the need for secure, efficient access in multi-tenant environments. Therefore, addressing these issues will require significant advancements in quantum data structures, error correction, and secure access protocols [304], all of which are still in the early stages.

7.2.6 Quantum Cloud for Quantum Machine Learning Applications

In the domain of classical computing, Machine Learning Operations (MLOps) streamline the machine learning (ML) application development and deployment with DevOps practices to enhance the efficiency of ML application workflows [305]. In the quantum computing realm, Quantum Machine Learning (QML) has emerged as a promising field, attracting significant interest from researchers [306]. The design of cloud-based systems tailored for QML applications is essential for their advancement. Adopting an approach akin to MLOps, termed Quantum Machine Learning Operations (QMLOps), could significantly streamline the development and management of QML applications. By applying the principles of MLOps to the quantum context, QMLOps aims to facilitate the seamless integration, deployment, and operation of QML workflows, thereby accelerating the maturation of quantum computing technologies and their application in solving complex computational problems. This convergence of quantum computing and machine learning within cloud environments underscores the potential for QMLOps to act as a trigger for innovation and efficiency in the realm of quantum machine learning.

7.2.7 LLM-guided Quantum Software and Resource Management

The emergence of large language models (LLMs) provides promising opportunities for advancing quantum software engineering and resource management in quantum computing environments. Current development workflows require specialized expertise in quantum computing and hardware-specific compilation, posing barriers for broader adoption. LLMs trained on high-quality quantum software datasets can potentially lower this barrier by assisting in generating, transpiling, and optimizing quantum circuits across heterogeneous backends [307–309]. Beyond software development, LLMs can be coupled with other machine learning approaches, such as deep reinforcement learning (see Chapter 5 and 6), to provide adaptive guidance for quantum task scheduling and resource allocation. For example, LLMs may serve as natural-language interfaces for specifying quantum workloads, which are then mapped to quantum-classical workflows by orchestration engines. This also raises critical research questions regarding the reliability of LLM-generated code, integration with quantum task scheduling policies, and the design of trustworthy co-pilots for quantum resource management. Addressing these challenges can significantly reduce the barrier to quantum application development and enable more efficient use of quantum computing resources.

7.2.8 Quantum-based Approaches for Cloud-Edge-IoT Resource Management

Quantum computing-based approaches also present potential opportunities for addressing resource management challenges in distributed computing environments, such as with cloud, edge, and Internet of Things (IoT) systems. Classical approaches to resource allocation, task offloading, and workload balancing across the cloud-edge-IoT continuum can face significant computational barriers when scaling to thousands or millions of connected devices with heterogeneous requirements. Quantum algorithms for optimization, such as the quantum approximate optimization algorithm (QAOA), offer potential advantages for solving complex resource allocation problems that can be modelled as quadratic unconstrained binary optimization (QUBO) or maximum cut problems [310, 311]. These quantum-enabled approaches can potentially address the resource management challenges, including task scheduling under deadline constraints,

energy-aware workload distribution, and service placement in cloud-edge computing environments. Future research should also explore quantum-inspired [312] and hybrid quantum-classical approaches that leverage near-term quantum processors for specific optimization subroutines while maintaining classical control frameworks.

7.2.9 Quantum Cloud Security and Privacy

The imperative to secure data and privacy within quantum cloud computing cannot be exaggerated, especially given quantum computing's capability to compromise conventional cryptographic techniques. There is a pressing need to develop security strategies rooted in quantum principles to protect the quantum cloud ecosystem. This includes a thorough exploration and mitigation of potential quantum-specific attack vectors. Additionally, Quantum Key Distribution (QKD) [313, 314] emerges as a pivotal research area in quantum cloud security, necessitating ongoing enhancements to ensure the secure transmission of encryption keys within cloud infrastructures. Despite progress in formulating attack models and devising corresponding defensive measures for quantum computing, current strategies only begin to address the breadth of potential security challenges. Several critical areas require further investigation, such as securing qubit technologies, quantum hardware and platforms, and preserving data security for quantum cloud applications [174]. Addressing these aspects of quantum cloud security is vital for advancing the field and ensuring the integrity and confidentiality of data within quantum computing environments.

Beyond these key challenges discussed, several emerging directions require further attention. First, integrating quantum cloud computing with the quantum internet [88, 92] introduces important challenges such as managing quantum state distribution, deploying quantum repeaters, and developing compatible quantum communication protocols. These are essential for building future distributed quantum systems. Second, combining quantum computing with the edge computing paradigm opens new use cases where limited devices, such as sensors or mobile systems, offload quantum tasks to cloud-based quantum processors. This can support efficient hybrid computing in real-world environments. Finally, as QCC technologies grow, social and ethical issues, such

as protecting user privacy, ensuring transparency in quantum decision-making, and providing fair access to quantum resources, must also be considered. These directions are important for building a secure, usable, and inclusive quantum cloud ecosystem.

7.3 Final Remarks

Quantum cloud computing has emerged to democratize access to quantum computational resources, enabling researchers and organizations to harness quantum capabilities without the prohibitive costs and complexities of maintaining dedicated quantum hardware. However, the unique characteristics of quantum systems, including noise sensitivity, limited coherence times, and heterogeneous quantum systems, present unprecedented resource management challenges that cannot be addressed by conventional cloud computing approaches. In this thesis, we investigated the intricate interplay between quantum hardware constraints and cloud resource management to develop adaptive and intelligent orchestration frameworks that optimize both performance and resource utilization in quantum cloud environments. The frameworks, algorithms, and simulation tools presented in this thesis achieve better resource efficiency and task execution quality while accommodating the inherent limitations of NISQ-era quantum systems. Our serverless quantum computing architecture enables seamless integration of quantum functions within classical workflows, while our deep reinforcement learning-based approaches provide adaptive solutions for dynamic quantum task scheduling and fidelity-aware orchestration. Furthermore, our comprehensive simulation framework offers researchers essential tools for developing and validating quantum cloud resource management algorithms without requiring expensive physical quantum hardware access. Research on quantum cloud resource management, such as presented in this thesis, will enable quantum cloud providers to efficiently manage quantum workloads in highly heterogeneous, dynamic, and noise-prone quantum computing environments at scale. Moreover, these research outcomes can further advance the practical deployment of quantum technologies and accelerate the development of quantum computing ecosystems.

Bibliography

[1] N. Quetschlich, L. Burgholzer, and R. Wille, "MQT Bench: Benchmarking Software and Design Automation Tools for Quantum Computing," *Quantum*, vol. 7, p. 1062, Jul. 2023.

[2] P. W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, Oct. 1997.

[3] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the 28th ACM Symposium on Theory of computing - STOC '96*, vol. 75. New York, USA: ACM, Jun. 1996, pp. 212–219.

[4] E. Farhi, J. Goldstone, and S. Gutmann, "A Quantum Approximate Optimization Algorithm," Tech. Rep., Nov. 2014. [Online]. Available: https://arxiv.org/abs/1411.4028

[5] A. Peruzzo, J. McClean, P. Shadbolt, M. H. Yung, X. Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, "A variational eigenvalue solver on a photonic quantum processor," *Nature Communications*, vol. 5, no. May, 2014.

[6] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, pp. 195–202, Sep. 2017.

[7] Y. Zhang and Q. Ni, "Recent advances in quantum machine learning," *Quantum Engineering*, vol. 2, no. 1, Mar. 2020.

[8] A. Bayerstadler, G. Becquin, J. Binder, T. Botter, H. Ehm, T. Ehmer, M. Erdmann, N. Gaus, P. Harbach, M. Hess, J. Klepsch, M. Leib, S. Luber, A. Luckow, M. Mansky, W. Mauerer, F. Neukart, C. Niedermeier, L. Palackal, R. Pfeiffer, C. Polenz, J. Sepulveda, T. Sievers, B. Standen, M. Streif, T. Strohm, C. Utschig-Utschig, D. Volz, H. Weiss, and F. Winter, "Industry quantum computing applications," *EPJ Quantum Technology*, vol. 8, no. 1, p. 25, Dec. 2021.

[9] B. Research, "Quantum Computing: Technologies and Global Markets to 2030," BCC Publishing, Tech. Rep. IFT149D, Jun. 2025. [Online]. Available: https://www.bccresearch.com/market-research/information-technology/quantum-computing-technologies-and-global-markets.html

[10] H. Soller, M. Gschwendtner, S. Shabani, and W. Svejstrup, "The Year of Quantum: From concept to reality in 2025," McKinsey & Company, Tech. Rep., Jun. 2025. [Online]. Available: https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/the-year-of-quantum-from-concept-to-reality-in-2025

[11] D. Herman, C. Googin, X. Liu, Y. Sun, A. Galda, I. Safro, M. Pistoia, and Y. Alexeev, "Quantum computing for finance," *Nature Reviews Physics*, vol. 5, no. 8, pp. 450–465, Jul. 2023.

[12] D. J. Egger, C. Gambella, J. Marecek, S. McFaddin, M. Mevissen, R. Raymond, A. Simonetto, S. Woerner, and E. Yndurain, "Quantum computing for finance: state-of-the-art and future prospects," *IEEE Transactions on Quantum Engineering*, vol. 1, pp. 1–24, 2020.

[13] Y. Cao, J. Romero, J. P. Olson, M. Degroote, P. D. Johnson, M. Kieferová, I. D. Kivlichan, T. Menke, B. Peropadre, N. P. D. Sawaya, S. Sim, L. Veis, and A. Aspuru-Guzik, "Quantum chemistry in the age of quantum computing," *Chemical Reviews*, vol. 119, no. 19, pp. 10 856–10 915, Oct. 2019.

[14] C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," *Theoretical Computer Science*, vol. 560, pp. 7–11, Dec. 2014.

[15] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, Aug. 2018.

[16] IBM, "IBM Quantum Computing Services," 2023. [Online]. Available: https://quantum-computing.ibm.com/

[17] Microsoft, "Azure Quantum," 2023. [Online]. Available: https://azure.microsoft.com/en-us/services/quantum

[18] "Cloud Quantum Computing Service - Amazon Braket - AWS," 2024. [Online]. Available: https://aws.amazon.com/braket/

[19] P. J. Karalekas, N. A. Tezak, E. C. Peterson, C. A. Ryan, M. P. Da Silva, and R. S. Smith, "A quantum-classical cloud platform optimized for variational hybrid algorithms," *Quantum Science and Technology*, vol. 5, no. 2, pp. 0–13, 2020.

[20] I. Faro, I. Sitdikov, D. G. Valiñas, F. J. M. Fernandez, C. Codella, and J. Glick, "Middleware for Quantum: An orchestration of hybrid quantum-classical systems," in *2023 IEEE International Conference on Quantum Software (QSW)*. Chicago, IL, USA: IEEE, Jul. 2023, pp. 1–8.

[21] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles, "Variational quantum algorithms," *Nature Reviews Physics*, vol. 3, no. 9, pp. 625–644, Aug. 2021.

[22] J. L. Hevia, G. Peterssen, and M. Piattini, "QuantumPath : A quantum software development platform," *Software: Practice and Experience*, vol. 2021, no. December, pp. 1–14, Dec. 2021.

[23] V. Stirbu, O. Kinanen, M. Haghparast, and T. Mikkonen, "Qubernetes: Towards a unified cloud-native execution platform for hybrid classic-quantum computing," *Information and Software Technology*, vol. 175, p. 107529, Nov. 2024.

[24] B. Weder, J. Barzen, M. Beisel, F. Bühler, D. Georg, F. Leymann, and L. Stiliadou, "Qunicorn: A Middleware for the Unified Execution Across Heterogeneous Quantum Cloud Offerings," Nov. 2024. [Online]. Available: http://arxiv.org/abs/2411.06889

[25] E. Giortamis, F. Romão, N. Tornow, D. Lugovoy, and P. Bhatotia, "Orchestrating Quantum Cloud Environments with Qonductor," Aug. 2024, arXiv:2408.04312 [quant-ph]. [Online]. Available: http://arxiv.org/abs/2408.04312

[26] C. Piveteau and D. Sutter, "Circuit Knitting With Classical Communication," *IEEE Transactions on Information Theory*, vol. 70, no. 4, pp. 2734–2745, Apr. 2024.

[27] F. Bova, A. Goldfarb, and R. G. Melko, "Commercial applications of quantum computing," *EPJ Quantum Technology*, vol. 8, no. 1, p. 2, Dec. 2021.

[28] E. Moguel, J. Rojo, D. Valencia, J. Berrocal, J. Garcia-Alonso, and J. M. Murillo, "Quantum service-oriented computing: current landscape and challenges," *Software Quality Journal*, vol. 30, no. 4, pp. 983–1002, Dec. 2022.

[29] A. W. Cross, L. S. Bishop, S. Sheldon, P. D. Nation, and J. M. Gambetta, "Validating quantum computers using randomized model circuits," *Physical Review A*, vol. 100, no. 3, p. 032328, Sep. 2019.

[30] R. LaRose, "Overview and comparison of gate level quantum software platforms," *Quantum*, vol. 3, p. 130, Mar. 2019.

[31] G. S. Ravi, K. N. Smith, P. Murali, and F. T. Chong, "Adaptive job and resource management for the growing quantum cloud," in *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*. Broomfield, CO, USA: IEEE, Oct. 2021, pp. 301–312.

[32] A. Wack, H. Paik, A. Javadi-Abhari, P. Jurcevic, I. Faro, J. M. Gambetta, and B. R. Johnson, "Quality, Speed, and Scale: three key attributes to measure the performance of near-term quantum computers," Oct. 2021, arXiv:2110.14108. [Online]. Available: http://arxiv.org/abs/2110.14108

[33] P. Castro, V. Ishakian, V. Muthusamy, and A. Slominski, "The rise of serverless computing," *Communications of the ACM*, vol. 62, no. 12, pp. 44–54, Nov. 2019.

[34] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, Jan. 2011.

[35] R. Mahmud, S. Pallewatta, M. Goudarzi, and R. Buyya, "iFogSim2: An extended iFogSim simulator for mobility, clustering, and microservice management in edge and fog computing environments," *Journal of Systems and Software*, vol. 190, no. 111351, pp. 1–17, 2022.

[36] P. S. Souza, T. Ferreto, and R. N. Calheiros, "EdgeSimPy: Python-based modeling and simulation of edge computing resource management policies," *Future Generation Computer Systems*, vol. 148, pp. 446–459, Nov. 2023.

[37] A. Citynow, "Enabling Serverless Quantum Computing with QuaO," Nov. 2023. [Online]. Available: https://citynow.asia/press/release/Enabling-Serverless-Quantum-Computing-with-QuaO

[38] R. Gobato, "Calculations Using Quantum Chemistry for Inorganic Molecule Simulation BeLi2SeSi," *Science Journal of Analytical Chemistry*, vol. 5, no. 5, p. 76, 2017.

[39] A. B. Magann, M. D. Grace, H. A. Rabitz, and M. Sarovar, "Digital quantum simulation of molecular dynamics and control," *Physical Review Research*, vol. 3, no. 2, p. 023165, Jun. 2021.

[40] Y. Cao, J. Romero, and A. Aspuru-Guzik, "Potential of quantum computing for drug discovery," *IBM Journal of Research and Development*, vol. 62, no. 6, pp. 6:1–6:20, Nov. 2018.

[41] M. Zinner, F. Dahlhausen, P. Boehme, J. Ehlers, L. Bieske, and L. Fehring, "Quantum computing's potential for drug discovery: Early stage industry dynamics," *Drug Discovery Today*, vol. 26, no. 7, pp. 1680–1688, Jul. 2021.

[42] K. Batra, K. M. Zorn, D. H. Foil, E. Minerali, V. O. Gawriljuk, T. R. Lane, and S. Ekins, "Quantum Machine Learning Algorithms for Drug Discovery Applications," *Journal of Chemical Information and Modeling*, vol. 61, no. 6, pp. 2641–2647, Jun. 2021.

[43] A. A. Saki, Alam, Suresh, and S. Ghosh, "A Survey and Tutorial on Security and Resilience of Quantum Computing," *Proceedings of the European Test Workshop*, vol. 2021-May, 2021.

[44] L. Huang, H. Zhou, K. Feng, and C. Xie, "Quantum random number cloud platform," *npj Quantum Information*, vol. 7, no. 1, p. 107, Dec. 2021.

[45] Y. Li, Y. Fei, W. Wang, X. Meng, H. Wang, Q. Duan, and Z. Ma, "Quantum random number generator using a cloud superconducting quantum computer based on source-independent protocol," *Scientific Reports*, vol. 11, no. 1, p. 23873, Dec. 2021.

[46] V. Kumar, J. B. B. Rayappan, R. Amirtharajan, and P. Praveenkumar, "Quantum true random number generation on IBM's cloud platform," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 8, pp. 6453–6465, Sep. 2022.

[47] R. Berganza Gomez, C. O'Meara, G. Cortiana, C. B. Mendl, and J. Bernabe-Moreno, "Towards AutoQML: A Cloud-Based Automated Circuit Architecture Search Framework," in *2022 IEEE 19th International Conference on Software Architecture Companion, ICSA-C 2022*, 2022, pp. 129–136, arXiv: 2202.08024.

[48] C. Gong, Z. Dong, A. Gani, and H. Qi, "Quantum k-means algorithm based on trusted server in quantum cloud computing," *Quantum Information Processing*, vol. 20, no. 4, p. 130, Apr. 2021.

[49] S. Fadli, B. S. Rawal, and A. Mentges, "Hybrid Quantum Machine learning using Quantum Integrated Cloud Architecture (QICA)," in *2023 International Conference on Computing, Networking and Communications (ICNC)*. Honolulu, HI, USA: IEEE, Feb. 2023.

[50] M. Hibat-Allah, M. Mauri, J. Carrasquilla, and A. Perdomo-Ortiz, "A framework for demonstrating practical quantum advantage: comparing quantum against classical generative models," *Communications Physics*, vol. 7, no. 1, p. 68, Feb. 2024.

[51] M. Kaiiali, S. Sezer, and A. Khalid, "Cloud computing in the quantum era," in *2019 IEEE Conference on Communications and Network Security (CNS)*, vol. 2019-Janua. IEEE, Jun. 2019, pp. 1–4.

[52] H. Soeparno and A. S. Perbangsa, "Cloud Quantum Computing Concept and Development: A Systematic Literature Review," in *Procedia Computer Science*, vol. 179. Elsevier B.V., 2021, pp. 944–954.

[53] F. Leymann, J. Barzen, M. Falkenthal, D. Vietz, B. Weder, and K. Wild, "Quantum in the Cloud: Application Potentials and Research Opportunities," in *Proceedings of the 10th International Conference on Cloud Computing and Services Science*. SCITEPRESS - Science and Technology Publications, Mar. 2020, pp. 9–24.

[54] N. C. for Reviews and Dissemination, "The database of abstracts of reviews of effects (DARE) york: university of york. Effectiveness matters 6(2)," 2002.

[55] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele University and University of Durham, UK, Technical Report EBSE-2007-01, Sep. 2007.

[56] B. Kitchenham, R. Pretorius, D. Budgen, O. Pearl Brereton, M. Turner, M. Niazi, and S. Linkman, "Systematic literature reviews in software engineering – A tertiary study," *Information and Software Technology*, vol. 52, no. 8, pp. 792–805, Aug. 2010.

[57] S. S. Gill, A. Kumar, H. Singh, M. Singh, K. Kaur, M. Usman, and R. Buyya, "Quantum computing: A taxonomy, systematic review and future directions," *Software - Practice and Experience*, vol. 52, no. 1, pp. 66–114, 2022.

[58] M. A. Serrano, J. A. Cruz-Lemus, R. Pérez-Castillo, and M. Piattini, "Quantum Software Components and Platforms: Overview and Quality Assessment," *ACM Computing Surveys*, 2022.

[59] A. A. Khan, A. Ahmad, M. Waseem, P. Liang, M. Fahmideh, T. Mikkonen, and P. Abrahamsson, "Software architecture for quantum computing systems — a systematic review," *Journal of Systems and Software*, vol. 201, p. 111682, Jul. 2023.

[60] Z. Yang, M. Zolanvari, and R. Jain, "A Survey of Important Issues in Quantum Computing and Communications," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 1059–1094, 2023.

[61] M. G. Matthew Brotherton, "A survey of quantum computing for cloud security," *Future Connected Technologies*, pp. 1–34, May 2023.

[62] M. Golec, E. S. Hatay, M. Golec, M. Uyar, M. Golec, and S. S. Gill, "Quantum cloud computing: Trends and challenges," *Journal of Economy and Technology*, vol. 2, pp. 190–199, Nov. 2024.

[63] M. Caleffi, M. Amoretti, D. Ferrari, J. Illiano, A. Manzalini, and A. S. Cacciapuoti, "Distributed quantum computing: A survey," *Computer Networks*, vol. 254, p. 110672, Dec. 2024.

[64] J. Illiano, M. Caleffi, A. Manzalini, and A. S. Cacciapuoti, "Quantum Internet protocol stack: A comprehensive survey," *Computer Networks*, vol. 213, p. 109092, Aug. 2022.

[65] M. Lewis and F. Glover, "Quadratic unconstrained binary optimization problem preprocessing: Theory and empirical analysis," *Networks*, vol. 70, no. 2, pp. 79–97, Jun. 2017.

[66] G. Kochenberger, J.-K. Hao, F. Glover, M. Lewis, Z. Lü, H. Wang, and Y. Wang, "The unconstrained binary quadratic programming problem: a survey," *Journal of Combinatorial Optimization*, vol. 28, no. 1, pp. 58–81, Apr. 2014.

[67] R. Orus, S. Mugel, and E. Lizaso, "Quantum computing for finance: Overview and prospects," *Reviews in Physics*, vol. 4, p. 100028, Nov. 2019.

[68] M. T. West, S.-L. Tsang, J. S. Low, C. D. Hill, C. Leckie, L. C. L. Hollenberg, S. M. Erfani, and M. Usman, "Towards quantum enhanced adversarial robustness in machine learning," *Nature Machine Intelligence*, vol. 5, no. 6, pp. 581–589, May 2023.

[69] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, Jun. 2012.

[70] M. T. West, S. M. Erfani, C. Leckie, M. Sevior, L. C. L. Hollenberg, and M. Usman, "Benchmarking adversarially robust quantum machine learning at scale," *Physical Review Research*, vol. 5, no. 2, p. 023186, Jun. 2023.

[71] M. Grassl and M. Rötteler, "Quantum error correction and fault tolerant quantum computing," *Encyclopedia of Complexity and Systems Science*, vol. Not available, pp. 7324–7342, 2009.

[72] R. Buyya, S. N. Srirama, G. Casale, R. Calheiros, Y. Simmhan, B. Varghese, E. Gelenbe, B. Javadi, L. M. Vaquero, M. A. Netto, A. N. Toosi, M. A. Rodriguez, I. M. Llorente, S. De Capitani Di Vimercati, P. Samarati, D. Milojicic, C. Varela, R. Bahsoon, M. Dias De Assuncao, O. Rana, W. Zhou, H. Jin, W. Gentzsch, A. Y. Zomaya, and H. Shen, "A manifesto for future generation cloud computing: Research directions for the next decade," *ACM Computing Surveys*, vol. 51, no. 5, Jan. 2019.

[73] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, Jun. 2009.

[74] V. Yussupov, J. Soldani, and U. B. A. B. F. Leymann, "From Serverful to Serverless: A Spectrum of Patterns for Hosting Application Components," in *Proceedings of the 11th International Conference on Cloud Computing and Services Science (CLOSER 2021)*. SciTePress, May 2021, pp. 268–279.

[75] A. Mampage, S. Karunasekera, and R. Buyya, "A Holistic View on Resource Management in Serverless Computing Environments: Taxonomy and Future Directions," *ACM Computing Surveys*, vol. 54, no. 11s, pp. 1–36, Jan. 2022.

[76] H. Shafiei, A. Khonsari, and P. Mousavi, "Serverless Computing: A Survey of Opportunities, Challenges, and Applications," *ACM Computing Surveys*, pp. 1–31, 2022.

[77] V. Yussupov, J. Soldani, U. Breitenbücher, A. Brogi, and F. Leymann, "FaaSten your decisions: A classification framework and technology review of function-as-a-Service platforms," *Journal of Systems and Software*, vol. 175, May 2021.

[78] Q. Zhu, X. Yu, Y. Zhao, A. Nag, and J. Zhang, "Resource allocation in quantum-key-distribution- secured datacenter networks with cloud–edge collaboration," *IEEE Internet of Things Journal*, vol. 10, pp. 10 916–10 932, Jun. 2023.

[79] A. K. Singh, D. Saxena, J. Kumar, and V. Gupta, "A Quantum Approach Towards

the Adaptive Prediction of Cloud Workloads," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 12, pp. 2893–2905, Dec. 2021.

[80] H. Nejatollahi, N. Dutt, S. Ray, F. Regazzoni, I. Banerjee, and R. Cammarota, "Post-quantum lattice-based cryptography implementations," *ACM Computing Surveys*, vol. 51, pp. 1–41, Jan. 2019.

[81] E. Zeydan, J. Baranda, and J. Mangues-Bafalluy, "Post-quantum blockchain-based secure service orchestration in multi-cloud networks," *IEEE Access*, vol. 10, pp. 129 520–129 530, 2022.

[82] C. Duran, R. Carrasco, I. Soto, I. Galeas, J. Azocar, V. Pena, S. Lara-Salazar, and S. Gutierrez, "Quantum algorithms: applications, criteria and metrics," *Complex & Intelligent Systems*, vol. 9, pp. 6373–6392, May 2023.

[83] Z. Yang, H. Alfauri, B. Farkiani, R. Jain, R. D. Pietro, and A. Erbad, "A Survey and Comparison of Post-Quantum and Quantum Blockchains," *IEEE Communications Surveys & Tutorials*, vol. 26, no. 2, pp. 967–1002, 2024.

[84] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic Mapping Studies in Software Engineering," Jun. 2008.

[85] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*. London, United Kingdom: ACM, May 2014, pp. 1–10.

[86] F. Gemeinhardt, A. Garmendia, M. Wimmer, B. Weder, and F. Leymann, "Quantum Combinatorial Optimization in the NISQ Era: A Systematic Mapping Study," *ACM Computing Surveys*, vol. 56, no. 3, pp. 1–36, Mar. 2024.

[87] F. Truger, J. Barzen, M. Bechtold, M. Beisel, F. Leymann, A. Mandl, and V. Yussupov, "Warm-Starting and Quantum Computing: A Systematic Mapping Study," *ACM Computing Surveys*, vol. 56, no. 9, pp. 1–31, Oct. 2024.

[88] A. Singh, K. Dev, H. Siljak, H. D. Joshi, and M. Magarini, "Quantum Internet—Applications, Functionalities, Enabling Technologies, Challenges, and Research Directions," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2218–2247, 2021.

[89] J. Gambetta, "IBM's roadmap for scaling quantum technology," Sep. 2020. [Online]. Available: https://research.ibm.com/blog/ibm-quantum-roadmap

[90] J. Barzen, F. Leymann, M. Falkenthal, D. Vietz, B. Weder, and K. Wild, "Relevance of Near-Term Quantum Computing in the Cloud: A Humanities Perspective," *Communications in Computer and Information Science*, vol. 1399 CCIS, pp. 25 – 58, 2021.

[91] J. Rojo, E. Moguel, D. Valencia, J. Berrocal, J. García-Alonso, and J. M. Murillo, "Trials and tribulations of developing hybrid quantum-classical microservices systems," in *CEUR Workshop Proceedings*, vol. 3008, 2021, pp. 38–53, arXiv: 2105.04421.

[92] M. Caleffi, A. S. Cacciapuoti, and G. Bianchi, "Quantum internet: From communication to distributed computing," in *Proceedings of the 5th ACM International Conference on Nanoscale Computing and Communication, NANOCOM 2018*. ACM Press, Sep. 2018.

[93] S. Wehner, D. Elkouss, and R. Hanson, "Quantum internet: A vision for the road ahead," *Science*, vol. 362, no. 6412, p. eaam9288, Oct. 2018.

[94] Q. Zhu, Y. Zhao, X. Yu, and J. Zhang, "Quantum Cloud with Communication, Computing, Caching, and Cipher (4C) Resource Coordination," in *Proceedings of the 2023 International Conference on Optical Network Design and Modeling, ONDM 2023*, G. T, L.-L. D, M.-M. C, V. L, J. L, and M. P, Eds. Institute of Electrical and Electronics Engineers Inc., 2023, pp. 1–3.

[95] L. Nallamothula, "Quantum Ecosystem Development Using Advanced Cloud Services," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12845 LNCS, pp. 163 – 171, 2021.

[96] H. Singh and A. Sachdev, "The Quantum way of Cloud Computing," in *2014 International Conference on Reliability Optimization and Information Technology (ICROIT)*. Faridabad, Haryana, India: IEEE, Feb. 2014, pp. 397–400.

[97] S. J. Devitt, "Performing quantum computing experiments in the cloud," *Physical Review A*, vol. 94, no. 3, pp. 1–13, 2016.

[98] C. Gonzalez, "Cloud based QC with Amazon Braket," *Digitale Welt*, vol. 5, no. 2, pp. 14–17, Apr. 2021.

[99] A. R. R. Carvalho, H. Ball, M. J. Biercuk, M. R. Hush, and F. Thomsen, "Error-Robust Quantum Logic Optimization Using a Cloud Quantum Computer Interface," *Physical Review Applied*, vol. 15, no. 6, Jun. 2021.

[100] S. Seo and J. Bae, "Measurement Crosstalk Errors in Cloud-Based Quantum Computing," *IEEE Internet Computing*, vol. 26, no. 1, pp. 26–33, 2022.

[101] S. Resch and U. R. Karpuzcu, "Benchmarking Quantum Computers and the Impact of Quantum Noise," *ACM Computing Surveys*, vol. 54, no. 7, 2022.

[102] G. G. Guerreschi, J. Hogaboam, F. Baruffa, and N. P. D. Sawaya, "Intel Quantum Simulator: a cloud-ready high-performance simulator of quantum circuits," *Quantum Science and Technology*, vol. 5, no. 3, p. 034007, May 2020.

[103] N. P. de Leon, K. M. Itoh, D. Kim, K. K. Mehta, T. E. Northup, H. Paik, B. S. Palmer, N. Samarth, S. Sangtawesin, and D. W. Steuerman, "Materials challenges and opportunities for quantum computing hardware," *Science*, vol. 372, no. 6539, Apr. 2021.

[104] G. Li, A. Wu, Y. Shi, A. Javadi-Abhari, Y. Ding, and Y. Xie, "On the Co-Design of Quantum Software and Hardware," in *Proceedings of the Eight Annual ACM International Conference on Nanoscale Computing and Communication*, ser. NANOCOM '21. ACM, Sep. 2021.

[105] M. De Stefano, D. Di Nucci, F. Palomba, D. Taibi, and A. De Lucia, "Towards Quantum-algorithms-as-a-service," in *Proceedings of the 1st International Workshop*

on Quantum Programming for Software Engineering. Singapore Singapore: ACM, Nov. 2022, pp. 7–10.

[106] J. Obst, J. Barzen, M. Beisel, F. Leymann, M. Salm, and F. Truger, "Comparing Quantum Service Offerings," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*. Bellevue, WA, USA: IEEE, Sep. 2023, pp. 181–184.

[107] A. Ahmad, A. B. Altamimi, and J. Aqib, "A reference architecture for quantum computing as a service," *Journal of King Saud University - Computer and Information Sciences*, vol. 36, no. 6, p. 102094, Jul. 2024.

[108] J. Garcia-Alonso, J. Rojo, D. Valencia, E. Moguel, J. Berrocal, and J. M. Murillo, "Quantum Software as a Service Through a Quantum API Gateway," *IEEE Internet Computing*, vol. 26, no. 1, pp. 34–41, 2022.

[109] P. Dreher and M. Ramasami, "Prototype container-based platform for extreme quantum computing algorithm development," *2019 IEEE High Performance Extreme Computing Conference, HPEC 2019*, 2019.

[110] I. Kumara, W.-J. Van Den Heuvel, and D. A. Tamburri, "QSOC: Quantum Service-Oriented Computing," *Communications in Computer and Information Science*, vol. 1429 CCIS, pp. 52 – 63, 2021.

[111] K. A. Britt and T. S. Humble, "High-Performance Computing with Quantum Processing Units," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 13, no. 3, pp. 1–13, Jul. 2017.

[112] T. Pfandzelter, A. Dhakal, Frachtenberg, D. Emmot, N. Hogade, R. P. H. Enriquez, G. Rattihalli, D. Bermbach, and D. Milojicic, "Kernel-as-a-Service: A Serverless Programming Model for Heterogeneous Hardware Accelerators," in *Proceedings of the 24th International Middleware Conference*, ser. Middleware '23. New York, NY, USA: Association for Computing Machinery, 2023, pp. 192–206.

[113] N. Maring, A. Fyrillas, M. Pont, E. Ivanov, E. Bertasi, M. Valdivia, and J. Senellart, "One Nine Availability of a Photonic Quantum Computer on the Cloud Toward

HPC Integration," in *Proceedings - 2023 IEEE International Conference on Quantum Computing and Engineering, QCE 2023*, M. H, A. Y, D. A, and B. G, Eds., vol. 2. IEEE Press, 2023, pp. 112 – 116.

[114] IBM Quantum, "IBM Quantum Development Roadmap 2022," 2022. [Online]. Available: https://www.ibm.com/quantum/roadmap

[115] M. Yarter, G. Uehara, and A. Spanias, "Investigating a Quantum Cloud Paradigm with Quantum Neural Networks," in *2023 IEEE 66th International Midwest Symposium on Circuits and Systems (MWSCAS)*, Aug. 2023, pp. 900–903, iSSN: 1558-3899.

[116] P. J. J. O'Malley, R. Babbush, I. D. Kivlichan, Romero, R. Barends, J. Kelly, P. Roushan, A. Tranter, N. Ding, B. Campbell, Y. Chen, Z. Chen, Chiaro, A. G. Fowler, E. Jeffrey, E. Lucero, A. Megrant, J. Y. Mutus, M. Neeley, C. Neill, C. Quintana, D. Sank, A. Vainsencher, J. Wenner, T. C. White, Coveney, H. Neven, A. Aspuru-Guzik, and J. M. Martinis, "Scalable Quantum Simulation of Molecular Energies," *Phys. Rev. X*, vol. 6, no. 3, p. 031007, Jul. 2016.

[117] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," *Nature*, vol. 549, no. 7671, pp. 242–246, Sep. 2017.

[118] A. Rossi, P. G. Baity, V. M. Schäfer, and M. Weides, "Quantum computing hardware in the cloud: Should a computational chemist care?" *International Journal of Quantum Chemistry*, vol. 121, no. 14, May 2021.

[119] Y. Kim, A. Eddins, S. Anand, K. X. Wei, E. Van Den Berg, S. Rosenblatt, H. Nayfeh, Y. Wu, M. Zaletel, K. Temme, and A. Kandala, "Evidence for the utility of quantum computing before fault tolerance," *Nature*, vol. 618, no. 7965, pp. 500–505, Jun. 2023.

[120] X.-q. Fang and R.-h. Shi, "Cloud-assisted quantum primitive protocols and applications," *Physica Scripta*, vol. 98, no. 9, p. 095114, Aug. 2023.

[121] C. Gong, J. Du, Z. Dong, Z. Guo, A. Gani, L. Zhao, and H. Qi, "Grover algorithm-based quantum homomorphic encryption ciphertext retrieval scheme in quantum cloud computing," *Quantum Information Processing*, vol. 19, no. 3, p. 105, Mar. 2020.

[122] A. EL Azzaoui, P. K. Sharma, and J. H. Park, "Blockchain-based delegated Quantum Cloud architecture for medical big data security," *Journal of Network and Computer Applications*, vol. 198, p. 103304, Feb. 2022.

[123] V. S. Barletta, D. Caivano, A. Lako, and A. Pal, "Quantum as a Service Architecture for Security in a Smart City," *Communications in Computer and Information Science*, vol. 1871 CCIS, pp. 76 – 89, 2023.

[124] Y. Zhang, Y.-x. Bian, Q. Fan, and J. Chen, "Quantum Solution for the 3-SAT Problem Based on IBM Q," *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, vol. 322 LNICST, pp. 410 – 423, 2020.

[125] V. Hassija, V. Chamola, A. Goyal, S. S. Kanhere, and N. Guizani, "Forthcoming applications of quantum computing: peeking into the future," *IET Quantum Communication*, vol. 1, no. 2, pp. 35–41, Dec. 2020.

[126] G. Cloud, "IonQ quantum computer available through Google Cloud," 2024. [Online]. Available: https://cloud.google.com/blog/products/compute/ionq-quantum-computer-available-through-google-cloud

[127] Aqt, "AQT realizes the first general-purpose quantum computer," 2024. [Online]. Available: https://www.aqt.eu/

[128] A. Computing, "Atom Computing," 2024. [Online]. Available: https://atom-computing.com/

[129] D.-W. System, "D-Wave Systems Leap - Quantum Computing Service," 2022. [Online]. Available: https://www.dwavesys.com/solutions-and-products/cloud-platform/

[130] Google, "Google Quantum AI," 2024. [Online]. Available: https://quantumai.google/

[131] OCQ, "OxfordQuantumCircuits," 2024. [Online]. Available: https://oxfordquantumcircuits.com/

[132] Pasqal, "Programmable atomic arrays - PASQAL," 2024. [Online]. Available: https://www.pasqal.com/

[133] QCI, "Dirac-3 - Quantum Computing Inc." [Online]. Available: https://quantumcomputinginc.com/offerings/quantum-computing/dirac-3

[134] Quantinuum, "Quantinuum Cloud Service," 2022. [Online]. Available: https://www.quantinuum.com/

[135] QuEra, "Quantum computing with neutral atoms - QuEra," 2024. [Online]. Available: https://www.quera.com/

[136] D. Voorhoede, "Quantum Inspire," 2024. [Online]. Available: https://www.quantum-inspire.com/

[137] Xanadu, "Xanadu Quantum Cloud Service," 2024. [Online]. Available: https://www.xanadu.ai/

[138] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, and W. D. Oliver, "A quantum engineer's guide to superconducting qubits," *Applied Physics Reviews*, vol. 6, no. 2, p. 021318, Jun. 2019.

[139] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell, B. Burkett, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, A. Dunsworth, E. Farhi, B. Foxen, A. Fowler, C. Gidney, M. Giustina, R. Graff, K. Guerin, S. Habegger, M. P. Harrigan, M. J. Hartmann, A. Ho, M. Hoffmann, T. Huang, T. S. Humble, S. V. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, J. Kelly, P. V. Klimov, S. Knysh, A. Korotkov, F. Kostritsa, D. Landhuis, M. Lindmark, E. Lucero, D. Lyakh, S. Mandrà, J. R. McClean, M. McEwen, A. Megrant, X. Mi, K. Michielsen, M. Mohseni, J. Mutus, O. Naaman, M. Neeley, C. Neill, M. Y. Niu, E. Ostby, A. Petukhov, J. C. Platt, C. Quintana, E. G. Rieffel, P. Roushan, N. C. Rubin, D. Sank, K. J. Satzinger, V. Smelyanskiy, K. J. Sung, M. D. Trevithick, A. Vainsencher, B. Villalonga, T. White, Z. J. Yao, P. Yeh,

A. Zalcman, H. Neven, and J. M. Martinis, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, Oct. 2019.

[140] J. J. Pla, K. Y. Tan, J. P. Dehollain, W. H. Lim, J. J. L. Morton, D. N. Jamieson, A. S. Dzurak, and A. Morello, "A single-atom electron spin qubit in silicon," *Nature*, vol. 489, no. 7417, pp. 541–545, Sep. 2012.

[141] E. Vahapoglu, J. P. Slack-Smith, R. C. C. Leon, W. H. Lim, F. E. Hudson, T. Day, J. D. Cifuentes, T. Tanttu, C. H. Yang, A. Saraiva, N. V. Abrosimov, H.-J. Pohl, M. L. W. Thewalt, A. Laucht, A. S. Dzurak, and J. J. Pla, "Coherent control of electron spin qubits in silicon using a global field," *npj Quantum Information*, vol. 8, no. 1, Nov. 2022.

[142] S. Slussarenko and G. J. Pryde, "Photonic quantum information processing: A concise review," *Applied Physics Reviews*, vol. 6, no. 4, Oct. 2019.

[143] E. T. Campbell, "Enhanced Fault-Tolerant Quantum Computing in d-Level Systems," *Physical Review Letters*, vol. 113, no. 23, Dec. 2014.

[144] M. Ringbauer, M. Meth, L. Postler, R. Stricker, R. Blatt, P. Schindler, and T. Monz, "A universal qudit quantum processor with trapped ions," *Nature Physics*, vol. 18, no. 9, pp. 1053–1057, Jul. 2022.

[145] J. M. Arrazola, V. Bergholm, K. Brádler, T. R. Bromley, M. J. Collins, I. Dhand, A. Fumagalli, T. Gerrits, A. Goussev, L. G. Helt, J. Hundal, T. Isacsson, R. B. Israel, J. Izaac, S. Jahangiri, R. Janik, N. Killoran, S. P. Kumar, J. Lavoie, A. E. Lita, D. H. Mahler, M. Menotti, B. Morrison, S. W. Nam, L. Neuhaus, H. Y. Qi, N. Quesada, A. Repingon, K. K. Sabapathy, M. Schuld, D. Su, J. Swinarton, A. Száva, K. Tan, P. Tan, V. D. Vaidya, Z. Vernon, Z. Zabaneh, and Y. Zhang, "Quantum circuits with many photons on a programmable nanophotonic chip," *Nature*, vol. 591, no. 7848, pp. 54–60, Mar. 2021.

[146] 1Qcloud, "1Qloud optimization platform - 1qbit," 2024. [Online]. Available: https://1qbit.com/1qloud/

[147] IBM, "IBM Cloud," 2024. [Online]. Available: https://www.ibm.com/cloud

[148] C. Linnhoff-Popien, "PlanQK — Quantum Computing Meets Artificial Intelligence: How to make an ambitious idea reality," *Digitale Welt*, vol. 4, no. 2, pp. 28–35, Apr. 2020.

[149] Q. Cloud, "QEMIST cloud – good chemistry co," 2024. [Online]. Available: https://goodchemistry.com/qemist-cloud/

[150] Strangeworks, "Strangeworks quantum computing platform," 2022. [Online]. Available: https://app.quantumcomputing.com/

[151] Z. Ai, "Orquestra platform — zapata.ai," 2023. [Online]. Available: https://zapata.ai/platform-orquestra/

[152] B. Weder, U. Breitenbucher, F. Leymann, and K. Wild, "Integrating Quantum Computing into Workflow Modeling and Execution," in *2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC)*. Leicester, UK: IEEE, Dec. 2020, pp. 279–291.

[153] T. Xin, S. Huang, S. Lu, K. Li, Z. Luo, Z. Yin, J. Li, D. Lu, G. Long, and B. Zeng, "NMRCloudQ: a quantum cloud experience on a nuclear magnetic resonance quantum computer," *Science Bulletin*, vol. 63, no. 1, pp. 17–23, Jan. 2018.

[154] S. Blinov, B. Wu, and C. Monroe, "Comparison of cloud-based ion trap and superconducting quantum computer architectures," *AVS Quantum Science*, vol. 3, no. 3, 2021.

[155] G. S. Ravi, K. N. Smith, P. Gokhale, and F. T. Chong, "Quantum Computing in the Cloud: Analyzing job and machine characteristics," in *Proceedings of the 2021 IEEE International Symposium on Workload Characterization (IISWC)*, Storrs, CT, USA, Nov. 2021, pp. 39–50.

[156] M. Salm, J. Barzen, U. Breitenbücher, F. Leymann, B. Weder, and K. Wild, "The NISQ analyzer: automating the selection of quantum computers for quantum algorithms," in *Service-oriented Computing*, S. Dustdar, Ed. Cham: Springer Inter-

national Publishing, 2020, vol. 1310, pp. 66–85, series Title: Communications in Computer and Information Science.

[157] C. Cicconetti, M. Conti, and A. Passarella, "Resource Allocation in Quantum Networks for Distributed Quantum Computing," in *Proceedings of the 2022 IEEE International Conference on Smart Computing (SMARTCOMP)*. Helsinki, Finland: IEEE, Jun. 2022, pp. 124–132.

[158] M. Zhang, Y. Fu, J. Wang, and J. Lai, "Research on task scheduling scheme for quantum computing cloud platform," in *Proceedings of the 2022 6th International Conference on Cloud and Big Data Computing*. Birmingham United Kingdom: ACM, Aug. 2022, pp. 7–11.

[159] C. Cicconetti, M. Conti, and A. Passarella, "Service differentiation and fair sharing in distributed quantum computing," *Pervasive and Mobile Computing*, vol. 90, p. 101758, Mar. 2023.

[160] T. Li and Z. Zhao, "Moirai: optimizing quantum serverless function orchestration via device allocation and circuit deployment," in *2024 IEEE International Conference on Web Services (ICWS)*, Jul. 2024, pp. 707–717.

[161] S. Niu and A. Todri-Sanial, "How Parallel Circuit Execution Can Be Useful for NISQ Computing?" in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, Mar. 2022, pp. 1065–1070, arXiv: 2112.00387.

[162] P. Das, S. S. Tannu, P. J. Nair, and M. Qureshi, "A case for multi-programming quantum computers," in *Proceedings of the Annual International Symposium on Microarchitecture, MICRO*. IEEE Computer Society, Oct. 2019, pp. 291–303, iSSN: 10724451.

[163] Y. Ohkura, T. Satoh, and R. Van Meter, "Simultaneous Execution of Quantum Circuits on Current and Near-Future NISQ Systems," *IEEE Transactions on Quantum Engineering*, vol. 3, pp. 1–10, 2022.

[164] T. Nguyen, D. Arya, M. Doherty, N. Herrmann, J. Kuhlmann, F. Preis, P. Scott, and S. Yin, "Software for massively parallel quantum computing," Dec. 2022,

arXiv:2211.13355 [quant-ph]. [Online]. Available: http://arxiv.org/abs/2211.13355

[165] J. Alvarado-Valiente, J. Romero-Álvarez, E. Moguel, J. García-Alonso, and J. M. Murillo, "Orchestration for quantum services: The power of load balancing across multiple service providers," *Science of Computer Programming*, vol. 237, p. 103139, Oct. 2024.

[166] H. Chen, H. Jia, X. Wu, X. Wang, and M. Wang, "Quantum Token for Network Authentication," *Proceedings - 2021 IEEE International Conference on Web Services, ICWS 2021*, pp. 688–692, 2021.

[167] V. Lipinska, J. Ribeiro, and S. Wehner, "Secure multiparty quantum computation with few qubits," *Physical Review A*, vol. 102, no. 2, pp. 1–15, 2020.

[168] P. Wallden and E. Kashefi, "Cyber security in the quantum era," *Communications of the ACM*, vol. 62, no. 4, pp. 120–120, Mar. 2019.

[169] Y. Baseri, V. Chouhan, and A. Ghorbani, "Cybersecurity in the quantum era: assessing the impact of quantum computing on infrastructure," Apr. 2024, arXiv:2404.10659 [cs]. [Online]. Available: http://arxiv.org/abs/2404.10659

[170] D. J. Bernstein and T. Lange, "Post-quantum cryptography," *Nature*, vol. 549, no. 7671, pp. 188–194, Sep. 2017.

[171] J. A. Buchmann, D. Butin, F. Göpfert, and A. Petzoldt, "Post-quantum cryptography: state of the art," in *The New Codebreakers*, P. Y. A. Ryan, D. Naccache, and J.-J. Quisquater, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, vol. 9100, pp. 88–108, series Title: Lecture Notes in Computer Science.

[172] Z. Iftikhar, M. Iftikhar, and M. A. Shah, "Quantum Safe Cloud Computing Using Hash-based Digital Signatures," in *2021 26th International Conference on Automation and Computing (ICAC)*. Portsmouth, United Kingdom: IEEE, Sep. 2021, pp. 1–6.

[173] B. Hrdá and S. Wessel, "Confidential Quantum Computing," in *Proceedings of the 18th International Conference on Availability, Reliability and Security*. Benevento Italy: ACM, Aug. 2023, pp. 1–10.

[174] S. Ghosh, S. Upadhyay, and A. A. Saki, "A primer on security of quantum computing hardware," Apr. 2025, arXiv:2305.02505 [quant-ph]. [Online]. Available: http://arxiv.org/abs/2305.02505

[175] A. Singh and K. Chatterjee, "Cloud security issues and challenges: A survey," *Journal of Network and Computer Applications*, vol. 79, pp. 88–115, Feb. 2017.

[176] A. A. Saki, A. Suresh, R. O. Topaloglu, and S. Ghosh, "Split Compilation for Security of Quantum Circuits," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2021, pp. 1–7.

[177] B. Harper, B. Tonekaboni, B. Goldozian, M. Sevior, and M. Usman, "Crosstalk attacks and defence in a shared quantum computing environment," 2024, version Number: 1. [Online]. Available: https://arxiv.org/abs/2402.02753

[178] S. Upadhyay and S. Ghosh, "Robust and Secure Hybrid Quantum-Classical Computation on Untrusted Cloud-Based Quantum Hardware," in *Proceedings of the 11th International Workshop on Hardware and Architectural Support for Security and Privacy*, ser. HASP '22. ACM, Oct. 2022.

[179] H.-L. Huang, Q. Zhao, X. Ma, C. Liu, Z.-E. Su, X.-L. Wang, L. Li, N.-L. Liu, Sanders, and others, "Experimental blind quantum computing for a classical client," *Physical review letters*, vol. 119, no. 5, p. 050503, 2017.

[180] J. Li, Y. Zhao, Y. Yang, and Y. Lin, "Verifiable quantum cloud computation scheme based on blind computation," *IEEE Access*, vol. 8, pp. 56 921–56 926, 2020.

[181] X. Chen, B. Cheng, Z. Li, X. Nie, N. Yu, M.-H. Yung, and X. Peng, "Experimental cryptographic verification for near-term quantum cloud computing," *Science Bulletin*, vol. 66, no. 1, pp. 23–28, Jan. 2021.

[182] Y. Ma, E. Kashefi, M. Arapinis, K. Chakraborty, and M. Kaplan, "QEnclave - A practical solution for secure quantum cloud computing," *npj Quantum Information*, vol. 8, no. 1, p. 128, Nov. 2022.

[183] J. L. L. Huang and V. C. Emeakaroha, "Performing Distributed Quantum Calculations in a Multi-cloud Architecture Secured by the Quantum Key Distribution Protocol," *SN Computer Science*, vol. 5, no. 4, 2024.

[184] K. Phalak, A. A. Saki, M. Alam, R. O. Topaloglu, and S. Ghosh, "Quantum PUF for Security and Trust in Quantum Computing," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 11, no. 2, pp. 333–342, Jun. 2021.

[185] S. S. Tannu and M. Qureshi, "Ensemble of diverse mappings: Improving reliability of quantum computers by orchestrating dissimilar mistakes," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2019, pp. 253–265.

[186] Z. Wang, Y. Sheng, N. Koirala, K. Basu, T. Jung, C.-C. Lu, and W. Jiang, "PristiQ: A Co-Design Framework for Preserving Data Security of Quantum Learning in the Cloud," in *Proceedings of IEEE Computer Society Annual Symposium on VLSI, ISVLSI*, T. H and B. J, Eds. IEEE Computer Society, 2024, pp. 403 – 408.

[187] N. Acharya and S. M. Saeed, "A Lightweight Approach to Detect Malicious/Unexpected Changes in the Error Rates of NISQ Computers," in *Proceedings of the 39th International Conference on Computer-Aided Design*, ser. ICCAD '20. New York, NY, USA: Association for Computing Machinery, 2020, event-place: Virtual Event, USA.

[188] B. Kitchenham and P. Brereton, "A systematic review of systematic review process research in software engineering," *Information and Software Technology*, vol. 55, no. 12, pp. 2049–2075, Dec. 2013.

[189] J. Quan, Q. Li, C. Liu, J. Shi, and Y. Peng, "A simplified verifiable blind quantum computing protocol with quantum input verification," *Quantum Engineering*, vol. 3, no. 1, pp. 1–10, 2021.

[190] P. Griffin and R. Sampat, "Quantum Computing for Supply Chain Finance," in *Proceedings of the 2021 IEEE International Conference on Services Computing (SCC)*, Chicago, IL, USA, Sep. 2021, pp. 456–459.

[191] D. Deutsch and R. Jozsa, "Rapid solution of problems by quantum computation," *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, vol. 439, no. 1907, pp. 553–558, Dec. 1992.

[192] M. A. Serrano, R. Pérez-Castillo, and M. Piattini, Eds., *Quantum Software Engineering*. Cham: Springer International Publishing, 2022.

[193] F. Gemeinhardt, A. Garmendia, and M. Wimmer, "Towards Model-Driven Quantum Software Engineering," in *2021 IEEE/ACM 2nd International Workshop on Quantum Software Engineering (Q-SE)*. Madrid, Spain: IEEE, Jun. 2021, pp. 13–15.

[194] IBM Quantum, *Qiskit Textbook*. Online: IBM, 2022.

[195] G. Aleksandrowicz, T. Alexander, P. Barkoutsos, L. Bello, Y. Ben-Haim, D. Bucher, F. J. Cabrera-Hernández, and et. al., "Qiskit: An Open-source Framework for Quantum Computing," Jan. 2019. [Online]. Available: https://zenodo.org/record/2562111

[196] Cirq Developers, "Cirq Framework," 2023. [Online]. Available: https://zenodo.org/record/5182845

[197] Microsoft, "Q# Quantum Programming Language," 2021. [Online]. Available: https://github.com/microsoft/qsharp-language

[198] R. S. Smith, M. J. Curtis, and W. J. Zeng, "A Practical Quantum Instruction Set Architecture," Tech. Rep., Aug. 2016, arXiv: 1608.03355. [Online]. Available: http://arxiv.org/abs/1608.03355

[199] N. Killoran, J. Izaac, N. Quesada, V. Bergholm, M. Amy, and C. Weedbrook, "Strawberry Fields: A Software Platform for Photonic Quantum Computing," *Quantum*, vol. 3, p. 129, Mar. 2019.

[200] V. Bergholm, J. Izaac, M. Schuld, and et. al., "PennyLane: Automatic differentiation of hybrid quantum-classical computations," Tech. Rep., Nov. 2018, arXiv: 1811.04968. [Online]. Available: http://arxiv.org/abs/1811.04968

[201] X. Fu, J. Yu, X. Su, H. Jiang, H. Wu, and et. al., "Quingo: A Programming Framework for Heterogeneous Quantum-Classical Computing with NISQ Features," *ACM Transactions on Quantum Computing*, vol. 2, no. 4, pp. 1–37, 2021.

[202] D. Ittah, T. Häner, V. Kliuchnikov, and T. Hoefler, "QIRO: A Static Single Assignment-based Quantum Program Representation for Optimization," *ACM Transactions on Quantum Computing*, vol. (Just Accepted), pp. 1–29, Feb. 2022.

[203] A. Mccaskey, T. Nguyen, A. Santana, D. Claudino, T. Kharazi, and H. Finkel, "Extending C++ for Heterogeneous Quantum-Classical Computing," *ACM Transactions on Quantum Computing*, vol. 2, no. 2, pp. 1–36, Jul. 2021.

[204] J. Schleier-Smith, V. Sreekanti, A. Khandelwal, J. Carreira, N. J. Yadwadkar, R. A. Popa, J. E. Gonzalez, I. Stoica, and D. A. Patterson, "What serverless computing is and should become," *Communications of the ACM*, vol. 64, no. 5, pp. 76–84, 2021.

[205] S. Eismann, J. Scheuner, E. van Eyk, M. Schwinger, J. Grohmann, N. Herbst, C. L. Abad, and A. Iosup, "Serverless Applications: Why, When, and How?" *IEEE Software*, vol. 38, no. 1, pp. 32–39, Jan. 2021.

[206] J. Scheuner and P. Leitner, "Function-as-a-Service performance evaluation: A multivocal literature review," *Journal of Systems and Software*, vol. 170, p. 110708, 2020.

[207] M. Cerezo, G. Verdon, H.-Y. Huang, L. Cincio, and P. J. Coles, "Challenges and opportunities in quantum machine learning," *Nature Computational Science*, vol. 2, no. 9, pp. 567–576, Sep. 2022.

[208] A. Cross, L. Bishop, J. Smolin, and J. Gambetta, "Open Quantum Assembly Language," Tech. Rep., Jul. 2017, arXiv: 1707.03429. [Online]. Available: http://arxiv.org/abs/1707.03429

[209] P. Leitner, E. Wittern, J. Spillner, and W. Hummer, "A mixed-method empirical study of Function-as-a-Service software development in industrial practice," *Journal of Systems and Software*, vol. 149, pp. 340–359, Mar. 2019.

[210] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, "DevOps," *IEEE Software*, vol. 33, no. 3, pp. 94–100, May 2016.

[211] P. C. Lotshaw, T. Nguyen, A. Santana, A. McCaskey, R. Herrman, J. Ostrowski, G. Siopsis, and T. S. Humble, "Scaling quantum approximate optimization on near-term hardware," *Scientific Reports*, vol. 12, no. 1, p. 12388, Jul. 2022.

[212] M. Van Steen and A. Tanenbaum, *Distributed systems, 4th edition*, 2023.

[213] V. Giménez-Alventosa, G. Moltó, and M. Caballer, "A framework and a performance assessment for serverless MapReduce on AWS Lambda," *Future Generation Computer Systems*, vol. 97, pp. 259–274, 2019.

[214] E. Younis and C. Iancu, "Quantum Circuit Optimization and Transpilation via Parameterized Circuit Instantiation," Jun. 2022, arXiv:2206.07885. [Online]. Available: http://arxiv.org/abs/2206.07885

[215] M. Golec, G. K. Walia, M. Kumar, F. Cuadrado, S. S. Gill, and S. Uhlig, "Cold Start Latency in Serverless Computing: A Systematic Review, Taxonomy, and Future Directions," Oct. 2023, arXiv:2310.08437 [cs]. [Online]. Available: http://arxiv.org/abs/2310.08437

[216] A. Ellis, "OpenFaaS - Serverless Functions Made Simple," 2022. [Online]. Available: https://github.com/openfaas/faas

[217] M. Herrero-Collantes and J. C. Garcia-Escartin, "Quantum random number generators," *Reviews of Modern Physics*, vol. 89, no. 1, p. 15004, Feb. 2017.

[218] D. M. Greenberger, "GHZ (Greenberger—Horne—Zeilinger) Theorem and GHZ States," in *Compendium of Quantum Physics*, D. Greenberger, K. Hentschel, and F. Weinert, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 258–263.

[219] D. Claudino, A. J. McCaskey, and D. I. Lyakh, "A backend-agnostic, quantum-classical framework for simulations of chemistry in C ++," *ACM Transactions on Quantum Computing*, 2022.

[220] S. Sivarajah, S. Dilkes, A. Cowtan, W. Simmons, A. Edgington, and R. Duncan, "t|ket: a retargetable compiler for NISQ devices," *Quantum Science and Technology*, vol. 6, no. 1, Jan. 2021.

[221] S. Sim, Y. Cao, J. Romero, P. D. Johnson, and A. Aspuru-Guzik, "A framework for algorithm deployment on cloud-based quantum computers," Tech. Rep., Oct. 2018, arXiv: 1810.10576. [Online]. Available: http://arxiv.org/abs/1810.10576

[222] M. Grossi, L. Crippa, A. Aita, G. Bartoli, V. Sammarco, E. Picca, N. Said, F. Tramonto, and F. Mattei, "A Serverless Cloud Integration For Quantum Computing," Tech. Rep., Jul. 2021, arXiv: 2107.02007. [Online]. Available: http://arxiv.org/abs/2107.02007

[223] N. Moll, P. Barkoutsos, L. S. Bishop, J. M. Chow, A. Cross, D. J. Egger, S. Filipp, A. Fuhrer, J. M. Gambetta, M. Ganzhorn, A. Kandala, A. Mezzacapo, P. Müller, W. Riess, G. Salis, J. Smolin, I. Tavernelli, and K. Temme, "Quantum optimization using variational algorithms on near-term quantum devices," *Quantum Science and Technology*, vol. 3, no. 3, p. 030503, Jul. 2018.

[224] E. P. Debenedictis, "A Future with Quantum Machine Learning," *Computer*, vol. 51, no. 2, pp. 68–71, Feb. 2018.

[225] R. S. K. V and N. P. Kavya, "A Review on Progress and Problems of Quantum Computing as a Service (QCaaS) in the Perspective of Cloud Computing," *Global Journal of Computer Science and Technology: B Cloud and Distributed*, vol. 15, no. 4, 2015.

[226] L. Ma and L. Ding, "Hybrid quantum edge computing network," in *Quantum Communications and Quantum Imaging XX*. San Diego, United States: SPIE, Oct. 2022, p. 29.

[227] A. Furutanpey, T. Vienna, and J. Barzen, "Architectural Vision for Quantum Computing in the Edge-Cloud Continuum," in *Proceedings of the 2nd IEEE International Conference on Quantum Software (QSW)*. Chicago, IL, USA: IEEE, Jul. 2023, pp. 88–103.

[228] H. Wang, T. Liu, B. Kim, C.-W. Lin, S. Shiraishi, J. Xie, and Z. Han, "Architectural Design Alternatives Based on Cloud/Edge/Fog Computing for Connected

Vehicles," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2349–2377, 2020.

[229] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge Computing: Vision and Challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[230] X. Qiu, W. Zhang, W. Chen, and Z. Zheng, "Distributed and Collective Deep Reinforcement Learning for Computation Offloading: A Practical Perspective," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 5, pp. 1085–1101, May 2021.

[231] C. Sonmez, A. Ozgovde, and C. Ersoy, "EdgeCloudSim: An environment for performance evaluation of edge computing systems: EdgeCloudSim: An environment for performance evaluation of edge computing systems," *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 11, p. e3493, Nov. 2018.

[232] T. Qayyum, A. W. Malik, M. A. Khan Khattak, O. Khalid, and S. U. Khan, "FogNetSim++: A Toolkit for Modeling and Simulation of Distributed Fog Environment," *IEEE Access*, vol. 6, pp. 63 570–63 583, 2018.

[233] E. Altman, K. R. Brown, G. Carleo, L. D. Carr, E. Demler, C. Chin, B. DeMarco, S. E. Economou, M. A. Eriksson, K.-M. C. Fu, M. Greiner, K. R. Hazzard, R. G. Hulet, A. J. Kollár, B. L. Lev, M. D. Lukin, R. Ma, X. Mi, S. Misra, C. Monroe, K. Murch, Z. Nazario, K.-K. Ni, A. C. Potter, P. Roushan, M. Saffman, M. Schleier-Smith, I. Siddiqi, R. Simmonds, M. Singh, I. Spielman, K. Temme, D. S. Weiss, J. Vučković, V. Vuletić, J. Ye, and M. Zwierlein, "Quantum Simulators: Architectures and Opportunities," *PRX Quantum*, vol. 2, no. 1, p. 017003, Feb. 2021.

[234] S. Diadamo, J. Notzel, B. Zanger, and M. M. Bese, "QuNetSim: A Software Framework for Quantum Networks," *IEEE Transactions on Quantum Engineering*, vol. 2, pp. 1–12, 2021.

[235] T. Coopmans, R. Knegjens, A. Dahlberg, D. Maier, L. Nijsten, J. de Oliveira Filho, M. Papendrecht, J. Rabbie, F. Rozpedek, M. Skrzypczyk, L. Wubben, W. de Jong,

D. Podareanu, A. Torres-Knoop, D. Elkouss, and S. Wehner, "NetSquid, a NETwork Simulator for QUantum Information using Discrete events," *Communications Physics*, vol. 4, no. 1, 2021.

[236] G. Dagkakis and C. Heavey, "A review of open source discrete event simulation software for operations research," *Journal of Simulation*, vol. 10, no. 3, pp. 193–206, Aug. 2016.

[237] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (IoT): a vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013.

[238] C. Puliafito, E. Mingozzi, F. Longo, A. Puliafito, and O. Rana, "Fog computing for the internet of things: a survey," *ACM Transactions on Internet Technology*, vol. 19, no. 2, pp. 1–41, May 2019.

[239] M. Laroui, B. Nour, H. Moungla, M. A. Cherif, H. Afifi, and M. Guizani, "Edge and fog computing for IoT: a survey on current research activities & future directions," *Computer Communications*, vol. 180, pp. 210–231, Dec. 2021.

[240] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: a toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, Sep. 2017.

[241] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized Computation Offloading Performance in Virtual Edge Computing Systems Via Deep Reinforcement Learning," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4005–4018, Jun. 2019.

[242] L. Sun, X. Jiang, H. Ren, and Y. Guo, "Edge-cloud computing and artificial intelligence in internet of medical things: architecture, technology and application," *IEEE Access*, vol. 8, pp. 101 079–101 092, 2020.

[243] G. Qu, N. Cui, H. Wu, R. Li, and Y. Ding, "ChainFL: a simulation platform for

joint federated learning and blockchain in edge/cloud computing environments," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 5, pp. 3572–3581, May 2022.

[244] A. W. Malik, T. Qayyum, A. U. Rahman, M. A. Khan, O. Khalid, and S. U. Khan, "xFogSim: a distributed fog resource management framework for sustainable IoT services," *IEEE Transactions on Sustainable Computing*, vol. 6, no. 4, pp. 691–702, Oct. 2021.

[245] T. Jones, A. Brown, I. Bush, and S. C. Benjamin, "QuEST and High Performance Simulation of Quantum Computers," *Scientific Reports*, vol. 9, no. 1, p. 10736, Jul. 2019.

[246] H. Bian, J. Huang, J. Tang, R. Dong, L. Wu, and X. Wang, "PAS: A new powerful and simple quantum computing simulator," *Software: Practice and Experience*, vol. 53, no. 1, pp. 142–159, 2023.

[247] J. Brennan, L. O'Riordan, K. Hanley, M. Doyle, M. Allalen, D. Brayford, L. Iapichino, and N. Moran, "QXTools: A Julia framework for distributed quantum circuit simulation," *Journal of Open Source Software*, vol. 7, no. 70, p. 3711, Feb. 2022.

[248] A. Dahlberg and S. Wehner, "SimulaQron - A simulator for developing quantum internet software," *Quantum Science and Technology*, vol. 4, no. 1, pp. 0–15, 2019.

[249] D. S. Steiger, T. Häner, and M. Troyer, "ProjectQ: An Open Source Software Framework for Quantum Computing," *Quantum*, vol. 2, p. 49, Jan. 2018.

[250] J. Johansson, P. Nation, and F. Nori, "QuTiP: An open-source Python framework for the dynamics of open quantum systems," *Computer Physics Communications*, vol. 183, no. 8, pp. 1760–1772, Aug. 2012.

[251] G. F. Riley and T. R. Henderson, "The ns-3 Network Simulator," in *Modeling and Tools for Network Simulation*, K. Wehrle, M. Güneş, and J. Gross, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 15–34.

[252] A. Varga, "OMNeT++," in *Modeling and Tools for Network Simulation*, K. Wehrle, M. Güneş, and J. Gross, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 35–59.

[253] W. Tang, T. Tomesh, M. Suchara, J. Larson, and M. Martonosi, "CutQC: using small Quantum computers for large Quantum circuit evaluations," in *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, Virtual USA, Apr. 2021, pp. 473–486.

[254] C. Aravanis, G. Korpas, and J. Marecek, "Transpiling Quantum Circuits using the Pentagon Equation," Sep. 2022, arXiv:2209.14356 [quant-ph]. [Online]. Available: http://arxiv.org/abs/2209.14356

[255] B. Weder, J. Barzen, F. Leymann, and D. Vietz, "Quantum Software Development Lifecycle," in *Quantum Software Engineering*, M. A. Serrano, R. Pérez-Castillo, and M. Piattini, Eds. Cham: Springer International Publishing, 2022, pp. 61–83.

[256] A. Anand, J. Romero, M. Degroote, and A. Aspuru-Guzik, "Noise Robustness and Experimental Demonstration of a Quantum Generative Adversarial Network for Continuous Distributions," *Advanced Quantum Technologies*, vol. 4, no. 5, pp. 1–11, 2021.

[257] D. Mills, S. Sivarajah, T. L. Scholten, and R. Duncan, "Application-Motivated, Holistic Benchmarking of a Full Quantum Computing Stack," *Quantum*, vol. 5, p. 415, Mar. 2021.

[258] E. Pelofske, A. Bärtschi, and S. Eidenbenz, "Quantum Volume in Practice: What Users Can Expect From NISQ Devices," *IEEE Transactions on Quantum Engineering*, vol. 3, pp. 1–19, 2022.

[259] C. A. Waldspurger and W. E. Weihl, "Lottery Scheduling: Flexible Proportional-Share Resource Management," in *Proceedings of the 1st USENIX Conference on Operating Systems Design and Implementation*, pp. 1–es.

[260] Y. Kim, C. J. Wood, T. J. Yoder, S. T. Merkel, J. M. Gambetta, K. Temme, and A. Kan-

dala, "Scalable error mitigation for noisy quantum circuits produces competitive expectation values," *Nature Physics*, Feb. 2023.

[261] "Australian National Quantum Strategy," May 2023. [Online]. Available: https://www.industry.gov.au/publications/national-quantum-strategy

[262] S. Pirandola, U. L. Andersen, L. Banchi, M. Berta, D. Bunandar, R. Colbeck, D. Englund, T. Gehring, C. Lupo, C. Ottaviani, J. L. Pereira, M. Razavi, J. Shamsul Shaari, M. Tomamichel, V. C. Usenko, G. Vallone, P. Villoresi, and P. Wallden, "Advances in quantum cryptography," *Advances in Optics and Photonics*, vol. 12, no. 4, p. 1012, Dec. 2020.

[263] A. M. Dalzell, S. McArdle, M. Berta, P. Bienias, C.-F. Chen, A. Gilyén, C. T. Hann, M. J. Kastoryano, E. T. Khabiboulline, A. Kubica, G. Salton, S. Wang, and F. G. S. L. Brandão, "Quantum algorithms: A survey of applications and end-to-end complexities," Oct. 2023, arXiv:2310.03011 [quant-ph]. [Online]. Available: http://arxiv.org/abs/2310.03011

[264] H. v. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, ser. AAAI'16. Phoenix, Arizona: AAAI Press, 2016, pp. 2094–2100.

[265] D. Yi, X. Zhou, Y. Wen, and R. Tan, "Efficient Compute-Intensive Job Allocation in Data Centers via Deep Reinforcement Learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 6, pp. 1474–1485, Jun. 2020.

[266] M. Goudarzi, M. S. Palaniswami, and R. Buyya, "A Distributed Deep Reinforcement Learning Technique for Application Placement in Edge and Fog Computing Environments," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2021.

[267] M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining Improvements in Deep Reinforcement Learning," Oct. 2017, arXiv:1710.02298 [cs]. [Online]. Available: http://arxiv.org/abs/1710.02298

[268] Z. Chen, J. Hu, G. Min, C. Luo, and T. El-Ghazawi, "Adaptive and Efficient Resource Allocation in Cloud Datacenters Using Actor-Critic Deep Reinforcement Learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 8, pp. 1911–1923, Aug. 2022.

[269] R. Kaewpuang, M. Xu, D. Niyato, H. Yu, Z. Xiong, and J. Kang, "Stochastic qubit resource allocation for quantum cloud computing," in *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*. Miami, FL, USA: IEEE, May 2023, pp. 1–5.

[270] N. Ngoenriang, M. Xu, S. Supittayapornpong, D. Niyato, H. Yu, Xuemin, and Shen, "Optimal stochastic resource allocation for distributed quantum computing," Sep. 2022, arXiv:2210.02886 [cs]. [Online]. Available: http://arxiv.org/abs/2210.02886

[271] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[272] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," Dec. 2013, arXiv:1312.5602 [cs]. [Online]. Available: http://arxiv.org/abs/1312.5602

[273] P. Moritz, R. Nishihara, S. Wang, A. Tumanov, R. Liaw, E. Liang, M. Elibol, Yang, M. I. Jordan, and I. Stoica, "Ray: a distributed framework for emerging AI applications," in *Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, Carlsbad, CA, USA, Oct. 2018, pp. 561–577.

[274] K. De Asis, J. F. Hernandez-Garcia, G. Z. Holland, and R. S. Sutton, "Multi-step reinforcement learning: a unifying algorithm," in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence and 30th Innovative Applications of Artificial Intelligence Conference and 8th AAAI Symposium on Educational Advances in Artificial Intelligence*. New Orleans, Louisiana, USA: AAAI Press, 2018.

[275] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17. Sydney, NSW, Australia: JMLR, 2017, pp. 449–458.

[276] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *Proceedings of the 4th International Conference on Learning Representations, ICLR 2016*, San Juan, Puerto Rico, 2016.

[277] M. Fortunato, M. G. Azar, B. Piot, J. Menick, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin, C. Blundell, and S. Legg, "Noisy networks for exploration," in *Proceedings of the 6th International Conference on Learning Representations, ICLR 2018*, Vancouver, Canada, 2018.

[278] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica, "Tune: a research platform for distributed model selection and training," in *ICML 2018 AutoML Workshop*. Stockholm, Sweden: PMLR, Jul. 2018, arXiv:1807.05118 [cs].

[279] H. Abraham, AduOffei, R. Agarwal, G. Agliardi, M. Aharoni, I. Y. Akhalwaya, G. Aleksandrowicz, T. Alexander, M. Amy, S. Anagolum, E. Arbel, A. Asfaw, A. Athalye, A. Avkhadiev, C. Azaustre, and et al., "Qiskit: An Open-source Framework for Quantum Computing," 2021. [Online]. Available: https://github.com/Qiskit

[280] E. Liang, R. Liaw, P. Moritz, R. Nishihara, R. Fox, K. Goldberg, J. E. Gonzalez, M. I. Jordan, and I. Stoica, "RLlib: abstractions for distributed reinforcement learning," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80. Stockholm, Sweden: PMLR, 2018.

[281] Y. Fan, B. Li, D. Favorite, N. Singh, T. Childers, P. Rich, W. Allcock, M. E. Papka, and Z. Lan, "DRAS: Deep Reinforcement Learning for Cluster Scheduling in High Performance Computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 4903–4917, Dec. 2022.

[282] C. Portmann and R. Renner, "Security in quantum cryptography," *Reviews of Modern Physics*, vol. 94, no. 2, Jun. 2022.

[283] N. Saurabh, S. Jha, and A. Luckow, "A Conceptual Architecture for a Quantum-HPC Middleware," in *2023 IEEE International Conference on Quantum Software (QSW)*. Chicago, IL, USA: IEEE, Jul. 2023, pp. 116–127.

[284] T. Beck, A. Baroni, R. Bennink, G. Buchs, E. A. C. Pérez, M. Eisenbach, R. F. Da Silva, M. G. Meena, K. Gottiparthi, P. Groszkowski, T. S. Humble, R. Landfield, K. Maheshwari, S. Oral, M. A. Sandoval, A. Shehata, I.-S. Suh, and C. Zimmer, "Integrating quantum computing resources into scientific HPC ecosystems," *Future Generation Computer Systems*, vol. 161, pp. 11–25, Dec. 2024.

[285] A. Kurlej, S. Alterman, and K. Obenland, "Performance of algorithms for emerging ion-trap quantum hardware," *Future Generation Computer Systems*, vol. 160, pp. 654–665, Nov. 2024.

[286] J.-S. Chen, E. Nielsen, M. Ebert, V. Inlek, K. Wright, V. Chaplin, A. Maksymov, E. Páez, A. Poudel, P. Maunz, and J. Gamble, "Benchmarking a trapped-ion quantum computer with 29 algorithmic qubits," Aug. 2023, arXiv:2308.05071 [quant-ph]. [Online]. Available: http://arxiv.org/abs/2308.05071

[287] A. Esposito, J. R. Jones, S. Cabaniols, and D. Brayford, "A Hybrid Classical-Quantum HPC Workload," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*. Bellevue, WA, USA: IEEE, Sep. 2023, pp. 117–121.

[288] J. Li, Y. Song, Y. Liu, J. Pan, L. Yang, T. Humble, and W. Jiang, "QuSplit: Achieving Both High Fidelity and Throughput via Job Splitting on Noisy Quantum Computers," Jan. 2025, arXiv:2501.12492 [quant-ph]. [Online]. Available: http://arxiv.org/abs/2501.12492

[289] S. Zhang, C. Wang, and A. Y. Zomaya, "Robustness Analysis and Enhancement of Deep Reinforcement Learning-based Schedulers," *IEEE Transactions on Parallel and Distributed Systems*, pp. 1–12, 2022.

[290] W. Luo, J. Zhao, T. Zhan, and Q. Guan, "Adaptive Job Scheduling in Quantum Clouds Using Reinforcement Learning," Jun. 2025, arXiv:2506.10889 [cs]. [Online]. Available: http://arxiv.org/abs/2506.10889

[291] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," Aug. 2017, arXiv:1707.06347 [cs]. [Online]. Available: http://arxiv.org/abs/1707.06347

[292] M. Kordzanganeh, M. Buchberger, B. Kyriacou, M. Povolotskii, W. Fischer, A. Kurkin, W. Somogyi, A. Sagingalieva, M. Pflitsch, and A. Melnikov, "Benchmarking Simulated and Physical Quantum Processing Units Using Quantum and Hybrid Algorithms," *Advanced Quantum Technologies*, vol. 6, no. 8, p. 2300043, Aug. 2023.

[293] A. Cross, A. Javadi-Abhari, T. Alexander, N. De Beaudrap, L. S. Bishop, S. Heidel, C. A. Ryan, P. Sivarajah, J. Smolin, J. M. Gambetta, and B. R. Johnson, "OpenQASM 3: A Broader and Deeper Quantum Assembly Language," *ACM Transactions on Quantum Computing*, vol. 3, no. 3, pp. 1–50, Sep. 2022.

[294] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," Oct. 2018, arXiv:1506.02438 [cs]. [Online]. Available: http://arxiv.org/abs/1506.02438

[295] D. Bruneo, "A stochastic model to investigate data center performance and QoS in IaaS cloud computing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 3, pp. 560–569, Mar. 2014.

[296] M. S. Yoon, A. E. Kamal, and Z. Zhu, "Adaptive data center activation with user request prediction," *Computer Networks*, vol. 122, pp. 191–204, Jul. 2017.

[297] O. Bel and M. Kiran, "Simulators for quantum network modeling: a comprehensive review," *Computer Networks*, vol. 263, p. 111204, May 2025.

[298] B. Wickremasinghe, R. N. Calheiros, and R. Buyya, "CloudAnalyst: a CloudSim-based visual modeller for analysing cloud computing environments and applica-

tions," in *2010 24th IEEE International Conference on Advanced Information Networking and Applications*. Perth, Australia: IEEE, 2010, pp. 446–452.

[299] B. Johnson, I. Faro, M. Behrendt, and J. Gambetta, "Introducing Quantum Serverless, a new programming model for leveraging quantum and classical resources," May 2021. [Online]. Available: https://research.ibm.com/blog/quantum-serverless-programming

[300] A. Eddins, M. Motta, T. P. Gujarati, S. Bravyi, A. Mezzacapo, C. Hadfield, and S. Sheldon, "Doubling the Size of Quantum Simulators by Entanglement Forging," *PRX Quantum*, vol. 3, no. 1, p. 010309, Jan. 2022.

[301] Z. Davarzani, M. Zomorodi, and M. Houshmand, "A hierarchical approach for building distributed quantum systems," *Scientific Reports*, vol. 12, no. 1, p. 15421, Sep. 2022.

[302] L. Gyongyosi and S. Imre, "Scalable distributed gate-model quantum computers," *Scientific Reports*, vol. 11, no. 1, p. 5172, Feb. 2021.

[303] G. Yuan, Y. Chen, J. Lu, S. Wu, Z. Ye, L. Qian, and G. Chen, "Quantum computing for databases: overview and challenges," 2024, arXiv: 2405.12511 [cs.DB]. [Online]. Available: https://arxiv.org/abs/2405.12511

[304] M. Salam and M. Ilyas, "Quantum computing challenges and solutions in software industry—a multivocal literature review," *IET Quantum Communication*, vol. Not available, p. Not available, Jun. 2024.

[305] D. Kreuzberger, N. Kühl, and S. Hirschl, "Machine Learning Operations (MLOps): Overview, Definition, and Architecture," *IEEE Access*, vol. 11, pp. 31 866–31 879, 2023.

[306] E. H. Houssein, Z. Abohashima, M. Elhoseny, and W. M. Mohamed, "Machine learning in the quantum realm: The state-of-the-art, challenges, and future vision," *Expert Systems with Applications*, vol. 194, p. 116512, May 2022.

[307] N. Dupuis, L. Buratti, S. Vishwakarma, A. V. Forrat, D. Kremer, I. Faro, R. Puri, and J. Cruz-Benito, "Qiskit code assistant: training LLMs for generating quantum

computing code," May 2024, arXiv:2405.19495 [quant-ph]. [Online]. Available: http://arxiv.org/abs/2405.19495

[308] L. Jern, V. Uotila, C. Yu, and B. Zhao, "Agent-Q: fine-tuning large language models for quantum circuit generation and optimization," Sep. 2025, arXiv:2504.11109 [quant-ph]. [Online]. Available: http://arxiv.org/abs/2504.11109

[309] C. Campbell, H. M. Chen, W. Luk, and H. Fan, "Enhancing LLM-based quantum code generation with multi-agent optimization and quantum error correction," Jul. 2025, arXiv:2504.14557 [quant-ph]. [Online]. Available: http://arxiv.org/abs/2504.14557

[310] D. Van Huynh, O. A. Dobre, and T. Q. Duong, "Optimal service placement for 6G edge computing with quantum-centric optimization in real quantum hardware," *IEEE Communications Letters*, vol. 29, no. 3, pp. 448–452, Mar. 2025.

[311] C. Mastroianni, F. Plastina, J. Settino, and A. Vinci, "Variational Quantum Algorithms for the Allocation of Resources in a Cloud/Edge Architecture," *IEEE Transactions on Quantum Engineering*, vol. 5, pp. 1–18, 2024.

[312] S. Chen, Z. Li, B. Yang, and G. Rudolph, "Quantum-Inspired Hyper-Heuristics for Energy-Aware Scheduling on Heterogeneous Computing Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 6, pp. 1796–1810, Jun. 2016.

[313] M. Mehic, M. Niemiec, S. Rass, J. Ma, M. Peev, A. Aguado, V. Martin, S. Schauer, A. Poppe, C. Pacher, and M. Voznak, "Quantum Key Distribution: A Networking Perspective," *ACM Computing Surveys*, vol. 53, no. 5, 2020.

[314] Y. Cao, Y. Zhao, Q. Wang, J. Zhang, and Ng, "The Evolution of Quantum Key Distribution Networks: On the Road to the Qinternet," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 2, pp. 839–894, 2022.